U20CS855 MOBILE AND PERVASIVE COMPUTING

OBJECTIVES:

- To understand the fundamental concepts of mobile computing
- To acquire knowledge on mobile technologies and networking
- To know the essentials of pervasive computing

PREREQUISITE:

There are many benefits to **mobile computing** including the ability to get directions, entertain yourself when bored, do business, and more, including: Connectivity.

UNIT I PERVASIVE COMPUTING 9

Basics and vision – Architecture and Applications requirements – Smart devices and operating systems, secure services – Smart mobiles, cards and device networks.

UNIT II MOBILE APPLICATIONS 9

History – Mobile Ecosystem – Designing for context – Mobile strategy – Mobile applications – Information Architecture – Design – Mobile Web apps vs Native Apps – Adapting to devices – Supporting devices – Application development on Android and iPhone.

UNIT III MEDIUM ACCESS AND TELECOMMUNICATIONS 9

Frequencies – Signals – Antennas – Signal propagation – Media Access Control: Motivation, SDMA, FDMA, TDMA, CDMA – GSM: Mobile services, System architecture, Protocols, Localization and calling, Handover – GPRS.

UNIT IV WIRELESS NETWORKS 9

Infrared vs radio transmission – Infrastructure and ad hoc networks – WLAN, IEEE 802.11 standards protocols. Piconet- Bluetooth-architecture and services.Wireless Broadband networks and satellites networks.

UNIT V MOBILE NETWORK AND TRANSPORT LAYERS 9

Mobile IP – DHCP – Routing in Mobile ad hoc networks – TCP improvements – TCP over 2.5/3G.

TOTAL: 45 PERIODS

OUTCOMES:

- Know the basics of mobile computing with its security and standards
- Explicate the emerging mobile technologies
- Describe the concept of WLAN and mobile network
- Identify the fundamentals of pervasive computing
- Elucidate the applications and operating systems of pervasive computing

TEXT BOOKS:

1. Stefan Poslad, “Ubiquitous Computing: Smart Devices, Environments and Interactions”, Wiley, 2009.
2. Brian Fling, “Mobile Design and Development”, O’Reily, 2009.
3. Jochen Schiller, “Mobile Communications”, 2nd ed., Pearson Education, 2003.

REFERENCES:

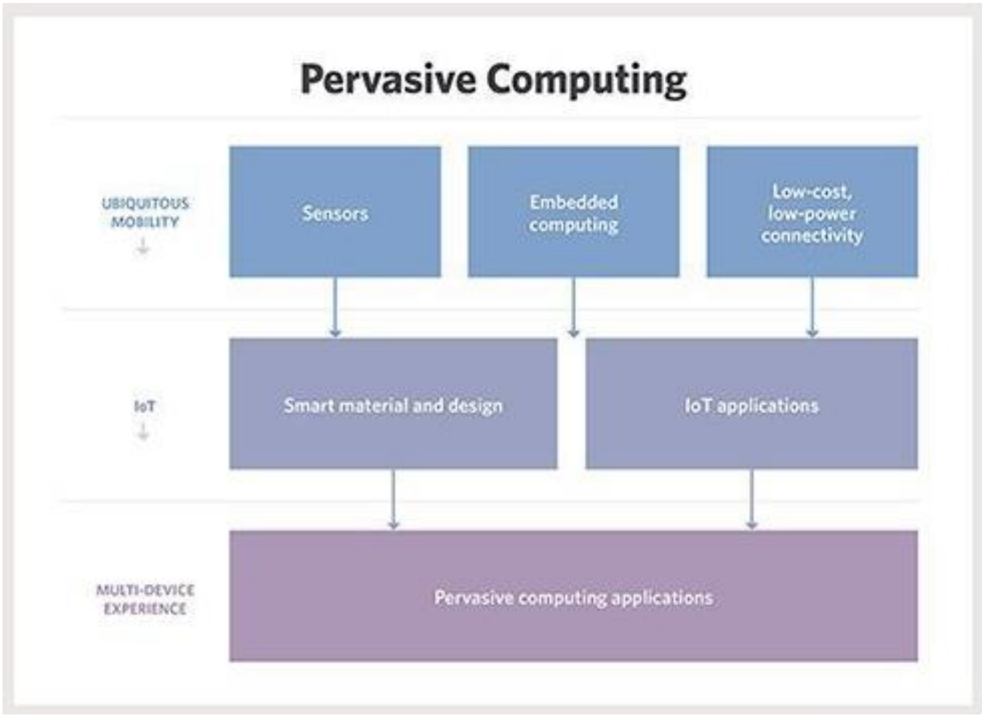
1. ZigurdMednieks, Laird Dornin, G,BlakeMeike and Masumi Nakamura “Programming Android”, O’Reilly, 2011.
2. Reto Meier, “Professional Android 2 Application Development”, Wrox Wiley, 2010.
3. Alasdair Allan, “iPhone Programming”, O’Reilly, 2010.
4. Wei-Meng Lee, “Beginning iPhone SDK Progrmming with Objective-C”, Wrox Wiley, 2010.
5. Asoke K Talukder, Hasan Ahmed, Roop R Yavagal, “Mobile Computing”, 2nd ed, Tata McGraw Hill, 2010.
6. Pei Zheng, Lionel M. Ni, “Smart Phone & Next Generation Mobile Computing”, Morgan Kaufmann, 2006.
7. Frank Adelstein, Sandeep KS Gupta, Golden Richard, “Fundamentals of Mobile and Pervasive Computing”, Tata McGraw-Hill, 2005.
8. UweHansmann, LotharMerk, Martin S. Nicklons and Thomas Stober, “Principles of Mobile Computing”, Springer, 2003.
9. JochenBurkhardt et al, Pervasive Computing: Technology and Architecture of Mobile Internet Applications, Pearson Education, 2002.

UNIT I PERVASIVE COMPUTING

Basics and vision – Architecture and Applications requirements – Smart devices and operating systems, secure services – Smart mobiles, cards and device networks.

1.1 Basics

Pervasive computing, or ubiquitous computing, integrates connectivity functionalities into all of the objects in our environment so they can interact with one another, automate routine tasks, and require minimal human effort to complete tasks and follow machine instructions.



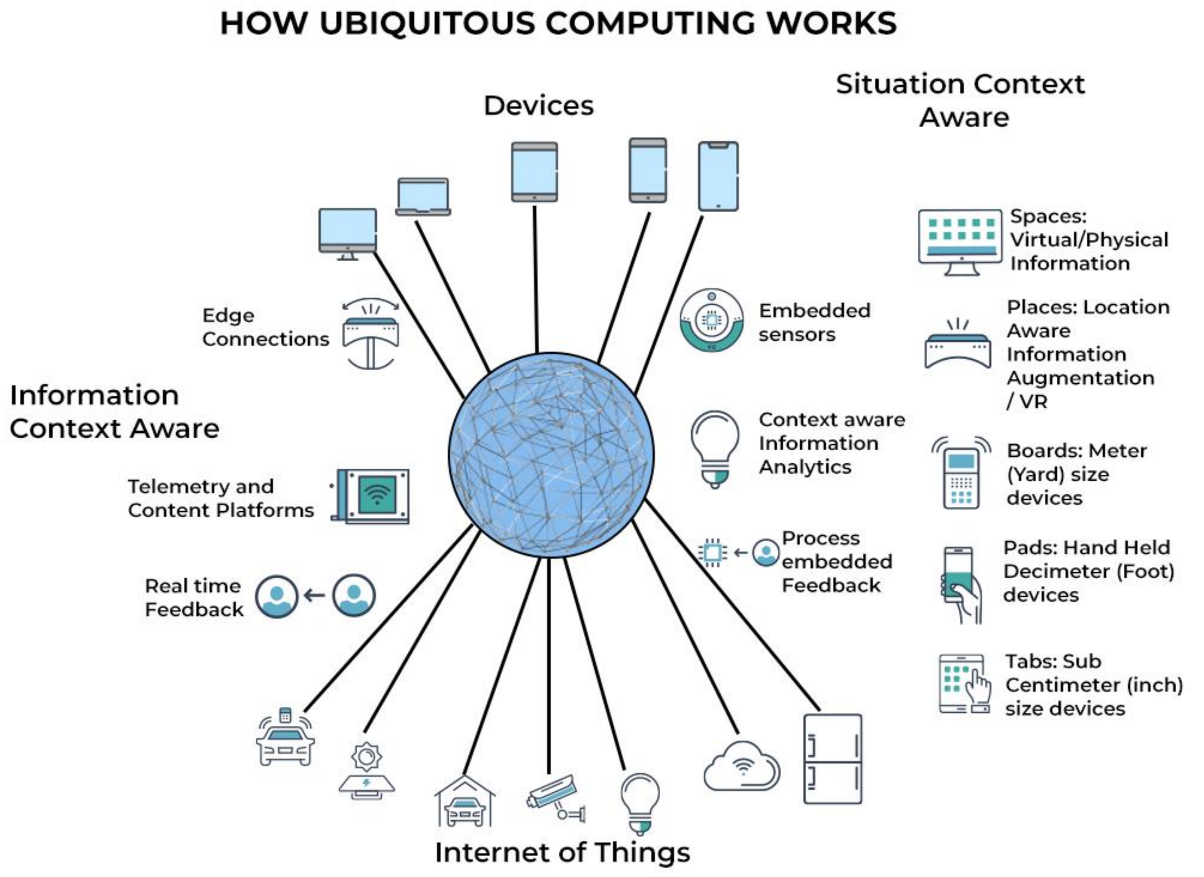
Pervasive computing, also known as ubiquitous computing, refers to the growing pattern of embedding computing capacity (generally in the form of microprocessors) into ordinary items to make them communicate effectively and accomplish helpful tasks while reducing the end subscriber's need to communicate with computers. Pervasive computing tools are network-connected and always on.

The late 1980s were the birthplace of ubiquitous computing. The term "ubiquitous computing" was coined in 1988 by Mark Weiser, CTO at Xerox PARC (or the Palo Alto Research Center). He co-authored some of the first studies on the subject alongside John Seely Brown, chief scientist and director of PARC. Weiser is credited for coining the term "ubiquitous computing" and diving into its underlying difficulties.

The proliferation of interconnected devices in work, home, and transportation environments is called ubiquitous computing or ambient computing. These integrated technologies would make these settings and transportation methods far more engaging and practical via contextual data collection, application, and seamless payment mechanisms.

This is a system in which computer network technology permits itself to exist discreetly in the background of the consumers' awareness. The seamless integration of computers into our regular activities and physical settings is a core concept in the ubiquitous computing paradigm. Ubiquitous computing tries to make the computer "invisible" by enabling these embedded processors to detect and respond to their application environment autonomously.

Pervasive computing, as opposed to desktop computing, may operate with any device, anytime, in any location, and datatype across networks and can transfer duties from one computer to another when a consumer walks from vehicles to the workplace. Laptops, notebooks, smartphones, tablets, wearable devices, and sensors are ubiquitous computing devices (fleet management and pipeline components, lighting systems, and appliances).


HOW UBIQUITOUS COMPUTING WORKS
Devices
Situation Context Aware
Spaces: Virtual/Physical Information
Edge Connections
Embedded sensors
Places: Location Aware Information Augmentation / VR
Information Context Aware
Context aware Information Analytics
Boards: Meter (Yard) size devices
Telemetry and Content Platforms
Process embedded Feedback
Pads: Hand Held Decimeter (Foot) devices
Real time Feedback
Tabs: Sub Centimeter (inch) size devices
Internet of Things

A great example of a ubiquitous computing system is an autonomous vehicle that recognizes its authorized passenger via smartphone vicinity, docks and charges itself when required, and handles the emergency response, toll, and fast-food payments efficiently by interfacing with the infrastructure. It entails linking electrical equipment, including installing microprocessors to transfer data. Devices that employ ubiquitous computing are always available and fully connected.

Ubiquitous computing keeps a close focus on learning by decreasing computer complexity and enhancing efficiency while utilizing computing for everyday activities. Ubiquitous computing, which is sometimes regarded as the successor to mobile computing, usually entails wireless communication and networking technologies, mobile devices, embedded systems, wearable computers, radio frequency ID (RFID) tags, middleware, and software agents.

Internet connectivity, speech recognition, and artificial intelligence (AI) features are frequently added. It incorporates computers into everyday items, allowing people to connect with information-processing equipment more easily and freely than they do now, regardless of place or context.

The expression is further described as a computing environment where each instructor and student has their own internet-connected, private mobile computing device that they may use at home and in the classroom. While computers were formerly large and heavy equipment, ubiquitous computing is based on the downsizing of electronics, which allows for the integration of computer technologies into lightweight handheld devices and various other ubiquitously emerging situations.

1.2 Vision

The vision of pervasive computing is to enhance our lives in various ways:

- **Improved efficiency and productivity:** Automation and intelligence can streamline tasks and optimize resource use.
- **Enhanced healthcare and well-being:** Sensors and monitoring systems can track health metrics, provide personalized care, and assist in emergencies.
- **Smarter cities and infrastructure:** Traffic management, energy efficiency, and disaster response can be improved through interconnected systems.
- **Personalized experiences:** Technology can adapt to individual needs and preferences, creating a more comfortable and convenient environment.

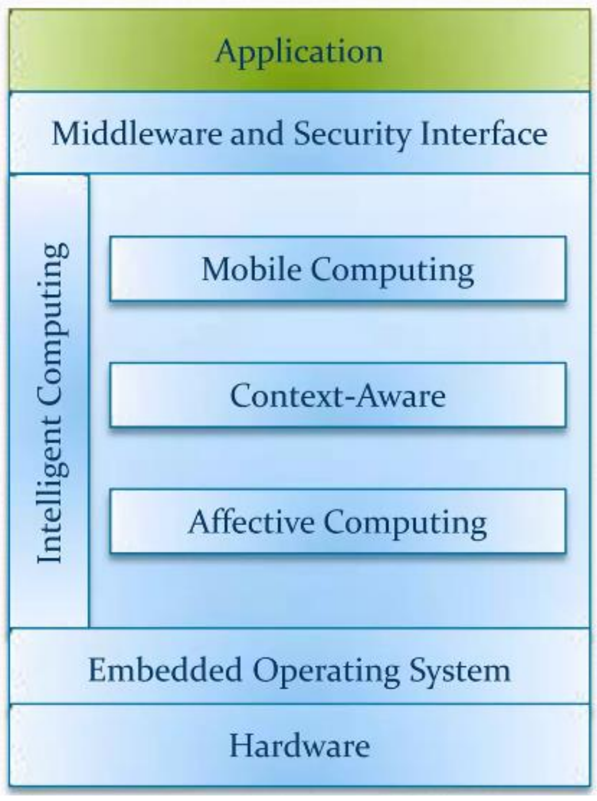
1.3 Characteristics of pervasive computing:

- **Miniaturization and low power consumption:** Tiny sensors and chips can be embedded into almost anything.

- **Wireless communication:** Devices can seamlessly connect and share data without wires.
- **Context awareness:** Systems can understand their environment and user behavior to adapt and provide relevant assistance.
- **Invisibility:** Technology seamlessly blends into the background, becoming an extension of our natural environment.

1.4 Architecture of Pervasive Computing

The architecture of pervasive computing is multifaceted and can be viewed from different layers, but it generally consists of five main components:

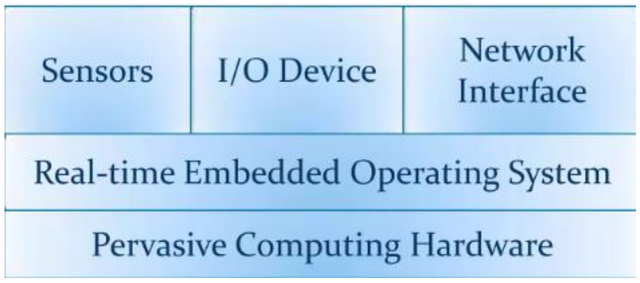


1.4.1. Sensing Layer:

This layer comprises various sensors and actuators embedded in the environment and devices. These sensors collect data about the physical world, such as temperature, light, movement, and sound. Actuators, on the other hand, can perform actions based on the collected data, like controlling lights, adjusting temperature, or triggering alarms. Examples of sensing devices include:

- **Temperature sensors:** Used in smart thermostats and wearables.
- **Motion sensors:** Used in security systems and automatic lighting.

- **Cameras:** Used for video surveillance, facial recognition, and gesture recognition.
- **Microphones:** Used for voice commands and environmental monitoring.


Sensors
I/O Device
Network Interface
Real-time Embedded Operating System
Pervasive Computing Hardware

1.4.2. Communication Layer:

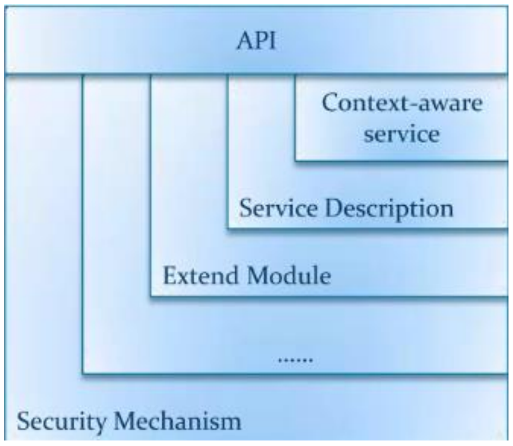
This layer enables data exchange between different devices and components of the system. It utilizes various communication protocols, including:

- **Wireless technologies:** Wi-Fi, Bluetooth, RFID, and cellular networks.
- **Wired connections:** Ethernet and USB for high-bandwidth applications.
- **Short-range communication:** NFC for data exchange between devices in close proximity.

1.4.3. Middleware Layer:

This layer acts as a bridge between the sensing and processing layers. It performs various tasks, including:

- **Data filtering and aggregation:** Preprocessing and preparing raw data for further analysis.
- **Context awareness:** Understanding the context in which the data was collected to provide relevant information or services.
- **Resource management:** Optimizing resource allocation and communication among devices.
- **Security and privacy:** Ensuring secure transmission and storage of data.



1.4.4. Processing Layer:

This layer is responsible for analyzing the collected data and making intelligent decisions. It can involve:

- **Edge computing:** Performing basic processing on devices at the edge of the network for real-time response.
- **Cloud computing:** Offloading complex computations to powerful servers in the cloud for further analysis and decision-making.
- **Machine learning and artificial intelligence:** Utilizing algorithms to learn from data and make predictions or recommendations.

1.4.5. Application Layer:

This layer sits at the top of the architecture and interacts directly with users. It includes various applications and services that utilize the data and insights generated by the lower layers. Examples include:

- **Smart home applications:** Controlling lights, temperature, appliances, and security systems.
- **Wearable devices:** Tracking health metrics, providing fitness coaching, and enabling hands-free interactions.
- **Context-aware advertising and recommendations:** Delivering personalized information and services based on user needs and preferences.
- **Smart transportation systems:** Optimizing traffic flow, providing real-time travel information, and enabling autonomous vehicles.

1.5 Applications Requirements of Pervasive Computing

Pervasive computing applications have unique requirements due to their inherent characteristics of being embedded, interconnected, and context-aware. These requirements can be broadly categorized into four main areas:

1. Technical Requirements:

- **Miniaturization:** Sensors, actuators, and processing units need to be small and energy-efficient to be embedded in everyday objects.
- **Connectivity:** Reliable and secure communication channels are crucial for data exchange between devices and the cloud.

- **Heterogeneity:** Systems must be able to seamlessly integrate and communicate with various devices and platforms, regardless of their underlying technology.
- **Real-time processing:** Certain applications require real-time data analysis and response, demanding efficient processing power at the edge or in the cloud.
- **Context awareness:** Applications need to understand the surrounding environment and user behavior to provide relevant and personalized services.

2. Security and Privacy Requirements:

- **Data protection:** Secure storage and transmission of sensitive data collected by sensors and actuators are paramount.
- Privacy preservation: User privacy needs to be protected through anonymization, access control, and other mechanisms.
- Trustworthiness: Users must trust the system's reliability, accuracy, and fairness in its decision-making processes.
- Transparency and control: Users should be informed about how their data is collected and used, and have control over their privacy settings.

3. Quality of Service Requirements:

- **Availability:** Systems need to be highly available and reliable, with minimal downtime or service disruptions.
- **Scalability:** Applications should be able to adapt and scale to handle increasing data volumes and user demands.
- **Responsiveness:** Systems must respond to user interactions and events in a timely and efficient manner.
- **Accuracy and reliability:** Data collected and analyzed by the system needs to be accurate and reliable to ensure valid decision-making and service delivery.

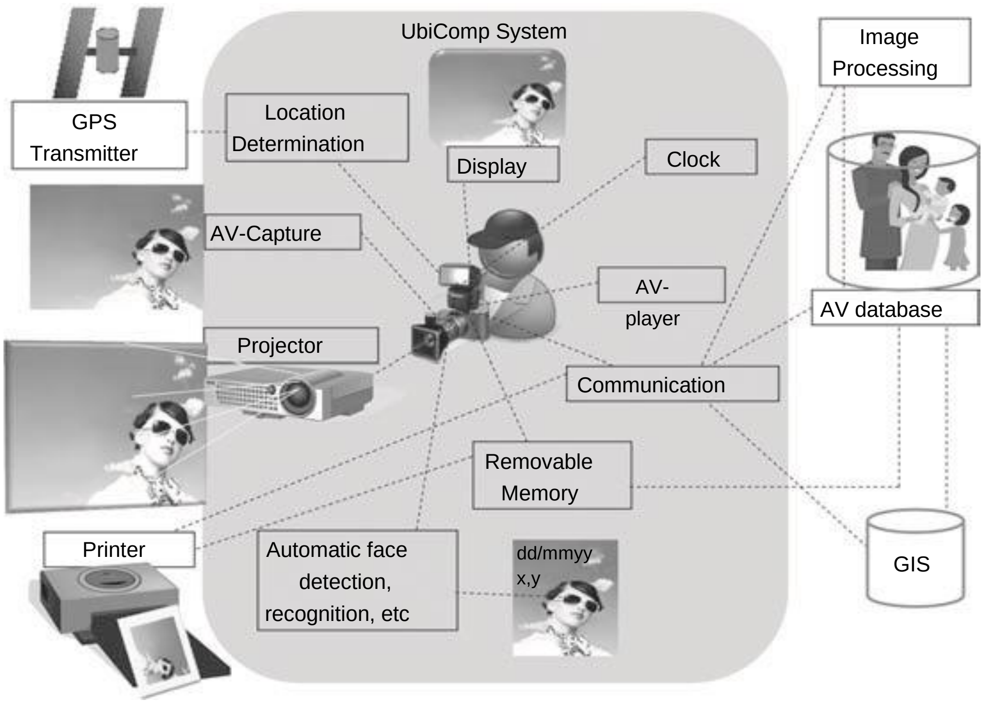
4. User Experience Requirements:

- **Seamlessness and transparency:** Technology should blend seamlessly into the background and not intrusively disrupt the user experience.
- **Personalization:** Applications should adapt to individual preferences and needs, providing a personalized experience.
- **Learnability and usability:** Interfaces and interactions should be intuitive and easy to

learn and use for users of diverse technical backgrounds.

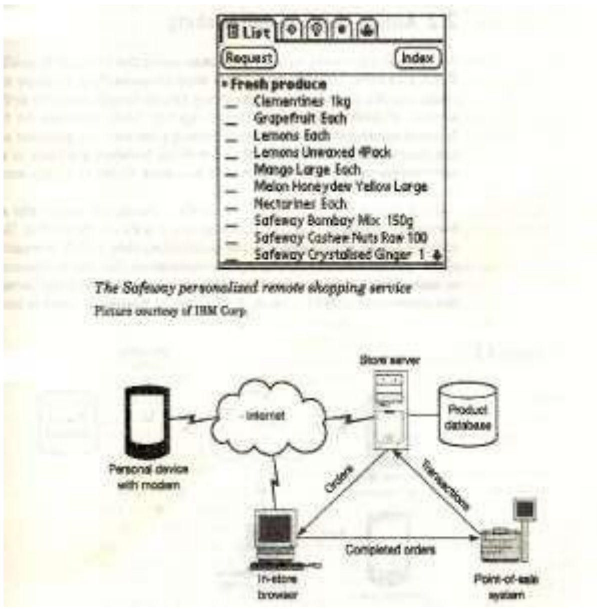
- **Affordability and accessibility:** Pervasive computing solutions need to be affordable and accessible to a broad range of users.

1.6 Applications of Pervasive Computing



Example of a pervasive computing application

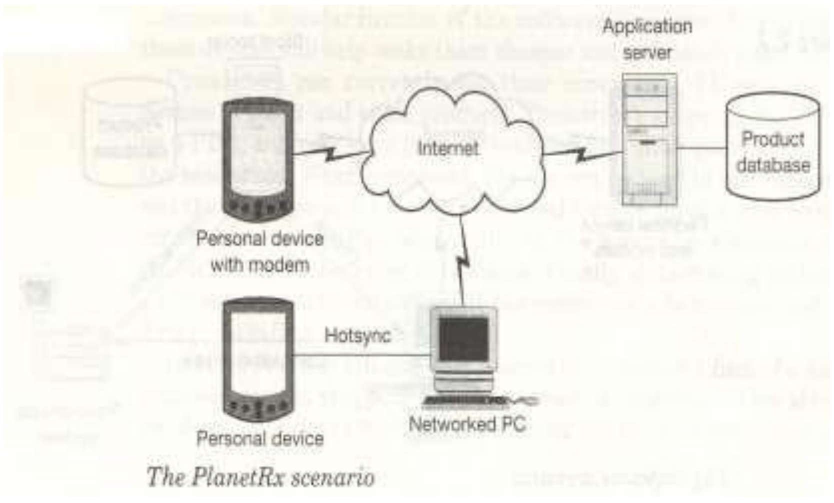
1.6.1 Retail


List
Request
Index
• Fresh produce
Clementines 1kg
Grapefruit Each
Lemons Each
Lemons Unwaxed 4Pack
Mango Large Each
Melon Honeydew Yellow Large
Nectarines Each
Safeway Bombay Mix 150g
Safeway Cashew Nuts Raw 100
Safeway Crystalised Ginger 1

The Safeway personalized remote shopping service
Picture courtesy of IBM Corp.

Retail

1.6.2 Air Line Check in and Booking:



Air Line in and Booking

1.6.3 Sales Force Automation:

- Mobile workers relied on their portable computers in order to access and process data on the road
- Availability of wireless modems has enabled them to travel and allow them to access to enterprise data
- Mobile professionals to use the phone book and calendar while working out of office and stay in contact via email.
- Used to access mission-critical data – contracts, technical descriptions, small database and graphics etc at anytime & anywhere.
- Used to control the delivery of goods, update work assignment, submit orders, even billing info while on road.

1.6.4 Healthcare:

- Modern medicine already depends on a wide range of computerized devices, sensors, actors.
- Clinical professional learn about new methods and how to use them
- Access to laboratory results and surgical reports as well as ordering processes and physician directory look ups can be improved
- With respect to security, integrity of data and privacy assured though hardware, software and system design.
- Identification of patients at all levels of treatment and medical record keeping

- Patient and clinical data must be exchangeable and accessible whenever needed but with restriction
- Modern healthcare systems are now using smart cards for patients and professionals.

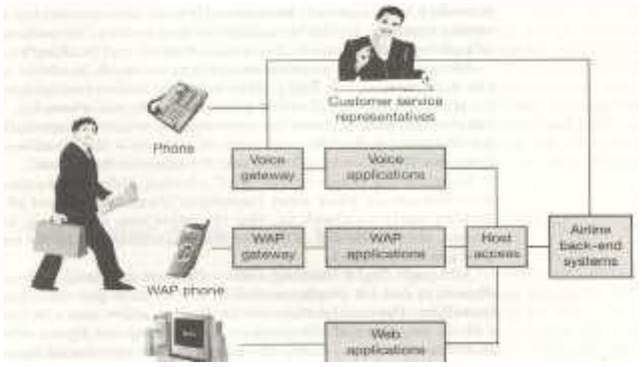
1.6.5 Tracking:

- Use of barcodes has revolutionized processes in many industries.
- Provide fast and accurate identification of goods during transportation.
- Barcodes on all products.
- One dimensional barcode – encodes only few characters.
- Two dimensional barcode – several hundreds of characters of information to be stored..
- Allows tracking of goods – eg- airline luggage.
- Tags are cheaper attached to each luggage and detected at certain points on the journey – enables airline to track individual pieces of luggage from check-in to baggage claim.

1.6.6 Car Information System:

Car manufacturers are very interested in using pervasive computing technology. Nowadays cars are equipped with more than 30 microprocessors. To connect the car with outside world, OGSI (open Service Gateway Initiative) is an industry group which defines features such as downloading software, application lifecycle management and gateway circuitry etc, which is implemented in JAVA,

1.6.7 Email Access via WAP and Voice:



Email Access via WAP and Voice

1.7 Smart devices

Smart devices are characterized by the ability to execute multiple, possibly concurrent, applications, supporting different degrees of mobility and customization and by supporting intermittent remote service access and operating according to local resource constraints.

Smart devices tend be owned, operated, configured and under the control of individual human users, e.g., personal computers (PC), phones, cameras, games consoles, set top boxes and other computer peripheral devices such as printers, external disk drives, etc.

Some of the more complex devices such as laptops can themselves be considered as aggregates of several smart devices.

1.7.1 Characteristics of service access used by smart devices

Feature	Requirements	Design
General Characteristics for smart devices, environments and interaction		
Multiple applications accessed via a range of devices	Devices execute multiple local processes; act as portals to access preconfigured sets of multiple remote services	Support different mixes of local versus remote processing
Simple service access	Services should be simple to initiate and to operate	Modularisation and abstraction of ICT resources
Minimum configuration and maintenance	Device access to local resources and remote services.	Self discovery networks, services, etc. Use of appliance, utility self management, etc models
Shared resources & services	Support access to resources by multiple users	Concurrency control
Internal Smart Device Properties		
Open, Interoperable, Heterogeneous	Users prefer a choice of service instances, access devices and networks	Use virtualisation and mediation combined with abstraction to support these
Dynamic service provision	Discover, select, compose & invoke services anywhere, anytime from anywhere	Discover, compose, invoke & maintain services

	relevant	
Mobility	Range of mobile services and users depending on the application	Mobility support to: route to mobile receivers; to discover mobile access nodes
Volatile remote service access	Intermittent access is due to network failures, dynamic service access, concurrency, upgrading, etc.	Design to execute Self sufficiently and in an intermittent off line mode
Local resource constraints	Some types of mobile & embedded access devices have energy, data storage, display and input constraints	Adapt operation to limited ICT resource context, limited local energy self sufficiency
External System Interaction		
ICT Context aware	Awareness of local resource availability Volatile remote resource availability	OS versus application support
User context aware (Personalized)	Smart devices are often personalized but users may interact with many other impersonal services. User Context is dynamic varies with activity, time, etc.	Devices act as a source for a personal space that can expand into other devices. Support for dynamic direct, indirect user profiling
Physical Context aware	Devices are active, situated in a relatively passive physical world environment	Limited awareness of the physical world

1.7.2 Hardware and Battery

Today, lithium ion (Li ion) batteries can be found in all sorts of electronic equipment. These batteries are lighter and have better energy density, resulting in more power delivered. The weight of a NiCad battery for a five-year-old mobile phone is often higher than the total weight of a modern mobile phone, including the Li ion battery. While the latter does have a lower capacity, it still offers longer talk time because of the reduced power requirements of modern devices.

Table gives an estimate of the expected standby and talk time for a mobile phone when used with typical batteries available today. The data is taken from the specification of three batteries of comparable size. The latest in battery technology is the emergence of lithium polymer cells, which use a gel material for the electrolyte. The batteries are made from a few thin and flexible layers, and do not require a leakproof casing. This means the batteries can be made in almost any shape or size.

Expected lifetime for NiCad, NiMH, and Li ion batteries

Chemistry	**Standby time (h)**	**Talk time (m)**
NiCad	12-27	85-160
NiMH	16-37	110-210
Li ion	21-50	170-225

1.7.3 Displays

- LCDs are already replacing the bulky cathode ray tubes.
- Larger and more readable
- Dramatic weight, size, and power consumption benefits of LCD technology outweigh their relatively high cost.
- Today's PDAs usually feature dual-scan (DSTN) displays that control individual display elements via passive matrix addressing.
- This technology consumes considerably less power than the thin-film transistor (TFT) active matrix technology.
- This latter technology is more expensive, but is capable of significantly superior display performance and thus is generally used in portable computers.
- Better and thinner displays will be available in the future based on the light-emitting organic diode (OLED) or light-emitting polymer (LEP) technologies.
- OLED technology was invented about 15 years ago.
- It only recently became commercially attractive when the initial problems with the expected life and efficiency were solved.
- Instead of crystalline semiconductor material, organic compounds are used.
- The simplified manufacturing process of smaller structures and a rich selection of

organic compounds enable OLEDs to be built in almost any size and colour.

- This will eventually allow manufacturers to create extremely thin displays that are flexible enough to be bent and shaped as required.
- Other new display technologies, such as chip-on-glass (CoG) and liquidcrystal-on-glass (LCoG), integrate the picture elements with transistors on a layer of glass.
- This allows manufacturing of extremely small displays, with a pixel size of only 10 micrometers.
- In contrast to regular small displays like those on the back of a camcorder, the microdisplays usually require some form of magnification.
- They can be found, for example, in projection systems and in headmounted displays used with wearable computers.

1.7.4 Memory

- Memory is becoming cheaper, while the demand from applications is growing.
- Development is driven in part by smart phones, digital cameras, MP3 players and PDAs.
- For these mobile devices, the currently available technologies and their associated costs have reached a point where it is now feasible to integrate several megabytes of memory into a mobile device with an acceptable form factor.
- On PCs, permanent data can be stored on hard disk drives.
- For mobile devices, this is often not an option because neither the space nor the power supply is available.
- Recently, extremely small removable disk drives like the IBM Microdrive became available.
- Their capacity ranges between 340 MB and 1 GB, and is sufficient to store, for example, several hundred pictures when used in a digital camera.
- Other devices such as smart phones and PDAs store their operating system code and application data in non-volatile Flash memory and batterybacked random-access memory (RAM) instead.
- These semiconductor-based technologies require less power and offer faster access than disk drives.
- The typical capacity of built-in memory in mobile devices ranges from 2 to 16 MB.
- Expansion slots allow additional memory modules to be plugged into the device,

which in turn allow data exchange and replace removable media such as diskettes and CD-ROM for a PC.

1.7.5 Processors

- During the last couple of years, the clock rate of microprocessors and the processing power available from them has increased steadily.
- Rapid improvements in the CMOS manufacturing process have created
- ever-smaller structures and delivered higher and higher numbers of transistors per chip.
- At the same time, the processor core voltage was lowered from the industry standard 3.3 V in 1995 to 1.35 V in 2000.
- This means lower heat emissions, which in turn paves the way for new improvements like larger on-die caches.
- This, together with advances in packaging technologies, delivers the modern Central Processing Units (CPUs) found in mobile computers and PDAs today.

1.7.6 Intel's Speed Step technology

- Recent processors include improvements in power management.
- These processors are capable of changing their internal clock frequencies and core voltage to adapt to changes in power supply.
- Newer designs are even capable of switching parts of the CPU on or off depending on whether the current calculations require them to be available.
- One such design is the Speed Step technology from Intel.
- While the system is connected to an external power supply, the full clock rate and core voltage are available to the processor, resulting in the maximum performance.
- When running on batteries, the clock rate and core voltage of the processor are reduced, resulting in significant power savings.
- The transition between both modes is very fast and completely transparent to the user.
- During the boot cycle, the Crusoe processor loads its software into a section of the main memory.
- Frequently used code parts are optimized during run-time and kept in a separate cache.

- A technology called LongRun promises to reduce the power consumption even more by reducing the processor's voltage on the fly when the processor is idle.
- The big advantage of this approach is that the Crusoe processor can be used to emulate almost any other processor and uses only a few watts, even with high clock rates.

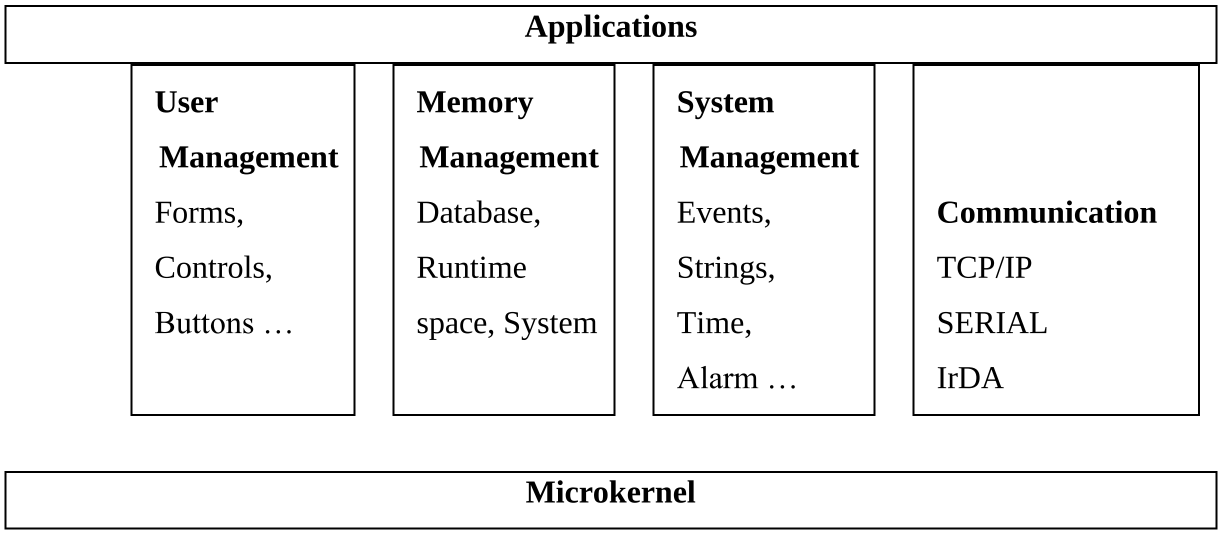
1.8 Operating systems

- The core functionality of every pervasive computing device is determined by its operating system.
- The major differences of operating systems for pervasive devices from the user's point of view are the human-machine interface, and the speed with which a task can be performed.
- For pervasive devices, there will likely be no equivalent to the Windows/Intel monopoly in the near future because pervasive devices do have a wide range of usages (from mobile phones to set-top boxes) with very constrained hardware

1.8.1 Palm OS:

Suitable and easy to use operating system for PDAs, optimized restricted features are available which leads to lower memory and CPU usage which results in longer battery life.

Features: Enhanced Communication Support and Multimedia with Mobile Phones



Palm OS

User Management:

- Single user operating system

Task Management:

- One application runs at a time and can call other applications.

Power Management:

- Power modes (sleep, doze and running)

OS size:

- OS 3.5 is about 1.4 MB.

User Interface:

- It recognizes only the palm handwriting alphabets, one button access to applications; minimize taps for often used operations.

Memory Management:

- Applications should be well tested since if one application crashes then the system crashes.
- Thus memory is divided into dynamic heap which is execution based and clears on reset and storage is designed to hold permanent data.
- Software can be developed with both C and C++ in Palm OS.

1.8.2 EPOC:

The EPOC operating system was designed specifically for phones. There are two versions: EPOC16 for 16-bit processors and EPOC32 for 32-bit processors.

Core operating system functionality:

- Heavily Multitasking.
- The base layer provides the fundamental APIs.
- The middleware layer provides the graphics, data, and other components to support the graphical user interface and applications.
- EIKON is the system graphical user interface framework.
- **User Management:**

  Single user operating system
- **Task management**

  Provides multitasking with a pre-emptive, priority-driven scheduler.

- **User interface**

  The EPOC user interface supports display, keyboard, and sound. . It is also responsible for handling the data and command input. Figure shows the EPOC user interface of an Ericsson device with a map application.

- **Memory management**

  EPOC has a memory management unit (MMU) concept to provide separate address spaces for each application. These tools include design patterns, stack clean-up heap failure, and heap-checking tools.
- Programming languages supported by EPOC are C++, Java and OPL. C++ used to develop system development and high performance application programming.

1.8.3 Window CE:

Windows CE is an embedded operating system developed by Microsoft. Windows CE 3.0 offers real-time support, a smart card subsystem for PC/SC compliant readers, is Unicode based, and supports grayscale and color graphics up to 32-bit depth.

Windows CE is a modular operating system that can be configured by the device manufacturer. This is a result of the read-only memory (ROM)-based design of Windows CE, in contrast to more desktop-oriented, disk-based operating systems like Linux or BeOS. It can even be configured at runtime.

- The kernel provides memory management, task scheduling, and interrupt handling.
- The graphics/window/event manager (GWE) integrates the user interface functions of graphical output and user input.
- The object store is the persistent memory of Windows CE and includes files, the registry, and a database.
- Finally, the communication interfaces include infrared communication via IrDA, TCP/IP, and serial drivers.
- **User management:**

  Because Windows CE is designed for PDAs, it supports only one user.
- **Task management:**

  The task manager supports 32 simultaneous processes and an unlimited number of threads.
- **Operating system size:**

  The Windows CE footprint can be as small as 400 kb for the kernel, up to 3 MB with all modules, and up to 8 MB including Pocket Word and Internet Explorer.
- **User interface:**

  Windows CE provides menu controls, dialog boxes, an: icons, and supports sound.
- **Memory management:**

A protected virtual memory system that supports up to 32 MB memory per process protects applications against each other. There exists a special heap for the file system, registry, and object store that has a transaction service for ensuring data integrity. The object store can have a size up to 256 MB.

- **Security:**

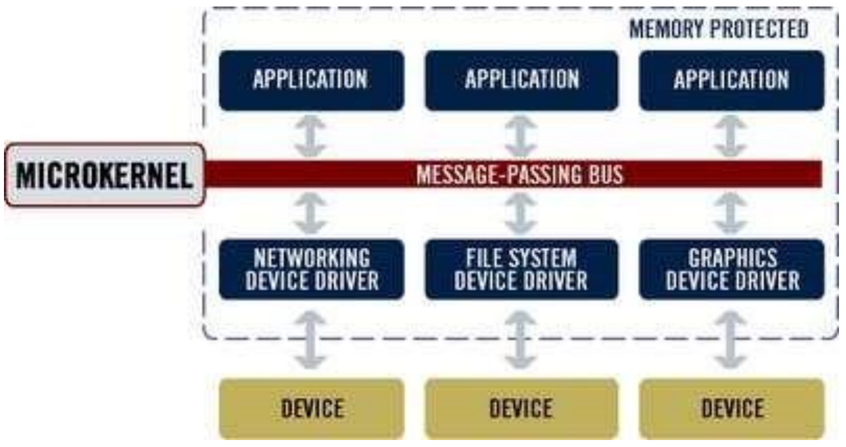
  Windows CE has support for cryptography with a cryptographic library (Cryptographic Application Programming Interface, CAPI) to securely store information in memory. The kernel-loader authentication program can use public key signatures to prevent unauthorized applications from running. Access to the data, however, will be slower because of the electrically erasable and programmable read-only memory (EEPROM) memory used instead of battery backed RAM.

- **Software development for Windows CE**

  Since Windows CE is based on the Win32 API development tools, such as Visual C++ or Visual Basic, available for this API.

1.8.4 QNX Neutrino:

QNX is a real time operating system consisting of microkernel surrounded by a collection of optimal processes that provides UNIX based system services. Due to microkernel architecture even if the file system driver or network driver crashes, still the system will work which leads to stable system. QNX is very well suitable for car devices.



QNX Neutrino

- **User management:**

  QNX supports single user.

- **Task management:**

  Supports real multitasking.

- **User interface:**

  Consist of micro graphical user interfaces and widgets for easy interface.
- **Memory management:**

  QNX has an MMU concept for separation of address space of applications. Different application runs on different threads.
- Software development for QNX is in C language.

1.8.5 Be OS:

Be OS is highly optimized for multimedia application. It posses sound and graphic processor. It deals with 64 bit file system

The architecture is based on a symmetric multiprocessor model, allowing each processor full access to resources and also it provides pre-emptive multitasking and pervasive multithreading (raping switching between several task).

- **User management:**

  It supports multiuser as like standard operating system.
- **Task management:**

  Pre-emptive multitasking by pervasive threads enables the task management much speeder.
- **Memory management:**

  Provides memory protection between applications and virtual memory support.
- Software development is done using C/C++.

1.8.6 Embedded Linux:

Embedded Linux is a stripped down operating system with special support to pervasive devices. Mainly used for handheld devices.

The core features are Configurable kernel, Scalability and Networking.

- **User management:**

  It supports multiuser as like standard operating system.
- **Task management:**

  Preemptive multitasking with optional real time scheduler is implemented.
- **Operating System Size:**

  Depending on the configuration, the size of the kernel can range from 200 KB to several megabytes.

- **User Interface:**

  x-Window system for user interface and have striped down version to save memory.
- **Memory management:**

  Supports MMUs to provide memory protection between applications and virtual memory for paging memory to hard disc.
- Software development is done using C/C++ and JAVA.

1.9 Secure services

1.9.1. Security Concept

Identification:

Various identification methods are used in pervasive devices. One is user have to type the one which is stored in the device otherwise the user can be identified using certificates.

Authentication:

The most common way of authenticating peers is by using user name and password. For a client server connection authentication is ensured by establishing secure socket layer. WTLS is used in WAP phones. Wireless Authentication Module (WIM) is for smart cards. SIMs will be authenticating the mobile terminal in GSM network.

When smart card at the client and authentication software is installed in the server then the server will produce a challenge to the client and in return the client gives the signature and the certificate which will be checked by the server.

Authorization:

Authorization is based on from which device the user is accessing. In a banking transaction, If the user access from PC using user identification and password and this will be the same for user from WAP phone, PDAs but when user is from normal telephone then there is no entry for user identification and password.

Transaction Authorization:

The applications such as home-banking, placing orders, brokerage should authorize every individual transaction.

Digital Signature Endorsed by a Password:

Digital signature ensures that the particular request or message comes from the authorized user since the user will be having the unique key from which a token will be generated and the token generates the required signature which will be passed to the server for verifying the transaction.

Transaction Authorization Number:

TAN (in blocks) is a secret number generated for legitimate users in an organization. The users have to acknowledge immediately after receiving the TAN and the user should ensure that it will be kept secret. Whenever the user contacts the server, TAN should be entered which will be compared with the stored one.

Non-Repudiation:

User A denying user B and user B denying user A is called non-repudiation. This can be avoided by using efficient digital signature.

1.9.2. Device Security

Trojan Horses

A **Trojan horse**, or **Trojan**, is a non-self-replicating type of malware which gains privileged access to the operating system while appearing to perform a desirable function but instead drops a malicious payload, often including a backdoor allowing unauthorized access to the target's computer. These backdoors tend to be invisible to average users, but may cause the computer to run slow. Trojans do not attempt to inject themselves into other files like a computer virus. Trojan horses may steal information, or harm their host computer systems. Trojans may use drive-by downloads or install via online games or internet-driven applications in order to reach target computers.

Security in WAP phones

WTLS is the layer that provides most of the security functionalities for WAP applications. These functionalities include client-server mutual authentication, privacy, data integrity, and non-repudiation.

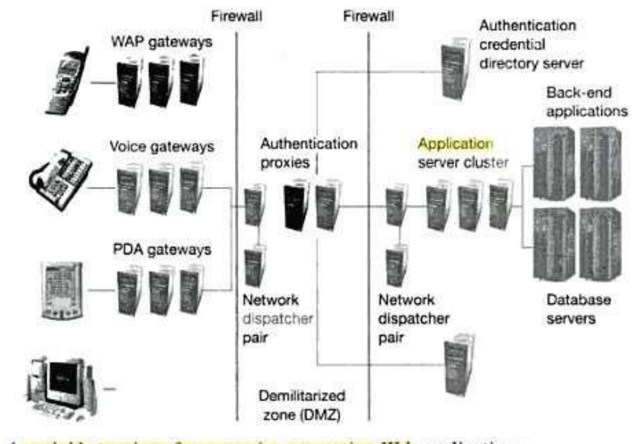
WTLS and TLS:

The design of WTLS is based upon TLS (Transport Layer Security) that is in turn built upon SSL (Secure Socket Layer). TLS has become de facto security protocol for ensuring end-to-end security for Internet communications. Similar to TLS, WTLS requires the client and the server negotiate and agree on a set of security parameters during the handshake before the communicate channel can be established. Once handshake succeeds, the client and the server can exchange information using the secrets known to both ends of the channel. Since WTLS resembles TLS so much, one could consider that the WTLS provides the same level of security as TLS does. However, due to the limitations of wireless communications and the modifications WTLS made to accommodate to these limitations, it has been shown that WTLS is vulnerable to a variety of known attacks such as plaintext recovery attacks and datagram truncation attacks.

Security in PDAs

Downloading and installing arbitrary software makes prone Trojan horse attacks.

1.9.3. Server Side Security



A scalable topology for pervasive computing Web applications

Server Side Security

5.7.4. Cryptographic Algorithms:

A cryptographic algorithm, or cipher, is a mathematical function used in the encryption and decryption process. A cryptographic algorithm works in combination with a key—a word, number, or phrase—to encrypt the plaintext. The same plaintext encrypts to different ciphertext with different keys. The security of encrypted data is entirely dependent on two things: the strength of the cryptographic algorithm and the secrecy of the key. A cryptographic algorithm, plus all possible keys and all the protocols that make it work comprise a cryptosystem

Symmetric cryptographic algorithm:

In secret key cryptography, a single key is used for both encryption and decryption. The sender uses the key (or some set of rules) to encrypt the plaintext and sends the cipher text to the receiver. The receiver applies the same key to decrypt the message and recover the plaintext. Because a single key is used for both functions, secret key cryptography is also called symmetric encryption. With this form of cryptography, it is obvious that the key must be known to both the sender and the receiver; that, in fact, is the secret. The biggest difficulty with this approach, of course, is the distribution of the key.

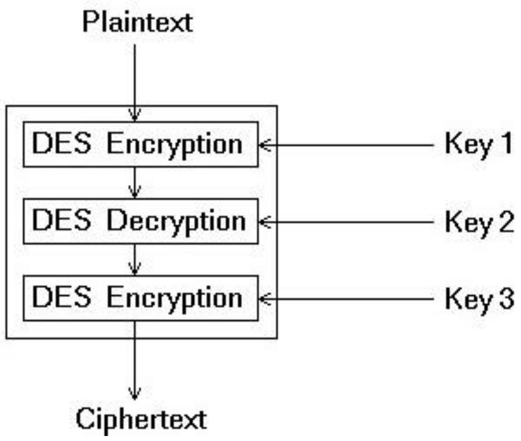
Asymmetric cryptographic algorithm:

Public or asymmetric key cryptography involves the use of key pairs: one private key and one public key. Both are required to encrypt and decrypt a message or transmission. The private key, not to be confused with the key utilized in private key cryptography, is just that, private. It is not to be shared with anyone. The owner of the key is responsible for securing it in such a manner that it will not be lost or compromised. On the other hand, the public key is just that, public. Public key cryptography intends for public keys to be accessible to all users. In fact, this is what makes the system strong. If a person can access anyone public key easily, usually via some form of directory service, then the two parties can communicate securely and with little effort, i.e. without a prior key distribution arrangement.

Data Encryption Standard (DES):

DES is the archetypal block cipher — an algorithm that takes a fixed-length string of plaintext bits and transforms it through a series of complicated operations into another cipher text bit string of the same length. In the case of DES, the block size is 64 bits. DES also uses a key to customize the transformation, so that decryption can supposedly only be performed by those who know the particular key used to encrypt. The key ostensibly consists of 64 bits; however, only 56 of these are actually used by the algorithm. Eight bits are used solely for checking parity, and are thereafter discarded. Hence the effective key length is 56 bits

Triple Data Encryption Standard (Triple DES):

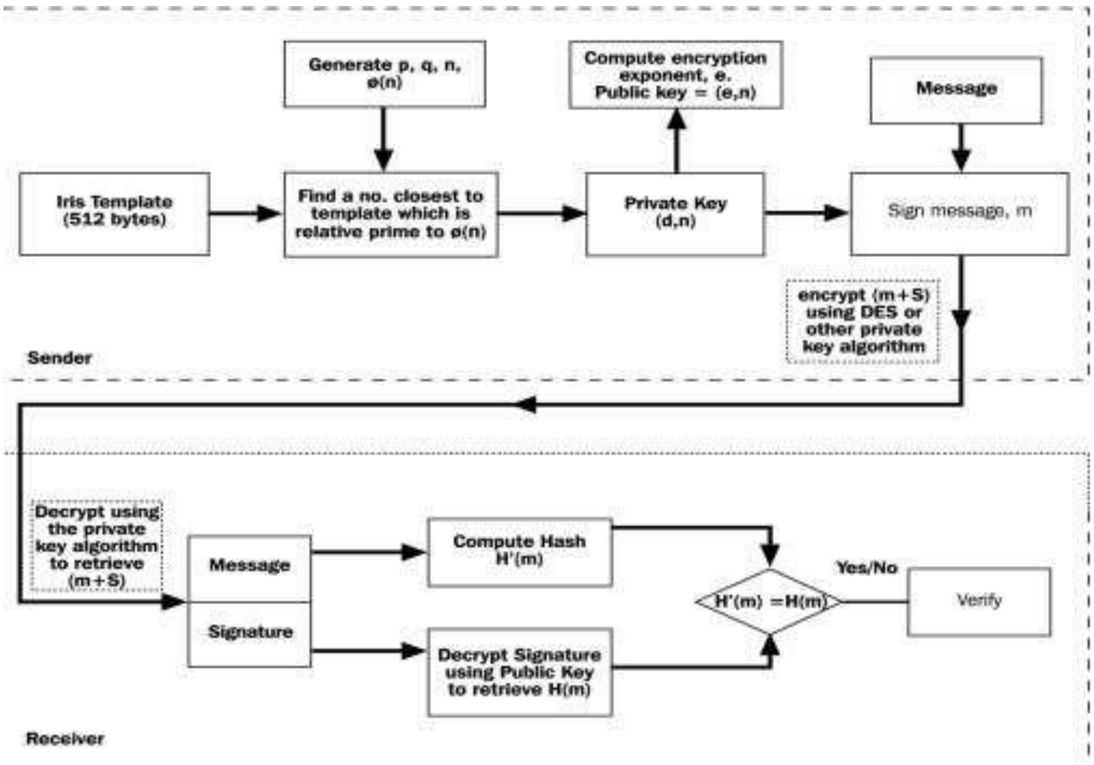


Triple Data Encryption

AES:

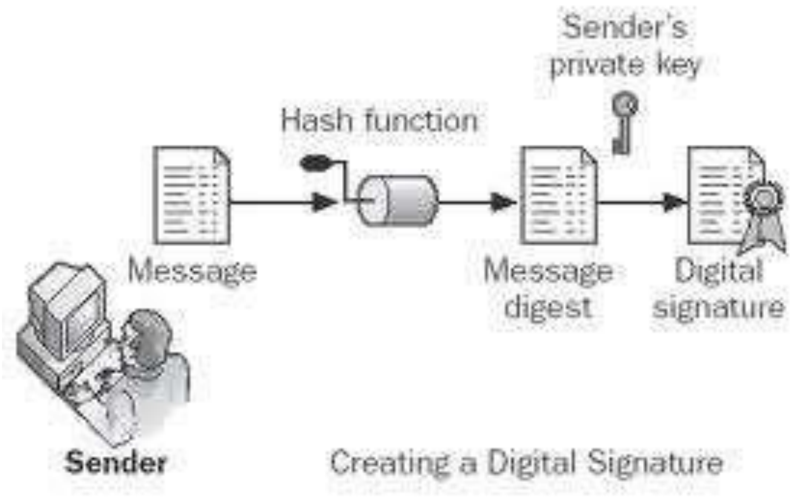
AES is based on a design principle known as a substitution-permutation network, and is fast in both software and hardware. Unlike its predecessor DES, AES does not use a Feistel network. AES is a variant of Rijndael which has a fixed block size of 128 bits, and a key size of 128, 192, or 256 bits.

RSA:


Generate p, q, n, ø(n)
Compute encryption exponent, e. Public key = (e,n)
Message
Iris Template (512 bytes)
Find a no. closest to template which is relative prime to ø(n)
Private Key (d,n)
Sign message, m
encrypt (m+S) using DES or other private key algorithm
Sender
Decrypt using the private key algorithm to retrieve (m+S)
Message
Signature
Compute Hash H'(m)
Decrypt Signature using Public Key to retrieve H(m)
H'(m) =H(m)
Yes/No
Verify
Receiver

RSA

Digital Signature:



Digital Signature

Elliptic Curve Cryptography:

Public-key cryptography is based on the intractability of certain mathematical problems. Early public-key systems are secure assuming that it is difficult to factor a large integer composed of two or more large prime factors.

For elliptic-curve-based protocols, it is assumed that finding the discrete logarithm of a random elliptic curve element with respect to a publicly known base point is infeasible. The size of the elliptic curve determines the difficulty of the problem.

The primary benefit promised by ECC is a smaller key size, reducing storage and transmission requirements—i.e., that an elliptic curve group could provide the same level of security afforded by an RSA-based system with a large modulus and correspondingly larger key—e.g., a 256-bit ECC public key should provide comparable security to a 3072-bit RSA public key.

For current cryptographic purposes, an elliptic curve is a plane curve which consists of the points satisfying the equation

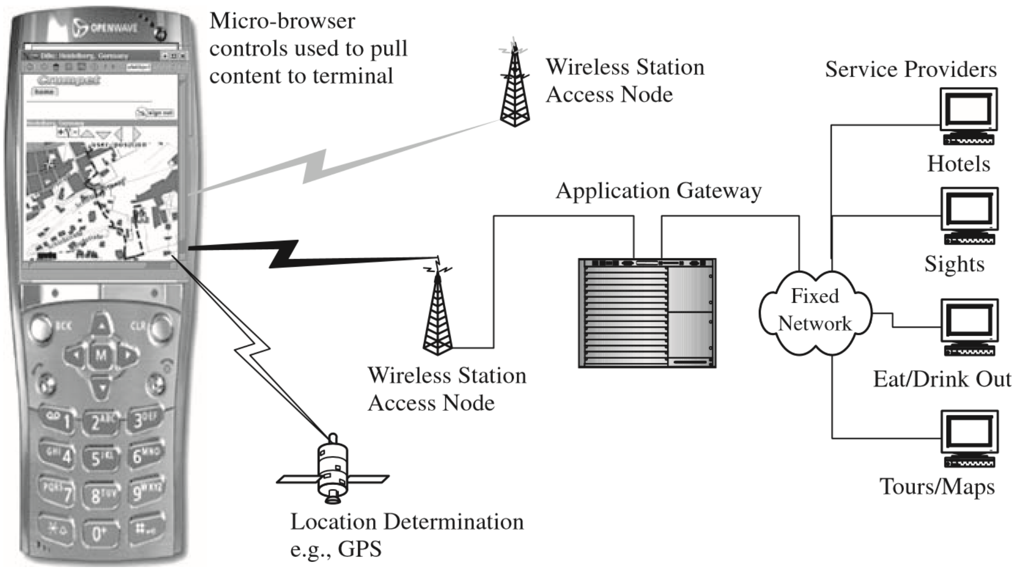
$$y^2 = x^3 + ax + b,$$

along with a distinguished point at infinity, denoted ∞. (The coordinates here are to be chosen from a fixed finite field of characteristic not equal to 2 or 3, or the curve equation will be somewhat more complicated.)

1.10 Smart mobiles

The rise of smart mobiles has fundamentally reshaped our interaction with the world around us. These ubiquitous devices are no longer just phones; they are powerful mini-computers embedded in our pockets, constantly gathering data, analyzing information, and connecting us to a vast network of services and experiences.

In the realm of pervasive computing, smart mobiles play a central role as gateways to a seamlessly interconnected world, where physical and digital realms merge to create intelligent environments that adapt to our needs and preferences.


Micro-browser controls used to pull content to terminal
Wireless Station Access Node
Service Providers
Hotels
Application Gateway
Sights
Fixed Network
Eat/Drink Out
Wireless Station Access Node
Tours/Maps
Location Determination e.g., GPS

Thin client server architecture example, a micro browser running on a mobile device is used to retrieve content over a wireless network

The Capabilities of Smart Mobiles in Pervasive Computing:

- **Data Acquisition:** Smart mobiles are equipped with an arsenal of sensors, including cameras, microphones, gyroscopes, and GPS, capable of capturing real-time data about the user's environment and activities. This data can be used to personalize experiences, trigger context-aware actions, and fuel the intelligence of pervasive systems.
- **Connectivity and Communication:** Smart mobiles act as communication hubs, seamlessly connecting to other devices, networks, and services. Wi-Fi, Bluetooth, and cellular technologies enable data exchange, real-time interaction with smart environments, and access to cloud-based services.
- **Computation and Processing:** Modern smartphones boast powerful processors and significant RAM, enabling them to handle complex tasks on-the-go. This processing power allows for real-time data analysis, decision-making, and intelligent responses within the context of pervasive environments.
- **User Interface and Interaction:** Smart mobiles offer intuitive touch interfaces and voice assistants, making interaction with pervasive systems natural and convenient. These interfaces allow users to control devices, access information, and initiate actions within the interconnected environment.

Applications of Smart Mobiles in Pervasive Computing:

- **Smart Homes and Cities:** Smart mobiles can act as the control center for intelligent homes, managing lighting, temperature, appliances, and security systems based on user preferences and real-time data. In smart cities, they can access public transportation information, optimize traffic flow, and provide personalized city services.
- **Context-Aware Services:** With access to location data and user activity patterns, smart mobiles can trigger context-aware services like automatic parking suggestions, personalized restaurant recommendations, or relevant news updates based on the user's surroundings.
- Health and Wellness: Smart mobiles can act as personal health assistants, monitoring fitness metrics, providing health coaching, and enabling remote patient monitoring for improved healthcare delivery.
- **Education and Learning:** Mobile devices can personalize learning experiences, provide access to educational resources on-demand, and facilitate collaboration between students and educators in intelligent learning environments.

Challenges and Considerations:

- **Privacy and Security:** The vast amount of data collected by smart mobiles raises concerns about privacy and security. Robust data protection measures, user control over data sharing, and secure communication protocols are essential to address these concerns.
- **Battery Life and Energy Efficiency:** Mobile devices in pervasive computing environments will require longer battery life to support continuous data collection and interaction. Advancements in battery technology and energy-efficient algorithms are crucial.
- **Accessibility and Inclusiveness:** Pervasive computing solutions should be designed with accessibility in mind, ensuring these technologies are equally accessible and beneficial for users with diverse abilities and needs.
- **Ethical Considerations:** The integration of smart mobiles into pervasive environments raises ethical questions about data ownership, algorithmic bias, and the potential for manipulation or surveillance. Careful consideration and ethical frameworks are needed to ensure these technologies are used responsibly and for the benefit of all.

1.11 Cards

- **Smart cards:** These embedded chips store secure information like identification credentials, payment details, or medical records. They enable secure transactions, access control, and personalized services within pervasive environments.
- **RFID tags:** These tiny radio-frequency identification tags store unique identifiers and can be attached to objects or embedded in them. They facilitate automatic identification, tracking, and inventory management in various applications.
- **SIM cards:** These Secure Identity Modules connect mobile devices to cellular networks, allowing for voice and data communication. They also store subscriber information and authentication credentials.

Benefits:

- **Security:** Cards provide secure storage and transmission of sensitive data, minimizing the risk of unauthorized access or fraud.
- **Convenience:** They offer contactless interactions and eliminate the need for bulky wallets or multiple access tokens.
- **Personalization:** Cards can store user preferences and enable personalized experiences within smart environments.

1.12 Device networks

1.12.1 PROTOCOLS

Standardized protocols are basic requirements for pervasive computing devices. Wireless protocols supporting IP (Mobile IP) is needed for existing pervasive devices. Another issue is regarding the consistency of database and their data. Connected pervasive computing devices to the distributed environment and their services are important in pervasive domain. Data delivery, in time and integrity is maintained by transaction protocols. Protocols such as WAP and Bluetooth having role in pervasive domain.

5.6.1 Wireless Protocols:

Wireless protocols are enhanced mainly for PDAs and Mobile phones. The protocols such as WAP, Bluetooth, IrDA, Object Exchange Protocol (OBEX) and other mobile phone technologies are concentrated.

WAP/WML:

- WAP is a technology which mainly pronounces on rapid access to internet.
- WAP works for Ericsson, Nokia, Motorola.
- Integrates telephony services with browser technology.
- WAP application includes e-commerce, online banking and messaging.
- WAP is similar to HTTP and then optimized for narrow band channel.
- WML is used for textual format and WBXML is used for compressed binary format.

OBEX:

- OBEX was originally designed for IrDA later due to creation of Bluetooth it was emerged as high level protocol.
- OBEX simplifies communication enabled application by using push and pull commands.
- Two models- The Session Model and The Object Model.
- Session Model: Constructs the dialog between two devices by packet based client/server request/response model.
- Object Model: It contains information about the object being sent with header and the header has name, length, descriptive text and the object body.
- Security is provided using challenge response scheme.

Bluetooth:

- It is a short range communication and data exchange.
- The various characteristics are

- Frequency Band: operates at 2.45 GHz.
- Security: Authentication based on private key and encryption.
- Transmitting Capability: It is omnidirectional and range upto 10m.
- Bandwidth: Data Transfer Rate of 1 Mbps.
- Speech: Support digital speech channel.
- Cost: Reasonable Price.

IrDA:

- Pervasive Computing device are IrDA-Data and Infrared Mobile Communication (IrMC)**.**
- Frequency Band: It is a physical transport medium.
- Security: Higher level protocol service.
- Transmitting Capabilities: Point-point connection with narrow angle between sender and receiver. Range of communication is about 0-30 cm.
- Bandwidth: Data rate-4 Mbps Speech: one digital speech channel.
- Cost: very cheap.

1.12.2 Mobile Phone Technologies:

Cellular system for mobile communication:

Mobile environment is in need of electromagnetic waves at frequency 1Ghz. Small antennas emit only limited power and thus mobile and cellular environment is designed as such the transmitters are placed in small grid like space where the frequency in one grid is reused in another grid which is placed far away and thus reusability is enhanced.

The grid is called as cell. Base station within the cell and the mobile station should be connected with stronger signal.

Features	**1G**	**2G**	**2+G**	**3G**
Protocols	AMPS, C-Net	GSM, TDMA, CDMA	GPRS, HSCSD, EDGE	UMTS, WCDMA
Technology	Analog Circuit Switched	Digital Circuit Switched	Digital or Packet Switched	Digital Packet Switched
Speech Quality	Poor	High	High	High

Bandwidth	Low	Low	Medium	High
Security	None	Low to High	High	High

1.12.3 Mobile Internet Protocol:

Once the mobile connection is available, the next step is to access internet and thus in mobile environment we are moving to IPv6 working on Mobile IP.

Standard internet protocol and mobile devices:

The standard protocol used to access internet is IPv4 which provides with unique address to nodes. Similar to postal addresses IP addressing is provided to each node. The data will be in the form of packets and the packet contains source address, source port, and destination address and destination port.

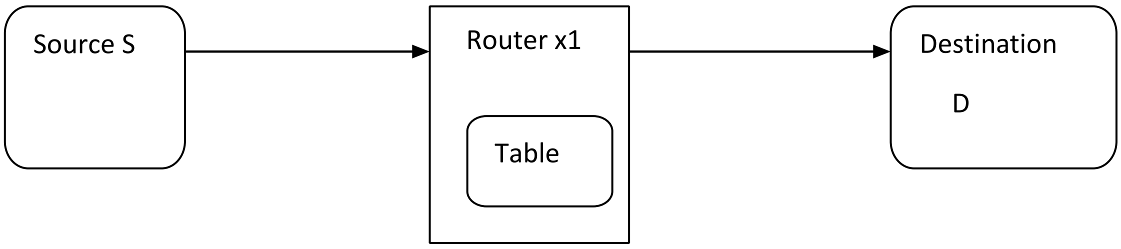


Fig. 5.9 Standard Internet Protocol and Mobile device

Steps to have connection to a mobile node:

- Discover a care-of-address
- Register the care-of address
- Tunnel the care-of address

Changes to Internet Protocol Version 6:

IPv6 includes the features of IPv4 but the significance is address configuration and neighbour discovery. Ipv6 expects all nodes to implement strong encryption and authentication features. Supports mobility at greater extent.

1.12.4 Synchronization and replication Protocols:

Synchronization also called as replication in concern with database. For instance, if the calendar application is stored in the PC and PDAs. The updation in one device related to the application should be reflected in the other device also.

Steps in Synchronization Protocol:

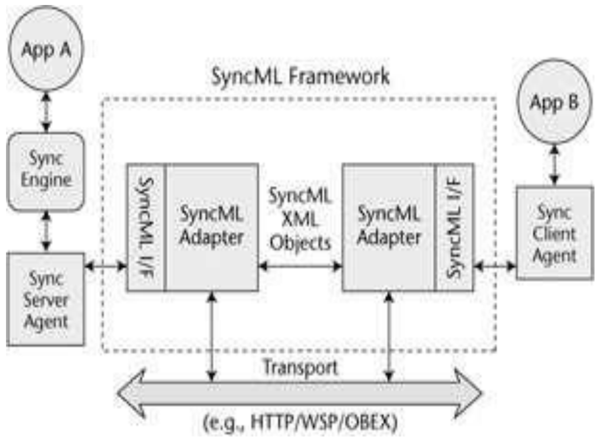
- **Presynchronization:** Before the actual synchronization authentication (authenticates client and server), authorization (allowed to perform action) and determination of device capabilities (maximum buffer size) should be done.

- **Synchronization:** Data exchanged here, al; local Ids are mapped with global Ids. Only updated entries are allowed to exchange.

SyncML

Scope of SyncML

- XML based framework for data synchronization
- Message oriented data exchange protocol
- Transport agnostic
- Universal deployment
- Extension for device management



SyncML Framework

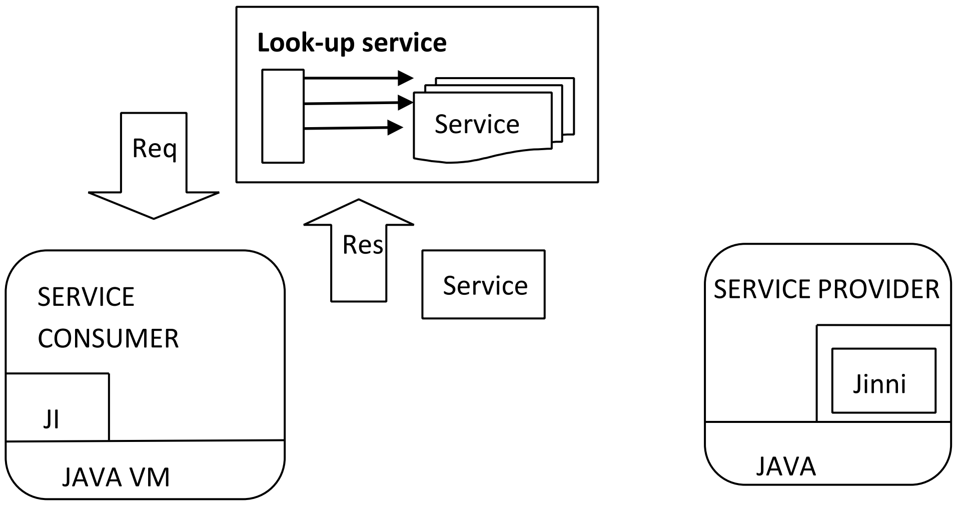
1.12.5 Distributed Services:

Jini:

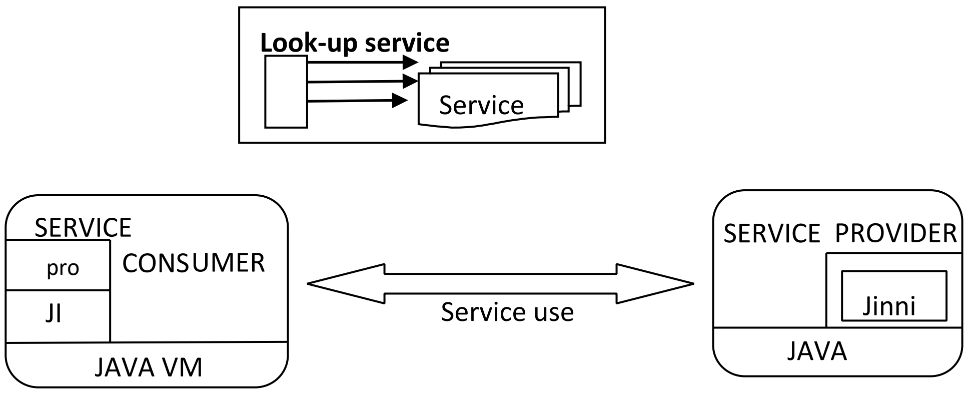
A Jini system consists of the following parts:

- A set of components that provides an infrastructure for federating services in a distributed system.
- A programming model that supports and encourages the production of reliable distributed services.
- Services that can be made part of a federated Jini system and that offer functionality to any other member of the federation.
- **Discovery:** finds other members in the jinni community and joins to enable networking capability.
- **Look-up:** Search based on object type.
- **Leasing:** Stable and self healing. Resources are only leased since each have some timestamp.
- **Remote Events:** Sending notification to the participants.
- **Transactions:** similar to database transaction.

Jini scenario for accessing a remote service- first step



Jini scenario for accessing a remote service- second step



1.12.6 Message and Transaction Based Protocol:

Messaging and Transaction Technology:

- Assured Message delivery and atomic operation are the two different technologies. Standards are needed in order to implement these technologies.
- SOAP provides message exchanging structure and sends information between peers.
- Message queuing is another concept which emphasis on storing of messages and as soon as the connection established, the message will be delivered. This is suitable for asynchronous network.

- Topology can be build with multiple queues. One to one, one to many, many to one.

UNIT II MOBILE APPLICATIONS

History – Mobile Ecosystem – Designing for context – Mobile strategy – Mobile applications – Information Architecture – Design – Mobile Web apps vs Native Apps – Adapting to devices – Supporting devices – Application development on Android and iPhone.

2.1 History

The evolution of mobile networks, the devices that run on them, and the services we use every day have evolved at an amazing rate, from the early phones that looked more like World War II field radios to the ultra-sleek fashion statements of today.

If there is one basic principle about mobile, it is that everything is the way it is for a reason. It might not be a good reason, but a reason exists nonetheless. It is the history, or the context, of the medium that gives mobile designers and developers the patience and passion needed to deal with the frequent issues they face in the mobile ecosystem. The mobile industry is a difficult one to jump into without patience and passion.

2.1.1 In the Beginning

For those of us who are older—that is, over the age of 30—when we think of what a telephone is and try to picture it, we might think of the phone illustrated in Figure 1-1.

The telephone is undoubtedly one of the greatest inventions of mankind. It revolutionized communications, enabling us to reach across great distances and share thoughts, ideas, and dreams with our fellow man, making the world a much smaller place in the process. In fact, the telephone is probably one the most defining technologies of the twentieth century and the most commonly used electronic device in the world today.

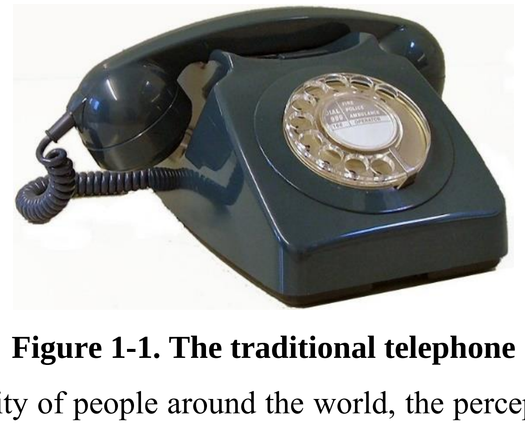


Figure 1-1. The traditional telephone

For the vast majority of people around the world, the perception of what a “phone” is and what it can do hasn’t changed from this iconic image—something you hold up to your ear and talk into—but when those under the age of 20 picture a telephone, they might think of an image similar to the one shown in Figure 1-2.

Carrier 7:43 PM

The New York Times

mobile.nytimes.com/

Google

The New York Times

Jun 7, 2009, 10:25 PM EDT

New York 73°F, Partly Cloudy

Tony Awards »

LATEST NEWS

1. U.S. Weighs Intercepting North Korea Shipments

The Obama administration signaled that it was seeking a way to interdict North Korean shipments suspected of carrying weapons or nuclear technology.

2. U.S. to Propose Wider Oversight of Compensation

Under the plan, companies that received two rounds of federal bailouts would submit major executive pay changes for approval

Figure 1-2. A modern mobile phone

Although the modern mobile phone is a distant cousin to the telephone, it is a communication and information device. It is nearly always connected to the Internet, even if you don’t have a web browser open. You can send and receive voice and text messages. You can purchase goods and services without opening your wallet. Plus, it can locate which street corner you are standing on and tell you what is nearby all in a fraction of an instant.

In fact, the modern mobile phone is capable of doing nearly everything you can do with a desktop computer, but with the potential for more meaningful relevance to our daily activities. The mobile phone is not merely a telephone. In fact, modern mobile devices deliver on the long-overdue promises that technology will make our lives easier, but without the cable clutter that drives someone like my wife nuts.

Thinking of mobile devices more as personal computers and less as telephones is a difficult shift in perception. The mobile industry of today has somewhat of a split personality—each side with its own conflicting interests: the first half being the telecom infrastructure and the people who run it, required for everything to work but only focused on the network; and the other consisting of the devices we carry, focused on how and when we interact with the network. And yet a third personality is the Web, the repository of the world’s knowledge that we seek to use in the context of our daily activities.

2.1.2 The Evolution of Devices

Mobile technology has gone through many different evolutions to get to where it is today. In the industry, we often refer to these evolutions as “generations” or simply “G,” which refers to the maturity and capabilities of the actual cellular networks. The cellular network is only one element of the overall mobile ecosystem. To make sense of it all, you cannot rely on the common convention of network generations, as those milestones are too focused on the network and not the true cultural milestone that is shifting how people use technology.

Rather, you need to segment the history of mobile into five distinct eras of devices.

2.1.2.1 The Brick Era

The first era I call the Brick Era (1973–1988). Nothing is more emblematic of this era than the Motorola DynaTAC (see Figure 1-3) introduced in 1983.

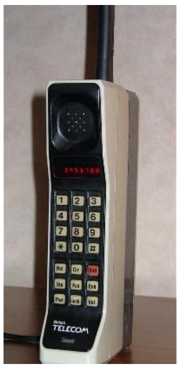


Figure 1-3. The Motorola DynaTAC 8000X was the first mobile phone to receive FCC acceptance, in 1983; DynaTAC was actually an abbreviation of Dynamic Adaptive Total Area Coverage

Brick Era phones proved useful only to those who truly needed constant communication, such as stockbrokers or those who worked in the field, such as salespeople or real estate agents; because they were so enormous and so expensive, they were far too impractical for the majority of us.

After mobile technology started racing down the motorways of the world, more cellular radio towers were needed to provide constant coverage. As more towers went up, the power demands of the devices went down. In other words, the closer you were to a tower, the smaller the device you needed.

The proliferation of mobile technology in this era opened the gates to mobile devices of today. What started as a bulky luxury item became something everyone could fit into their budgets and their pockets.

2.1.2.2 The Candy Bar Era

The second era, the Candy Bar Era (1988–1998), represented one of the more significant leaps in mobile technology. “Candy bar” is the actual term used to describe the long, thin, rectangular form factor of the majority of mobile devices used during the Candy Bar Era and even today (see Figure 1-4).

At this point, network operators started to see the clear value (and big profits) in their burgeoning cellular networks, and a “perfect storm” ensued. The network shifted to second-generation (2G) technology, starting in Finland in 1991. The density of cellular sites caused by increased usage decreased the power demands of the device, making it small enough to fit in your pocket.

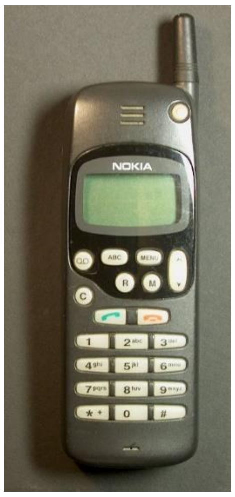


Figure 1-4. A Nokia candy bar phone

This era didn't just user in portability. For the first time, people started to realize that mobile phones could do more than make voice calls. Candy bar phones—so commonly associated with 2G GSM (Global System for Mobile communications) networks—included SMS (Short Message Service) capabilities.

Initially, the idea behind SMS was for the mobile operator to send subscribers a notification of a new voicemail, or other short notifications. But in the early 1990s, due to oversights by mobile operators, text messages were not charged to consumers. Mobile-savvy Europeans quickly realized that they could send messages to their friends for free when voice calls were still fairly expensive by today's standards. Thus, today's abbreviated text language, which is limited to 140 characters, was born.

2.1.2.3 The Feature Phone Era

The third era, the Feature Phone Era (1998–2008), wasn't nearly as radical a technological leap as the leap from the Brick Era to the Candy Bar Era, but it was an important evolution nonetheless. Up to this point, mobile phones had done three things: make voice calls, send text messages, and play the Snake game. The Feature Phone Era (see Figure 1-5) opened the floodgates to a variety of applications and services on the phone, like listening to music and taking photos, and introduced the use of the Internet on a phone.

Figure 1-5. The Motorola RAZR, probably the most iconic device from the Feature Phone Era

During this era, GSM network providers added GPRS (General Packet Radio Service), allowing packet-switched data services. This network evolution is most often referred to as 2.5G, or halfway between 2G and 3G networks. Network providers offering CDMA and other TDMA-based networks followed suit with similar packet-switched data services soon after.

At last, the Web had reached mobile devices, but due to high prices, poor marketing, and inconsistent rendering, no one was using it. Instead, mobile companies were focusing on creating downloadable ringtones, wallpapers, games, and applications to sell through network operator portals.

2.1.2.4 The Smartphone Era

The Smartphone Era occurred at the same time as the third and fifth eras and spans from around 2002 to the present. What is and isn't a smartphone (see Figure 1-6) has never been defined, which explains the overlap in chronology. Although smartphones have all the same capabilities of a feature phone, like making a phone call, sending an SMS, taking a picture, and accessing the mobile web, most smartphones are distinctive in that they use a common operating system, a larger screen size, a QWERTY keyboard or stylus for input, and Wi-Fi or another form of high-speed wireless connectivity.

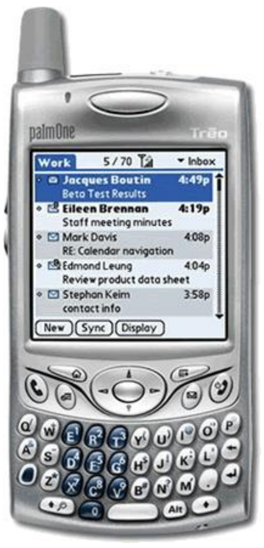


Figure 1-6. Early smartphones came from companies like Nokia, Handspring, and Research in Motion (RIM)

Although there have been many different flavors of smartphone, I see this era more as a technological bridge. Most notable devices of this era seemed to try to be something that they weren't.

For example, the Nokia 9000 series of "Communicator" smartphones looked like a feature phone on the outside, but you when held one on its side, it could be opened like a clamshell to reveal a large screen and keyboard—obviously in an attempt to position the phone as a micro laptop. This is something Microsoft also attempted with its initial Windows CE platform, which would later become Windows Mobile.

Handspring, on the other hand, combined a Palm OS–based PDA with a phone module to create PDA-style smartphones, which would later become the popular Treo line of smartphones. Research in Motion applied its background in two-way paging to create the first BlackBerry, which would later be used to "push" email to corporate citizens in a pager-like fashion.

But even with all this effort, smartphones failed to pique the public's interest and create demand, capturing—at best estimates—just 10% to 15% of the global mobile phone market share.

Meanwhile, feature phone manufacturers continued to evolve their devices, merging the capabilities of PDAs or limited desktop computing with traditional feature phones. Most notably, Symbian—initially a joint venture of mobile device makers Nokia, Motorola, Ericsson, and Psion, and now fully owned by Nokia—created the Symbian OS, a smartphone operating system containing common libraries, tools, and frameworks.

The Symbian OS is used for a variety of mobile devices, the most recognizable of which are the Nokia S60 (still referred to by the defunct Series 60 label), and popular models like the 6260 and N95, both devices that look more like phones than computers.

I think Nokia defined the devices of the Smartphone Era. They created great telephones, an amazing framework for creating cool applications and services, and a reusable infrastructure to innovate.

And although new phones continue to emerge based on the smartphone model, I feel like they will continue to be usurped by the fifth and final era: the Touch Era.

2.1.2.5 The Touch Era

Mobile devices started as simple portable telephones, but they evolved. Messaging was added to mobile capabilities, but mobile devices were still just person-to-person communication tools. We saw networks improve and data speeds increase, which allowed for more technology and more features each year, crammed into smaller and smaller packages. Mobile devices got smarter by learning from desktop computing, truly becoming personal computers, but people weren't interested.

Until recently, the history of mobile has been borrowing from other mediums, learning and growing along the way, but never quite creating an identity of its own. But that all changed.

Though the majority of technology within the iPhone and its lower-cost successor, the iPhone 3G/3GS, had already been available from several manufacturers for some time, what was so notable about the iPhone was how it changed everyday perceptions of what mobile technology can do. It wasn't a phone, it wasn't a computer: it was something else entirely.

Mobile devices of the Touch Era are a completely new medium capable of offering real people new and exciting ways to interact and understand information. The devices of tomorrow will be able to leverage location, movement, and the collective knowledge of mankind, to provide people's lives with greater meaning.

And what is so exciting is that "tomorrow" is actually happening right now.

2.2 Mobile Ecosystem

Mobile is an entirely unique ecosystem and, like the Internet, it is made up of many different parts that must all work seamlessly together. However, with mobile technology, the parts are different, and because you can use mobile devices to access the Internet, that means that not only do you need to understand the facets of the Internet, but you also need to understand the mobile ecosystem.

Think of the mobile ecosystem instead as a system of layers, as shown in Figure 2-1. Each layer is reliant on the others to create a seamless, end-to-end experience. Although not every piece of the puzzle is included in every mobile product and service, for the majority of the time, they seem to add complexity to our work, regardless of whether we expressly put them there.

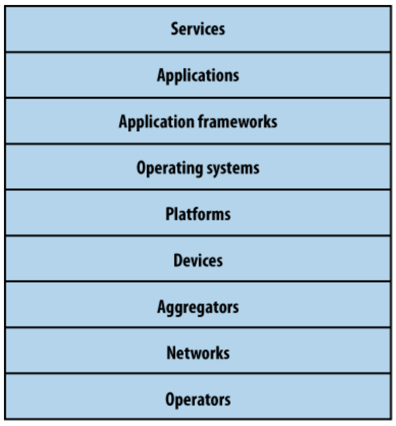


Figure 2-1. The layers of the mobile ecosystem

The following sections expand on each of these layers and the roles they play in the mobile ecosystem.

2.2.1 Operators

- The base layer in the mobile ecosystem is the operator. Operators go by many names, depending on what part of the world you happen to be in or who you are talking to. Operators can be referred to as Mobile Network Operators (MNOs); mobile service providers, wireless carriers, or simply carriers; mobile phone operators; or cellular companies. In the mobile community, we officially refer to them as operators, though in the United States, there is a tendency to call them carriers.
- Operators are what essentially make the entire mobile ecosystem work. They are the gatekeepers to the kingdom. They install cellular towers, operate the cellular network, make services (such as the Internet) available for mobile subscribers, and they often maintain relationships with the subscribers, handling billing and support, and offering subsidized device sales and a network of retail stores.
- The operator's role in the ecosystem is to create and maintain a specific set of wireless services over a reliable cellular network. That's it. However, to grow the mobile market over the past decade, the operator has been required to take a greater role in the mobile ecosystem, doing much more than just managing the network. For example, they have had to establish trust with subscribers to handle the billing relationship and to offer devices, content, and services that often compete with their partners, who are people like us and who want to create content and services for mobile devices.

2.2.2 Networks

- Operators operate wireless networks. Remember that cellular technology is just a radio that receives a signal from an antenna. The type of radio and antenna determines the capability of the network and the services you can enable on it.
- You’ll notice that the vast majority of networks around the world use the GSM standard (see Table 2-2 for an explanation of these acronyms), using GPRS or GPRS EDGE for 2G data and UMTS or HSDPA for 3G. We also have CDMA (Code Division Multiple Access) and its 2.5G hybrid CDMA2000, which offers greater coverage than its more widely adopted rival. So in places like the United States or China, where people are more spread out, CDMA is a great technology. It uses fewer towers, giving subscribers fewer options as they roam networks.

Table 2-2. GSM mobile network evolutions

2G	Second generation of mobile phone standards and technology	Theoretical max data speed
GSM	Global System for Mobile communications	12.2 KB/sec
GPRS	General Packet Radio Service	Max 60 KB/sec
EDGE	Enhanced Data rates for GSM Evolution	59.2 KB/sec
HSCSD	High-Speed Circuit-Switched Data	57.6 KB/sec
3G	Third generation of mobile phone standards and technology	Theoretical max data speed
W-CDMA	Wideband Code Division Multiple Access	14.4 MB/sec

3G	Third generation of mobile phone standards and technology	Theoretical max data speed
UMTS	Universal Mobile Telecommunications System	3.6 MB/sec
UMTS-TDD	UMTS +Time Division Duplexing	16 MB/sec
TD-CDMA	Time Divided Code Division Multiple Access	16 MB/sec
HSPA	High-Speed Packet Access	14.4 MB/sec
HSDPA	High-Speed Downlink Packet Access	14.4 MB/sec
HSUPA	High-Speed Uplink Packet Access	5.76 MB/sec

- Like all things in mobile, we like to merge a lot of technology into overly simplistic terms, which tends to create a lot of confusion. So when we say 3G, for example, we often aren't talking about just the capabilities of the network, but the devices that run on it.
- Although the core technology that empowers voice communication has stayed relatively the same, network generations are most often used to describe the data speeds the network is capable of delivering.

2.2.3 Devices

- What you call phones, the mobile industry calls handsets or terminals. These are terms that I think are becoming outdated with the emergence of wireless devices that rely on operator networks, but do not make phone calls. The number of these "other" devices is a small piece of the overall pie right now, but it's growing rapidly.
- Let's focus on the biggest slice of the device pie—mobile phones. As of 2008, there are about 3.6 billion mobile phones currently in use around the world; just more than half the planet's population has a mobile phone (see Figure 2-2).

- Most of these devices are feature phones, making up the majority of the marketplace. Smartphones make up a small sliver of worldwide market share and maintain a healthy percentage in the United States and the European Union; smartphone market share is growing with the introduction of the iPhone and devices based on the Android platform. As next-generation devices become a reality, the distinction between feature phones and smartphones will go away. In the next few years, feature phones will largely be located in emerging and developing markets. Figure 2-3 shows a breakdown of devices.

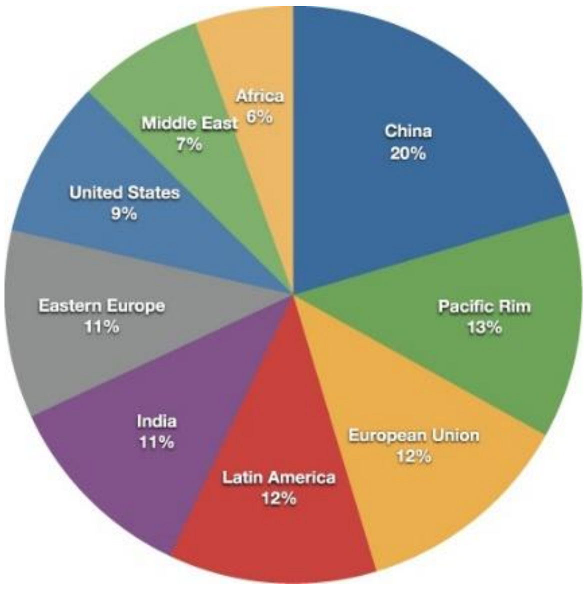


Figure 2-2. Mobile devices around the world

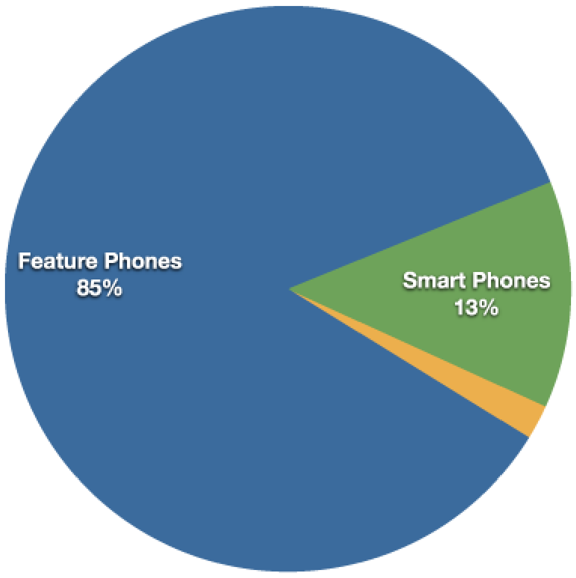


Figure 2-3. Breakdown of devices

- This brings us to the greatest challenge the mobile ecosystem currently faces: device fragmentation, a term used to describe how mobile devices interpret industry specifications differently, causing different mobile devices to display content inconsistently. Despite what you may know or have heard, you can take a deep breath and relax. Device fragmentation is a topic we will clear up completely in the following chapters.

2.2.4 Platforms

- A mobile platform's primary duty is to provide access to the devices. To run software and services on each of these devices, you need a platform, or a core programming language in which all of your software is written. Like all software platforms, these are split into three categories: licensed, proprietary, and open source.

2.2.4.1 Licensed

- Licensed platforms are sold to device makers for nonexclusive distribution on devices. The goal is to create a common platform of development Application Programming Interfaces (APIs) that work similarly across multiple devices with the least possible effort required to adapt for device differences, although this is hardly reality. Following are the licensed platforms:

  Java Micro Edition (Java ME)

  - Formerly known as J2ME, Java ME is by far the most predominant software platform of any kind in the mobile ecosystem. It is a licensed subset of the Java platform and provides a collection of Java APIs for the development of software for resource-constrained devices such as phones.

  Binary Runtime Environment for Wireless (BREW)

  - BREW is a licensed platform created by Qualcomm for mobile devices, mostly for the U.S. market. It is an interface-independent platform that runs a variety of application frameworks, such as C/C++, Java, and Flash Lite.

  Windows Mobile

  - Windows Mobile is a licensable and compact version of the Windows operating system, combined with a suite of basic applications for mobile devices that is based on the Microsoft Win32 API.

  LiMo

  - LiMo is a Linux-based mobile platform created by the LiMo Foundation. Although Linux is open source, LiMo is a licensed mobile platform used for mobile devices. LiMo includes SDKs for creating Java, native, or mobile web applications using the WebKit browser framework.

2.2.4.2 Proprietary

- Proprietary platforms are designed and developed by device makers for use on their devices. They are not available for use by competing device makers. These include:

  Palm

  - Palm uses three different proprietary platforms. Their first and most recognizable is the Palm OS platform based on the C/C++ programming language; this was initially developed for their Palm Pilot line, but is now used in low-end smartphones such as the Centro line. As Palm moved into higher-end smartphones, they started using the Windows Mobile-based platform for devices like the Treo line. The most recent platform is called webOS, is based on the WebKit browser framework, and is used in the Prē line.

BlackBerry

- Research in Motion maintains their own proprietary Java-based platform, used exclusively by their BlackBerry devices.

iPhone

- Apple uses a proprietary version of Mac OS X as a platform for their iPhone and iPod touch line of devices, which is based on Unix.

2.2.4.3 Open Source

- Open source platforms are mobile platforms that are freely available for users to download, alter, and edit. Open source mobile platforms are newer and slightly controversial, but they are increasingly gaining traction with device makers and developers. Android is one of these platforms. It is developed by the Open Handset Alliance, which is spearheaded by Google. The Alliance seeks to develop an open source mobile platform based on the Java programming language.

2.2.5 Operating Systems

- It used to be that if a mobile device ran an operating system, it was most likely considered a smartphone. But as technology gets smaller, a broader set of devices supports operating systems.
- Operating systems often have core services or toolkits that enable applications to talk to each other and share data or services. Mobile devices without operating systems typically run “walled” applications that do not talk to anything else.

- Although not all phones have operating systems, the following are some of the most common:

Symbian

- Symbian OS is a open source operating system designed for mobile devices, with associated libraries, user interface frameworks, and reference implementations of common tools.

Windows Mobile

- Windows Mobile is the mobile operating system that runs on top of the Windows Mobile platform.

Palm OS

- Palm OS is the operating system used in Palm’s lower-end Centro line of mobile phones.

Linux

- The open source Linux is being increasingly used as an operating system to power smartphones, including Motorola’s RAZR2.

Mac OS X

- A specialized version of Mac OS X is the operating system used in Apple’s iPhone and iPod touch.

Android

- Android runs its own open source operating system, which can be customized by operators and device manufacturers.

- You might notice that many of these operating systems share the same names as the platforms on which they run. Mobile operating systems are often bundled with the platform they are designed to run on.

2.2.6 Application Frameworks

- Often, the first layer the developer can access is the application framework or API released by one of the companies mentioned already. The first layer that you have any control over is the choice of application framework.
- Application frameworks often run on top of operating systems, sharing core services such as communications, messaging, graphics, location, security, authentication, and many others.

2.2.6.1 Java

- Applications written in the Java ME framework can often be deployed across the majority of Java-based devices, but given the diversity of device screen size and processor power, cross-device deployment can be a challenge.

- Most Java applications are purchased and distributed through the operator, but they can also be downloaded and installed via cable or over the air.

2.2.6.2 S60

- The S60 platform, formerly known as Series 60, is the application platform for devices that run the Symbian OS. S60 is often associated with Nokia devices—Nokia owns the platform—but it also runs on several non-Nokia devices. S60 is an open source framework.
- S60 applications can be created in Java, the Symbian C++ framework, or even Flash Lite.

2.2.6.3 BREW

- Applications written in the BREW application framework can be deployed across the majority of BREW-based devices, with slightly less cross-device adaption than other frameworks.
- However BREW applications must go through a costly and timely certification process and can be distributed only through an operator.

2.2.6.4 Flash Lite

- Adobe Flash Lite is an application framework that uses the Flash Lite and ActionScript frameworks to create vector-based applications. Flash Lite applications can be run within the Flash Lite Player, which is available in a handful of devices around the world.
- Flash Lite is a promising and powerful platform, but there has been some difficulty getting it on devices. A distribution service for applications written in Flash Lite is long overdue.

2.2.6.5 Windows Mobile

- Applications written using the Win32 API can be deployed across the majority of Windows Mobile-based devices. Like Java, Windows Mobile applications can be downloaded and installed over the air or loaded via a cable-connected computer.

2.2.6.6 Cocoa Touch

- Cocoa Touch is the API used to create native applications for the iPhone and iPod touch. Cocoa Touch applications must be submitted and certified by Apple before being included in the App Store. Once in the App Store, applications can be purchased, downloaded, and installed over the air or via a cable-connected computer.

2.2.6.7 Android SDK

- The Android SDK allows developers to create native applications for any device that runs the Android platform. By using the Android SDK, developers can write applications in C/C++ or use a Java virtual machine included in the OS that allows the creation of applications with Java, which is more common in the mobile ecosystem.

2.2.6.8 Web Runtimes (WRTs)

- Nokia, Opera, and Yahoo! provide various Web Runtimes, or WRTs. These are meant to be miniframeworks, based on web standards, to create mobile widgets. Both Opera's and Nokia's WRTs meet the W3C-recommended specifications for mobile widgets.
- Although WRTs are very interesting and provide access to some device functions using mobile web principles, I've found them to be more complex than just creating a simple mobile web app, as they force the developer to code within an SDK rather than just code a simple web app. And based on the number of mobile web apps written for the iPhone versus the number written for other, more full-featured WRTs, I don't think I'm alone in thinking this. Nonetheless, it is a move in the right direction.

2.2.6.9 WebKit

- With Palm's introduction of webOS, a mobile platform based on WebKit, and given its predominance as a mobile browser included in mobile platforms like the iPhone, Android, and S60, and that the vast majority of mobile web apps are written specifically for WebKit, I believe we can now refer to WebKit as a mobile framework in its own right.
- WebKit is a browser technology, so applications can be created simply by using web technologies such as HTML, CSS, and JavaScript. WebKit also supports a number of recommended standards not yet implemented in many desktop browsers.
- Applications can be run and tested in any WebKit browser, desktop, or mobile device.

2.2.6.10 The Web

- The Web is the only application framework that works across virtually all devices and all platforms. Although innovation and usage of the Web as an application framework in mobile has been lacking for many years, increased demand to offer products and services outside of operator control, together with a desire to support more devices in shorter development cycles, has made the Web one of the most rapidly growing mobile application platforms to date.

2.2.7 Applications

- Application frameworks are used to create applications, such as a game, a web browser, a camera, or media player. Although the frameworks are well standardized, the devices are not.
- The largest challenge of deploying applications is knowing the specific device attributes and capabilities. For example, if you are creating an application using the Java ME application framework, you need to know what version of Java ME the device supports, the screen dimensions, the processor power, the graphics capabilities, the number of buttons it has, and how the buttons are oriented.
- Multiply that by just a few additional handsets and you have hundreds of variables to consider when building an application. Multiply it by the most popular handsets in a single market and you can easily have a thousand variables, quickly dooming your application's design or development.
- Although mobile applications can typically provide an excellent user experience, it almost always comes at a fantastic development cost, making it nearly impossible to create a scalable product that could potentially create a positive return on investment.
- A common alternative these days is creating applications for only one platform, such as the iPhone or Android. By minimizing the number of platforms the developer has to support and utilizing modern application frameworks, the time and cost of creation go down significantly. This strategy may be perfectly acceptable to many, but what about the rest of the market? Surely people without a more costly smartphone should be able to benefit from mobile applications, too.

2.2.8 Services

- Finally, we come to the last layer in the mobile ecosystem: services. Services include tasks such as accessing the Internet, sending a text message, or being able to get a location—basically, anything the user is trying to do.
- What makes the mobile environment such a complicated space to design and develop for are these layers, which the user must wade through in order to accomplish a simple task like "I want to send a text message," "I want to get on the Web," and "I want to access Google." The user has so many opportunities for failure that creating a valuable experience is virtually impossible.
- But we've already seen the future of mobile development, in the form of the iPhone. The iPhone attempts to solve many of the problems facing the mobile ecosystem, from

how people interact with their phones, to where we buy our phones, to what type of applications we will pay money for, to the level of technology standards we can support on constrained devices. What makes the iPhone special is how it attempts change on virtually all fronts, something no other device, or company for that matter, has been able to do previously.

- Now Apple has done it, the gates are wide open for anyone. People in the industry aren't as jaded anymore, and there is a feeling of excitement and optimism. Although many of the problems in the mobile ecosystem are yet to be fixed and we still have plenty of nonsense to contend with, we can see the light. We can see the path to innovation, to creating applications and services that can quite literally reach the entire planet and quite possibly change the world.

2.3 Designing for Context

In late 2008, I was in Berlin doing a mobile workshop at the Web 2.0 Expo. Having never been to Berlin, I did what I always do in new cities that I visit—I explored. I enjoy just walking aimlessly around a new city with no particular destination or plan. Not only is it a relaxing way to see the sights, but I find amazing things that aren't on any tour or in any guidebook. There was just one problem with my plan: everything I explored was in German.

Because I know only about five and a half words in German, this made exploration more challenging. Although I really enjoyed seeing Berlin, my first of couple days there were an empty experience. I could certainly see what was right in front of me, but I didn't have any understanding of what I was looking at. I found myself constantly wondering, "What is this place? Who is this a statue of and why is it here? What significance does it play? What are these people doing? Why is this important?"

At the end of my first day, I found myself in my hotel room visiting Wikipedia so that I could read about what I had seen that day. Unfortunately, I couldn't remember all the sites I had seen. The next day was my workshop, so I had only a little time to wander, but I started taking photos of plaques and historical markers with the camera in my phone so I could translate them later that night. I thought it was an inspired idea, but it proved to be too difficult to make out the text and then enter it into an online translator. My third and final full day in Berlin I had the entire day to explore. To make the most of it, I completely gave up on my futile attempt to avoid incurring international data roaming charges and started using my iPhone as my own personal tour guide.

I wandered aimlessly like I did the previous days, but this time, as soon as I found myself in front of an interesting cultural landmark, I pulled out my phone and started using the location features of the device to detect my location and show me information, in English, about the nearby sites and how far away they were from my precise location. I would read the history of the landmark, what it meant to the German people, and learn about other nearby landmarks that were similar. My day of sightseeing, the locations I visited, and the experiences I had were defined by the information I received on my mobile device.

There I was, the geekiest tourist you've ever seen, standing in front of a great historical landmark, looking down at my phone reading the Wikipedia entry about it (see Figure 4-1).

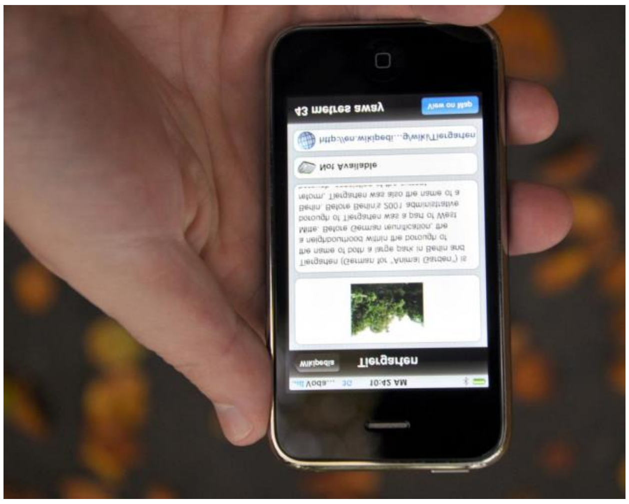


Figure 4-1. My mobile device enhanced my understanding of the landmarks in Berlin, in my own language

It might have appeared stupid to others and maybe it made me stand out as a tourist; if it did, I didn't care. My experience of Berlin was immensely richer because I had an understanding, or context, and I was experiencing it in my immediate surroundings. The irony is that as a "mobile guy," it would take me several months to realize that my experience was almost the perfect example of mobile context.

2.3.1 Thinking in Context

- Context is probably the most used, underestimated, and misunderstood concept in mobile. I think of it as the chewy nougat in the center of a good candy bar. Sure, the candy bar would be good without it, but it's that little extra bit that makes the candy bar an incredible experience; no one quite knows what it is, but everyone knows that it tastes good. Actually, that example probably doesn't help demystify context as much as it makes you crave a candy bar. So let me explain it another way.

- I define context in two ways. There is “Context” with a big C and “context” with a little c. These are often used interchangeably with no preference or distinction. Although they are the same word, they have two different implied meanings. This isn’t to say that the case of a letter makes one more important than the other. It only helps to make a distinction between lofty big ideas (big C) and the more practical and more invisible intention of use (little c).

2.3.1.1 Context with a Capital C

- Context with a capital C is how the users will derive value from something they are currently doing, or in other words, the understanding of circumstance. It is the mental model they will establish to form understanding. For example, standing in front of the remnants of the Berlin Wall and reading about the history on my phone is adding Context to my task. It enhances my experience and awareness of my surroundings in a significant way.

- I refer to this as “providing Context” as in “this information is providing me Context or better understanding of what this moment in time means to me.” Context enables us to better understand a person, a place, a thing, a situation, or even an idea by adding information to it.

2.3.1.1.1 The distracted driver

- One example of putting Context to use is the Distracted Driver campaign that ran in New Zealand.[6] The government saw an increasing problem with drivers having “diverted attention,” causing as many as 26 fatalities in a single year. Research found that tasks such as receiving an incoming text message seriously diverted drivers’ attention and increased the chances of an accident. In an effort to build awareness, the New Zealand Transport Authority asked Clemenger BBDO to develop a campaign to demonstrate that sending text messages while driving can be dangerous.

- Do you remember hearing the stories of those gory highway safety scare films from the 1960s, such *Options to Live*, *Wheels of Tragedy*, or *Mechanized Death*? It used to be that you had to show teenagers extremely violent and horrific images to illustrate the importance of safe driving. But Clemenger BBDO was able to use Context to prove a point instead of blood, guts, and dismemberment.

- The result was a website showing a driver casually behind the wheel, almost as though you were looking at a live webcam from inside a car somewhere in New Zealand (Figure 4-2).

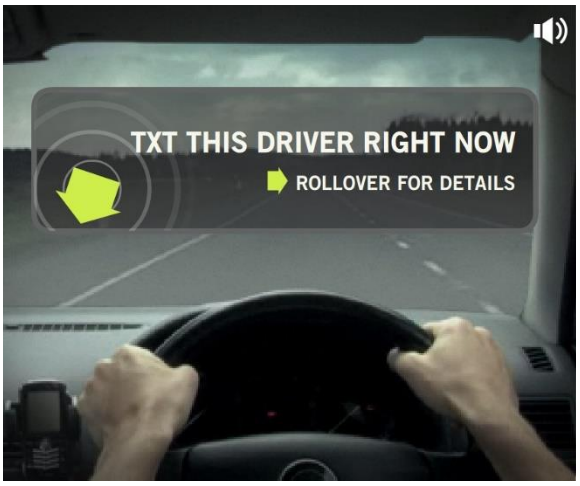


Figure 4-2. The distracted driver

- You are then encouraged to send a text message from your phone to the driver seen in the website (Figure 4-3).

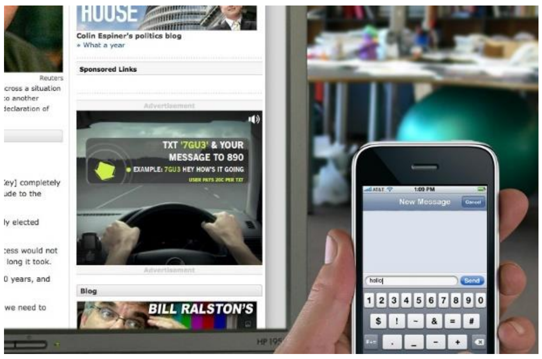


Figure 4-3. People are encouraged to send a text message to the distracted driver

- In a few seconds, the driver in the video would receive the text message, picking up his phone to read the message you just sent him (Figure 4-4).

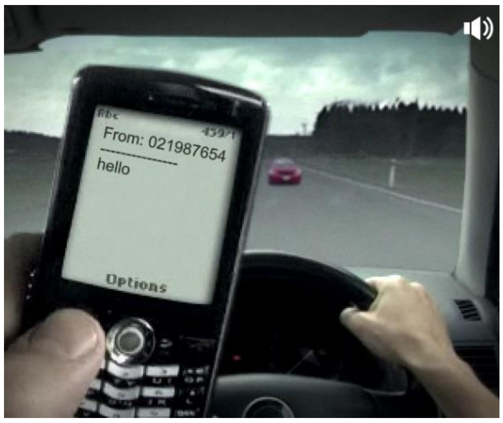


Figure 4-4. The distracted driver receives the text message

- As the driver (and you) is reading the message, he slowly heads into oncoming traffic and crashes (Figure 4-5).

- This merger of mobile, web, and video technology creates a dramatic and compelling case for when you should read your text messages. By using Context to form understanding—in this case, of the situation of being behind the wheel, followed by an event (the crash) that would be impossible to duplicate in real life, the user is able to participate in a hypothetical situation in real time. The user becomes part of the experience. Without Context—in this case, the ability to interact with the driver using our mobile phone—our understanding of what transpired would be diminished. We would simply be watching one of those shocking highway safety films, with no personal or emotional stake in what just happened.

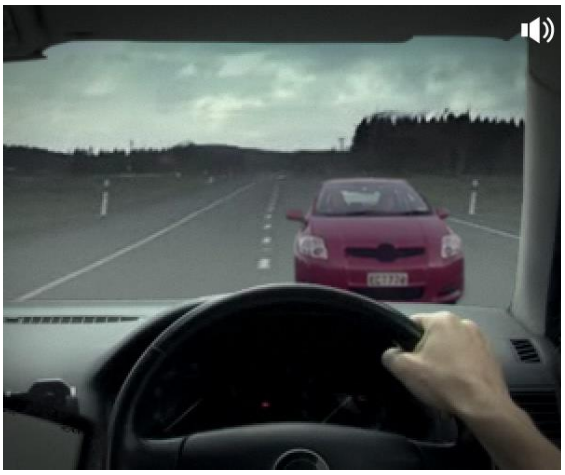


Figure 4-5. The distracted driver crashes due to your text message. Nice work!

2.3.1.1.2 The eRuv project

- Another fascinating example is the mobile eRuv project (http://www.dziga.com/eruv/), which Elliot Malkin created in New York City. An *eruv* is a wire boundary that often surrounds orthodox Jewish communities (see Figure 4-6). Every Saturday, the Jews observe the Sabbath, or the day of rest. During the Sabbath, Jews are unable to perform any work of any kind; for those that observe the stricter concepts, this means that you can't carry anything outside of your private domain, such as carrying your house keys outside of your house. The eruv designates a conceptual area within a public space, as a shared private space for the community where Jews can still observe the Sabbath, carrying their keys with them from their home, but without committing sin.

- Fifty years ago, a large area of Manhattan was designated as an eruv without wires, using the Third Avenue elevated train on the West and the East River on the East (Figure 4-7).

Figure 4-6. A modern-day eruv in the Upper West Side of Manhattan

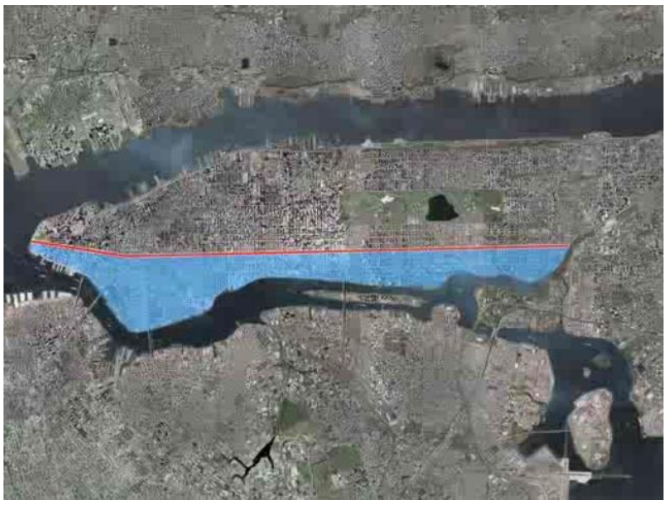


Figure 4-7. The pre-1955 eruv, using physical and natural landmarks instead of wires

- In 1955, the train line was dismantled and replaced with a subway, and thereby the eruv was dismantled, too. To experiment with these concepts and what a modern eruv might look like, Elliot used QR, or quick response, codes (also known as semacodes in the United States) to designate the boundaries of the old Third Avenue eruv (as shown in Figure 4-8). QR codes are two-dimensional barcodes that can be read by the cameras within mobile phones, and display text, or contain a URL to a mobile web page or photo. In this case, Elliot showed a picture of what the eruv looked like 50 years ago.

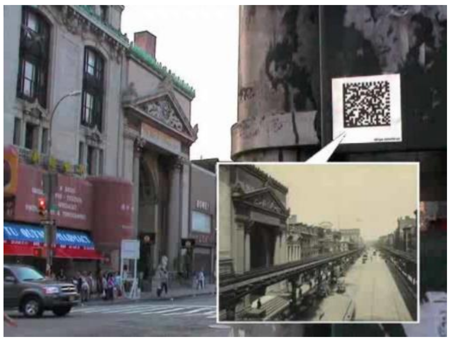


Figure 4-8. An eruv built with QR codes, instead of wires, showing 50-year-old location information

- The eRuv project shows that physical locations can contain information, too. In this case, location contains the information long since gone with the passage of time; it is shown using mobile technology that can be extracted from a mental model of how space is perceived around us.

2.3.1.2 Context with a Lowercase c

- On the other hand, context with a lowercase c is the mode, medium, or environment in which we perform a task or the circumstances of understanding. The following sections look at the three types of context with a little c.

2.3.1.2.1 The present location or physical context

- Let's start with physical or environmental context. Where I am, my location more often than not has meaning. My physical context influences my actions; whether I'm at home, in the office, in a car, on a bus or train, walking the streets of Berlin, etc., each environment will dictate how I access information and therefore how I derive value from it.
- The easiest example might be using a mobile device in a car versus a bus or train. My car is private, so the information I want to extract might be far more personal in nature, but I can't easily input or extract information from a screen because I'm focused on not running off the road and ending up like the distracted driver. On a bus or train, I have little or no privacy, so the information I seek will likely be casual and less meaningful, but because I have no other tasks to perform, I can focus on using the device.
- One interesting mashup of the real world augmented by information from the Web is the Android application Wikitude, which allows you to use the camera in your phone

and literally overlay information about the landmarks in front of you, as shown in Figure 4-9.

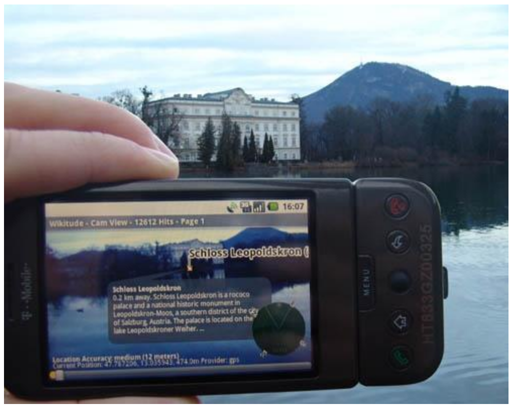

Wikitude - Cam View - 12612 Hits - Page 1
Schloss Leopoldskron
Schloss Leopoldskron
0.2 km away. Schloss Leopoldskron is a rococo palace and a national historic monument in Leopoldskron-Moos, a southern district of the city of Salzburg, Austria. The palace is located on the lake Leopoldskroner Weiher...
Location Accuracy: medium (12 meters)

Figure 4-9. The Android application Wikitude presenting information from the Web in the context the user's present location

2.3.1.2.2 Our present device of access, or media context

- Next let's talk about media. We talked about mass media earlier and the various benefits each medium offers. What we didn't discuss is why people use one medium over another and how each medium provides users different levels of value. For example, a printed newspaper is rich in content, but offers little value to our current task or location. A newspaper is a fixed artifact of the past, given the time it takes to print and distribute. By the time you read it during your morning coffee, it is already out of date.
- Mobile devices may not be as rich in content, but they can provide information on the present, which is a lengthy way of saying that the mobile context offers value that the printed context simply cannot. However, media context isn't just about the immediacy of the information we receive—it also can be used to engage audiences in real time, something that other mediums just can't do.
- The Estonian government will be putting the concept of media context to the test in their 2011 parliamentary elections, allowing citizens to vote for their leaders using SMS. In this case, the government can tabulate results instantly. But imagine a day when citizens can vote on local or national issues in real time, eschewing having to wait for traditional media to report on the effect of their vote, instead seeing the results in real time, as it happens.
- There are already many opting to use the mobile media context in order to be heard. On the immensely popular television show *American Idol*, more votes were cast using a

mobile phone in 2009 (178 million total text message votes) than votes cast in the 2008 presidential election (131 million ballots cast).

2.3.1.2.3 Our present state of mind, or modal context

- The third and final type of context with a little c is modal context, or our presence of mind when we perform a task. Now this one is by far the hardest to explain. Against my mother's better judgment, I decided I would be a geeky technologist when I grew up, rather than a brainy psychologist, but I'll try to explain it as best I can.
- Each time we are exposed to various stimuli, our brains go to work trying to understand the signals our senses send to our brain. Through some miracle of the central nervous system, these signals are converted into reactions. Sometimes they are involuntary physical reactions, like pulling your hand away from a flame, and sometimes they are mental reactions, which we call thoughts and reasoning, like asking yourself, "Should I cross the street now or later?"
- We go about our lives and make thousands of little decisions every day that we don't even consciously consider—perhaps habitual or perhaps instinctual. We make maybe a dozen or so inconsequential decisions like "What should I eat for lunch?" Things that seem important at the time, but we promptly forget. And maybe once or twice a day various decisions string together to form a bigger, more meaningful thought such as "Should I buy it now or later?"
- As humanity becomes more infused with information and ubiquitous access to it, for better or worse we begin to rely on information to aid us in making these decisions, usually for the more complex ones, and increasingly for the more minute.
- Our present state of mind is probably the greatest influencer of what, when, and where we perform tasks. Driven by need, want, or desire we make choices that attempt to accomplish goals—sometimes lofty ones, but more often than not trivial ones. This modal context is at the heart of all deliberate action or inaction.
- This concept is in desperate need of an example, so let me use my trip to Berlin once more. While in Berlin, I wanted to gain understanding of how a single city came to be so emblematic of the cultural division that for over half century influenced the rest of the world, and more importantly how the city overcame these differences to become something new.

- Is there something I can learn from the transformation of Berlin culture and apply to my own life? Is there something I can teach my child, who will otherwise read about the events that occurred in this great city only in history books?
- My presence of mind was to somehow transform artifacts of events and of a people into some type of information that is relevant to me in my own life. Presence of mind drives our actions, to consciously or unconsciously create circumstances in which we gain understanding through the acquisition of information.

2.3.2 Taking the Next Steps

- Wow, that was a little painful, but at least I made my mother proud. All right, let's put all the head-shrinking behind us and get to the practical stuff. The reason to dive into all this big C/little c nonsense is to illustrate that what has meaning to the user and what has meaning to us as the creators of experiences is different in terms of context. Can you guess which one is which?
- You guessed it: users really don't care about physicality, media, or modes, but that doesn't mean that we can ignore them. Just the opposite—good design means that these elements should be invisible to the user. It is our job to create intuitive experiences using technology to anticipate and solve problems for the user through the fewest, deliberate actions. Users tend not to care about what they can't see, which works out for us because making great experiences pays our bills.
- Mobile devices, unlike any other medium, present an amazing opportunity to create contextual, meaningful experiences unlike anything we've ever seen. The trick is mastery, which like all good things, requires time, patience, and understanding. If you can unlock the state of mind of your users and start thinking in their context, understanding how a mobile experience will add value to their lives, you will have the ever-elusive "killer app."
- Do this by thinking of your apps in the different contexts your users will likely encounter:
    - Who are your users? What do you know about them? What type of behavior can you assume or predict?
    - What is happening? What are the circumstances in which they will best absorb the content you intend to present?

    - When will they interact? When they are home and have large amounts of time? At work, where they have short periods of focus? During idle periods, while waiting for a train?
    - Where are they? Are they in a public space or a private space? Are they inside or outside? Is it day or is it night?
    - Why will they use your app? What value will they gain from your content or services in their present situation?
    - How are they using their mobile devices? Are they held in the hand or in the pocket? How are they holding it? Open or closed? Portrait or landscape?
- If you have ever studied journalism, you might notice that the previous is actually the inverted pyramid of telling a news story, expressing who, when, where, what, and how in the first paragraph. The intent is to define the circumstances, or rhetoric, of how we communicate and understand ideas. Interestingly, these concepts go back thousands of years to the works of philosophers Aristotle (384–322 BC) and later Cicero (106–43 BC).

2.4 Mobile Strategy

It starts with developing a mobile strategy. By the phrase "mobile strategy," what I mean is "how much time, effort, and money it will cost you." Have no illusions: mobile design and development don't come cheap. If you target only one platform, you might possibly see success with relatively no risk.

However, once you start attempting to scale that strategy, you will quickly find yourself exponentially increasing the amount of time and effort required to be successful. With the wrong strategy, it can be hard to justify and even harder to monetize the costs of mobile.

I recommend that you view your mobile strategy as a movement that can be transformational to your project, company, or organization. It is about discovering how to infuse a new medium into your business and build an innovative platform that will take you not just into the next year, but into the next decade. For example:

- For a sales-based organization, you could use the mobile web to get information to your salespeople in the field, allowing them to bring up live pricing and estimates while they are at lunch with a client.

- For health care, you could use mobile devices to provide records; access to formularies, policies, and procedures; even patient charts—and all at the patient's bedside.
- For real estate, you can provide potential buyers with live listings of similar properties nearby, information about the house, school information, the average home values in the neighborhood, and mortgage calculators—all while touring a property.
- For local governments, you can use mobile technology for mass transit, to increase awareness or community participation, to elicit feedback on public works projects, and to weigh in on important issues facing the community.
- For retailers, you can provide instant points of purchase without customers having to go to the register, provide ongoing customer support, link online customer reviews and price matching information—all while the customer is holding a product in her hand.
- For arts and entertainment, you can provide information (or even games) about upcoming events and other background information that is all tied into an address book, allowing the customer to plan a fun night out with friends.

The list could go on and on. You might notice that the common thread in each of the opportunities listed here is putting the user in the center of the experience, which is what mobile technology does. Empowering your users empowers the future of your business. And at the center of that transaction is this little always-on, always-connected device that we call the mobile phone.

2.4.1 New Rules

Mobile is a different medium and is governed by a different set of rules. Great mobile products are able to adapt or even look beyond traditional strategy to identify new and unique ways to address both the challenges and the benefits that the mobile medium has to offer.

2.4.1.1 Rule #1: Forget What You Think You Know

The first step is to set aside what you think you know. I've talked to several executives and even developers who are quick to explain that they ignore mobile because of this reason or that reason. Their reason is usually based on outdated or even incorrect information that is guiding their assumptions about the medium.

The truth is that the mobile industry is a highly competitive one, with many big companies and many investors and stakeholders. This produces an enormous amount of press releases filled with speculation and empty promises. With so much hot air being released, some is certain to make its way to the tech journals, blogs, or a casual chat with a knowledgeable technologist. Remember the massive amounts of speculation during the dot-com boom? The mobile industry makes the early dot-coms look like amateurs of spin in comparison.

Following are some tips and tricks to help you forget what you know:

- Leave your baggage at the door. Forget what you think you know about mobile. It is most likely incorrect.
- Don't try to emulate what has been done before and put that in front of your user. Focus on what is right for your user, not what is right for someone else's user.
- Start at the beginning. Even if a project has been in development for a while, you can still start fresh. Provide new perspectives, and breathe life into the project right from the start.

2.4.1.2 Rule #2: Believe What You See, Not What You Read

In mobile, any argument can be made, and for a few thousand dollars you can buy a report or white paper that supports your argument. This doesn't make an argument true, but it certainly can be convincing. And with so many people looking at mobile as an additional revenue stream, but completely clueless as to its inner workings, people are grasping for any information they can find—however inaccurate.

Over the years, I've seen the industry sway from opportunity to opportunity to no avail. Each year is named the year of something. Five years ago it was the "year of messaging." The next year it was video, then mobile social networks (cutely labeled MoSoSo, for "mobile social software"), then it was the mobile widgets, then it was the return of the downloadable app (popular eight years ago), then back to messaging. It is maddening.

There are several firms that provide detailed analysis and insight into the mobile ecosystem, but their data is either very costly or too abstract to create buy-in, forcing us to deduce our own conclusions anyway.

Following are some tips and tricks for believing what you see and not what you read:

- Don’t trust any report, fact, or figure that is more than a year or two old. It is most likely wrong. For example, the majority of assumptions about mobile development pre-iPhone are no longer applicable.
- Perform contextual inquiries, not focus groups. Go to your users and ask them questions in person, in their context, not yours. They often have a lot to say; listen and keep an open mind.
- Record everything. Nothing makes your case like your users’ own words. They have a funny way of reducing company politics and focusing back on the user.
- Don’t forget to innovate. Try new things, be bold, and don’t be afraid to fail.
- The best strategy succeeds even if it fails. Have a contingency plan. If your plan fails to meet expectations, how can you reuse what you’ve learned or done on something else?

2.4.1.3 Rule #3: Constraints Never Come First

This is the tricky one, even for the most seasoned mobile expert. Mobile is a highly constrained medium with many technical obstacles. It is hard not to have constraints of the medium, like devices, networks, or frameworks, influence your decisions.

Creating a great mobile strategy takes a passion for research, some vision, a little risk, and an optimistic, even foolhardy belief that you can make a difference. Unfortunately, these are not the traits you typically find in the halls of corporations. The larger the organization, the more difficult the challenge. We will get to how to circumnavigate this shortcoming later, but for now just try not to cripple yourself from the start.

The moral of my story: don’t invite *that* guy to the room. Mobile projects can be hard to kick off. Start with the big ideas and don’t let the many constraints of the medium kill your project before you’ve begun.

Following are tips and tricks for how to make the constraints not come first:

- Don’t invite that guy to an early-stage brainstorming session. Leave him out of the picture until you have something more concrete that will get him excited about the idea. If you don’t know how to do this, go to the people that have been around for a while and ask. They will most likely know how to get those guys excited about your idea.

- Hopefully it doesn't need to be said, but don't *be* that guy. Refer back to Rule #1 and forget what you think you know. It's probably wrong, and your team is trying to make you even more successful. Let the team do this.
- If you are concerned about the constraints of the mobile medium, know that there will always be constraints in mobile. Get over it. It isn't a deal breaker. Just make sure you aren't the deal breaker. Focus on strategy first, what they user needs, and lay down the features; then, if the constraints become an issue, fall back to the user goals. There is always an alternative.

2.4.1.4 Rule #4: Focus on Context, Goals, and Needs

We already discussed context abstractly. By now you understand that context is crucial to creating any mobile product strategy, but how do you find it? When designing a mobile strategy, predicting the user's context is very hard. It is a moment in time, with so many variables.

We can mentally picture only so many circumstances that could impact the equation before our heads feel like they're going to explode. It isn't impossible, but it isn't easy.

Although product managers might enjoy trying to plan out every conceivable variable at the end of a complex chain, filling our inbox with specifications, user stories, and edge cases, let's start at the beginning of the chain and reverse-engineer it.

Assume for a second that the user's context is based on action. If actions stem from goals and goals are a manifestation of needs, then the user's needs would be the root of the user's context. Needs, unlike context or even goals, are something we can safely predict by looking at some basic information about our user.

Let's use a simple example here: everyone has to eat. It is a basic physical need that everybody must fulfill on a daily basis. The goals associated with this need could be a variety of things, like wanting a particular cuisine, eating only healthy foods, and what to shop for at the local market. The context adds even more variables to our goals, like where I am and what is available around me.

Following are tips on how to focus on context, goals, and needs:

- Defining the users' context is the first thing to do. Without it, you don't have a mobile strategy; you have only a plan of action.

- Uncover the users' goals, and then try and understand how the users' context alters their goals.
- With goals understood, figure out the tasks the users want to perform.
- Look for ways to filter content by context, such as location, media, and model.

2.4.1.5 Rule #5: You Can't Support Everything

One of the first steps in building a mobile strategy is taking an honest look at what devices you can support. I'm here to tell you that you can't support everything.

There are literally hundreds of various device models sold around the world each year. And there are dozens of browsers, each with their own quirks. Think about testing a desktop site for Internet Explorer 6, a web browser notorious for its awful web standards support but immense in its market share, meaning that you have to do a lot of things wrong in order to serve the greatest slice of the market.

Mobile development has the same challenge—but don't forget to multiply the pesky browsers by a factor of 10. Supporting and testing for even a small percentage of the devices in the market will bury your resources before you've got your mobile product to market.

Here are some tips and tricks to remember when it comes to device support:

- Don't kill yourself by trying to support everything. Start with the devices that best represent your core customer.
- Remember, the most popular device or the one that's easiest to develop for might not always be the best device for your project.
- Check your server logs for the devices currently accessing your site. These are the first devices to target.
- Go to your operator store and do a little market research to find out the recommended devices for your target customer.

2.4.1.6 Rule #6: Don't Convert, Create

The question I am most frequently asked is "How do I convert my product to mobile?" and my answer is that you don't. Simply porting a site, application, or game to the mobile medium is a big mistake. Although there is plenty that we can learn from products in other media, the mobile market is unique in its challenges and its benefits. The best place to begin a mobile strategy is by creating a product, not simply trying to reimagine one for small screens.

Remember the early days of website design? We had this incredible new publishing medium, and as more people got online, the need for websites and web pages increased. Many looked to print designers to help create web content, often making sites that were more like glossy brochures than the web pages of today, filled with images, animations, and even sound. They took ages to load, and the text was too hard to read and difficult to search through. It took years to undo the numerous bad habits that we so quickly adopted—and quite nearly killed the Web before it even began. It was all because we designed by the rules of the wrong medium.

Here are some tips and tricks for how to create instead of convert:

- Understand your user and his context. Having an idea of how and when users will access your content will aid in understanding how to best create a tailored mobile experience.
- Don't forget that mobile is a unique medium with its own benefits. Don't try to simply apply the same rationale to your mobile strategy as you would your web or print strategy.

2.4.1.7 Rule #7: Keep It Simple

And if there was only one rule that I could impart to you, something that would summarize all of the previous rules, it would be this one: keep it simple.

Mobile devices are simple, but not stupid. In other words, they are actually very intelligent computers, but people want to use them in a simple way. People don't want you to offer all the features of your existing application or web app. They want simple features that address basic needs and nothing more. The user must deal with many constraints; therefore, we need to show restraint in the mobile products we build.

Adding feature after feature is an easy trap to fall into, but keeping things as simple as you can, from the structure to your design and the devices you support, has numerable benefits. Staying simple means that you'll have far fewer problems from start to finish, making it easier to get it to market, and meaning you'll learn more from your users sooner, in turn meaning that you can iterate and evolve your product faster, more cheaply, and better than others.

Tips and tricks include:

- Seriously, keep it simple! It can be a big challenge, especially for larger organizations, but try to limit the features to only those that are most crucial to your users. Never put your corporate goals or objectives before the users' interests.
- Try to determine the need that will motivate users to act or interact, and build the experience around that and nothing else.

2.4.2 Summary

Bad mobile strategies often start with bad assumptions. No one in mobile claims to know everything about mobile, it is simply too large of an ecosystem to keep tabs on all the facets. I've found the safe bet is to start with an understanding of your users' needs and to follow the seven rules covered in this chapter:

1. Forget what you think you know.
2. Believe what you see, not what you read.
3. Constraints never come first.
4. Focus on context, goals, and needs.
5. You can't support everything.
6. Don't convert, create.
7. Keep it simple.

2.5 Mobile Applications

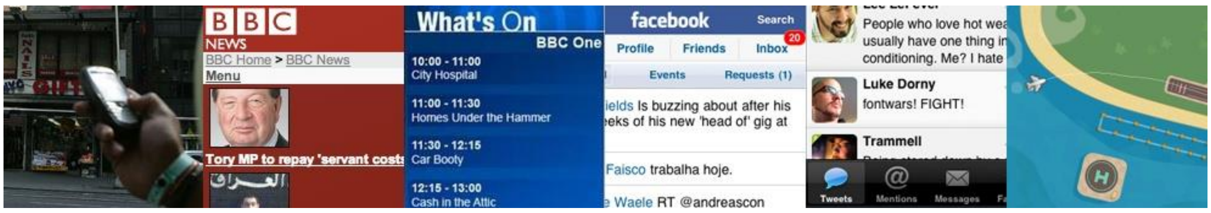
As we've already discussed, the mobile ecosystem is a large and deep pool. In fact, it probably isn't a pool, but an ocean—a really, really big ocean. An ocean is a good metaphor to put the different types of mobile applications in context. You see, in order to traverse an ocean, you need a sturdy boat. Boats of course come in all sorts of shapes and sizes, each with their pros and cons. Some are fast and agile, but carry little cargo. Others are large and lumbering, but can carry tons of people or cargo.

Mobile applications aren't that much different from boats in this seafaring example. You have a number of choices in what medium you use to address your goals, each with their own pros and cons. Some are quick to create but accessible to fewer users. Others address a larger market, but are far more complex and costly.

2.5.1 Mobile Application Medium Types

The *mobile medium type* is the type of application framework or mobile technology that presents content or information to the user. It is a technical approach regarding which type of medium to use; this decision is determined by the impact it will have on the user experience. The technical capabilities and capacity of the publisher also factor into which approach to take.

Figure 6-1 illustrates the spectrum of mobile media; it starts with the basic text-based experiences and moves on to the more immersive experiences.



BBC NEWS
BBC Home > BBC News
Menu
Tory MP to repay 'servant costs
What's On
BBC One
10:00 - 11:00 City Hospital
11:00 - 11:30 Homes Under the Hammer
11:30 - 12:15 Car Booty
12:15 - 13:00 Cash in the Attic
facebook
Search
Profile
Friends
Inbox
Events
Requests (1)
Faisco trabalha hoje.
People who love hot wea usually have one thing in conditioning. Me? I hate
Luke Dorny
fontwars! FIGHT!
Trammell
Tweets
Mentions
Messages

Figure 6-1. Multiple mobile application medium types

2.5.1.1 SMS

The most basic mobile application you can create is an SMS application. Although it might seem odd to consider text messages applications, they are nonetheless a designed experience. Given the ubiquity of devices that support SMS, these applications can be useful tools when integrated with other mobile application types.

SMS applications can be both "free," meaning that there is no additional charge beyond the text message fees an operator charges, or "premium," meaning that you are charged an additional fee in exchange for access to premium content.

A great example of how SMS adds incredible value would be Twitter, where users can receive SMS alerts from their friends and post to their timeline from any mobile device, or the SMS-to-Billboard that BBC World News put up in Midtown Manhattan (Figure 6-2).

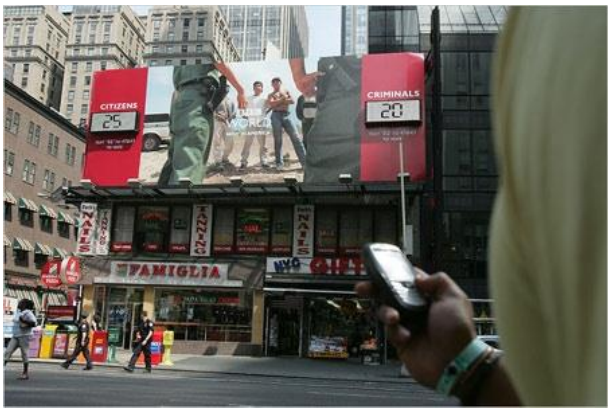


Figure 6-2. An SMS application to interact with a billboard in Manhattan

Pros

The pros of SMS applications include:

- They work on any mobile device nearly instantaneously.
- They're useful for sending timely alerts to the user.
- They can be incorporated into any web or mobile application.
- They can be simple to set up and manage.

Cons

The cons of SMS applications include:

- They're limited to 160 characters.
- They provide a limited text-based experience.
- They can be very expensive.

2.5.1.2 Mobile Websites

As you might expect, a mobile website is a website designed specifically for mobile devices, not to be confused with viewing a site made for desktop browsers on a mobile browser. Mobile websites are characterized by their simple "drill-down" architecture, or the simple presentation of navigation links that take you to a page a level deeper, as shown in Figure 6-3.

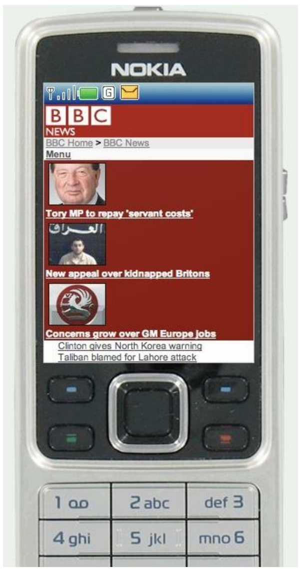

NOKIA
BBC
NEWS
BBC Home > BBC News
Menu
Tory MP to repay 'servant costs'
العراق
New appeal over kidnapped Britons
Concerns grow over GM Europe jobs
Clinton gives North Korea warning
Taliban blamed for Lahore attack
1 oo
2 abc
def 3
4 ghi
5 jkl
mno 6

Figure 6-3. An example of a mobile website

Mobile websites often have a simple design and are typically informational in nature, offering few—if any—of the interactive elements you might expect from a desktop site. Mobile websites have made up the majority of what we consider the mobile web for the past decade, starting with the early WML-based sites (not much more than a list of links) and moving to today's websites, with a richer experience that more closely resembles the visual aesthetic users have come to expect with web content.

Pros

The pros of mobile websites are:

- They are easy to create, maintain, and publish.
- They can use all the same tools and techniques you might already use for desktop sites.
- Nearly all mobile devices can view mobile websites.

Cons

The cons of mobile websites are:

- They can be difficult to support across multiple devices.
- They offer users a limited experience.
- Most mobile websites are simply desktop content reformatted for mobile devices.
- They can load pages slowly, due to network latency.

2.5.1.3 Mobile Web Widgets

Largely in response to the poor experience provided by the mobile web over the years, there has been a growing movement to establish mobile widget frameworks and platforms. For years the mobile web user experience was severely underutilized and failed to gain traction in the market, so several operators, device makers, and publishers began creating widget platforms (Figure 6-4) to counter the mobile web's weaknesses.

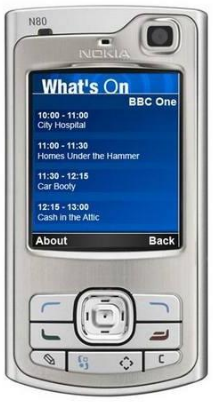


Figure 6-4. An example mobile web widget

Basically, mobile web widgets are small web applications that can't run by themselves; they need to be executed on top of something else. I think one reason for all the confusion around what is a mobile web widget is that this definition can also encompass any web application that runs in a browser. Opera Widgets, Nokia Web RunTime (WRT), Yahoo! Blueprint, and Adobe Flash Lite are all examples of widget platforms that work on a number of mobile handsets. Using a basic knowledge of HTML (or vector graphics in the case of Flash), you can create compelling user experiences that tap into device features and, in many cases, can run while the device is offline.

Pros

The pros of mobile web widgets are:

- They are easy to create, using basic HTML, CSS, and JavaScript knowledge.
- They can be simple to deploy across multiple handsets.
- They offer an improved user experience and a richer design, tapping into device features and offline use.

Cons

The cons of mobile web widgets are:

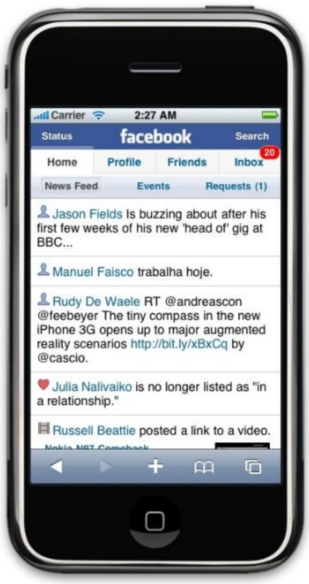
- They typically require a compatible widget platform to be installed on the device.
- They cannot run in any mobile web browser.
- They require learning additional proprietary, non-web-standard techniques.

2.5.1.4 Mobile Web Applications

Mobile web applications are mobile applications that do not need to be installed or compiled on the target device. Using XHTML, CSS, and JavaScript, they are able to provide an application-like experience to the end user while running in any mobile web browser. By "application-like" experience, I mean that they do not use the drill-down or page metaphors in which a click equals a refresh of the content in view. Web applications allow users to interact with content in real time, where a click or touch performs an action within the current view.

The history of how mobile web applications came to be so commonplace is interesting, and is one that I think can give us an understanding of how future mobile trends can be assessed and understood. Shortly after the explosion of Web 2.0, web applications like Facebook, Flickr, and Google Reader hit desktop browsers, and there was discussion of how to bring those same web applications to mobile devices. The Web 2.0 movement brought user-centered design principles to the desktop web, and those same principles were sorely needed in the mobile web space as well.

The challenge, as always, was device fragmentation. The mobile browsers were years behind the desktop browsers, making it nearly impossible for a mobile device to render a comparable experience. While XHTML support had become fairly commonplace across devices, the rendering of CSS2 was wildly inconsistent, and support for JavaScript, necessary or simple DHTML, and Ajax was completely nonexistent.



Jason Fields Is buzzing about after his first few weeks of his new 'head of' gig at BBC...

Manuel Faisco trabalha hoje.

Rudy De Waele RT @andreascon @feebeyer The tiny compass in the new iPhone 3G opens up to major augmented reality scenarios http://bit.ly/xBxCq by @cascio.

Julia Nalivaiko is no longer listed as "in a relationship."

Russell Beattie posted a link to a video.

Figure 6-5. The Facebook mobile web app

Pros

The pros of mobile web applications are:

- They are easy to create, using basic HTML, CSS, and JavaScript knowledge.
- They are simple to deploy across multiple handsets.
- They offer a better user experience and a rich design, tapping into device features and offline use.
- Content is accessible on any mobile web browser.

Cons

The cons of mobile web applications are:

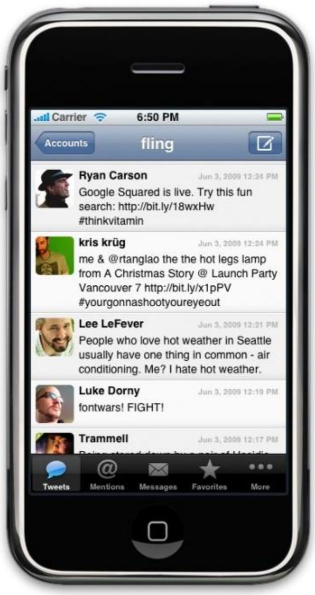
- The optimal experience might not be available on all handsets.
- They can be challenging (but not impossible) to support across multiple devices.
- They don’t always support native application features, like offline mode, location lookup, filesystem access, camera, and so on.

2.5.1.5 Native Applications

The next mobile application medium is the oldest and the most common; it is referred to as native applications, which is actually a misnomer because a mobile web app or mobile web widget can target the native features of the device as well. These applications actually should be called “platform applications,” as they have to be developed and compiled for each mobile platform.

These native or platform applications are built specifically for devices that run the platform in question. The most common of all platforms is Java ME (formerly J2ME). In theory, a device written as a Java ME MIDlet should work on the vast majority of feature phones sold around the world. The reality is that even an application written as a Java ME MIDlet still requires some adaptation and testing for each device it is deployed on.

In the smartphone space, the platform SDKs get much more specific. Although many smartphones are also powered by Java, an operating system layer and APIs added to allow developers to more easily offload complex tasks to the API instead of writing methods from scratch. In addition to Java, other smartphone programming languages include versions of C, C++, and Objective-C (Figure 6-6).



Ryan Carson Jun 3, 2009 12:24 PM
Google Squared is live. Try this fun search: http://bit.ly/18wxHw #thinkvitamin

kris krüg Jun 3, 2009 12:24 PM
me & @rtanglao the the hot legs lamp from A Christmas Story @ Launch Party Vancouver 7 http://bit.ly/x1pPV #yourgonnashootyoureyeout

Lee LeFever Jun 3, 2009 12:21 PM
People who love hot weather in Seattle usually have one thing in common - air conditioning. Me? I hate hot weather.

Luke Dorny Jun 3, 2009 12:19 PM
fontwars! FIGHT!

Trammell Jun 3, 2009 12:17 PM

Figure 6-6. A native application in the iPhone

Pros

The pros of native applications include:

- They offer a best-in-class user experience, offering a rich design and tapping into device features and offline use.
- They are relatively simple to develop for a single platform.
- You can charge for applications.

Cons

The cons of native applications include:

- They cannot be easily ported to other mobile platforms.
- Developing, testing, and supporting multiple device platforms is incredibly costly.
- They require certification and distribution from a third party that you have no control over.
- They require you to share revenue with the one or more third parties.

2.5.1.6 Games

The final mobile medium is games, the most popular of all media available to mobile devices. Technically games are really just native applications that use the similar platform SDKs to create immersive experiences (Figure 6-7). But I treat them differently from native applications for two reasons: they cannot be easily duplicated with web technologies, and porting them to multiple mobile platforms is a bit easier than typical platform-based applications.

Figure 6-7. An example game for the iPhone

The reason games are relatively easy to port (“relatively” being the key word), is that the bulk of the gaming experience is in the graphics and actually uses very little of the device APIs. The game mechanics are the only thing that needs to adapted to the various platforms. Like in console gaming, there are a great number of mobile game porting shops that can quickly take a game written in one language and port it to another.

Pros

The pros of game applications are:

- They provide a simple and easy way to create an immersive experience.
- They can be ported to multiple devices relatively easily.

Cons

The cons of game applications are:

- They can be costly to develop as an original game title.
- They cannot easily be ported to the mobile web.

2.6 Information Architecture

Your information architecture (also known as IA), is the foundation of your mobile product. A well-engineered product with good visual design can still fail because of poor information architecture. The truly successful mobile products always have a well-thought-out information architecture.

From a simple mobile website to an iPhone application, the mobile information architecture defines not just how your information will be structured, but also how people will interact with it. This is made especially tricky when you consider that different devices have different capabilities and therefore different interaction models. Take the way people interact with their devices: for example, a touch device on which the user literally points and clicks, or a more basic device on which the user uses the directional pad to navigate to the desired location.

2.6.1 What Is Information Architecture?

- The structural design of shared information environments
- The combination of organizations, labeling, search, and navigation systems within websites and intranets
- The art and science of shaping information products and experiences to support usability and findability
- An emerging discipline and community of practice focused on bringing principles of design and architecture to the digital landscape

Similar to how mobile technology has many facets, so does information architecture, as it is often used as an umbrella term to describe several unique disciplines, including the following:

Information architecture

- The organization of data within an informational space. In other words, how the user will get to information or perform tasks within a website or application.

Interaction design

- The design of how the user can participate with the information present, either in a direct or indirect way, meaning how the user will interact with the website of application to create a more meaningful experience and accomplish her goals.

Information design

- The visual layout of information or how the user will assess meaning and direction given the information presented to him.

Navigation design

- The words used to describe information spaces; the labels or triggers used to tell the users what something is and to establish the expectation of what they will find.

Interface design

- The design of the visual paradigms used to create action or understanding.

2.6.2 Mobile Information Architecture

Although information architecture has become a common discipline in the web industry, unfortunately, the mobile industry—like software—has only a handful of specialized mobile information architects. Although mobile information architecture is hardly a discipline in its own right, it certainly ought to be. This is not because it is so dissimilar from its desktop cousin, but because of context, added technical constraints, and needing to display on a smaller screen as much information as we would on a desktop.

2.6.2.1 Keeping It Simple

When thinking about your mobile information architecture, you want to keep it as simple as possible.

Support your defined goals

If something doesn't support the defined goals, lose it. Go back to your user goals and needs, and identify the tasks that map to them. Find those needs and fill them.

Clear, simple labels

Good trigger labels, the words we use to describe each link or action, are crucial in Mobile. Words like "products" or "services" aren't good trigger labels. They don't tell us anything about that content or what we can expect. Now, I would argue that good trigger labels are crucial in the Web as well, that we've become lazy and we assume so much about the user that we ignore the use of good trigger labels. Users have a much higher threshold of pain when clicking about on a desktop site or application, hunting and pecking for tasty morsels. Mobile performs short, to-the-point, get-it-quick, and get-out types of tasks. What is convenient on the desktop might be a deal breaker on mobile.

2.6.2.2 Site Maps

The first deliverable we use to define mobile information architecture is the site map. Site maps are a classic information architecture deliverable. They visually represent the relationship of content to other content and provide a map for how the user will travel through the informational space, as shown in Figure 7-4.

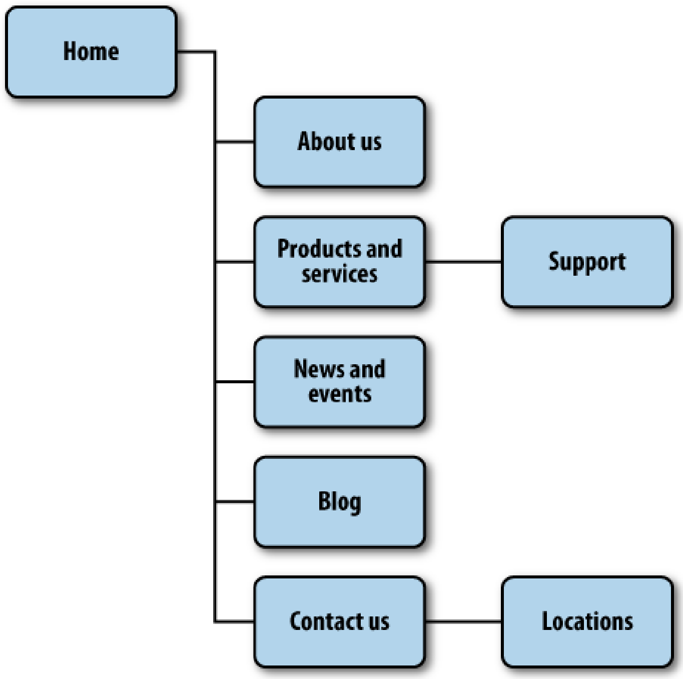


Figure 7-4. An example mobile site map

Mobile site maps aren't that dissimilar from site maps we might use on the Web. But there are a few tips specific to mobile that we want to consider.

Limit opportunities for mistakes

Imagine a road with a fork in it. We can go either left or right. The risk that we will make the wrong choice is only 50 percent, meaning that we have a better than good chance that we will get to where we want to go. But imagine three roads. Now our chances have dropped to 33 percent. Four roads drops your chances to 25 percent, and five roads takes you down to 20 percent. Now a 20 percent chance isn't great, but it isn't too bad, either.

In Figure 7-5, you can see a poorly designed mobile information architecture that too closely mimics its desktop cousin; it was not designed with the mobile user in mind.

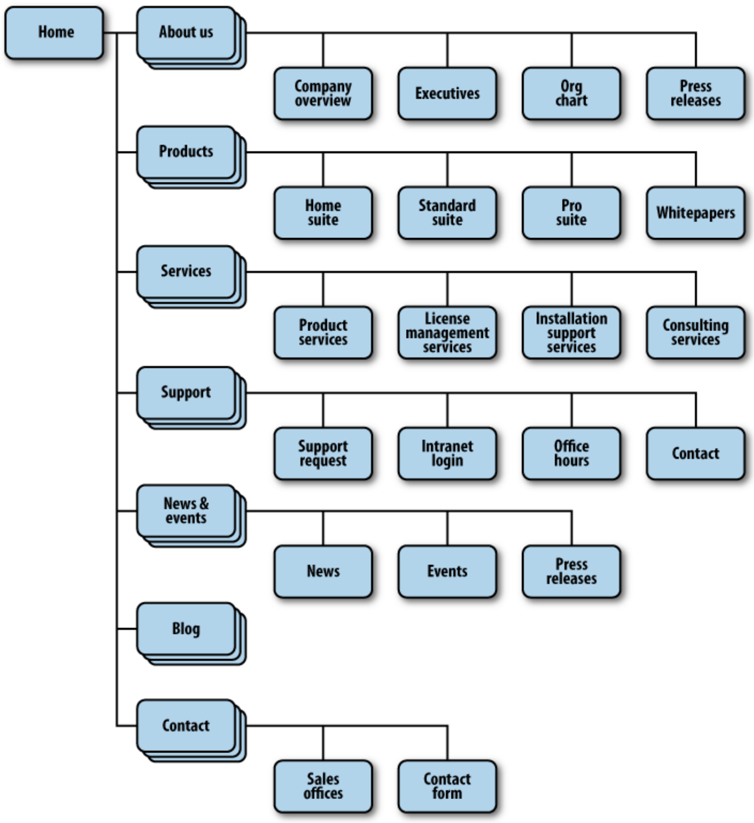


Figure 7-5. An example of a bad mobile information architecture that was designed with desktop users in mind rather than mobile users

But in mobile, we cannot make this assumption. In the mobile context, tasks are short and users have limited time to perform them. And with mobile websites, we can't assume that the users have access to a reliable broadband connection that allows them to quickly go back to the previous page. In addition, the users more often than not have to pay for each page view in data charges. So not only do they pay cash for viewing the wrong page by mistake, they pay to again download the page they started from: we can't assume that pages will be cached properly.

Confirm the path by teasing content

After the users have selected a path, it isn't always clear whether they are getting to where they need to be. Information-heavy sites and applications often employ nested or drill-down architectures, forcing the user to select category after category to get to their target. To reduce risking the user's time and money, we want to make sure we present enough information for the user to wade through our information architecture successfully. On the Web, we take these risks very lightly, but with mobile, we must give our users a helping hand. We do this by teasing content within each category—that is, providing at least one content item per category.

In Figure 7-6, you can see in a constrained screen that teasing the first few items of the page provides the user with a much more intuitive interface, immediately indicating what type of content the user can expect.

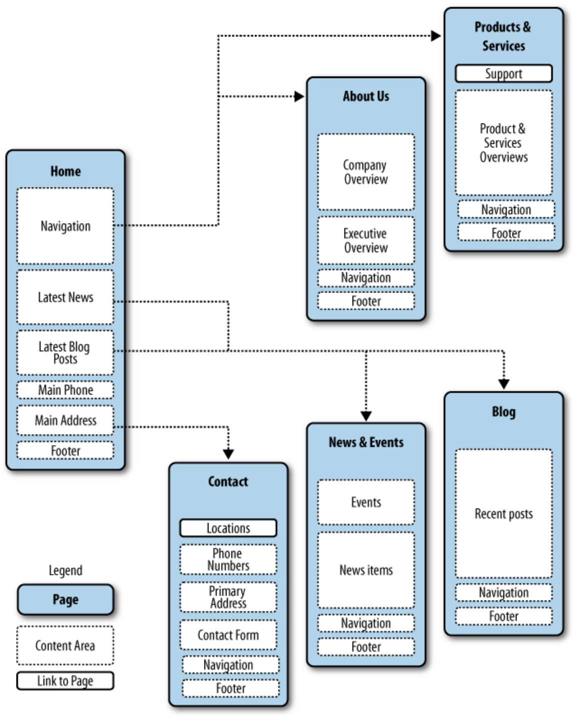


Figure 7-6. Teasing content to confirm the user's expectations of the content within

We immediately saw that users were finding content more quickly, driving up our sales. It was like night and day. Since those days, I've tested this principle on a variety of mobile sites—not just ringtones, but game catalogs, news sites, and regular old corporate websites. Each time, it has improved the conversion, getting users to the content they seek with the least amount of backtracking.

2.6.2.3 Clickstreams

Clickstream is a term used for showing the behavior on websites, displaying the order in which users travel through a site's information architecture, usually based on data gathered from server logs. Clickstreams are usually historical, used to see the flaws in your information architecture, typically using heat-mapping or simple percentages to show where your users are going. I've always found them to be a useful tool for rearchitecting large websites.

However, information architecture in mobile is more like software than it is the Web, meaning that you create clickstreams in the beginning, not the end. This maps the ideal path the user will take to perform common tasks. Being able to visually lay out the path users will take gives you a holistic or bird's-eye view of your mobile information architecture, just as a road map does. When you can see all the paths next to each other and take a step back, you start to see shortcuts and how you can get users to their goal faster or easier, as shown in Figure 7-7.

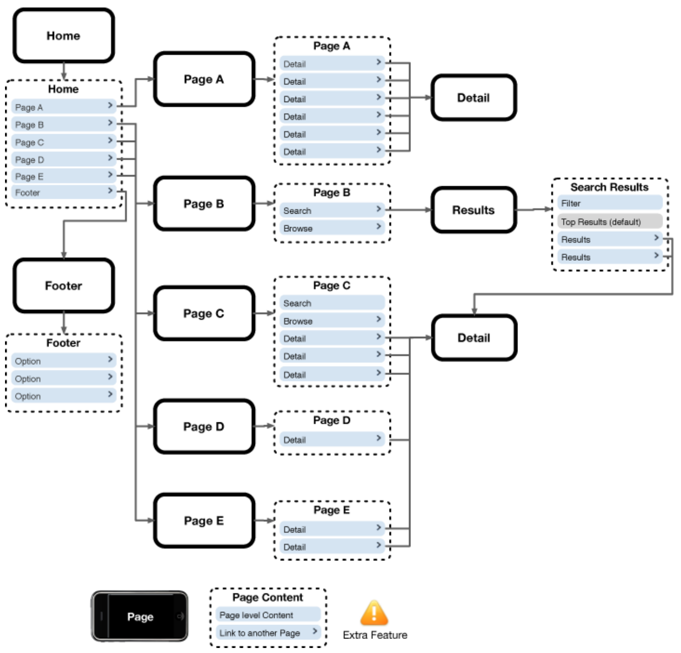


Figure 7-7. An example clickstream for an iPhone web application

Now the business analyst in you is probably saying, "Just create user or process flows," such as the esoteric diagram shown in Figure 7-8, which is made up of boxes and diamonds that look more like circuit board diagrams than an information architecture.

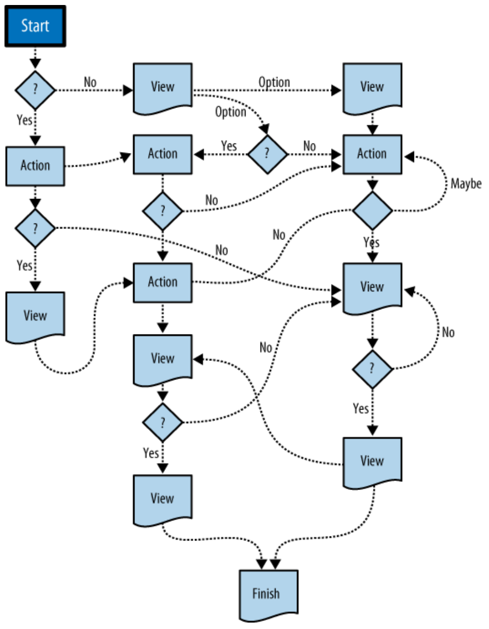


Figure 7-8. An example process flow diagram

If that is what your team prefers, then by all means, flow away. Personally, I like to present all of my information architecture deliverables from the perspective of the user, using the same metaphors she will use to make her way through my information architecture—in this case, either a screen or page metaphor.

2.6.2.4 Wireframes

The next information architecture tool at our disposal is wireframes. *Wireframes* are a way to lay out information on the page, also referred to as *information design*. Site maps show how our content is organized in our informational space; wireframes show how the user will directly interact with it. Wireframes are like the peanut butter to the site map jelly in our information architecture sandwich. It's the stuff that sticks. Wireframes like the one in Figure 7-9 serve to make our information space tangible and useful.

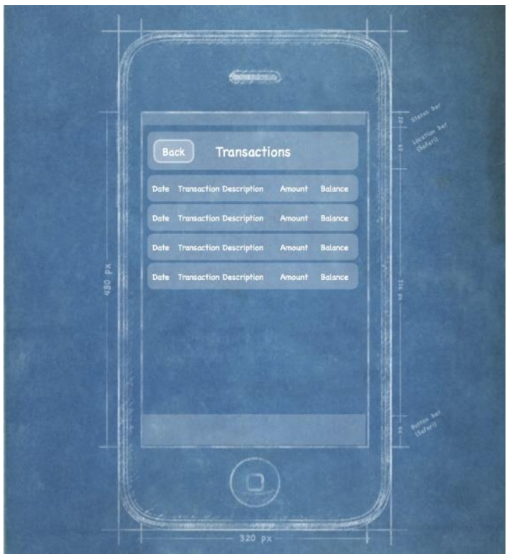

Back
Transactions
Date Transaction Description Amount Balance
Date Transaction Description Amount Balance
Date Transaction Description Amount Balance
Date Transaction Description Amount Balance
480 px
320 px

Figure 7-9. An example of an iPhone web application wireframe, intended to be low fidelity to prevent confusion of visual design concepts with information design concepts

Most common are what I call “in-place” interactions, or areas where the user can interact with an element without leaving the page. This can be done with Ajax or a little show/hide JavaScript. These interactions can include copious amounts of annotation, describing each content area in as much length as you can fit in the margins of the page, as shown in Figure 7-10.

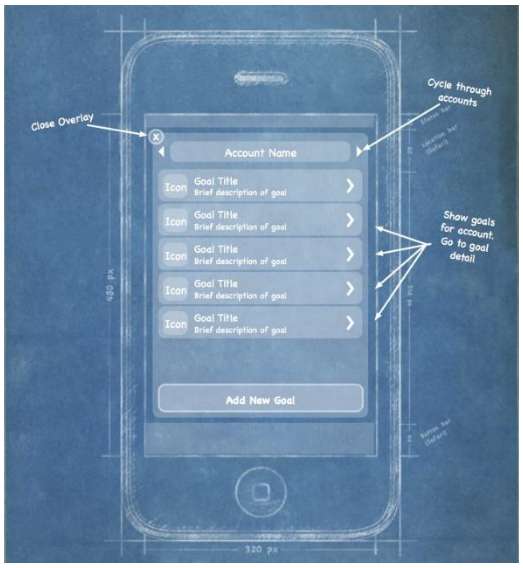

Close Overlay
Cycle through accounts
Account Name
Icon Goal Title Brief description of goal
Icon Goal Title Brief description of goal
Icon Goal Title Brief description of goal
Icon Goal Title Brief description of goal
Icon Goal Title Brief description of goal
Show goals for account. Go to goal detail
Add New Goal
480 px
320 px

Figure 7-10. Using annotations to indicate the desired interactions of the site or application

At this point, I highly recommend that you get some feedback from either others on your project or my most trusted reviewer, my wife. Well, not *my* wife, but someone you know and trust—and the less technical, the better. Have her review your work as you iterate through ideas.

2.6.2.5 Prototyping

As mentioned before, wireframes lack the capability to communicate more complex, often in-place, interactions of mobile experiences. This is where prototypes come in. Prototypes might sound like a scary (or costly) step in the process. Some view them as redundant or too time-consuming, preferring to jump in and start coding things. But as with wireframes, I’ve found that each product we’ve built out some sort of prototype has saved both time and money. The following sections discuss some ways to do some simple and fast mobile prototyping.

Paper prototypes

The most basic level we have is paper prototyping: taking our printed-out wireframes or even drawings of our interface, like the one shown in Figure 7-11, and putting them in front of people.

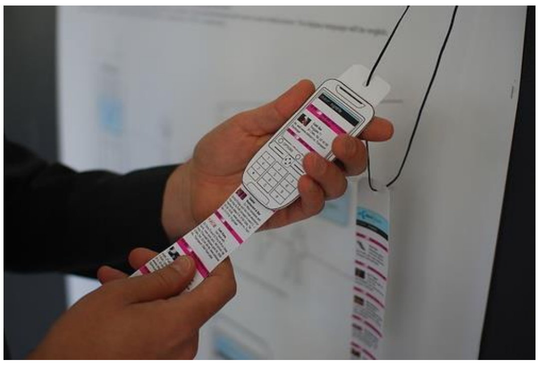


Figure 7-11. A paper prototype, where the interaction is nothing more than drawings on note cards

Create a basic script of tasks (hopefully based on either context or user input) and ask users to perform them, pointing to what they would do. You act as the device, changing the screens for them. For paper prototypes, I don’t recommend using full sheets of paper; instead try using small blank note cards, and for lower-end devices, use business-card-sized paper (Figure 7-12). The size matters and you’ll learn as much from how the user manages working with small media as you will what information is actually on it.

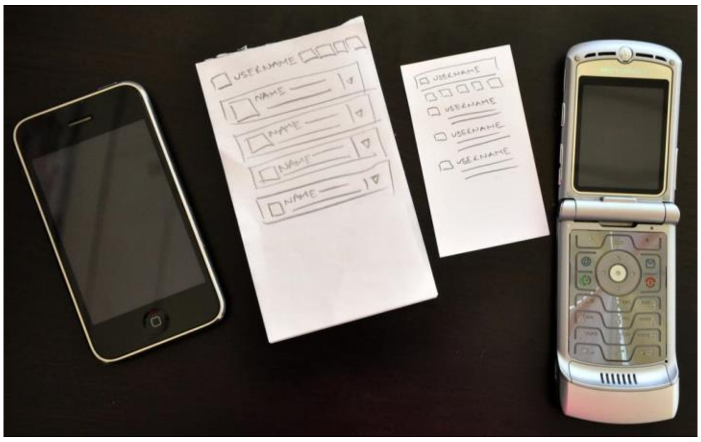


Figure 7-12. A touch interface paper prototype next to its smaller sibling

Context prototype

The next step is creating a context prototype (Figure 7-13). Take a higher-end device that enables you to load full-screen images on it. Take your wireframes or sketches and load them onto the device, sized to fill the device screen. Leave the office. Go for a walk down to your nearest café. Or get on a bus or a train. As you are traveling about, pull out your device and start looking your interface in the various contexts you find yourself currently in.

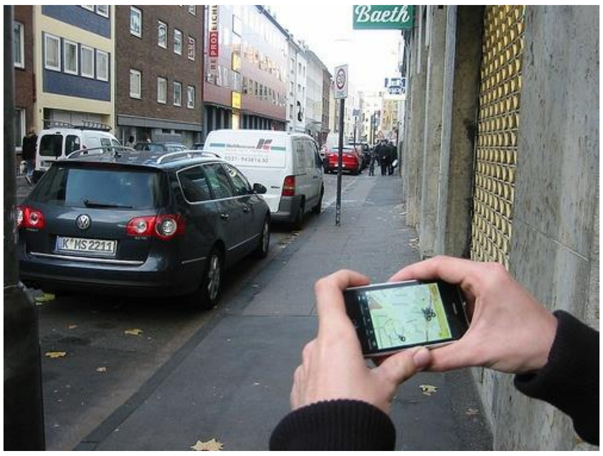
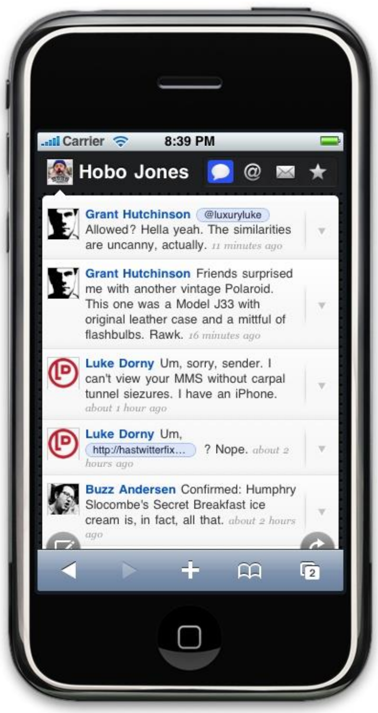


Figure 7-13. An example of a context prototype, or taking images loaded onto a device and testing them in the mobile context

Pay particular attention to what you are thinking and your physical behavior while you are using your interface and then write it down. If you are brave and don't have strict nondisclosure issues, ask the people around you to use it, too. I wouldn't bother with timing interactions or sessions, but try to keep an eye on a clock to determine how long the average session is.

HTML prototypes

The third step is creating a lightweight, semifunctional static prototype using XHTML, CSS, and JavaScript, if available. This is a prototype that you can actually load onto a device and produce the nearest experience to the final product, but with static dummy content and data (Figure 7-14). It takes a little extra time, but it is worth the effort.



Grant Hutchinson @luxuryluke Allowed? Hella yeah. The similarities are uncanny, actually. 11 minutes ago

Grant Hutchinson Friends surprised me with another vintage Polaroid. This one was a Model J33 with original leather case and a mittful of flashbulbs. Rawk. 16 minutes ago

Luke Dorny Um, sorry, sender. I can't view your MMS without carpal tunnel siezures. I have an iPhone. about 1 hour ago

Luke Dorny Um, http://hastwitterfix... ? Nope. about 2 hours ago

Buzz Andersen Confirmed: Humphry Slocombe's Secret Breakfast ice cream is, in fact, all that. about 2 hours ago

Figure 7-14. An XHTML prototype that you can actually interact with on real mobile devices

With a static XHTML prototype, you use all the device metaphors of navigation, you see how much content will really be displayed on screen (it is always less than you expect), and you have to deal with slow load times and network latency. In short, you will feel the same pains your user will go through.

2.6.2.6 Different Information Architecture for Different Devices

Depending on which devices you need to support, mobile wireframes can range from the very basic to the complex. On the higher-end devices with larger screens, we might be inclined to add more interactions, buttons, and other clutter to the screen, but this would be a mistake. Just because the user might have a more advanced phone, that doesn’t mean that he is giving you license to pack his screen with as much information as you can muster.

The motivations, goals, and how users will interact with a mobile experience are the same at the low end as they are on a high-end device. For the latter, you just have better tools to express the content. You can learn a lot from designing for the lower end first. The greatest challenge in creating valuable experiences is knowing when to lose what you don't need. You don't have a choice on lower-end devices—it must be simple. When designing for both, it is best to try and to keep your information architecture as close to each other as possible without sacrificing the user experience. They say that simple design is the hardest design, and this principle certainly is true when designing information architecture for mobile devices.

2.6.3 The Design Myth

A little secret about interactive design is that people don't respond to the visual aesthetic as much as you might think. What colors you use, whether you use square or rounded corners, or, gradients or flat backgrounds, helps build first impressions, but it doesn't do too much to improve the user's experience.

Just look at one of the top-selling iPhone Twitter applications, Tweetie, shown in Figure 7-15. Many consider Tweetie to be a "well-designed" application, but because it is built from the same API as all other iPhone applications, at first glance there is little that is actually visually distinctive between this and other applications. What makes this application "well designed" is how the content is applied to the context of the user—in other words, the mobile information architecture. In this example, the information design uses common layout metaphors, highlighted on the righthand side of Figure 7-15 to provide the user with familiar placement of common tasks, allowing the user to perform repetitive tasks common with most Twitter applications. The point is great information design is often mistaken for great visual design.

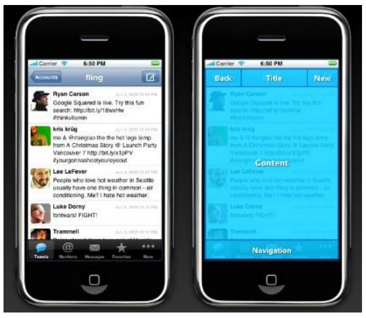


Figure 7-15. Comparing visual design to information design of the iPhone application Tweetie

Most non–information architects almost always do information architecture in some form or another; often, they don't even know they are doing it. They might do a few wireframes, or maybe a site map. Sometimes designers will jump in and incorporate information architecture deliverables directly into their designs. By not focusing on the information architecture exclusively from the start, you risk confusing your disciplines, your deliverables, and ultimately your direction. The more time you spend focusing on just your information architecture, the faster and less costly your project will be.

2.7 Design

When building mobile experiences, it is impossible to create a great experience without three ingredients: context, information architecture, and visual design. The visual design of your experience is the direct representation of everything underneath; it is the first impression the user will have. A great design gives the user high expectations of your site or application; a poor design leads to lower expectations.

Users' expectations translate to value and trust. In the mobile space, where content is often "free" (they still need to pay for data charges), users often have low expectations, due to the limitations of the canvas. Confusing site structures and slow download speeds reinforce those initial low expectations. In the downloadable application space, where application design can be much more robust, users must purchase applications almost sight unseen. For example, they may see just a small screenshot of your application or game. But if the application doesn't meet the higher expectations of the design, your application downloads will drop like a stone. The design, that first impression, determines right from the start if the user will spend five seconds, five minutes, or five hours with your product.

On computers, there is a strategy called "lowest common denominator": in order to reach the widest possible number of platforms, you create a product that works on the most common architectural components on all platforms (see Figure 8-1).

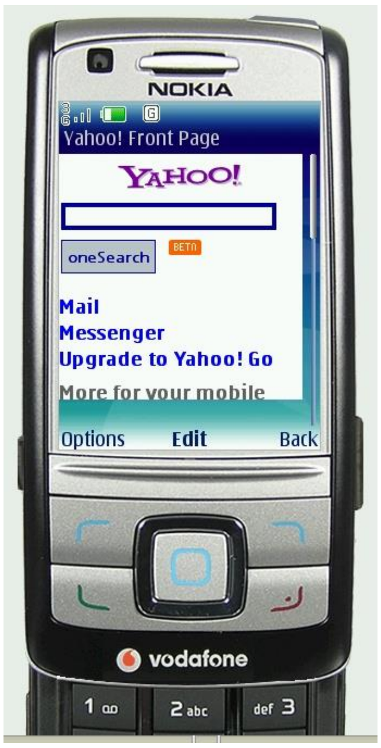


Figure 8-1. A lowest-common-denominator design

Typically, mobile design starts with the lowest common denominator. As a designer, you ask yourself, “How do I visually express this content across the most possible devices?” You start with the most basic of designs, catering to the limitations of the device. You try to pepper in some nice-looking elements until you’ve reached the extent that the device platform can tolerate. You are left with a Frankenstein-like design that only your mother could love.

2.7.1 Interpreting Design

Mobile design reminds me of a 25-year veteran graphic designer friend of mine. His days were spent creating print designs and advertisements, defining precisely what each design would be, down to the picas and points. His method of design meant creating a vision for how to communicate information or ideas in his head, and then duplicating that on the printed page. In his mind, it wasn’t right unless it was exactly like his original vision.

In the end, the graphic designer and I scrapped the work, and he provided me his pica-perfect designs as giant images, which I turned into a series of massive image maps. To this day, I do not work with graphic designers on web or mobile projects.

2.7.2 The Mobile Design Tent-Pole

In Hollywood, executives like to use the term “tent-pole” to describe their movies and television shows. The term has dual meanings: one is business, and the other creative. The business goal of a tent-pole production is to support or prop up the losses from other productions.

However, to create a tent-pole production, the creators involved must make an artistic work that they know will appeal to the largest possible audience, providing something for everyone. You probably know tent-pole movies as “blockbusters”; in television, they are known as “anchor” shows.

Apple’s App Store quickly changed that. You can clearly see that the best-selling games and applications for the iPhone are the ones with the best designs (Figure 8-2).

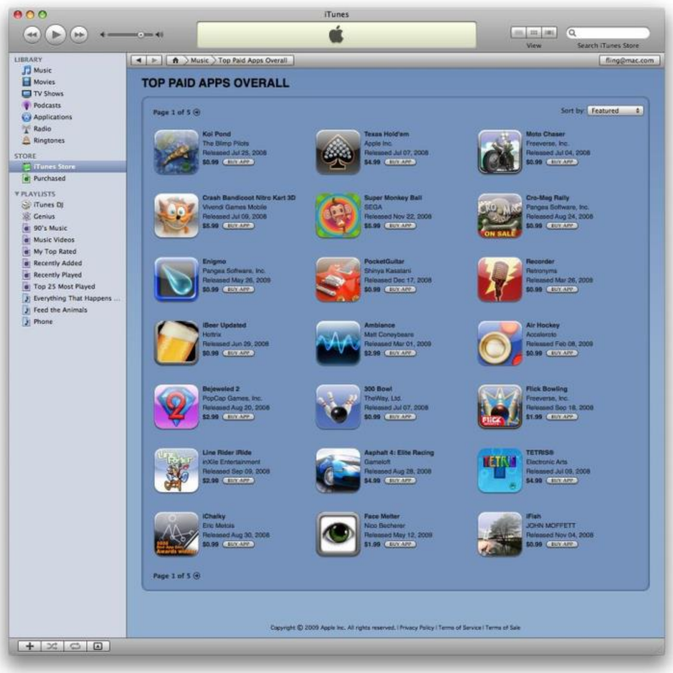
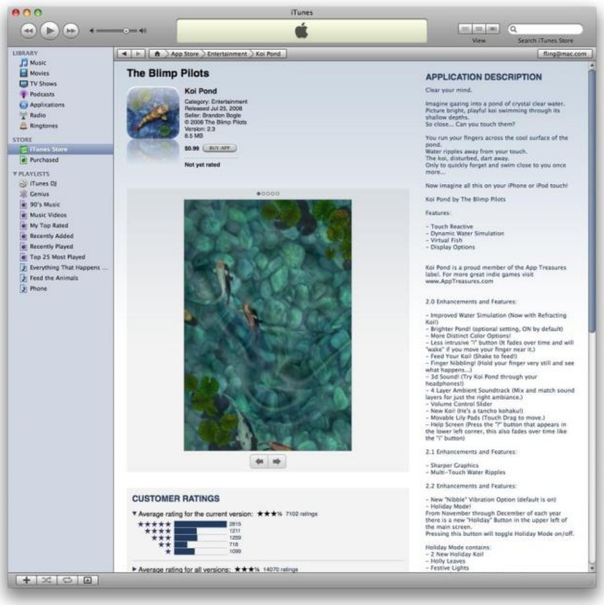


Figure 8-2. The app icon design greatly influences the user’s expectation of quality

Users look at multiple screenshots (Figure 8-3), read the user reviews, and judge the product based on the quality of its icon and of the screenshots before they buy.



The Blimp Pilots

Koi Pond
Category: Entertainment
Released Jul 25, 2008
Seller: Brandon Bogle
© 2008 The Blimp Pilots
Version: 2.3
8.5 MB
$0.99
Not yet rated

CUSTOMER RATINGS
▼ Average rating for the current version: ★★★½ 7102 ratings

★★★★★	2815
★★★★	1211
★★★	1259
★★	718
★	1099

▶ Average rating for all versions: ★★★½ 14070 ratings

APPLICATION DESCRIPTION
Clear your mind.

Imagine gazing into a pond of crystal clear water.
Picture bright, playful koi swimming through its shallow depths.
So close... Can you touch them?

You run your fingers across the cool surface of the pond.
Water ripples away from your touch.
The koi, disturbed, dart away.
Only to quickly forget and swim close to you once more...

Now imagine all this on your iPhone or iPod touch!

Koi Pond by The Blimp Pilots

Features:

- Touch Reactive
- Dynamic Water Simulation
- Virtual Fish
- Display Options

Koi Pond is a proud member of the App Treasures label. For more great indie games visit www.AppTreasures.com

2.0 Enhancements and Features:

- Improved Water Simulation (Now with Refracting Koi!)
- Brighter Pond! (optional setting, ON by default)
- More Distinct Color Options!
- Less intrusive "i" button (It fades over time and will "wake" if you move your finger near it.)
- Feed Your Koi! (Shake to feed!)
- Finger Nibbling! (Hold your finger very still and see what happens...)
- 3d Sound! (Try Koi Pond through your headphones!)
- 4 Layer Ambient Soundtrack (Mix and match sound layers for just the right ambiance.)
- Volume Control Slider
- New Koi! (He's a tancho kohaku!)
- Movable Lily Pads (Touch Drag to move.)
- Help Screen (Press the "?" button that appears in the lower left corner, this also fades over time like the "i" button)

2.1 Enhancements and Features:

- Sharper Graphics
- Multi-Touch Water Ripples

2.2 Enhancements and Features:

- New "Nibble" Vibration Option (default is on)
- Holiday Mode!
From November through December of each year there is a new "Holiday" Button in the upper left of the main screen.
Pressing this button will toggle Holiday Mode on/off.

Holiday Mode contains:
- 2 New Holiday Koi!
- Holly Leaves
- Festive Lights

Figure 8-3. Users are able to determine the quality of the app, largely influenced by the design, before they make a purchase

The Apple App Store is proving everyday that mobile design doesn't have to start with tent-pole lowest-common-denominator products—it can instead start with providing the best possible experience and tailoring that experience to the market that wants it most.

2.7.3 Designing for the Best Possible Experience

The lesson here is that although it may defy your business instincts to focus your product on just one device, in mobile development, the risks and costs of creating that tent-pole product are just too high. This lesson is so easily seen through bad or just plain uninspired mobile design. Asking creative people to create uninspiring work is a fast track to mediocrity.

Here is a design solution: design for the best possible experience. Actually, don't just design for it: focus on creating the best possible experience with unwavering passion and commitment. Iterate, tweak, and fine-tune until you get it right. Anything less is simply unacceptable. Do not get hindered by the constraints of the technology. Phrases like "lowest common denominator" cannot be part of the designer's vocabulary.

Your design—no, your work of art—should serve as the shining example of what the experience should be, not what it can be. Trying to create a mobile design in the context of the device constraints isn't where you start; it is where you should end.

2.7.4 The Elements of Mobile Design

Doing this involves knowing the six elements of mobile design that you need to consider, starting with the context and layering in visual elements or laying out content to achieve the design goal. Then, you need to understand how to use the specific tools to create mobile design, and finally, you need to understand the specific design considerations of the mobile medium.

2.7.4.1 Context

As the designer, it is your job to make sure that the user can figure out how to address context using your app. Make sure you do your homework to answer the following questions:

- Who are the users? What do you know about them? What type of behavior can you assume or predict about the users?
- What is happening? What are the circumstances in which the users will best absorb the content you intend to present?
- When will they interact? Are they at home and have large amounts of time? Are they at work where they have short periods of time? Will they have idle periods of time while waiting for a train, for example?
- Where are the users? Are they in a public space or a private space? Are they inside or outside? Is it day or is it night?
- Why will they use your app? What value will they gain from your content or services in their present situation?
- How are they using their mobile device? Is it held in their hand or in their pocket? How are they holding it? Open or closed? Portrait or landscape?

The answers to these questions will greatly affect the course of your design. Treat these questions as a checklist to your design from start to finish. They can provide not only great inspiration for design challenges, but justification for your design decisions later.

2.7.4.2 Message

Another design element is your message, or what you are trying to say about your site or application visually. One might also call it the “branding,” although I see branding and messaging as two different things. Your message is the overall mental impression you create explicitly through visual design. I like to think of it as the holistic or at times instinctual reaction someone will have to your design. If you take a step back, and look at a design from a distance, what is your impression? Or conversely, look at a design for 30 seconds, and then put it down. What words would you use to describe the experience?

For example, hold the book away from you and look at each of the designs in Figure 8-4; try not to focus too heavily on the content. What do each of these designs "say" to you?

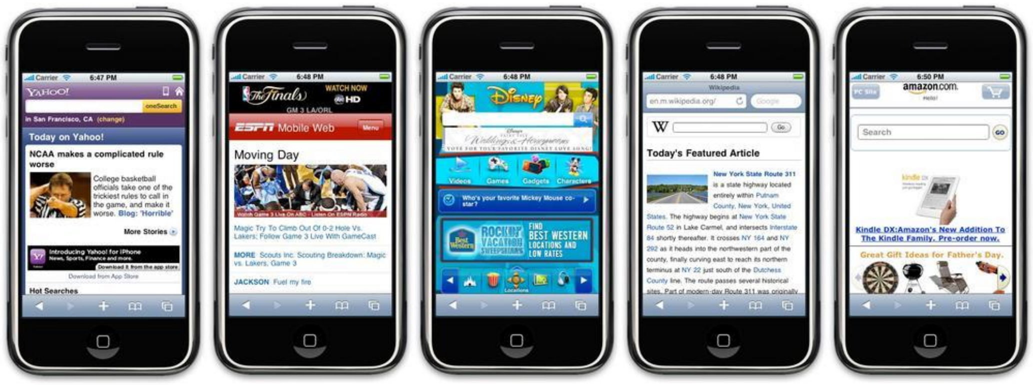


Figure 8-4. What is the message for each of these designs?

Which of the following designs provide a message? What do they say to you?

Yahoo!

Yahoo! sort of delivers a message. This app provides a clean interface, putting a focus on search and location, using color to separate it from the news content. But I'm not exactly sure what it is saying. Words you might use to describe the message are crisp, clean, and sharp.

ESPN

The ESPN site clearly is missing a message. It is heavily text-based, trying to put a lot of content above the fold, but doesn't exactly deliver a message of any kind. If you took out the ESPN logo, you likely would have indifferent expectations of this site; it could be about anything, as the design doesn't help set expectations for the user in any way. Words you might use to describe the message: bold, cluttered, and content-heavy.

Disney

Disney creates a message with its design. It gives you a lot to look at—probably too much—but it clearly tries to say that the company is about characters for a younger audience. Words you might use to describe the message: bold, busy, and disorienting.

Wikipedia

The Wikipedia design clearly establishes a message. With a prominent search and text-heavy layout featuring an article, you know what you are getting with this design. Words you might use to describe the message: clean, minimal, and text-heavy.

Amazon

Amazon sort of creates a message. Although there are some wasted opportunities above the fold with the odd ad placement, you can see that it is mostly about products (which is improved even more if you scroll down). Words you might use to describe the message: minimal but messy, product-heavy, and disorienting.

The words you might use to describe these designs might be completely different than mine—thus the beauty and the curse of visual design. The important thing isn't my opinion—it is the opinion of your user. Does the design convey the right message to your user in the right context? If you aren't sure, it might be a good time to find out.

2.7.4.3 Look and Feel

The concept of "look and feel" is an odd one, being subjective and hard to define. Typically, look and feel is used to describe appearance, as in "I want a clean look and feel" or "I want a usable look and feel." The problem is: as a mobile designer, what does it mean? And how is that different than messaging?

For example, in Figure 8-5 you can see the site Pattern Tap, which is a visual collection of many user interface patterns meant for websites and web applications, but there is no reason why it can't serve as inspiration for your mobile projects as well.

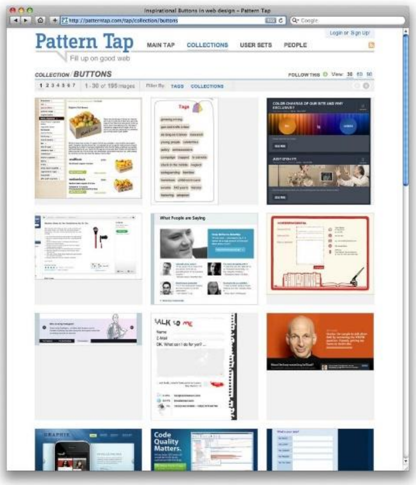
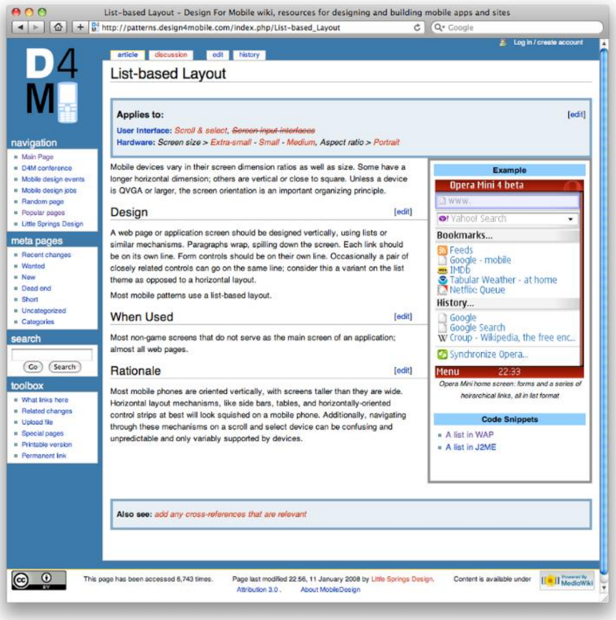


Figure 8-5. Pattern Tap shows a number of user interface patterns that help to establish look and feel

In Figure 8-6 you can see an example of a mobile design pattern at the Design4Mobile pattern library.



List-based Layout - Design For Mobile wiki, resources for designing and building mobile apps and sites

http://patterns.design4mobile.com/index.php/List-based_Layout

Log in / create account

article | discussion | edit | history

navigation
- Main Page
- D4M conference
- Mobile design events
- Mobile design jobs
- Random page
- Popular pages
- Little Springs Design

meta pages
- Recent changes
- Wanted
- New
- Dead end
- Short
- Uncategorized
- Categories

search

Go | Search

toolbox
- What links here
- Related changes
- Upload file
- Special pages
- Printable version
- Permanent link

List-based Layout

Applies to: [edit]

User Interface: *Scroll & select*, ~~*Screen input interfaces*~~

Hardware: *Screen size > Extra-small - Small - Medium, Aspect ratio > Portrait*

Mobile devices vary in their screen dimension ratios as well as size. Some have a longer horizontal dimension; others are vertical or close to square. Unless a device is QVGA or larger, the screen orientation is an important organizing principle.

Design [edit]

A web page or application screen should be designed vertically, using lists or similar mechanisms. Paragraphs wrap, spilling down the screen. Each link should be on its own line. Form controls should be on their own line. Occasionally a pair of closely related controls can go on the same line; consider this a variant on the list theme as opposed to a horizontal layout.

Most mobile patterns use a list-based layout.

When Used [edit]

Most non-game screens that do not serve as the main screen of an application; almost all web pages.

Rationale [edit]

Most mobile phones are oriented vertically, with screens taller than they are wide. Horizontal layout mechanisms, like side bars, tables, and horizontally-oriented control strips at best will look squished on a mobile phone. Additionally, navigating through these mechanisms on a scroll and select device can be confusing and unpredictable and only variably supported by devices.

Example

Opera Mini home screen: forms and a series of heirarchical links, all in list format

Code Snippets
- A list in WAP
- A list in J2ME

Also see: *add any cross-references that are relevant*

This page has been accessed 6,743 times. Page last modified 22:56, 11 January 2008 by Little Springs Design. Content is available under Attribution 3.0 . About MobileDesign

Figure 8-6. Design4Mobile provides a list of common mobile design patterns

The look and feel can either be consistent with the stock user interface elements that Apple provides; they can be customized, often retaining the "spirit" of Apple's original design; or an entirely new look and feel can be defined—this approach is often used for immersive experiences.

2.7.4.4 Layout

Layout is an important design element, because it is how the user will visually process the page, but the structural and visual components of layout often get merged together, creating confusion and making your design more difficult to produce.

For example, when I show my mobile design layouts as wireframes during the information architecture phase, I intentionally present them on blueprint paper, using handwriting fonts for my annotations (Figure 8-7). It also helps to say that this is not a design, it is a layout, so please give me feedback on the layout.

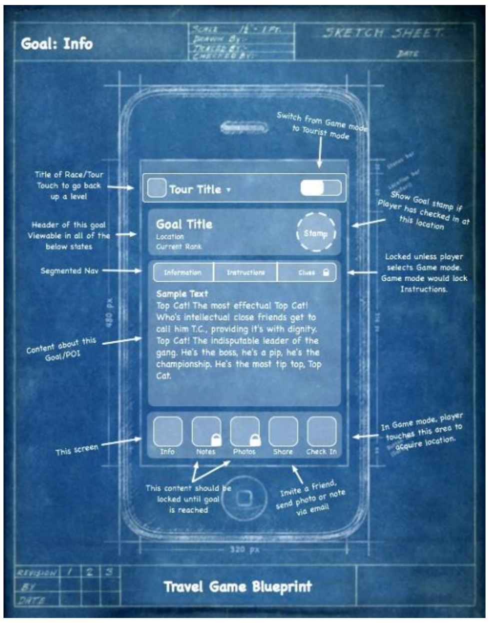

Goal: Info
SKETCH SHEET
Switch from Game mode to Tourist mode
Title of Race/Tour Touch to go back up a level
Tour Title
Header of this goal Viewable in all of the below states
Goal Title
Location
Current Rank
Stamp
Show Goal stamp if Player has checked in at this location
Segmented Nav
Information
Instructions
Clues
Locked unless player selects Game mode. Game mode would lock Instructions.
Sample Text
Top Cat! The most effectual Top Cat! Who's intellectual close friends get to call him T.C., providing it's with dignity. Top Cat! The indisputable leader of the gang. He's the boss, he's a pip, he's the championship. He's the most tip top, Top Cat.
Content about this Goal/POI
480 px
This screen
Info
Notes
Photos
Share
Check In
In Game mode, player touches this area to acquire location.
This content should be locked until goal is reached
Invite a friend, send photo or note via email
320 px
Travel Game Blueprint

Figure 8-7. Using a low-fidelity wireframe to define the layout design element before visual design begins

Different layouts for different devices

The second part of layout design is how to visually represent content. In mobile design, the primary content element you deal with the is navigation. Whether you are designing a site or app, you need to provide users with methods of performing tasks, navigating to other pages, or reading and interacting with content. This can vary, depending on the devices you support.

There are two distinct types of navigation layouts for mobile devices: touch and scroll. With touch, you literally point to where you want to go; therefore, navigation can be anywhere on the screen. But we tend to see most of the primary actions or navigation areas living at the bottom of the screen and secondary actions living at the top of the screen, with the area in between serving as the content area, like what is shown in Figure 8-8.

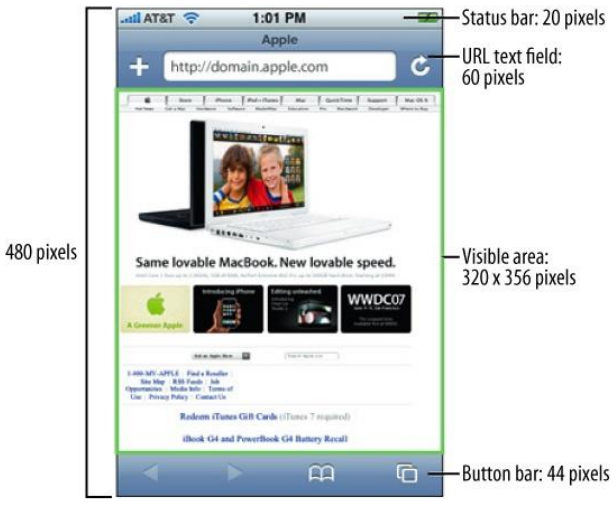


Figure 8-8. iPhone HIG, showing the layout dimensions of Safari on the iPhone

This is the opposite of the scroll navigation type, where the device's D-pad is used to go left, right, up, or down. When designing for this type of device, the primary and often the secondary actions should live at the top of the screen. This is so the user doesn't have to press down dozens of times to get to the important stuff. In Figure 8-9, you can actually see by the bold outline that the first item selected on the screen is the link around the logo.

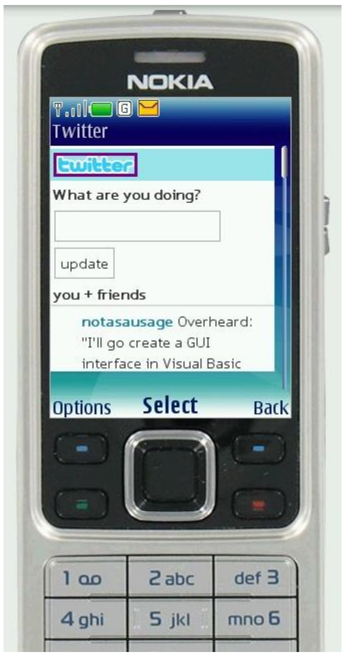


Figure 8-9. Example layout of a scroll-based application, where the user had to press the D-pad past each link to scroll the page

When dealing with scroll navigation, you also have to make the choice of whether to display navigation horizontally or vertically. Visually, horizontally makes a bit more sense, but when you consider that it forces the user to awkwardly move left and right, it can quickly become a bit cumbersome for the user to deal with. There is no right or wrong way to do it, but my advice is just to try and keep it as simple as possible.

Fixed versus fluid

Another layout consideration is how your design will scale as the device orientation changes, for example if the device is rotated from portrait mode to landscape and vice versa. This is typically described as either being fixed (a set number of pixels wide), or fluid (having the ability to scale to the full width of the screen regardless of the device orientation).

Orientation switching has become commonplace in mobile devices, and your design should always provide the user with a means to scale the interface to take full advantage of screen real estate.

2.7.4.5 Color

The fifth design element, color, is hard to talk about in a black-and-white book. Maybe it is fitting, because it wasn’t that long ago that mobile screens were available only in black and white (well, technically, it was black on a green screen). These days, we have nearly the entire spectrum of colors to choose from for mobile designs.

For an example of posterization, the technical term for when the gradation of tone is replaced with regions of fewer tones, see in Figure 8-10 how dramatically the color depth can affect the quality of a photo or gradient, producing banding in several parts in the image.

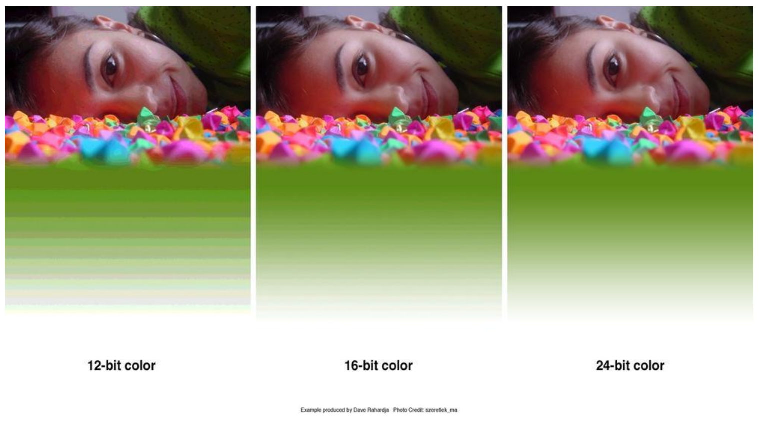

12-bit color
16-bit color
24-bit color
Example produced by Dave Rahardja Photo Credit: szeretlek_ma

Figure 8-10. An example of different levels of posterization that can occur across multiple device color depths

2.7.4.6 Typography

The sixth element of mobile design is typography, which in the past would bring to mind the famous statement by Henry Ford:

Any customer can have a car painted any color that he wants so long as it is black.

Traditionally in mobile design, you had only one typeface that you could use (Figure 8-12), and that was the device font. The only control over the presentation was the size.

The quick brown
fox jumped over
the lazy dog.

Figure 8-12. What most mobile designers think of when it comes to mobile typography

As devices improved, so did their fonts. Higher-resolution screens allowed for a more robust catalog of fonts than just the device font. First, let’s understand how mobile screens work.

2.7.4.7 Graphics

The final design element is graphics, or the images that are used to establish or aid a visual experience. Graphics can be used to supplement the look and feel, or as content displayed inline with the text.

For example, in Figure 8-17, you can see Ribot’s Little Spender application for the iPhone and the S60 platform. The use of graphical icons in the iPhone experience helps to establish a visual language for the user to interact with to quickly categorize entries. On the S60 application, the wallet photo in the upper-right corner helps communicate the message of the application to the user.

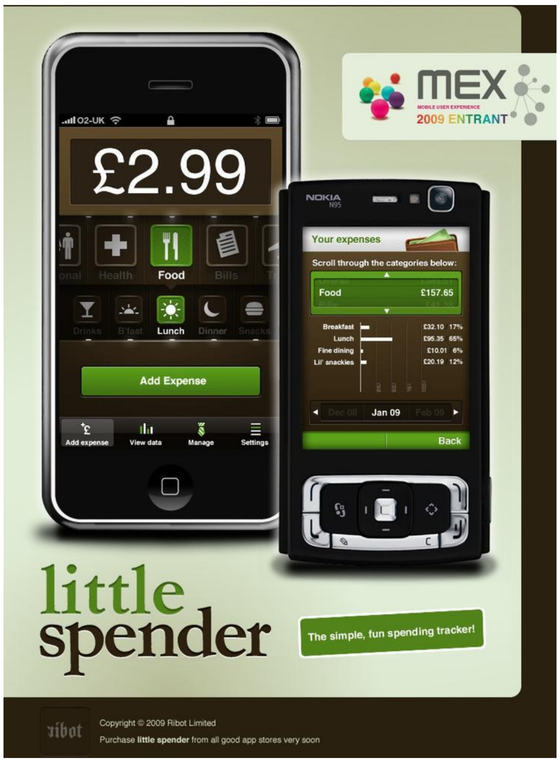


Figure 8-17. Ribot’s Little Spender application uses graphics to define the experience

2.7.5 Mobile Design Tools

Mobile design requires understanding the design elements and specific tools. The closest thing to a common design tool is Adobe Photoshop, though each framework has a different method of implementing the design into the application. Some frameworks provide a complete interface toolkit, allowing designers or developers to simply piece together the interface, while others leave it to the designer to define from scratch.

In Table 8-4, you can see each of the design tools and what interface toolkits are available for it.

Table 8-4. Design tools and interface toolkits

Mobile framework	Design tool	Interface toolkits
Java ME	Photoshop, NetBeans	JavaFX, Capuchin
BREW	Photoshop, Flash	BREW UI Toolkit, uiOne, Flash
Flash Lite	Flash	Flash Lite
iPhone	Photoshop, Interface Builder	iPhone SDK
Android	Photoshop, XML-based themes	Android SDK
Palm webOS	Photoshop, HTML, CSS, and JavaScript	Mojo SDK

Mobile framework	Design tool	Interface toolkits
Mobile web	Photoshop, HTML, CSS, and JavaScript	W3C Mobile Web Best Practices
Mobile widgets	Photoshop, HTML, CSS, and JavaScript	Opera Widget SDK, Nokia Web Runtime
Mobile web apps	Photoshop, HTML, CSS, and JavaScript	iUI, jQTouch, W3C Mobile Web App Best Practices

2.7.6 Designing for the Right Device

With the best possible experience at hand, take a moment to relish it. Remind yourself that you are working with a rapidly evolving medium and though it might not be possible for every user to experience things exactly the way you've intended, you've set the tone and the vision for how the application should look. The truly skilled designer doesn't create just one product—she translates ideas into experiences. The spirit of your design should be able to be adapted to multiple devices.

With a successful single device launch, you can start to adapt designs from the best possible experience to the second best possible experience, then the third, and fourth, and so on. The best possible experience is how it should be, so it serves as a reference point for how we will adapt the experience to suit more devices.

2.7.7 Designing for Different Screen Sizes

Mobile devices come in all shapes and sizes. Choice is great for consumers, but bad for design. It can be incredibly difficult to create that best possible experience for a plethora of different screen sizes. For example, your typical feature phone might only be 140 pixels wide, whereas your higher-end smartphone might be three to four times wider.

Landscape or portrait? Fixed width or fluid? Do you use one column or two? These are common questions that come up when thinking about your design on multiple screen sizes. The bad news is that there is no simple answer. How you design each screen of content depends on the scope of devices you look to support, your content, and what type of experience you are looking to provide. The good news is that the vast majority of mobile device screens share the same vertical or portrait orientation, even though they vary greatly in dimension, as shown in Figure 8-20.

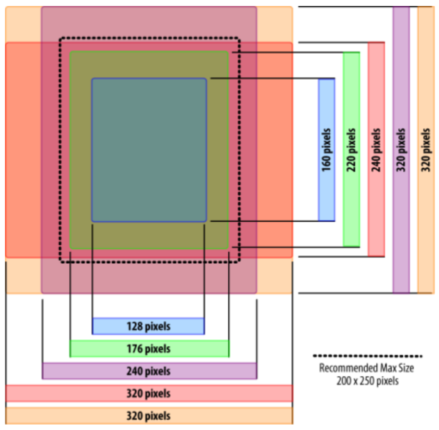


Figure 8-20. Comparing the various screen sizes

For years now, we’ve become used to placing less-crucial information along the sides of web pages. In software, tasks flow from left to right. With vertical designs, the goal is to think of your design as a cascade of content from top to bottom (Figure 8-21), similar to a newspaper. The most contextual information lives at the top, and the content consumes the majority of the screen. Any exit points live at the bottom. Mobile is no different.

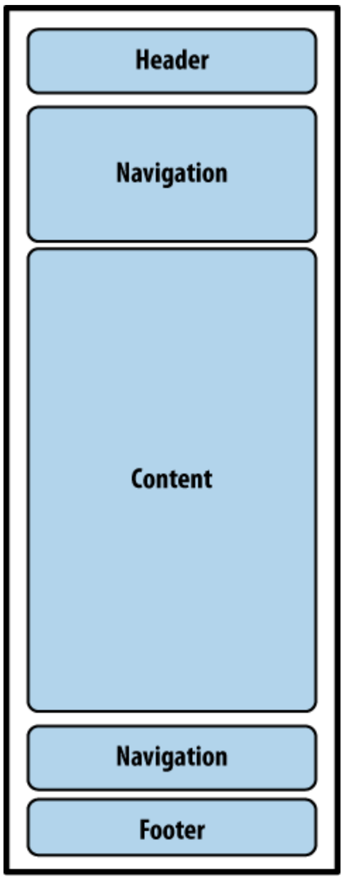


Figure 8-21. The typical flow of information on mobile devices

The greatest challenge to creating a design that works well on multiple screen sizes is filling the width. For content-heavy sites and applications, the width of mobile devices is almost the perfect readability, presenting not too many words per line of text. The problem is when you have to present a number of tasks or actions. The easiest and most compatible way is to present a stacked list of links or buttons, basically one action per line. It isn't the most effective use of space, but presenting too many actions on the horizontal axis quickly clutters the design—not to mention that it is more difficult to adapt to other devices.

As devices get larger, denser screens, you will see an increase in the use of touch, forcing the size of content to increase to fingertip size—typically 40 pixels wide and 40 pixels tall (Figure 8-22). This actually solves part of the horizontal axis problem, simply by making content larger for larger screens. Ironically, you can fit almost the same amount of usable content in an iPhone as you can a lower-end device.

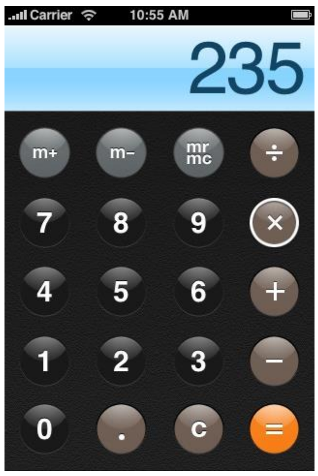


Figure 8-22. The iPhone calculator application uses common fingertip-size controls

Obviously, you can fit a lot more on screen with more advanced devices, but you want to avoid forcing the user to zoom in and out of your interfaces.

2.8 Mobile Web Apps vs Native Apps

When should you make a mobile web application versus a native mobile application, or a downloadable application designed for a specific platform, like a Java application or an iPhone? It wasn't that long ago that people didn't even bother asking the question; every mobile application was native.

The mobile web historically has been so horrendous to deal with that the only way to create a compelling experience was to go native. Assuming that native applications will immediately create compelling experience is misleading, though, as dealing with native applications across multiple platforms isn't exactly a walk in the park, either. Device fragmentation exists across devices, because if you are dealing with native APIs or mobile web browsers, it is an obstacle that cannot be avoided.

But like most things in the mobile ecosystem, the question of which path to choose comes down to money. Native applications can produce a lot of short-term revenue for developers and operators alike, especially with the proliferation of app stores. Though traffic to mobile websites and web applications produce a sizable chunk of operator revenue in data charges, developers assume they get nothing. This isn't to say that making money from the mobile web can't be done; it's just that few know how to choose the right path to monetization.

2.8.1 The Ubiquity Principle

Let me start by clearly stating where I stand on this issue: I believe that the mobile web is the only long-term commercially viable content platform for mobile devices. I have four key reasons to support this belief: fragmentation, the Web, control, and consumer expectations.

Fragmentation

First of all, we already know that mobile is a much larger playing field than desktop computing, but there is currently no economically feasible means to create native applications that can support the majority of the market. There isn't just a handful of platforms to contend with or a clear market leader, but literally hundreds (when you include all the variations), with no one vendor able to firmly claim itself king. Getting your application on one platform is a snap, but getting it on two is a challenge, five a costly headache, and supporting fifty virtually impossible.

The Web

The overall technology market is going to the Web. It is a highly vetted consumer medium that offers many pros and few cons. It is the only medium for information, applications, and services that has gone the distance for the last 15 years. With a new focus on advanced desktop web browser technology, which is poised to become more than just rendering text, the only native application that matters is the browser. The majority of digital innovation is occurring using the technologies of the Web, making the browser the central delivery mechanism for the applications and services of tomorrow. As more mobile browsers add services to detect location, acceleration, or use of the hardware, I predict that the need for native applications will be reduced to specific uses that really need the full capacity of the device, like games.

Control

Mobile application distribution cannot and will likely never be under the control of the developer. In other words, mobile application vendors always have to rely on middlemen to get their products to market and take a slice of their profits. This has been the case since the beginning of downloadable mobile applications, when they were under the tight control of the operator.

These days, we see that control shifting to the device and platform makers, making getting applications to market easier. But do not be fooled—the model is exactly the same. The purpose of your product is only to service them, boosting their bottom line. The lack of control or even influence of your primary distribution channel puts your product at risk almost from the start. Your product can be shelved after just a few days in the spotlight.

Without control, this reason alone means that the funding of creating mobile applications will always remain a small, high-risk investment. Without the funding to help weather market fluctuations or the willingness to invest in the truly innovative products, developers will be forced to continue knocking out small, pointless applications aimed at short-term revenue gain—novelty products that wear off quickly.

Consumer Expectations

My fourth belief is that consumers expect things to just work, and rightfully so. The challenge with native mobile applications is that the consumer may see an application that might look appealing to him, but if it isn't supported for his particular device, then not only is the sale lost, but often the customer is lost for good. This is one of the reasons that operators usually require applications sold on their marketplace to support their top 10 to 15 devices.

From the consumer's perspective, he spends good money on a device and wants content to support it. The lack of available content lowers the perceived value of the device. Consumers don't care what device or platform they have; they just want to participate in the same content and services that their friends are using. Because cross-platform support is so challenging, that is hardly ever the possible.

You can see these traffic trends in just about all mobile marketplaces. Numerous visits occur when a consumer has purchased a brand-new device, but then visits drop off precipitately in just days. After the device is just a month old, the likelihood that the user will return for more content is slim to none.

Ubiquity in the Mobile Web

The mobile web is the only platform that is available and works across all mobile devices, sharing the same set of standards and protocols with each other as well as the desktop web. The mobile web is also the only mobile distribution channel available to developers that they can control. It is the best way to bridge short, context-based mobile interactions with longer, desktop-based tasks.

The mobile web is the easiest platform to learn, the cheapest to produce, the most standardized, the most available, and the easiest to distribute. I call this the Ubiquity Principle: easier-to-produce quality content and services for the largest available market will always win.

The key word is “quality,” which the mobile web hasn’t had a lot of over the years. Although the mobile web has its own challenges with device fragmentation, the level of complexity to adapt to these challenges to produce the best possible experience is far lower than with native applications. In addition, these challenges are going away quickly and will be inconsequential in just a few years’ time.

2.8.2 When to Make a Native Application

I’m obviously a big supporter of the mobile web; however, I am the first one to admit that making a native application can be the best thing for a product. This is usually true when you need to take advantage of the features of a device that a mobile web browser does not allow.

The following sections discuss some of the key features you may be considering that almost immediately point toward creating a native application.

Charging for It

Nowhere is it written that you can’t charge people to use a mobile web app, but for some reason, people think you can’t or shouldn’t. Historically, charging for things on mobile devices has come down to two obstacles:

- Entering a credit card number in the mobile device is cumbersome and can be insecure on older devices. Typically, if you want to charge for it, you set up an arrangement with operators to do Billing on Behalf of (BoBo) so purchases just show up on the subscriber’s bill. This means you need to have BoBo arrangements with every carrier your application is on, which is its own headache. This is usually the preferred method, because a lot of people with phones don’t have credit cards, such as young people.

  Another route is storing subscribers’ credit card information on a secure website. The user is then able to log into her mobile profile and make purchases against it. This two-step process is less than ideal and basically means that people can’t just go and buy from their device.

- Operators want their cut. A native mobile application directory, either through the operator or through a device maker, always includes a means to collect payment for it. They typically just take their cut and pass on the rest of it to you, but it means that you must work within the rules and regulations of their marketplace. Getting

> onto the operator marketplace has historically been a colossal effort, requiring full-time staff just to negotiate terms. Since those days, device maker markets have made it much easier to get products to market, but are plagued by most, if not all, of the same problems and red tape.
>
> Applications and services that encroach on the operator or device maker's business model will likely be blocked or removed from sale. Mobile websites that earn too much money without the carrier getting a percentage have been shut down in the past, but you don't hear many of those stories these days.

The bottom line is that if you want to charge for a native application, you are going to have to work within the rules of the marketplace and give up a decent percentage of your profits.

Creating a Game

If your goal is to create a mobile game—one of the biggest mobile content markets—then you need to create a native application. Games are resource-intensive and almost always require the use of a device or platform API. Although there have been quite a few compelling games created using just web technologies, the competition in the native application game market is just too stiff. When users launch a game, they have some expectations of what it is going to look and act like. The mobile web is close to providing an analog experience, but is not quite there yet.

When making mobile games, you need to carefully consider which platforms to support. Luckily, there are a variety of game porting houses that can help get your game onto multiple platforms, but as you might imagine, it can be a costly endeavor in terms of money and time.

Using Specific Locations

The next feature is location, or being able to detect the users' locations by GPS or cell tower triangulation for the purpose of presenting users with information based on their current location. Traditionally, the only way to access users' locations is through native application APIs, but the W3C Geolocation API is quickly being incorporated into the most popular mobile browsers. Devices that run WebKit, like the iPhone or Android devices, as well as devices that run the Opera or Mozilla browsers, will all have the ability to detect user's locations.

I believe that location is the holy grail of the Web. If it can be made available to the web browser, then web publishers can begin to use location and context in new and interesting ways. Although it is not a technical hurdle, the issue has mostly been with privacy. We like to think of the web browser as an anonymous portal into the World Wide Web. Adding location means sharing sensitive information with websites, which could actually be quite dangerous. But for all location-aware applications, providing your location requires your direct authorization from the device for each instance, and you have the option to disable it entirely.

Using Cameras

The camera is another device function that can come in handy in your applications. Traditionally, mobile MMS (Multimedia Messaging Service) is used to handle mobile photo interactions. In other words, you take a photo, send it via MMS to a shortcode, then a server somewhere does something with the photo and sends an alert to you when it is done. This process is complicated, time-consuming, and fairly unreliable.

With access to the camera, native application developers can simplify the task to just taking a photo from within an application. The user then can do limited processing on the client side, sending to a server only if necessary, through a much more reliable HTTP connection. The W3C is working on an API to allow for camera access, but it has yet to be incorporated by the browser vendors.

The camera is useful in several types of mobile applications, from sharing snapshots or videos of friends to capturing important events as they happen and recording visual information, such as an important advertisement or sign found while out and about. The camera can even be used to process bar codes and then to redeem promotions. Not long from now, we could see cameras that are able to translate different printed languages just by holding our mobile camera up to a sign—a technology that is already popular in Japan.

Using Accelerometers

A popular feature in more recent mobile devices is the addition of an accelerometer, a small instrument that measures physical acceleration and gravity and sends data back to the device. The most common use of an accelerometer is to detect when the device is physically rotated, adjusting the display from vertical to horizontal orientation (or vice versa).

Using the accelerometer can be a benefit to mobile users, enabling them to interact with a device in a more natural way; being that the device is likely held in the hand, it can adjust content to suit its physical orientation, like rotating the screen, or detecting physical movement, and can therefore have limited prediction of the users' context. A simple example might be that if the user is walking, detected by slight movement or velocity, the user interface might display larger text than normal to make the experience easier for users on the move.

However, developers should be cautioned against relying too heavily on the accelerometer for frivolous interactions. Every mobile interaction should pass the "transit test." You should always assume that the user will interact with a mobile device the same way she would if she were sitting on a crowded bus or train. Ask yourself what the likelihood is that the user will shake the phone while standing in a packed subway or riding in a car. Typically, the likely answer is slim to none. So be sure to always include an alternative way to perform the same task that attracts less attention.

Accessing the Filesystems

Another reason you would want to create a native application is if you want to use the data stored on the device itself. This might be the user's address book, photos, an email message, or even data from another application.

Such filesystem access is obviously a big security and privacy issue. An errant application potentially could alter or delete your data, or worse. An infected application could use your contacts and constant connection to spread a virus to multiple phones, something that occurred quite often before widespread adoption of mobile application certification.

On the other hand, mobile devices are becoming highly personal, mobile computers that store an increasing amount of content and information about their owners and the owners' friends and business contacts. The idea of leveraging this information across applications is appealing. Though not without risk, using stored data is a powerful way to present contextually relevant information to the user.

Developers should be cautioned to use stored information only in limited doses. We've seen a trend with applications that leverage too much user data without the user's express permission as being mistakenly perceived as spamming or phishing information, even though the application was actually attempting to perform a valuable service. False perceptions of your application can and will significantly affect your ability to distribute it, either in loss of direct sales or possibly by being pulled altogether, if the operator receives enough complaints.

The rule of thumb here is to never do anything with the users' data without their express permission to do so: a precaution far too many mobile applications skip.

Again, we are seeing some progress by the W3C to establish a standard API for mobile vendors to follow, but we aren't there yet.

Offline Users

The final reason to make a native application is because you know the user is likely to be offline or out of range of a mobile network. For those of us who live in the city, that may seem like a rare occasion. Even for those who live in more rural areas, network dead spots are becoming increasingly rare. But going periods without a connection does of course happen frequently and your application should be designed to take this into account.

Think about the context of the user and when he is likely to use your application. The likelihood that a mobile game will be played on an airplane would probably be pretty high. A trail map application will likely be used in more remote areas, with less coverage. A mobile travel guide will probably be used in a foreign network, where roaming and international fees could occur. Each of these applications should have an offline mode in which the user can still perform the most common tasks of the application without the need of a wireless connection.

The mobile browsers that support HTML5 actually include the capability to create offline apps today, but it isn't made apparent to the user. As support for offline storage increases across multiple mobile browsers, we need to define metaphors and conventions for the users to know when their mobile web application will work when the device is offline.

I find that many native applications assume reliable network connectivity far too often. Applications are often designed under the best of circumstances, in a closed environment with a healthy wireless signal and a fast network. Mobile devices by nature move from the best of circumstances to the worst quickly. Native applications should always be tested under the worst possible conditions. For example, what happens if the user starts a task with a full signal, but tries to complete it with no signal?

Users don't think about being online or offline when they load native applications—they just expect them to work. It's our job to make sure that they do.

2.8.3 When to Make a Mobile Web Application

I believe that unless your application meets one of these native application criteria, you should not create a native application, but should instead focus on building a mobile web application. Like I said before, I'm a big fan of native applications and I feel that there are a lot of great innovative and market opportunities here, but mobile web apps are the only long-term viable platform for mobile content, services, and applications.

Native applications don’t service the user better in any significant way; they only add cost to your project, decrease your distribution channels, plus cause you to lose the ability to incrementally improve your application, lose control and profit, and add to the device fragmentation problem. Plenty of short-term opportunity exists with native apps, but not without great personal risk, not to mention damaging the long-term viability of the mobile content market.

The most interesting case for mobile web apps is actually composed of the reasons stated earlier. If those are the only reasons that you need to create a native app, then what happens if you take away those obstacles from the mobile browser? Something like that is being done by Palm’s webOS. They have created an entire mobile operating system built on WebKit, turning the phone into a web browser. Your “native applications” are just web applications.

Another innovation is the PhoneGap project, an open source effort that allows you to create native applications for iPhone, Android, and BlackBerry devices, exposing many device features like location and filesystem access to your web app. These applications can be distributed and sold in device marketplaces, but they share the same code and design. And because it is a web app, we can make a less capable version of our app available for free to lower-end mobile browsers. Build once, and deploy everywhere.

For those who have spent some time in mobile development in the past, you might have noticed the omission of “If you want to create a rich experience” from the reasons why to make a native app. Although there are certainly plenty of devices out there where this still might be the case, you can now create incredibly rich interfaces with mobile web apps. Not only can they be just as compelling as native apps, but they can work across multiple device frameworks with no alterations in code.

The rate of innovation for creating mobile web apps across every mobile device maker is at its highest level in years, but more important than that, for the first time device makers are all working toward achieving the exact same standards, which just happen to be the same standards as the desktop web. In addition, the devices that either lead in mobile web app innovation or support third-party browsers that do, are becoming the top-selling devices in multiple markets.

So, instead of asking yourself, “When to make a mobile web app?” I challenge you to start asking yourself “When *not* to make a mobile web app?”

2.9 Adapting to Devices

Not all mobile devices are created equal. Thus the age-old problem in mobile design and development: devices can be vastly different from each other. It would be easy if different devices simply supported different attributes—one supports CSS3 and one doesn’t. But it isn’t that easy. One device might support CSS3 and another device might support CSS3 poorly—or worse, incorrectly.

Honestly, this might not be a problem at all if we only had a few platforms or browsers to contend with. For example, how many big desktop platforms are there? Three: Windows, Macintosh, and Linux, with the first two making up almost 95 percent of the market. And how many big desktop web browsers are there? Four: Internet Explorer, Firefox, Safari, and Opera, with the first two making up nearly 90 percent of the market, the most recent versions of which all pass the Acid2 test for CSS2 support, effectively making them web-standards-compliant.

The mobile industry is an entirely different story altogether. In mobile, you have a half a dozen or so platforms, like the S60, iPhone, Android, BlackBerry, Windows Mobile, and LiMo smartphone platforms, plus a dozen or so feature phone platforms. Add the plethora of mobile web browsers that run on each of these platforms, for which less than 1 percent of all mobile browsers are able to pass the Acid2 test for CSS2 support, for example, and you can start to see that the mobile web is a very fragmented and difficult space to support.

Case in point: one would assume that if I enter *www.domain.com* into a web browser, I would be taken to the appropriate experience for my device and viewing context, right? This simple action is what we’ve promised to users regarding how the Web works. But of course in mobile, it isn’t that simple. Sometimes entering *www.domain.com* detects your device and sends you to the desired mobile experience. Sometimes we have to create *m.domain.com*, and the recent trend is to also create *iphone.domain.com*.

Where does it end? At what point do we just tell users that when they want to go to a site, they should enter a web address, regardless of their device? This is the topic of this chapter: adapting for multiple devices.

Detecting, adapting, and supporting multiple devices has historically been the worst pain point of mobile design and development. It usually requires a software system to perform a three-step process in order to render the best experience per device, creating plenty of opportunities for problems. Though there are several strategies to solve this problem, each of them work in this basic way:

Detect

The system must first detect the device requesting the content, known as device *detection*. This typically requires a valid and up-to-date device database, with all the pertinent information on the hundreds of devices on the market, or in some cases a very smart mobile web browser.

Adapt

It next takes the requested content and formats it properly to the constraints of the device, which is called *content adaptation*. This might include the screen size, device features, appropriately sized photos, supported media types, or web standards support. This is done by having an abstracted layer of content and a layer of presentation and media for each supported device.

Deliver

Finally, it must serve the content to the requesting device successfully, usually requiring testing each device or class of device that it is intended to support.

This entire process is often referred to as *multiserving*, *device adaptation*, or simply *adaptation*. As you might imagine, this process requires the consideration and adaptation of many variables. It is in no way an obstacle that cannot be surpassed, but it does add significant cost and complexity to a mobile project.

Luca Passani, who maintains the largest open source device database (a necessity in this process), puts it this way:

Multiserving is a necessity, but creating (and maintaining) more than three versions means going down a very slippery slope.

So what are we, as designers and developers, to do? We know that mobile technology is important—it is the platform for the future, and we want our content in the space—but dealing with all of these devices seems like a headache that we don’t exactly want or need.

I’ve said before that whenever there are multiple options in mobile, there are no right or wrong choices—just what is right for the user. Let me stress that point here again as well. Not only are there no right or wrong answers in how you adapt for different devices, but there is strong support and intense opposition for a number of different approaches. Members of both the web and mobile communities have an almost religious belief that one approach is better than another, either through dogmatic principle or through years of experience.

Even this book isn’t immune to this debate. Each of the technical reviewers invited to review this chapter offered different suggestions and techniques for how to solve the problems of adapting for devices. Some were based on how the adaptation should be, and some were based on how to get it done in today’s environment. If I, as someone with more than a decade of experience in the mobile ecosystem, struggle to make sense of this topic, then I can only imagine how you must feel. However, I can provide my own perspective on the topic, which is a bit unique.

I come from a family of entrepreneurs. My father was an inventor and business owner. Each of my six brothers (of which I am the youngest) either own or have owned their own businesses. You could easily say that entrepreneurship runs in my family. I’m not immune; I own my business now, but I certainly didn’t start there. Like most people, I’ve worked for a number of different companies, and many of them just happened to be mobile companies.

In fact, I’ve had the pleasure of working with some really great companies and brands over the course of my career in the mobile industry. This topic, adapting for multiple devices, has been the coffin nail for the majority of them. Not only did the costs run them out of the mobile business altogether and force them to sell off assets way below their value, but the companies also started to lose faith in the long-term viability of the medium.

As an entrepreneur, I cannot in good faith recommend to anyone something that I wouldn’t be willing to make myself. To me, this is the only principle that matters. It isn’t a matter of industry dogma: it is a matter of business ethics.

In my experience, adapting for multiple devices is the greatest risk that your project can make. Too many challenges are unforeseen, there is too much knowledge that takes too much time to learn, and the hidden costs can too easily multiply beneath you. As you might imagine, I get a bit conflicted, as I also believe that mobile will be the most powerful and defining medium of the next 25 years.

So how does one balance the necessity of multiserving while still keeping mobile investments of time and money under control? I've found five basic strategies to work around this seemingly insurmountable obstacle:

1. You can do nothing and wait for the mobile industry to become fully homogenized.
2. You can try to use a progressive enhancement technique to provide fallback experiences to a number of devices.
3. You can target only a handful of devices that support the standards you wish to support, knowing full well that this means making your mobile experience less accessible to your intended market.
4. You can adapt the experience based on the class of the requesting device, making many assumptions about its capabilities.
5. Provide an experience specific to each requesting device.

I will discuss all five approaches to adapting experiences for multiple devices. I will focus primarily on the mobile web, but there is no reason why most of this knowledge couldn't apply to all mobile experiences as well.

I will try to leave out debate as much as possible and just report what I believe that you need to know. There are of course many elements of this topic that could be expanded upon; in fact, an entire book could easily be written on just this one topic. But for the purposes of this book, I will introduce to you the general principles of adaptation and will leave it up to you to decide what is best for you, and more importantly, what is best for your users.

2.9.1 Why Is Adaptation a "Necessity"?

Is multiserving a "necessity" like Luca suggests, and if so, why? Yes, multiserving your content in one way or another is absolutely a necessity. If you have a website and you want to have a mobile website, web app, or even native application that shares content in some way with another context, you are going to need to figure out a strategy to make that happen.

In Figure 13-1, you can see a simple example of the software systems that someone might need if she wanted to have a desktop site, a mobile website, a mobile web app, and a native app. Each context is creating an standalone system, because more often than not, tying them all together would be costly in time and resources.

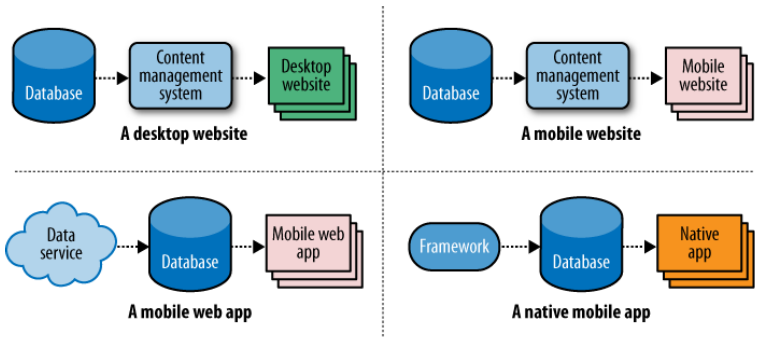


Figure 13-1. Serving multiple contexts can be redundant and expensive

I've found this to be the most common multiserving strategy with most companies I've worked with: creating an internal API or data service to get each system talking to each other, although more often than not the API is actually designed only to pull information from one source into another, and doesn't always send information as well, or at least not as easily. Getting information out of a central database through an API can be a snap, allowing users to create new sessions on the client, but storing data either locally or back into the context-specific database can be a hassle. The problems occur when trying to get each of these separate systems in sync.

Ideally, we could use one system for multiserving all of our content, as shown in Figure 13-2, with one central source of data outputting content to multiple contexts, using whatever external tools are needed.

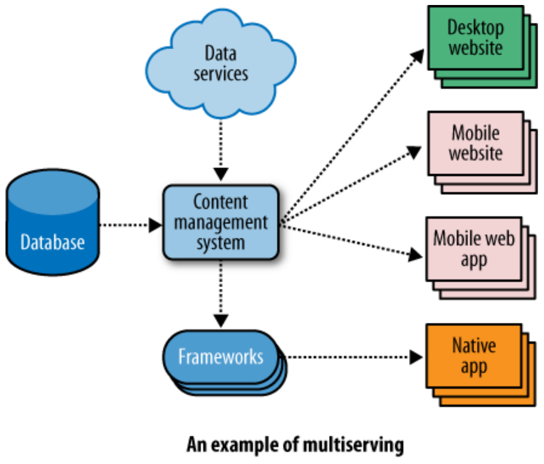


Figure 13-2. A simple example of a multiserving system

The question isn't whether it is a necessity; in today's environment, and with the growth of mobile devices starting to move beyond just phones, this isn't a problem that is going away anytime soon. The question is how to adapt for multiple devices with a minimal number of headaches. This brings us to our first philosophical debate: the adaptation/multiserving conundrum, and our first adaptation/multiserving strategy.

2.9.2 Strategy #1: Do Nothing

Our first of four multiserving strategies is to simply do nothing, or rather to wait for the technology to adapt to our principles—which actually isn't as foolish as it might sound.

In 2005, the W3C created the Mobile Web Initiative (MWI) to attempt to bring the standardization it achieved with the desktop web to mobile. This was a difficult challenge, because the W3C had not issued any specifications for mobile standards. The announcement of the effort included an introduction from Tim Berners-Lee about the importance of "One Web," a concept that has been a hotly debated topic in both the web and mobile communities ever since.

Initially, the W3C defined One Web this way:

One Web means making the same information and services available to users irrespective of the device they are using.

In other words, content should be published once and the device should be smart enough to know what to render; effectively, we wouldn't need to do anything more than ensure that our content meets the content standards. This actually makes a lot of sense, as we'd be able to write once and publish everywhere to multiple devices and multiple contexts.

This do-nothing approach is actually a multiserving system in its own right: it detects devices, though without the need of a device database; it adapts content based on the requested device through thoroughly separated content and presentation and renders it to the requesting device. The multiserving system in this scenario is the browser, assuming that it can intelligently render the appropriate experience that we define for it.

Five Assumptions About One Web

I'll be honest: this approach can work really well when you design your content with it in mind, but it isn't very flexible or robust, and it isn't without flaws:

- It assumes that your content for multiple contexts will be the same, when it usually isn't.
- It assumes that cost per kilobyte to the user is minimal or nonexistent. In other words, it assumes that the user is willing to pay for content not designed for her context.
- It assumes that a persistent and high-speed data network will always be available.

- It assumes that mobile browsers are smart and will support the same standards consistently, which isn't the case, at least today.
- It assumes that a technology-based principle should come before the needs of the user.

These five assumptions will go away in time. In fact, the iPhone and other modern mobile browsers have been designed specifically to address the first four assumptions, which is why Apple has described the iPhone as a "One Web" device.

The problem with the One Web principle is that fifth assumption. It can be dangerous to put the desire for efficient technology before the needs of users. Creating a great experience means finding the sweet spot between the constraints of the technology and the needs of the user.

For example, take the first assumption. Assuming that the needs of a user on a mobile device are the same or even similar to a user on a desktop computer would be foolhardy. Even the needs of a user with a limited phone device are different from those of a user with a smartphone. After you add additional devices, such as GPS units, gaming consoles, or even subnotebooks, you start to see that the features you might include for one device or one context might not apply to another.

Trying to state a principle saying, "It should be done this way" adds risk that people will create poorly adapted experiences. Trying to shoehorn the user into an experience designed for the wrong device for the sake of simple technology or even a high-minded principle is simply irresponsible. The user deserves more and it is our job to provide it.

The One Web Aftermath

After the One Web principle was announced, the mobile community, which had been working on this problem for years, exploded in vehement disagreement over the principle. Many people, including myself, called it unrealistic and said that it ignored the needs of mobile users. Meanwhile, the web community applauded the idea, wanting to see the mobile landscape become more aligned with the Web, and demanding smarter mobile web browsers, thus starting a mild feud that continues to this day.

After working closely with the mobile community, the W3C and the MWI revised the definition, and this is how it still reads today:

One Web means making, as far as is reasonable, the same information and services available to users irrespective of the device they are using. However, it does not mean that exactly the same information is available in exactly the same representation across all devices. The context of mobile use, device capability variations, bandwidth issues, and mobile network capabilities all affect the representation. Furthermore, some services and information are more suitable for and targeted at particular user contexts.

In other words, content should be published once and the device should be smart enough to know what to render, unless it can’t or doesn’t make sense to.

Using This Strategy with Media Queries

We’ve clearly seen in terms of usage with more advanced mobile browsers, such as WebKit or Opera, that when rendering normal web content at desktop quality users, when given the choice, prefer the mobile optimized version designed for the mobile content instead of the often more full-featured desktop version. This actually means that the One Web principle does work if the first four of the assumptions about One Web are true.

This do-nothing One Web approach to multiserving means that you don’t need to do anything more than add a line of code that points to a CSS file that defines the presentation for each supported device; called a Media Query, it is part of the CSS3 specification. It requires a higher-end mobile web browser that supports CSS3, like the iPhone, Android, Palm’s webOS, or Opera Mobile.

For example, if you wanted to load only a stylesheet for a device that is 320 pixels wide, you could define it this way:

```
<link media="screen and (device-width: 320px)" rel="stylesheet"

href="320styles.css" type="text/css" />
```

However, because many devices can alter their orientation to 480 pixels wide, you would use the following:

```
<link media="only screen and (max-device-width: 480px)" rel="stylesheet"

href="webapp.css" type="text/css" />
```

You can also layer in additional stylesheets for other devices that might fall outside of the traditional mobile device (in this example, devices with screens larger than 481 pixels):

```
<link media="only screen and (min-device-width: 481px)" rel="stylesheet"

href="desktop.css" type="text/css" />
```

You can use fairly complex media queries to fine-tune the presentation for a number of devices, as long as the devices support it.

2.9.3 Strategy #2: Progressive Enhancement

Though progressive enhancement is normally thought of as a development strategy, it can also be used as an adaptation strategy as well, going through each of the three steps of multiserving. Similar to "doing nothing," this approach assumes that the browser will be smart enough to detect, adapt, and deliver the right experience, but in this strategy we design a site to have several fallback points, supporting a larger number of devices and not requiring support of media queries.

For example, take Figure 13-3, where we can see our presentations layered on one another. At the bottommost level, we have a rich CSS3-based experience for our Class A browsers, including support for media queries. Above it, we have good CSS2 support, for Class B browsers, and so on, until we get to the top level, or our last fallback position of no styling whatsoever, for our Class F browsers. In this case, as long as our markup is semantically coded, it should still be usable on any device that can render HTML.

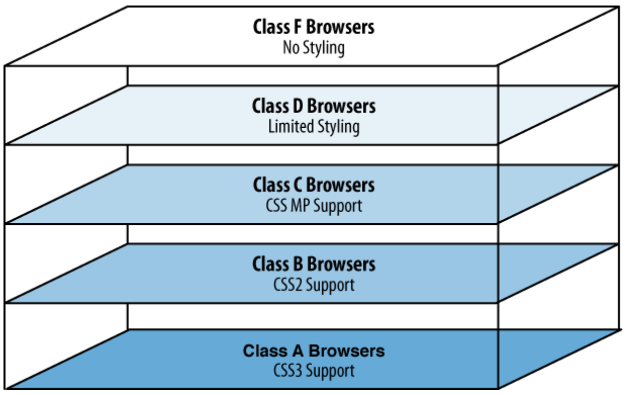


Figure 13-3. Using a progressive enhancement technique to establish several fallbacks to our presentation

The Handheld Media Type

In order to achieve this technique, we are going to rely on the browsers, but luckily there are couple of attributes in XHTML that will help us out. First, let's talk about the media type attributes, which are basically the parents of media queries. XHTML includes a number of media type attributes that you can add to your document to load different stylesheets for different contexts. For example, when we reference our stylesheet, we can define it as a screen, print, projection, or (the unfortunately named) handheld media type. For example:

```
<link media="screen" rel="stylesheet" href="desktop.css" type="text/css" />

<link media="handheld" rel="stylesheet" href="mobile.css" type="text/css" />
```

Or within a stylesheet or the `style` element, we could define it this way:

```
@media handheld {

  * { font-family: sans-serif }

}
```

All desktop browsers are smart enough to know that the screen media type is for them. It’s a little-known trick that most mobile browsers have supported the handheld media type for years. This means that we can use the same content and display it on multiple mediums with a single line of code.

However, there are a few catches. First, because the handheld stylesheet will be used on most mobile browsers, the stylesheet should be designed for the lowest common denominator, using basic style techniques and providing limited design opportunities. Mobile browsers that aren’t smart enough to support the handheld media type, usually falling into the Class D grouping, could be problematic to support with this approach. The second caveat is that more advanced mobile browsers will load the screen media type instead of the handheld media type because they can render desktop websites reasonably well.

Layering Multiple Stylesheets for Multiple Devices

We can use the media queries technique from our first strategy to target our Class A browsers:

```
<link media="only screen and (max-device-width: 480px)" rel="stylesheet"

href="class-a.css" type="text/css" />
```

This stylesheet would contain our more advanced styling, like rounded corners, heavy use of image replacement, and other advanced styling techniques, and would only be loaded for browsers that support CSS3.

Class B browsers typically, but not always, prefer to support the screen media type instead of the handheld media type. Therefore, we can add another stylesheet for these devices, with fairly basic CSS2 styles, some image replacement, and some placement, but nothing too aggressive:

```
<link media="screen" rel="stylesheet" href="class-b.css" type="text/css" />
```

For Class C and Class D browsers that support the handheld media type, we can then layer in another stylesheet. This time we use a very basic stylesheet. The simpler it is, the better it will render on devices:

```
<link media="handheld" rel="stylesheet" href="class-c.css" type="text/css" />
```

When we add all three of these together, we see something like this:

```
<link media="handheld" rel="stylesheet" href="class-c.css" type="text/css" />

<link media="screen" rel="stylesheet" href="class-b.css" type="text/css" />

<link media="only screen and (max-device-width: 480px)" rel="stylesheet"

href="class-a.css" type="text/css" />
```

The browsers will do their best to load and render the best stylesheet for the browser's capabilities. Now this is hardly a foolproof approach. It involves putting a lot of the burden on the mobile web browsers, which more often than not prove disappointing. However, this approach does work fairly well to reach most of the top devices accessing the mobile web today.

This adaptation strategy isn't the most popular approach with many in the mobile community, as it doesn't give you a lot of opportunities for fine-tuning your styles to the device. Imagine you have a popular browser that is actually quite capable but keeps loading the handheld stylesheet instead of your screen stylesheet. There isn't a lot that you can do to account for these oddities. This strategy can also present a user experience concern, in that you aren't providing the user with the ability to choose which experience he would prefer to load, as recommended by many mobile advocates.

I find this approach works extremely well for simple mobile web experiences, though in the interest of full disclosure, some of my technical reviewers discourage this approach, saying it is too unreliable. I think it all comes down to an issue of what browsers you plan to support. If you want to support the top browsers but still offer something for people with less-capable devices, this strategy might be the one for you. You certainly can't support everyone with it, but it is a quick and easy first step.

2.9.4 Strategy #3: Device Targeting

The third multiserving strategy is *device targeting*, or calling out specific devices by class or model and delivering an experience designed with that device in mind. In this strategy, we don't assume that browsers are trustworthy enough to get to where they need to be. Therefore, in this strategy, the first step of multiserving is to reliably detect the device.

Once the device and browser are detected, you route the user to the best experience. This can be done by checking the HTTP headers, starting with the `User-Agent` string, in order to recognize the device and browser and then deliver a device-specific site.

One of the more common examples here would be wanting to treat a device like the iPhone differently than the rest of your mobile devices; the markup and styles used to create the iPhone experience could be quite different than what you might use for all the other mobile devices. In these cases, each experience is often built as a standalone product, with little to no adaptation taking place.

Though you could certainly perform dynamic adaptation—something I will discuss more in the next strategy—in this strategy, it is common for developers to utilize a more simplistic approach, relying on the content management system or web application framework to create multiple experiences from a single data source. Create specific experiences for targeted devices, then route devices accordingly.

The Device Detection Dilemma

Device detection has been a technical challenge for many years. Although it's hardly a challenge to those experienced in mobile development, I feel that the mobile community severely underappreciates how much of a challenge this is for the rest of us.

Large publishers with experienced developers can certainly hack their web server configurations to detect and route devices, even though it still requires having an up-to-date device database with all the device profiles and appropriate techniques to recognize a device. And in larger organizations it can take months to prepare, test, and QA any major changes to the server directives. For smaller companies using shared hosting, altering the server configuration at the levels needed isn't even an option.

In my experience, only a handful of companies that I worked with ever employed a server-side device detection strategy. The vast majority of them were mobile companies whose business was to provide excellent mobile experiences to lots of devices.

This is obviously less than ideal. For the mobile community and the mobile web to flourish, these tools need to be available to all for a price everyone can afford. For many years, I’ve been arguing that device detection is a problem that we need to solve. The need to provide some level of adaptation will always exist in one way or another, and there will always be many devices and many contexts to which we will want to adapt. However, as mobile device and browser makers adhere more closely to the defined and accepted standards, the variation required will become almost inconsequential.

Device fragmentation will go away one day, but device detection will not. We will always require a method of easily routing experiences for the user. Ironically, this is actually the easiest problem to solve. It just takes people with the experience and commitment to make it happen.

Andy Moore’s Mobile Browser Detection

One of the more popular solutions to provide simple detection is the PHP script written by Andy Moore (http://www.detectmobilebrowsers.mobi). Based on the WordPress Mobile Plugin, which he also wrote, this script looks at the requesting user agent string and abstracts the data to match against the majority of mobile devices, then routes it to the mobile version of your site. All you need to do is include the file with the function, then call the function before your PHP pages do anything else:

```
include('mobile_device_detect.php');

mobile_device_detect();
```

The script is easy to install, works as advertised, is free for personal use, and is available for a small fee for commercial applications. Of course your application needs to be written in PHP for it to work, but for many smaller sites or web apps looking to do simple device detection, Andy’s script is the way to go.

Greg Mulmash’s Mobile Browser Detection

Greg’s script, also written in PHP, goes one step further. Like Andy’s script, it looks at the requesting user agent string and looks for data recognizable as a mobile device or browser.

Using the WURFL database of 6,750 different mobile browser User Agent IDs, this script caught 94.34 percent of them.

Simply create a PHP document with the following code in it:

```
function checkmobile(){

if(isset($_SERVER["HTTP_X_WAP_PROFILE"])) return true;

if(preg_match("/wap\.|\.wap/i",$_SERVER["HTTP_ACCEPT"])) return true;

if(isset($_SERVER["HTTP_USER_AGENT"])){

// Ignore iPhone and iPod touches

if(preg_match("/iphone/i",$_SERVER["HTTP_USER_AGENT"])) return false;

// Quick Array to kill out matches in the user agent

// that might cause false positives

$badmatches = array("Creative\
AutoUpdate","OfficeLiveConnector","MSIE\ 8\.0");

foreach($badmatches as
$badstring){if(preg_match("/".$badstring."/i",$_SERVER["HTTP_USER_AGENT"])) return
false;
}

// Now we'll go for positive matches

if(preg_match("/Creative\
AutoUpdate/i",$_SERVER["HTTP_USER_AGENT"])) return false;

$uamatches = array("midp", "j2me", "avantg", "docomo", "novarra", "palmos",
"palmsource", "240x320", "opwv", "chtml", "pda", "windows\ ce", "mmp\/",
"blackberry", "mib\/", "symbian", "wireless", "nokia", "hand", "mobi", "phone",
"cdm", "up\.b", "audio", "SIE\-", "SEC\-", "samsung", "HTC", "mot\-", "mitsu",
"sagem", "sony", "alcatel", "lg", "erics", "vx", "NEC", "philips", "mmm", "xx",
"panasonic", "sharp", "wap", "sch", "rover", "pocket", "benq", "java", "pt", "pg",
"vox", "amoi", "bird", "compal", "kg", "voda", "sany", "kdd", "dbt", "sendo",
"sgh", "gradi", "jb", "\d\d\di", "moto");

foreach($uamatches as $uastring){
if(preg_match("/".$uastring."/i",$_SERVER["HTTP_USER_AGENT"]))
return true;
}

}
return false;
}
```

Once this is in place, you can check for a mobile browser and, if it returns true, conditionally load your stylesheet. This technique is more reliable than using a media type, as only one stylesheet will be presented to mobile browsers:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>

<head>

<title>My Page Title</title>

<?phpif(checkmobile()){

echo "<link rel=\"stylesheet\" type=\"text/css\" href=\"/mobilestyle.css\">";

} else {

echo "<link rel=\"stylesheet\" type=\"text/css\" href=\"/regularstyle.css\">";

}

?>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>

...
```

Or as with Andy's script, you can also just redirect traffic to a mobile site that lives in an entirely different location:

```
<?php

if(checkmobile()) header("Location:http://m.domain.com");

?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html>

...
```

This script is completely free to use, but like Andy's script, requires that your code be in PHP.

The Switcher

The Switcher is a device detection layer to route mobile devices to mobile experiences, available in Java, PHP and .NET, provided by WURFL maintainer Luca Passani.

Whenever an HTTP request is received, the Switcher will analyze each header, checking to see whether it is a mobile device or browser. If it is, the Switcher will direct the requesting client to the URL that's most appropriate for it. It provides a great deal of customization to suit your specific needs.

It is available to purchase as full source code from http://www.passani.it/switcher/ for €500.

htaccess-Based Device Detection

Another option available to sites running on Apache-based web servers is to create an *.htaccess* file to perform a device lookup method similar to the previous solutions. *.htaccess* files are magic little files that allow you to do URL rewriting, redirection, and other handy tricks that help to provide a positive user experience.

This example is provided by Ryan Neudorf (http://ohryan.ca/blog/2009/02/18/revisiting-mobile-redirection-using-htaccess-rewrite-rules/):

```
# don't apply the rules if you're already in the mobile directory

# infinite loop

# you'll want to test against the host if you're using a subdomain



RewriteCond %{REQUEST_URI} !^/mobiledirectoryhere/.*$



# if the browser accepts these mime-types,

# it's definitely mobile, or pretending to be
```

```
RewriteCond %{HTTP_ACCEPT} "text\/vnd\.wap\.wml|application\/vnd\.wap\.xhtml\+xml"
[NC,OR]


# a bunch of user agent tests
RewriteCond %{HTTP_USER_AGENT} "sony|symbian|nokia|samsung|mobile|windows
ce|epoc|opera" [NC,OR]
RewriteCond %{HTTP_USER_AGENT} "mini|nitro|j2me|midp-|cldc-
|netfront|mot|up\.browser|up\.link|audiovox"[NC,OR]
RewriteCond %{HTTP_USER_AGENT}
"blackberry|ericsson,|panasonic|philips|sanyo|sharp|sie-"[NC,OR]
RewriteCond %{HTTP_USER_AGENT}
"portalmmm|blazer|avantgo|danger|palm|series60|palmsource|pocketpc"[NC,OR]
RewriteCond %{HTTP_USER_AGENT} "smartphone|rover|ipaq|au-
mic,|alcatel|ericy|vodafone\/|wap1\.|wap2\.|iPhone|android"[NC]
```

JavaScript-Based Device Detection

Yet another way to detect higher-end devices that support JavaScript is to look up the user agent and redirect if the value is true. For example, if we wanted to detect iPhones or iPod touches, we might add the following to the initial page of our domain:

```
<script type="text/javascript" charset="utf-8">
    if (navigator.userAgent.match(/AppleWebKit/i) &&
navigator.userAgent.match(/Mobile/i)) {
        window.location.replace('/path/to/iphone/site/');
     }
</script>
```

Although this approach is the easiest to add to your site, it only works with devices that support JavaScript, which aren't many. It gets tricky when you're trying to support multiple devices, and requires the mobile device to load the desktop content first before the redirect occurs, which costs the user time and money.

Reverse Device Detection

I suggested an alternative approach to the problem in an article I wrote several years ago; I call this alternative *reverse device detection*. In this technique, the primary content on your domain is your basic or lowest-common-denominator mobile content. This ensures that basic devices, such as low-end mobile devices, eBook readers, or other basic mobile devices, still have a usable experience.

Use JavaScript to detect the screen size, looking for screens at least 800 pixels wide, then redirect those users to the desktop site. Although it is not a popular idea to not have your primary site live at the root of your domain, this technique is actually quite efficient, because desktop browsers, or even high-end mobile browsers, are far fewer in number, more consistent, and much easier to detect.

Depending on how similar your mobile site architecture is to your desktop architecture, this technique can also have benefits such as making your site more accessible to those who use assistive devices as well as making your content more search engine friendly.

WordPress Mobile Plugin

As mentioned before, Andy Moore developed a plugin for the WordPress blogging tool, which is very popular and free. As an increasing number of people start to use WordPress for purposes beyond personal blogging, this plugin makes a simple and easy way to do device targeting. This script not only does the detection, but also provides light adapting as well.

You can see how simple it is from Andy's instructions:

1. Download the plugin (http://wordpressmobile.mobi/download.zip).
2. Unzip the download.
3. Upload *wordpress-mobile.php* to your *wp-content/plugins* folder.
4. Activate the plugin.
5. Configure the plugin to your needs.

That's it. Follow those steps and your WordPress site will automatically detect mobile devices and show them a mobile-ready version more suited to a small screen.

dotMobi WordPress Mobile Pack

James Pearce and the folks at dotMobi have taken WordPress integration one step further, turning WordPress into a full-fledged mobile content management system. Their solution provides the following features:

- Mobile switcher to detect mobile visitors and provide an appropriate experience
- Base mobile theme for quick-and-easy XHTML-MP compliance
- Extended mobile themes so that you can unleash your mobile creativity
- Transcoding and device adaptation to optimize the mobile experience
- DeviceAtlas integration for world-class adaptation
- Mobile admin panel for when posts can't wait
- Mobile ad widget to make you some money
- Barcode widget to help users bookmark your blog

The dotMobi WordPress Mobile Pack (http://mobiforge.com/running/story/the-dotmobi-wordpress-mobile-pack) is available for free.

Mobile Fu

Moving away from PHP-based solutions, Mobile Fu is a free plugin for Ruby on Rails (http://github.com/brendanlim/mobile-fu/tree/master). It will automatically detect mobile devices that access your Rails application, then dynamically define the stylesheet to be viewed depending on the requesting device.

And Many More…

This is just the beginning; new solutions to the problems of device targeting are emerging every month. Content management systems and web application frameworks are incorporating mobile templates and logic into their core. I imagine that one day soon, you won't be able to run a web-based publishing system without the ability to output a mobile version and route devices to it, built right in.

2.9.5 Strategy #4: Full Adaptation

The fourth and final multiserving strategy is *full content adaptation*, or the process of making extremely unique mobile experiences based on the device that is requesting the content—almost always dynamically. Like the other multiserving strategies, it starts with detecting the device that is requesting content and matching that to a valid user-agent string; then the system outputs markup, styles, and images generated exclusively for that device.

Let's say, for example, that we want to support 20 devices across multiple operators. These devices carry browsers that range from Class A to Class D, each with different screen sizes and device capabilities. Using a full adaptation strategy, we detect, adapt, and render, but then we take it a step further. For each request, we detect each of those device's user-agent strings against our device database and then dynamically adapt our base templates to suit that device class and render to the device.

With full adaptation, we dynamically create four specific experiences for each of our classes based on a number of templates and assets designed to degrade or adapt that would render a unique experience to each device.

In addition, we may have interclass optimizations. An example of this might be when you have the same device deployed on two different operator networks. This is a more common problem in the U.S. market, where we have more than just one type of network standard. Occasionally, multiple devices can share the same model number and even have similar user agents but render content differently. In this case, we need full adaptation to do detailed lookups and make sure that we detect the right device on the right network, then do interclass optimizations for that particular device.

How complex people perceive full adaptation to be will vary based on their development experience. I often stress how difficult adaptation is, and then a developer will always come up and offer a simple solution to solve it: "Couldn't you just rig the so-and-so to detect the doohickey, then render the whatchamacallit?" And every time my answer is the same: "You are exactly right. That would work." I pause so the developer can be proud of his inventive solution before I deal the soul-crushing blow of saying, "Now multiply that by the devices you plan to support."

Although content adaptation by itself is a relatively easy problem to solve, the challenge occurs when you multiply your solution by 10 or 20 devices, which are considered by most to be a modest regional launch. For every device you adapt for, you now have to support it, test it, and adjust code to account for it. Content adaptation can easily add an order of magnitude in complexity for every device you support. This of course quickly leads to cost overruns, reduced margins, and a consumption of valuable resources.

Working “On Deck”

The publishers that employ full adaptation are usually the ones working with operators to put their content on the operators’ mobile portal, often referred to as a “deck.” At this point, we haven’t discussed much about the last part of multiserving: delivery. To ensure that your detection and adaptation system works, you need to test and confirm that it is finally delivered to the device successfully. With most multiserving strategies, you can typically test anywhere from 10–20 different devices and cover all your bases. When you work with operators, this is a very different picture.

Operators often demand that in order for publishers to be on their deck, they must support two to three years’ worth of devices, which can easily be 20–40 different devices offered each year by each operator. At the very least, this means supporting 40–80 different devices, and at most, 60–120 devices for a single operator.

To put this into perspective, most markets around the world might have only two or three major or Tier-1 operators, so the total number of devices you might need to support is 40–80 device models, depending on the operator agreements. Plus, they all operate on the same GSM network standard, so you rarely have to test the same device twice. In the U.S. market, we have four Tier-1 operators, and as many Tier-2 or regional operators, and several different networks. So you might have 20–40 device models per operator, but because each of those devices is provisioned a bit differently at each operator, each device has to be tested for each operator.

This isn’t to say that you have to create 200 versions of the same site; typically, you only need to create a version for each device class. But if you plan to work with an operator, it does mean that you have to test on each device you plan to support.

Where these numbers start to hit your bottom line is in quality assurance (QA). Each release will need to be tested thoroughly before the operator allows you to put it into production. This means that getting a product out on multiple operators for multiple devices means a lot of time pushing buttons on many different devices.

Working “Off Deck”

Now there are actually plenty of cases where you might want to use a full adaptation strategy, or least a pseudoadaptation strategy, which is greater than just detecting and routing, but less than completely dynamic. This is normally referred to in the mobile community as being “off deck,” which usually means delivering content to multiple mobile devices over the Web and not through an operator.

The most common reasons to use a full adaptation strategy are:

- You want to deliver the best possible experience to a number of Class B or lower devices where other strategies don’t cut it.
- You want to support users outside of the United States, where higher-end devices constitute the majority.
- You want to do anything highly transactional, like billing or payments for content or services, for Class B or lower devices.
- You want to do SMS campaigns that terminate with a mobile website.
- You have a media rich experience, such as images, video, or audio, and need it to render properly on several devices.
- You want to support several nonphone mobile devices, like GPS units, e-book readers, portable gaming consoles, and so on.
- You want to support a number of different devices and contexts, either now or in the future.

None of these reasons are unreasonable; in fact, if you want to take mobile seriously and support anything lower than Class A devices, you probably need to take a look at a full adaptation strategy.

Luckily, we have several existing tools to solve this problem as well. Let’s briefly look at several of the common tools out there used for full adaptation.

WURFL

WURFL, or the Wireless Universal Resource File (http://wurfl.sourceforge.net), is an open source database of device profiles. It is a massive endeavor—the largest and one of the most active open source projects in mobile. WURFL founder Luca Passani defines the project like this:

WURFL is an XML configuration file which contains information about capabilities and features of many mobile devices.

As new devices are released, the community (often the handset makers themselves) profiles the device. The result is a detailed database of device attributes, from user agent strings to screen size dimensions, supported media types, and other device characteristics.

Usually developers use WURFL to load the device profiles into their server directives or to create their own conditional logic to dynamically load or replace content based on the requesting device. WURFL provides a free API for PHP-, .NET-, and Java-based applications to query the database and return device information (in the form of “capabilities,” where a capability represents a particular device property).

For example, we could use WURFL to detect device information in PHP this way:

```
<?php echo $_SERVER["HTTP_USER_AGENT"];

$requestingDevice = $wurflManager->getDeviceForHttpRequest($_SERVER);

?>



<ul>

<li>ID: <?php echo $requestingDevice->id ?> </li>

<li>Brand Name: <?php echo $requestingDevice->getCapability("brand_name") ?> </li>

<li>Model Name: <?php echo $requestingDevice->getCapability("model_name") ?> </li>

<li>Xhtml Preferred Markup:

<?php echo $requestingDevice->getCapability("preferred_markup") ?> </li>

<li>Resolution Width:

<?php echo $requestingDevice->getCapability("resolution_width") ?> </li>

<li>Resolution Height:

<?php echo $requestingDevice->getCapability("resolution_height") ?> </li>

</ul>
```

WURFL benefits from the feedback of thousands of mobile developers around the globe. Some of these developers are also contributors of device data and can add device information through the public interface available at http://wurflpro.com.

In addition to the APIs, the WURFL project offers developers tools such as WALL and WNG adaptation libraries, which we will come back to a bit later in this chapter.

DeviceAtlas

dotMobi's DeviceAtlas is a premium device database service that leverages multiple device databases, including WURFL. DeviceAtlas is meant to be used in one or all of the following ways:

As a test suite

A full-featured mobile web testing site that enables you to visit a site from a mobile device and run a series of tests

As a web interface to the database

Enables you to search for device properties and interact with the community

As a standards-based API

Enables you to embed in your application an API that is built on W3C recommendations

DeviceAtlas can be used in a variety of ways. For example, DeviceAtlas could be used to detect and redirect devices in PHP this way:

```
<?php
include 'Mobi/Mtld/DA/Api.php';


$s = microtime(true);


$memcache_enabled = extension_loaded("memcache");
$no_cache = array_key_exists("nocache", $_GET);
if ($memcache_enabled && !$no_cache) {
  $memcache = new Memcache;
  $memcache->connect('localhost', 11211);

  $tree = $memcache->get('tree');

}


if (!is_array($tree)) {

  $tree = Mobi_Mtld_DA_Api::getTreeFromFile("json/Sample.json");

  if ($memcache_enabled && !$no_cache) {

    $memcache->set('tree', $tree, false, 10);

  }

}


if ($memcache_enabled && !$no_cache) {

  $memcache->close();

}


$properties                    =                    Mobi_Mtld_DA_Api::getProperties($tree,
$_SERVER['HTTP_USER_AGENT']);

//further performance can be gained through caching the properties against the

user-agent as a key (since many requests are likely to come from one device during

its visit)


if($properties[mobileDevice] && !$properties[isBrowser]) {

    Header("Location: /mobi/");

}
```

```
?>
```

But DeviceAtlas could also be used as an adaptation tool. In this example, it is used to detect the device, then adapt an image to be optimized for the requesting mobile device:

```
<?php
include('imageAdaptation.php');
$i='imgc/eye.jpg'; // This is the main image that you want to see in your mobile
screen
$imgurl=convertImage($i); // $imgurl will hold the path of the converted image that
is suitable for your mobile
?>
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.0//EN"
"http://www.wapforum.org/DTD/xhtml-mobile10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>Test the API</title></head>
<body>
<img src="<?php echo $imgurl;?>" />
</body>
</html>
```

As Andrea Trasatti of dotMobi describes DeviceAtlas:

DeviceAtlas is a full ecosystem to test devices, store the results, and use them in your application, all in one place, all at your fingertips.

DeviceAtlas is available in a tiered plan, depending on how frequently you need to update your data. A free developer’s version is also available for testing at http://deviceatlas.com.

Volantis

One of the larger content adaptation vendors, Volantis—like many adaptation solution providers—maintains its own proprietary device database. It has its own testing laboratory, and tests and records all device capabilities to enable it to work with its primary product. Its device detection service is included in all its service plans, but expect the pricing to be out of reach for all but a few. Volantis does, however, provide an open source version of its Java-based software that includes an older version of its database.

From Volantis:

Volantis Mobility Server™ is a Java-based development and runtime platform, allowing web developers to build and run their own mobile Internet applications across over 6,200 devices.

Volantis Mobility Server is designed to reduce the complexity of managing mobile content, so that developers and content owners can more easily create innovative content and services and users can use the device of their choice to access it.

For those serious about full adaptation, especially in an enterprise environment, I recommend giving Volantis a look: http://www.volantis.com/volantis-mobility-server.

WALL and WNG

WALL (Wireless Abstraction Library) and WNG (WALL Next Generation) are adaptation libraries written by WURFL's Luca Passani. WNG is the more modern tool, building on the success of WALL as a way to abstract content and adapt it on the fly.

An example JSP page written with WNG might look like this:

```
<wng:head>
  <wng:title text="Radio WNG" />
  <wng:meta httpEquiv="Cache-Control" content="no-cache"/>
  <wng:css_style>
    <wng:css selector="body">
      <wng:css_property name="margin" value="0" />
      <wng:css_property name="border" value="0" />
      <wng:css_property name="font-family" value="arial, sans-serif" />
    </wng:css>
```

```
   <wng:css selector="a">
     <wng:css_property name="color" value="#666" />
     <wng:css_property name="font-weight" value="bold" />
     <wng:css_property name="text-decoration" value="none" />
   </wng:css>
   <wng:css selector=".label">
     <wng:css_property name="color" value="#222" />
     <wng:css_property name="margin-left" value="10px" />
     <wng:css_property name="font-size" value="10px" />
   </wng:css>
   <wng:css selector=".input">
     <wng:css_property name="color" value="#222" />
     <wng:css_property name="margin-left" value="10px" />
     <wng:css_property name="font-size" value="10px" />
   </wng:css>
  </wng:css_style>
</wng:head>
```

Once the device is detected, WNG dynamically replaces the tags with the markup appropriate for the device.

WALL and WNG are open source and available for free for both Java- and PHP-based servers. They are ideal for developers looking to build their own customized adaptation solution. Learn more at http://wurfl.sourceforge.net/java/tutorial.php and http://wurfl.sourceforge.net/wng/tutorial.php.

Yahoo! Blueprint

Yahoo! Blueprint is another adaptation service, which allows you to host applications on your own server but use the Blueprint XML markup language, loosely based on XForms, to adapt and deliver them to multiple devices, as shown in Figure 13-4.

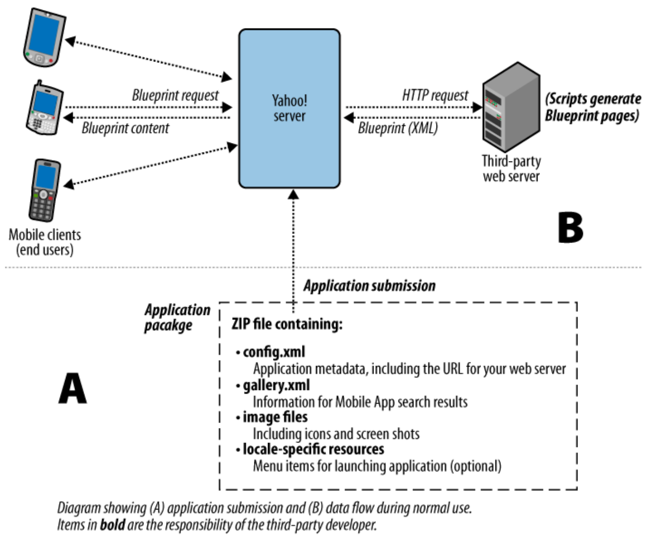

Blueprint request
Blueprint content
Yahoo! server
HTTP request
Blueprint (XML)
(Scripts generate Blueprint pages)
Third-party web server
Mobile clients (end users)
B
Application submission
Application pacakge
A
ZIP file containing:
• config.xml
Application metadata, including the URL for your web server
• gallery.xml
Information for Mobile App search results
• image files
Including icons and screen shots
• locale-specific resources
Menu items for launching application (optional)
Diagram showing (A) application submission and (B) data flow during normal use.
Items in bold are the responsibility of the third-party developer.

Figure 13-4. Using Yahoo! Blueprint to adapt and deliver to multiple devices

Blueprint is a free service from Yahoo! and can be a great way to build services on top of a robust and feature-complete platform. Read more about Blueprint at http://mobile.yahoo.com/devcenter.

Netbiscuits

Netbiscuits is a web service, like the others, that allows you to create and manage mobile websites and web applications and then detect, adapt, and deliver the experiences to the right devices (Figure 13-5).

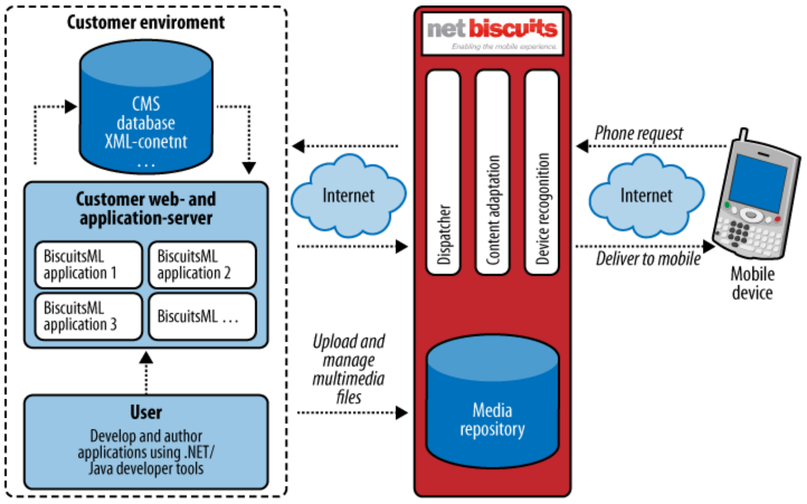


Figure 13-5. The Netbiscuits web service

It is available in JSP, PHP, and .NET as both a hosted and a custom API in different pricing tiers. Read more about Netbiscuits at http://www.netbiscuits.com.

MobileAware

MobileAware is an enterprise mobile adaptation system. Like the others, it can detect, adapt, and deliver experiences based on the requesting device (Figure 13-6).

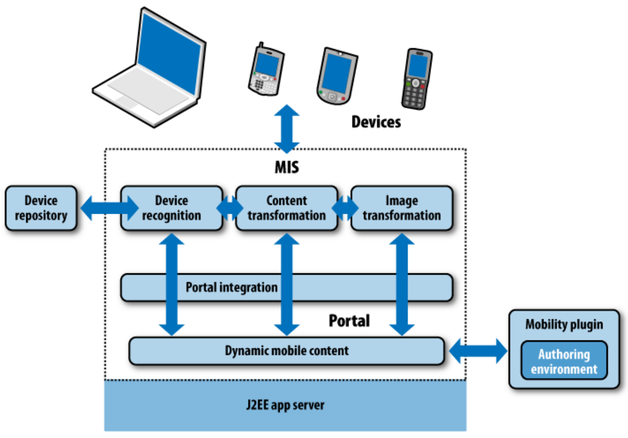


Figure 13-6. The MobileAware web service

It is available for Java-based servers for a fee. More information is available at http://www.mobileaware.com.

Mobify

The folks at Mobify have taken this process of adaptation and made it available for anyone. Using Mobify's web-based tool requires no setup or installation. You can simply point to the content from your desktop website and then apply styling to it. After you point your mobile domain to Mobify, you have an effortlessly adapted site that supports more than 4,000 devices.

The service is offered in tiered monthly pricing, starting with a free limited plan. This is by far the least expensive adaptation solution, and is fairly full-featured compared to the other solutions, which might cost you thousands.

2.9.6 What Domain Do I Use?

Ideally, the user should have to enter only your primary URL—for example, domain.com—and your multiserving system should route him to the appropriate experience for his device and context. There are some who believe that the users should control which experience they see, and therefore your mobile experiences should live at a separate URL. I am not one of those people.

I believe routing to the best experience should be the job of technology and give users the option to exit if they so choose. But unfortunately, due to some of the issues with sever-side device routing, as I mentioned before, a separate domain is the only available option. In this case, I offer the following solutions.

m.domain.com

The most common method is to prefix your site domain with a subdomain called "m"—for example, *m.your-domain-name.com*. Now that many sites are providing normal and iPhone versions of their sites, we have *m.domain.com* and iphone.domain.com. It's hardly consistent or the best experience for the user, but it is extremely simple to set up.

I recommend at the very least putting all of your mobile content in a single domain, and using one of the previous mentioned detection techniques to route to the best experience. This is preferable to using a URL that the user has to associate with their mobile device.

domain.com/mobile or domain.com/m

Before subdomains were all the rage, most sites used a folder, such as *domain.com/m*. Although it's not as common these days, I think this method works well when combined with subdomains to solve the problem with multiple mobile sites. For example, *m.domain.com* might point to your lowest-common-denominator site and m.domain.com/iphone would point to your iPhone site. Any additional versions you create later, such as *m.domain.com/android/*, might simply be added into folders on your m-domain. Using a little bit of simple (specifically reverse or *.htaccess*) device detection on the root of the *m-domain* would enable you to redirect your higher-end devices to the appropriate folder.

If you choose not to use folders on your primary domain, I highly recommend setting up *.htaccess* or server redirects on */m* and */mobile* folders to wherever you decide to put your site, just in case.

domain.mobi

The .mobi top-level domain is the ICANN-approved top-level domain specifically for mobile devices and gives publishers the option to use an alternate domain for their mobile site. Instead of using device detection, subdomains, or directors, mobile users go to *domain.mobi*.

If you use a .mobi domain, you can also route all traffic to it, using some simple device detection or server redirects. For example, you could establish server redirects: when a user enters *domain.com/mobile*, she is redirected to *domain.mobi*.

For example, you might consider the following domain mapping:

- *m.domain.com→domain.mobi*
- *domain.com/m→domain.mobi*
- *iphone.domain.com→domain.mobi/iphone*

2.9.7 Taking the Next Step

How do you feel about adapting to mobile devices now? I hope that I was able to report the available options with a minimum of propaganda. Like I said at the beginning: there are no right or wrong answers, only what makes the most sense for your users.

My advice is to start simple with something that can be done in, say, a long weekend. If that means setting up an adaptation system, then go for it. If it means doing something basic, that is OK, too. The most important thing is getting it mobile. Everything else will fall into place after that.

2.10 Supporting Devices

Imagine a restaurant where you could just walk up and order whatever you felt like—no menu, just anything you want. If you felt like eating a salad, the restaurant would make you a salad. If you felt like a steak, it would make you steak just the way you liked it. If you wanted a banana, peanut butter, and potato chip sandwich, it would make it. Or, if you felt like ratatouille niçoise, in which each ingredient is sautéed separately, layered together, then baked to perfection (the proper way to prepare this delicious dish, by the way), it would make that for you, too. If you are the only customer, then this restaurant would certainly be a dream come true, because the chef would have time to focus and prepare your dish to perfection. It would probably be one of the best meals you've ever had in your life.

But, of course, as more customers enter our fictional restaurant and as more custom orders are placed, the kitchen would become a nightmare. The chef, regardless of his training and expertise, would have to deal with so many variables that not only would it become unmanageable, but the quality would start to decrease. It becomes impossible to create so many variations and still maintain a high degree of quality.

This is the case with the task of testing mobile devices, and unfortunately, we are the chefs. Dealing with one device is easy, but as you start adding more and more devices the variables/variations become too great to manage. Maintaining quality and consistency moves further out of reach.

Many developers tell me, “No problem! Let’s be great on one and OK on another.” A wise strategy, maybe, for the mobile web, but for native applications, OK is often not quite good enough. In order to get your native applications downloaded onto your users’ devices, the applications will need to be certified and approved by at least one organization outside of your own—an opaque process regardless of what channel, platform, device, or operator you intend to work with.

Another lazy strategy is to dismiss personally testing devices altogether and assign them to a QA resource or outsource it entirely. Unfortunately, it is not that easy. Good mobile design and development requires that everyone be a part of ensuring that the work they do ends up on devices correctly, making sure that your experiences are usable and valuable to the user. Testing does not simply occur at the end of the project, but throughout the process.

Keep in mind that there is no one foolproof way of testing your work on devices. It takes flexibility and often some creativity to find the right solution for your resources, project, or company. This chapter discusses some of the various techniques and best practices for testing your work on devices and trying to cost-effectively support as many devices as you possibly can.

2.10.1 Having a Device Plan

I’ve mentioned numerous times in this book how challenging it is to support multiple devices. This is a problem that shouldn’t exist, but it does, and it likely always will. One day in the future, mobile phone platforms will be more alike than they currently are, but by that time we will have many more “mobile” devices to contend with.

I hate to break it to you, but the days of being able to simply test your product on two or three different browsers are over. The browsers and modes in which tomorrow’s users will access your content will increase. Our only hope for a sane testing cycle will be the Web and consistent support for web standards. But that is tomorrow; let’s talk about today.

Sorting out your device support strategy now prepares you for how you will span your content and services across multiple digital landscapes, including phones, desktops, always-on devices, the social web, and more. The list will only grow over time, and if the past is prologue to the future, then each of these new contextual landscapes will appear faster and faster.

2.10.2 Device Testing

So you have your device plan. Now how do you go about getting devices to test your work on? The simple answer would be to just go buy them, but you guessed it: it isn’t that easy. Many devices are subsidized through the operator. So buying a device at the advertised price means buying into a two-year contract that goes with it. Because you need only one contract per operator, this means you have to pay full price for each device you plan to support. At $500–$600 per device unsubsidized, the costs of having multiple devices adds up fast: yet another reason to have a well-researched device plan, so you know which devices to purchase.

Access to Devices

Gaining access to multiple devices is a challenge for every mobile design and developer. I recommend that everyone involved has at least one device, indicative of your primary device class, on the desk when working on the project. This will dramatically reduce the number of assumptions that you have to make during the design and development of the product and can ultimately reduce the amount of time spent testing. This of course can be a device shared among everyone in the office, but if you have multiple offices, or outsourced staff, this can be a problem.

There are alternatives to hands-on device testing, which are discussed later in this chapter. But nothing beats having the device in your hand at the moment of creation. Over the course of my career, I’ve done it all, and my best mobile products have been the ones where I had access to the intended devices. As you know by now, mobile design is about context, and context is not easily duplicated on the desktop. Testing your work outside the walls of your office will affect how you design.

For example, take an application that uses the accelerometer, like being able to shake the device to load a new page like the iPhone application Urban Spoon (Figure 15-1). How do you test an application where you interact with the mobile nature of the device, if you don’t have a device to shake?

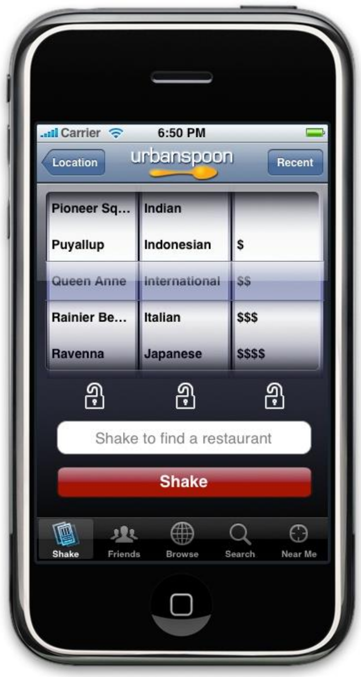


Figure 15-1. With Urban Spoon you shake to produce a result—something that is hard to test for without an actual device

Guerrilla testing

The guerrilla testing technique I’ve been known to recommend in mobile web workshops is to go down to the operator store and test your work on the devices in the store. It isn’t exactly something you can do during design or development, but you certainly can confirm how it works on several devices that you can hold in your hand all at once. And as you can find operator stores on just about every other block in most urban areas around the world, you are never very far from a “testing center.”

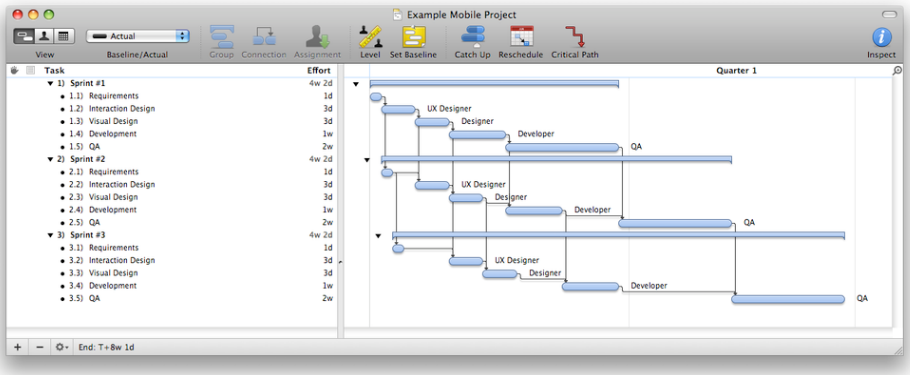
Mobile Monday device libraries

Another route I recommend is talking to your peers at your local Mobile Monday chapter. Mobile Monday is a group of mobilers that meet the first Monday of the month in cities around the world. If you live in a city that has a chapter, and it is the first Monday of the month, then you should put this book down and head over to a meeting. It is a great way to connect with your local mobile community. But for our purposes, you should propose to your local chapter that you start pooling together your collective devices. Create a spreadsheet of who has what devices on what network, then offer to share (and track) them amongst the group, like a library, checking out devices for a day, week, or month. This way, instead of one person buying dozens of devices, you can coordinate with each other so that everyone buys one or two devices and shares them among the group.

Estimating the Testing Effort

One of the worst mistakes you can make is underestimating the time and resources required to test your work on multiple devices. The rule of thumb is that device testing can take anywhere between two and four times the development effort, meaning that for every developer, you need at least two to three QA resources. Or if you are flying solo, then for every week of development, you can easily spend two testing and fixing on multiple devices. This of course depends greatly on your device plan and the range of devices you plan to support.

When developing a single mobile product or supporting a single device class, an extended testing effort can simply result in a delayed release. But when you are supporting multiple classes, the testing effort can hold up resources from moving on to the next phase of development, as shown in Figure 15-2.

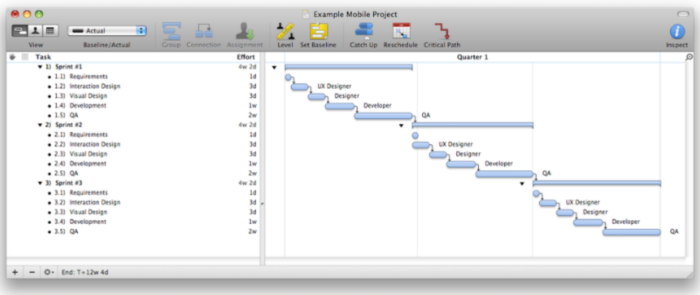


Task	Effort
1) Sprint #1	4w 2d
1.1) Requirements	1d
1.2) Interaction Design	3d
1.3) Visual Design	3d
1.4) Development	1w
1.5) QA	2w
2) Sprint #2	4w 2d
2.1) Requirements	1d
2.2) Interaction Design	3d
2.3) Visual Design	3d
2.4) Development	1w
2.5) QA	2w
3) Sprint #3	4w 2d
3.1) Requirements	1d
3.2) Interaction Design	3d
3.3) Visual Design	3d
3.4) Development	1w
3.5) QA	2w

End: T+8w 1d

Figure 15-2. An example mobile project plan, showing how lengthy QA cycles can delay phases

This is one reason why I typically recommend focusing on one class at a time, using sprint-like development process. A sprint is a unit of work from the scrum-based methodology that usually lasts anywhere from a week to a month. Support your primary devices in the first sprint, release it, then after a period of time move on to the second class in a second sprint. This frees up your resources and doesn't get them bogged down in development hell during QA. An example project plan is shown in Figure 15-3.



Task	Effort
1) Sprint #1	4w 2d
1.1) Requirements	1d
1.2) Interaction Design	3d
1.3) Visual Design	3d
1.4) Development	1w
1.5) QA	2w
2) Sprint #2	4w 2d
2.1) Requirements	1d
2.2) Interaction Design	3d
2.3) Visual Design	3d
2.4) Development	1w
2.5) QA	2w
3) Sprint #3	4w 2d
3.1) Requirements	1d
3.2) Interaction Design	3d
3.3) Visual Design	3d
3.4) Development	1w
3.5) QA	2w

Figure 15-3. An example mobile project plan, separating support into multiple sprints

But if sprints don't work for your project, remember to allow ample time after development is done for testing and device support.

Creating a Test Plan

Creating a test plan for mobile devices means testing for every possible problem that the phone might encounter and that might result in it failing to present the desired content to the user. Because these devices are never from a fixed list like we are accustomed to with the desktop web, that means this can be a pretty big list of things to test. Multiply that by a larger number of devices and you can start to see that this can take some time.

You can get away with lighter testing for mobile web products than native applications, as the latter will need to be certified by a third party in order to get released. So if you don't cover all your bases, your product could fail to pass certification. But in either case, you should create a test plan to make sure that you run your product against a number of routine areas in which the product might fail.

Functional tests

The first step is creating a test plan based on your feature list. Jot down all the features of your application and test them one by one on an actual device, if you have one on hand. Does the application pass or fail? If it fails, then write down what you did to cause it to fail. I encourage people to step away from their computers, go someplace where they can focus, and actually write down their notes on a piece of paper.

You get much better and more detailed feedback when there are fewer distractions. For mobile products, more detailed feedback and finding more bugs earlier on greatly reduces the time and efforts needed to get the product finished and released. I've found that it is easy to simply do a "once-over" that finds only a fraction of the issues you find with an in-depth testing phase.

Context tests

After your features are tested, and you know your application is functionally sound, it is time to move on to testing the mobile context—in other words, testing a few scenarios that can cause your product to fail because it is a mobile device.

Questions to ask about your product during context tests include:

- How does the user experience render on the device? Are there visible issues on the test device? If so, explain them in detail.
- Does it load quickly? Does it load correctly? How many seconds does it take before the user can perform an action? Are you showing a progress indicator if it takes more than a second or two to load? Be sure to test the device at the lowest possible speed. Turn off Wi-Fi, 3G, or any high-speed data connections and load your application using the basic GPRS connection.
- Can you use the physical features of the device as they are intended? Do the soft keys work correctly? Does the view change when you rotate the device orientation? Test any unique physical features that your test device might offer.
- Does it terminate correctly? When you place a phone call to the device, does the application terminate itself correctly? When you resume operation, does it retain your previous state? What about sending a text message? Does it interrupt the session, and can it then be resumed?
- What happens when the device loses its connection? Can users recover their sessions? Test this by starting an action, then go into an elevator or parking garage or someplace where you know you will lose the signal.
- If the device is in offline mode and won't be able to send or receive data, is the user presented with an error or an indication of how to rectify the problem?
- For intense data applications, does the application work when hopping from cell tower to cell tower? Test this by using the application in a car on a highway (with someone else driving, of course).

Creating a Test Portal

A simple but effective trick for mobile web device testing is creating a test portal—basically, a web page with a list of links on it to all your web pages, as shown in Figure 15-4. This provides easy access to all your development servers that you need to test.

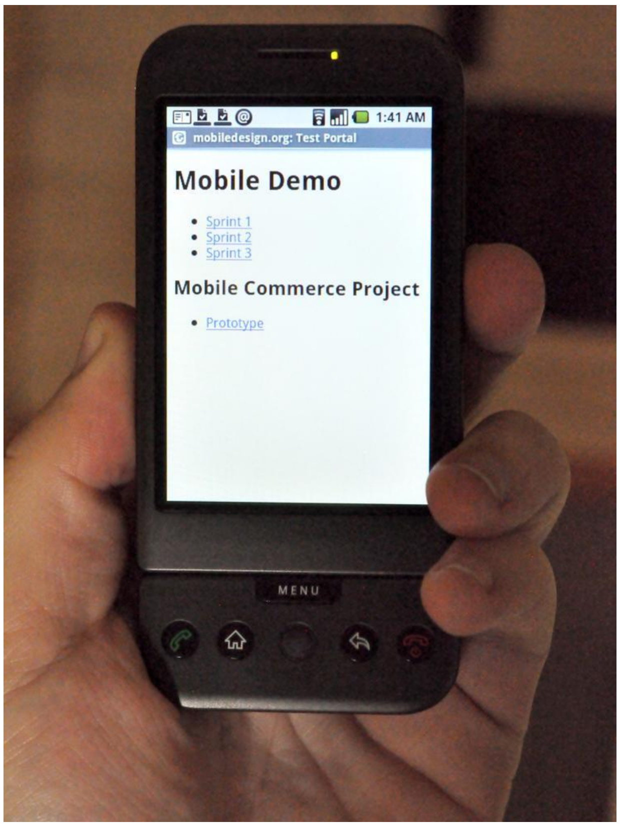


Figure 15-4. An example test portal

Typing URLs into multiple devices is time-consuming, especially development URLs, which can be long. Creating a test portal lets you add URLs for each new product that needs to be tested. You then bookmark that page on each test device, giving you quick access with a fraction of the typing.

Another helpful tool is a URL shortener, like TinyURL or bitly, which shorten long URLs to make them mobile-friendly. I actually set up my own service using a tool like Shorty, which you can install and run on your own servers, enabling you to define the short URL. With Shorty, for example, you can make a URL like *http://m.domain.com/testapp/* map to a longer development URL like *http://dev.domain.com/testapp/session?id=12345*.

2.10.3 Desktop Testing

A big advantage of mobile web products is that you can do the majority of your testing from your desktop before ever getting it on a device. You can verify the majority of your markup, styles, and JavaScript, as well doing functional tests on desktop browsers, before putting them on devices and doing your context tests. Desktop testing reduces the time span between developing a feature, testing a feature, and fixing a feature, and ultimately allows you to spend less time dealing with devices.

Frames

Many web browsers have a minimum window size that is larger than your average device screen, making it impractical for testing mobile websites or web apps in a desktop browser. This is where *inline framesets*, or *iframes*, come in handy. Create a web page with an iframe, specifying the dimensions to match your target mobile screen (as shown in the following code), and then add the URL to your mobile project:

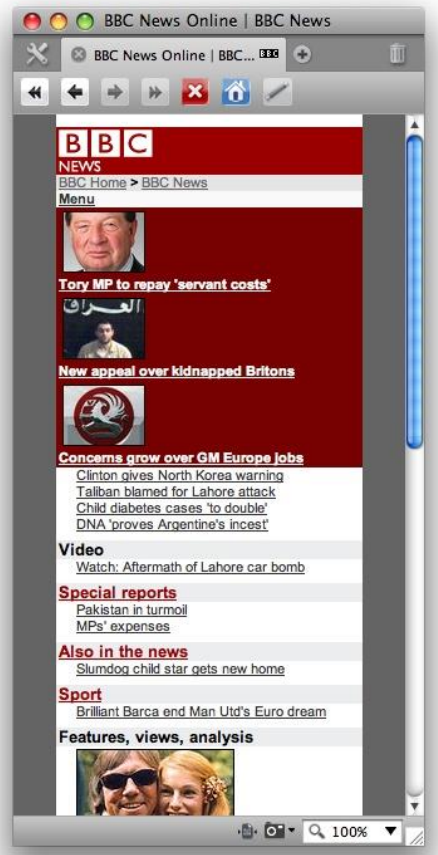
```
<iframe src ="mobile/index.html" width="240px" height="320px"

style="border:1px solid;"> </iframe>
```

Because you can embed multiple iframes in a single page, each pointing to the same URL, you can view how your page will look in multiple screen dimensions from a single page. For demonstration purposes, I often use an image of the target device as a background, positioning the iframe where the screen should go, presenting the mobile project as if it were on the device.

Opera

The Opera desktop browser has a Small Screen view (see Figure 15-5) that mimics a mobile screen, loading the handheld media type if available and presenting the page in a narrower format. Because Opera uses the same rendering engine for its desktop browser as the Opera Mobile product, what appears on one is likely to appear the same on the other. However, Opera Mini, the more popular of the two Opera browsers, uses a different rendering engine, but it still comes out close to what you see on the desktop.



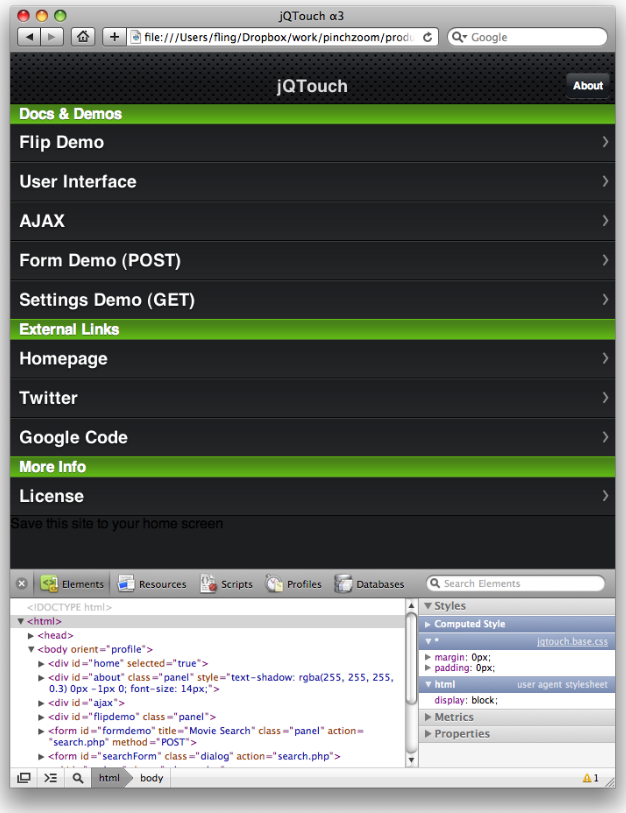
BBC NEWS
BBC Home > BBC News
Menu
Tory MP to repay 'servant costs'
New appeal over kidnapped Britons
Concerns grow over GM Europe jobs
Clinton gives North Korea warning
Taliban blamed for Lahore attack
Child diabetes cases 'to double'
DNA 'proves Argentine's incest'
Video
Watch: Aftermath of Lahore car bomb
Special reports
Pakistan in turmoil
MPs' expenses
Also in the news
Slumdog child star gets new home
Sport
Brilliant Barca end Man Utd's Euro dream
Features, views, analysis

Figure 15-5. Desktop testing using Opera’s Small Screen view

Opera’s Small Screen view is a great way to see how your work might look on a mobile device—to be taken with a grain of salt. Toggle the Small Screen view by going to View→Small Screen in Opera.

WebKit

The WebKit browser engine can also be used for desktop testing for WebKit-based mobile applications. You can download the nightly builds of WebKit and run them on your desktop, giving you a close representation of how it may render on the target device, as shown in Figure 15-6.



```
<!DOCTYPE html>
▼<html>
  ▶<head>
  ▼<body orient="profile">
    ▶<div id="home" selected="true">
    ▶<div id="about" class="panel" style="text-shadow: rgba(255, 255, 255,
     0.3) 0px -1px 0; font-size: 14px;">
    ▶<div id="ajax">
    ▶<div id="flipdemo" class="panel">
    ▶<form id="formdemo" title="Movie Search" class="panel" action=
     "search.php" method="POST">
    ▶<form id="searchForm" class="dialog" action="search.php">
```

```
▼ *
▶ margin: 0px;
▶ padding: 0px;
▼ html
display: block;
```

Figure 15-6. Desktop testing with the WebKit browser

Although the version of WebKit for the desktop isn't always the same as the version that's on mobile browsers, it still provides a great means of quickly testing your project prior to loading it onto actual devices.

In addition, WebKit has an excellent debugging tool, called the Web Inspector, which allows you to see how styles are rendered, offline storage information, the page weight, and the estimated time it takes to load resources, all of which are helpful for fine-tuning your mobile web projects, as shown in Figure 15-7.

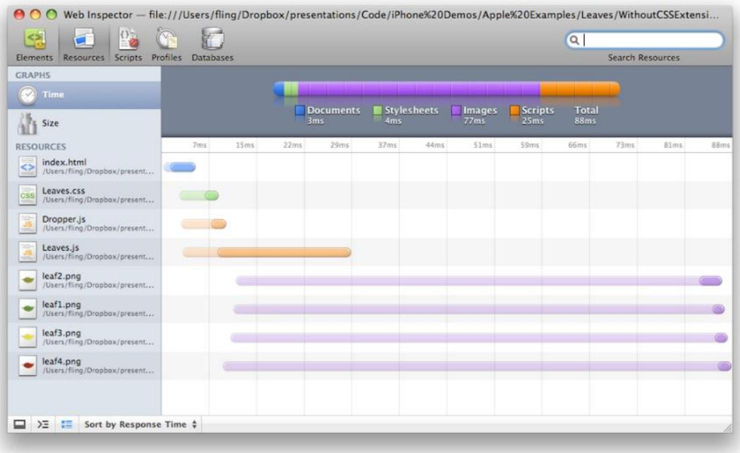

Web Inspector — file:///Users/fling/Dropbox/presentations/Code/iPhone%20Demos/Apple%20Examples/Leaves/WithoutCSSExtensi...
Elements
Resources
Scripts
Profiles
Databases
Search Resources
GRAPHS
Time
Size
Documents 3ms
Stylesheets 4ms
Images 77ms
Scripts 25ms
Total 88ms
7ms
15ms
22ms
29ms
37ms
44ms
51ms
59ms
66ms
73ms
81ms
88ms
RESOURCES
index.html /Users/fling/Dropbox/present...
Leaves.css /Users/fling/Dropbox/present...
Dropper.js /Users/fling/Dropbox/present...
Leaves.js /Users/fling/Dropbox/present...
leaf2.png /Users/fling/Dropbox/present...
leaf1.png /Users/fling/Dropbox/present...
leaf3.png /Users/fling/Dropbox/present...
leaf4.png /Users/fling/Dropbox/present...
Sort by Response Time

Figure 15-7. Using the Web Inspector to debug mobile web WebKit applications

Firefox

One challenge when using device detection techniques is making sure that they work! The Firefox User Agent Switcher extension lets you change the user agent information you send to the server (Figure 15-8). Once you add the data from the supported mobile user agents, you can test how each of your targeted sites renders for the requesting device.

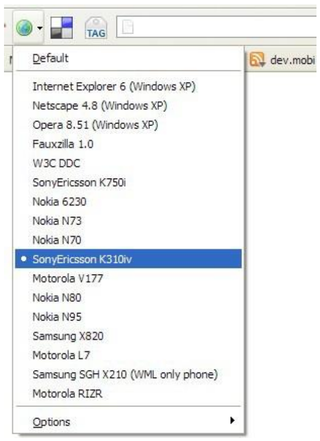


Figure 15-8. Using the Firefox User Agent Switcher to test device detection methods

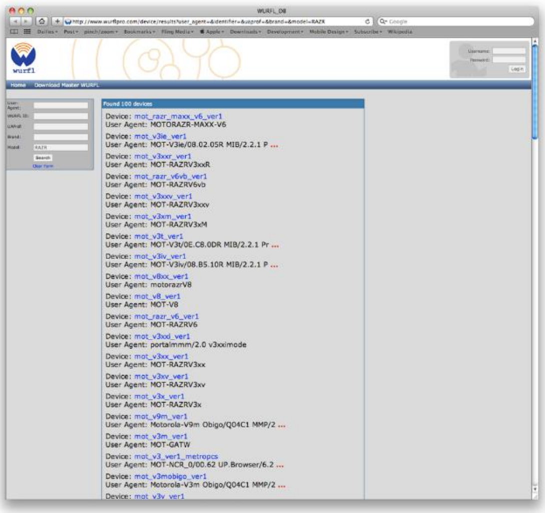
Because few mobile browsers are based on Mozilla's Gecko rendering engine, this form of desktop testing isn't so much for testing how content renders, but is instead for your device detection and content adaptation functional tests.

Also helpful are the Web Developers Toolbar and Firebug Firefox extensions to provide XHTML and CSS debugging, similar to the WebKit Web Inspector.

Collecting User Agents

Test detection and rendering of multiple devices on the desktop means having valid user agents on hand. Given the quantity of devices on the market, there are numerous user agents—sometimes for the same device. The best means of collecting user agents is the open source WURFL database, which contains a large number of user agents submitted by the community and from device makers themselves.

For example, in Chapter 2, I mentioned the numerous mobile browsers found on the Motorola RAZR. Each of those browsers likely has a unique user agent. Searching the WURFL database shows 100 devices and therefore 100 user agents for devices called the "RAZR" (Figure 15-9).



Found 100 devices

Device: mot_razr_maxx_v6_ver1
User Agent: MOTORAZR-MAXX-V6

Device: mot_v3ie_ver1
User Agent: MOT-V3ie/08.02.05R MIB/2.2.1 P ...

Device: mot_v3xxr_ver1
User Agent: MOT-RAZRV3xxR

Device: mot_razr_v6vb_ver1
User Agent: MOT-RAZRV6vb

Device: mot_v3xxv_ver1
User Agent: MOT-RAZRV3xxv

Device: mot_v3xm_ver1
User Agent: MOT-RAZRV3xM

Device: mot_v3t_ver1
User Agent: MOT-V3t/0E.C8.0DR MIB/2.2.1 Pr ...

Device: mot_v3iv_ver1
User Agent: MOT-V3iv/08.B5.10R MIB/2.2.1 P ...

Device: mot_v8xx_ver1
User Agent: motorazrV8

Device: mot_v8_ver1
User Agent: MOT-V8

Device: mot_razr_v6_ver1
User Agent: MOT-RAZRV6

Device: mot_v3xxi_ver1
User Agent: portalmmm/2.0 v3xximode

Device: mot_v3xx_ver1
User Agent: MOT-RAZRV3xx

Device: mot_v3xv_ver1
User Agent: MOT-RAZRV3xv

Device: mot_v3x_ver1
User Agent: MOT-RAZRV3x

Device: mot_v9m_ver1
User Agent: Motorola-V9m Obigo/Q04C1 MMP/2 ...

Device: mot_v3m_ver1
User Agent: MOT-GATW

Device: mot_v3_ver1_metropcs
User Agent: MOT-NCR_0/00.62 UP.Browser/6.2 ...

Device: mot_v3mobigo_ver1
User Agent: Motorola-V3m Obigo/Q04C1 MMP/2 ...

Device: mot_v3v_ver1

Figure 15-9. The 100 user agents for just the Motorola RAZR

You need only a small fraction of these user agents to test your device detection methods on a desktop browser, but when supporting popular devices like the RAZR, make sure to grab a few variations to make sure all RAZRs get to where they're supposed to go.

Simulators and Emulators

Almost every mobile framework comes with an emulator (Figure 15-10) that allows you to test your work in a virtual environment. Because the hardware in your computer is different than the hardware on the device, it has to run in an emulated environment, and can therefore cause inconsistencies between the emulated environment and the real environment.

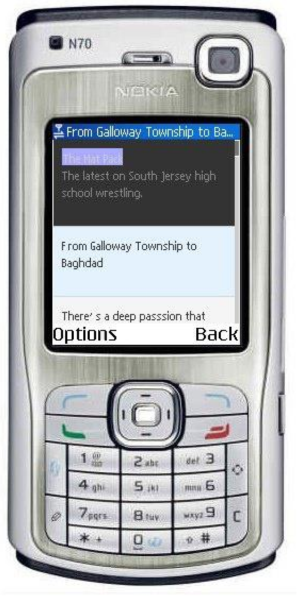
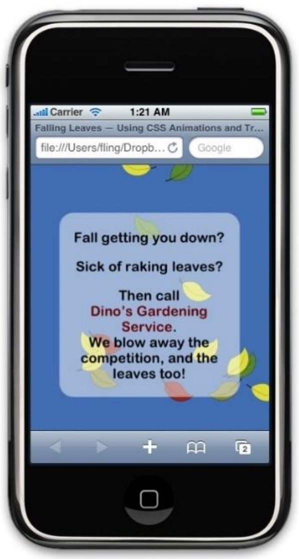

N70
NOKIA
From Galloway Township to Ba...
The Hat Pack
The latest on South Jersey high school wrestling.
From Galloway Township to Baghdad
There' s a deep passsion that
Options
Back

Figure 15-10. The dotMobi emulator

Unfortunately, every new device that is released doesn't necessarily get its own emulator. So if we use the RAZR as our example, emulators are only available for a small handful of the total number of RAZRs. In fact, there is no way we can test all available RAZRs using a traditional emulator model.

Furthermore, emulators are often out of sync with the actual devices, sometimes lagging several years behind. For these reasons, I usually recommend that you not rely on emulators for desktop testing, unless they are your only available option. For mobile web products, you can usually get just as accurate and far faster test cycles using desktop browsers as your first-pass and hands-on device testing for your second pass.

Unlike emulators, however, the iPhone Simulator (Figure 15-11) that comes with the free iPhone SDK is an accurate representation of the iPhone environment. Because the iPhone and iPod touch run on Mac OS X, the simulator is able to run the iPhone OS natively on your desktop, not virtually, as long as your desktop is running Mac OS X.


Carrier
1:21 AM
Falling Leaves — Using CSS Animations and Tr...
file:///Users/fling/Dropb...
Google
Fall getting you down?

Sick of raking leaves?

Then call Dino's Gardening Service. We blow away the competition, and the leaves too!

Figure 15-11. The iPhone Simulator, which gives you a highly accurate desktop testing solution

The iPhone Simulator also runs the same version of WebKit that you will find in the devices themselves. Given that the version of WebKit on the device has been customized for the platform, testing in the iPhone Simulator is a far more accurate means of testing mobile web products for the iPhone than simply using the desktop version of WebKit.

Remote Access

Remote access services let you remotely control an actual device through your desktop, but few exist; DeviceAnywhere is one. The application taps into the actual device, which you rent through the company's software, and it has most of the devices sold in North America and Europe (Figure 15-12).

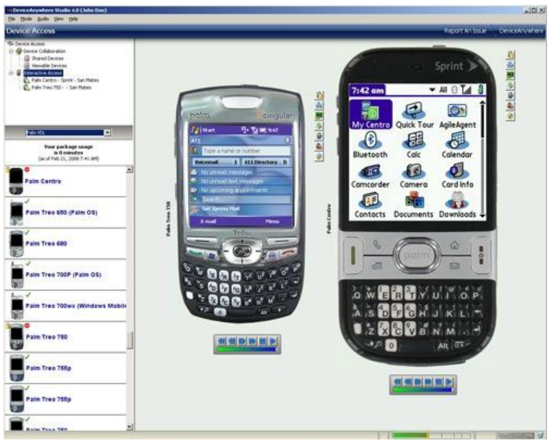


Figure 15-12. DeviceAnywhere

Remote access provides a compelling method for testing, with the added convenience of a desktop emulator, in addition to showing the displayed characteristics of the actual device. Remote access doesn’t replace the tactile lessons learned from using the device. However, for small publishers with limited resources, it solves the problem of supporting many devices and accessing devices in other countries on their native networks.

2.10.4 Usability Testing

Testing a mobile project with actual users always presents invaluable feedback—it gives you an outside perspective directly from your target users. The more you can test your project, the more data and insight you gain into the potential success or failure of your work. As usability guru Jared Spool of User Interface Engineering (http://www.uie.com) says, “Talking to two users is better than talking to one.”

However, usability testing is not unlike the age-old question “If a tree falls in the forest and there is no one to hear it, does it make a sound?” In psychology this is called reactivity, sometimes referred to as the Observer Effect. When people are observed, their behavior changes, due to the fact that they are being observed. In usability tests, it is common for participants to attempt to please the person conducting the test. Some participants prefer to give overly harsh criticism, trying to find the faults.

The goal of a usability test is to identify actionable faults in the system from subjective behavior and opinion. The researcher must find what data is, or will be, indicative of how the target user will use the system. Unfortunately, this is more art than science.

In mobile development, the challenges of the device can add additional subjectivity. I’ve found that traditional usability practices fail to address many concerns that the normal user has with mobile technology. Though I’m a big fan of usability testing, especially for mobile products, I find that there are too many variables to produce tangible and actionable results.

The best way to ensure a usable mobile product is to test early, often, and in context—the three things most usability tests skip. The goal is simply to get the experience in front of real users in some sort of analog of the target context. Alternate forms of usability testing, such remote testing, don't usually work well. Café testing—where you get casual unscheduled feedback from participants at a nearby café—works great, given the casual nature of the locale.

Mobile Usability Test Tips and Tricks

Conducting a great mobile usability test comes down to being prepared, keeping context in mind, and being casual and comfortable with the participants. Here are a few tips and tricks that I have found to work well:

Know your users

Knowing your target users before you start usability testing helps to draw them out while you perform the test. Keep it informal and casual. Don't make them feel like they are being tested. Your posture and attitude during the test will help to define the mood of the room.

Make no assumptions about your participants

Remember: there are no right or wrong answers in a usability test, and there are no right or wrong participants. Your job is to listen, watch, and learn from your participants. Every participant will tell you something new and valuable. Ask them about how they use their mobile devices and try to integrate this into your findings.

Test early

The earlier you can test, the better the product. Integrate user feedback into the entire design and development process. Usability testing just at the end of the project does no one any good. Ideally, there should be a user verification and feedback test after each phase of design and development.

Go to the user; don't have them come to you

Try to perform usability tests in the participants' home, office, or some sort of neutral location. You need to test mobile products in the wild, not in the lab.

Record everything

It can be a challenge to record screens of mobile devices, but it is important to record as much of the participants' behavior as possible. Small gooseneck digital video cameras can be positioned to capture the device screen as well as the users' reactions. There are a number of different approaches to mobile usability test camera "rigs"; my advice is to play around with the resources available to make them work for you. Just remember that it's more important to listen to your user than it is to fiddle around with technology. Keep it simple!

Have someone else record results

Whenever possible, have a designated note taker—a third person whose job is to record everything during the test. Her notes and extra perspective will come in handy later when reviewing the test material.

Test often

Usability testing is not a one-time event; it is something you need to integrate into your process. Plan to test as often as you possibly can.

Keep it simple

As I mentioned before, usability testing is more art than science. Keeping your tests simple and casual makes the process easier for you and your participants. It is easy to turn the simple act of talking to your users into an insurmountable chore, full of process and methodology recommended by the "experts." Go easy on yourself and just talk to your users.

Supporting devices, from device plans to usability tests, can certainly be a challenge, but it doesn't have to be. Be prepared, be smart about your plan, and after a few projects, you'll have it down pat.

2.11 Application development on Android and iPhone

Developing applications for Android and iPhone involves using different programming languages, development environments, and app distribution processes. Here's a brief overview of the key aspects for each platform:

2.11.1 Android Development:

Programming Language:

- Java/Kotlin: Historically, Android apps were primarily written in Java. However, Kotlin has gained popularity as an official language for Android development.

Integrated Development Environment (IDE):
- Android Studio: This is the official IDE for Android development. It supports both Java and Kotlin and provides a range of tools for designing, coding, testing, and debugging Android applications.

Framework:
- Android SDK (Software Development Kit): This includes a set of development tools, libraries, and APIs necessary for Android app development.

User Interface:
- XML Layouts: Android UI is defined using XML files.
- Activities and Fragments: The building blocks of Android UI.

Distribution:
- Google Play Store: Android apps are typically distributed through the Google Play Store.

2.11.2 iPhone (iOS) Development:

Programming Language:
- Swift/Objective-C: Swift is the modern and preferred language for iOS development. Objective-C is older but is still used in some existing projects.

Integrated Development Environment (IDE):
- Xcode: This is the official IDE for iOS development. It includes an interface builder, code editor, and other tools for developing iOS applications.

Framework:
- iOS SDK: Similar to Android SDK, the iOS SDK provides the tools and resources needed for iOS app development.

User Interface:
- Interface Builder: A visual design tool integrated into Xcode.

- **UIKit Framework:** Provides the core components for building iOS interfaces.

Distribution:
- **App Store:** iOS apps are distributed through the Apple App Store.

2.11.3 Cross-Platform Development:

If you want to develop applications that run on both Android and iOS with a single codebase, you might consider cross-platform frameworks. Popular options include:
- **React Native:** Uses JavaScript and React to build cross-platform apps.
- **Flutter:** Developed by Google, uses the Dart programming language.
- **Xamarin:** Uses C# and .NET to build cross-platform apps.

These frameworks aim to reduce development time and effort by allowing you to write code once and deploy it on multiple platforms.

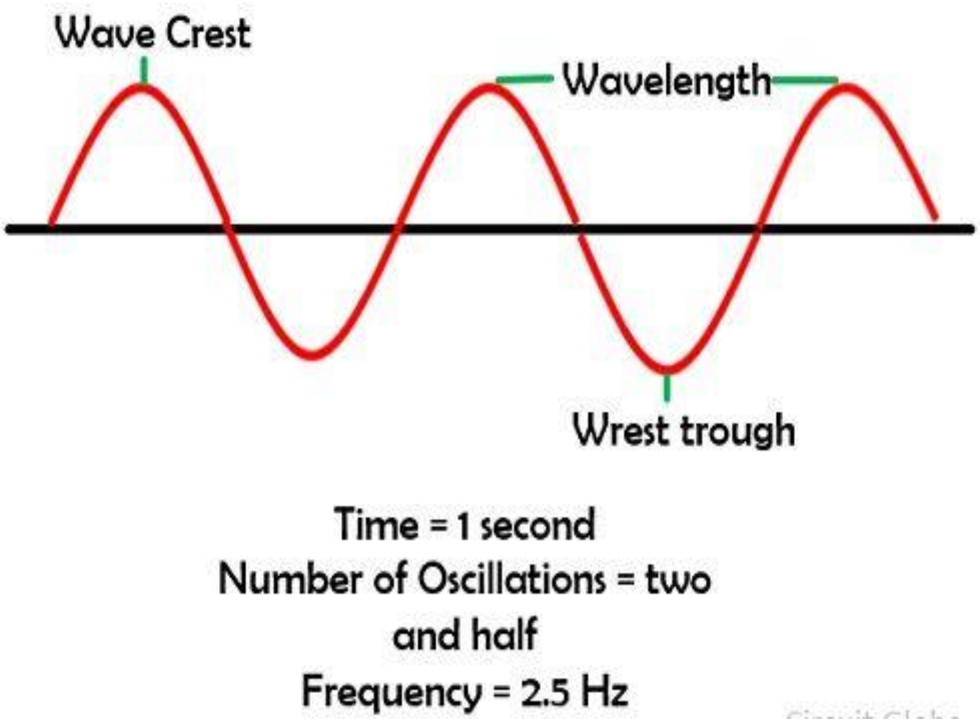
UNIT III MEDIUM ACCESS AND TELECOMMUNICATIONS

Frequencies – Signals – Antennas – Signal propagation – Media Access Control: Motivation, SDMA, FDMA, TDMA, CDMA – GSM: Mobile services, System architecture, Protocols, Localization and calling, Handover – GPRS.

3.1 Frequencies:

The frequency is the number of oscillations per unit time. It is used for defining the cyclic process like rotation, oscillation, wave etc. The completion of the cyclic process at particular interval of time is known as the frequency.

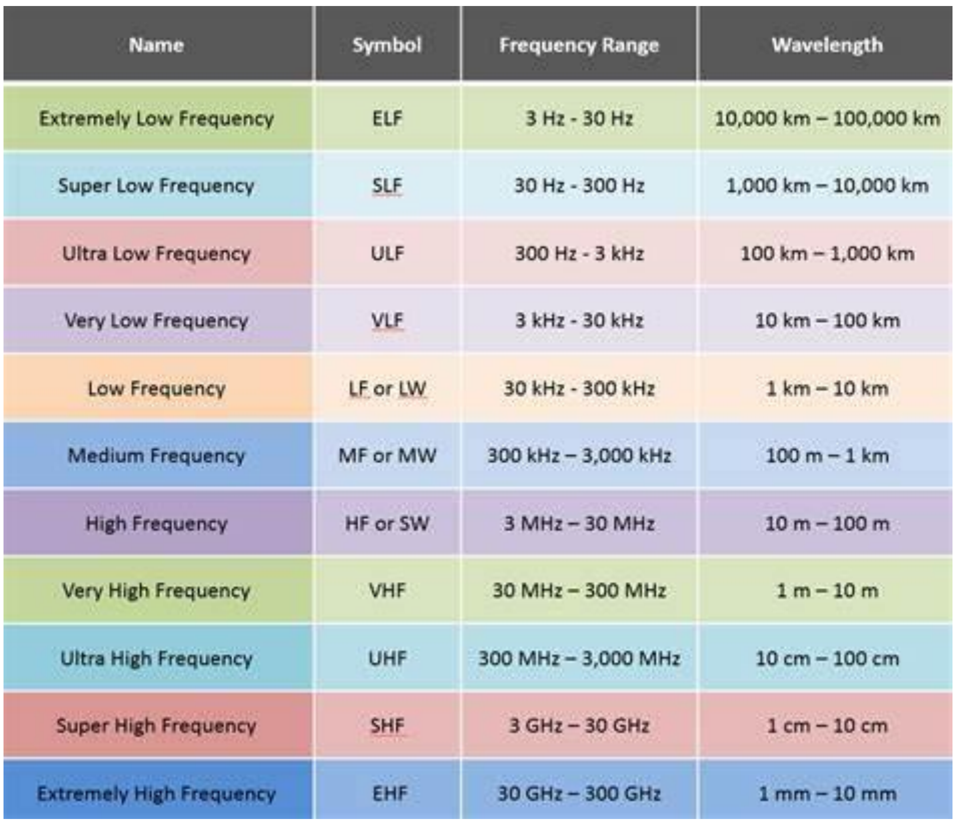
The SI unit of the frequency is Hertz. The symbol λ represents it. The one hertz means the wave completed one cycle in one second. The traditional unit for measuring the cyclic process is revolution per second which is equal to one hertz.



frequency-wave

The frequency is the parameter which explains the phenomenon of oscillatory and vibration like the mechanical vibration, sound signals, light, frequency waves etc. The term “period” represents the time required by the wave for one oscillation, i.e., it is inversely proportional to the frequency.

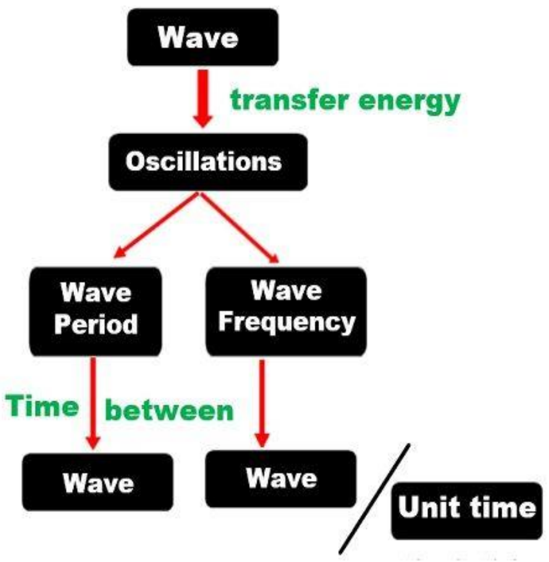
Frequency is the total number of oscillations per unit time. If we take the example of the flashes, then the period is the time between the two flashes. And the frequency is the total number of flashes per second.



Name	Symbol	Frequency Range	Wavelength
Extremely Low Frequency	ELF	3 Hz - 30 Hz	10,000 km – 100,000 km
Super Low Frequency	SLF	30 Hz - 300 Hz	1,000 km – 10,000 km
Ultra Low Frequency	ULF	300 Hz - 3 kHz	100 km – 1,000 km
Very Low Frequency	VLF	3 kHz - 30 kHz	10 km – 100 km
Low Frequency	LF or LW	30 kHz - 300 kHz	1 km – 10 km
Medium Frequency	MF or MW	300 kHz – 3,000 kHz	100 m – 1 km
High Frequency	HF or SW	3 MHz – 30 MHz	10 m – 100 m
Very High Frequency	VHF	30 MHz – 300 MHz	1 m – 10 m
Ultra High Frequency	UHF	300 MHz – 3,000 MHz	10 cm – 100 cm
Super High Frequency	SHF	3 GHz – 30 GHz	1 cm – 10 cm
Extremely High Frequency	EHF	30 GHz – 300 GHz	1 mm – 10 mm

Relation Between Wave δ Frequency

The wave is a kind of disturbance used for transferring the information. The information is transferred in the form of the oscillations. The wave period and the wave frequency are the two phenomena of the oscillations. The wave period is the difference between the wave and the wave frequency is the number of waves per unit's time.



Types of Frequency

The frequency is mainly classified into two categories.

1. Angular Frequency – The angular frequency shows the number of revolution at the fixed interval of time. The unit of angular frequency is Hertz. The relation between the frequency and angular frequency is expressed as;

$$f = \frac{\omega}{2\pi}$$

Where, ω – angular frequency

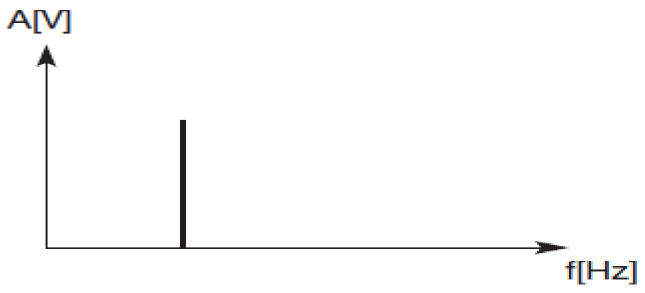
2. Spatial Frequency – The frequency which depends on the spatial coordinate is known as the spatial frequency. It is inversely proportional to the wavelength. The spatial frequency measures the characteristic of the structure that is periodic in space.

3.2 Signals:

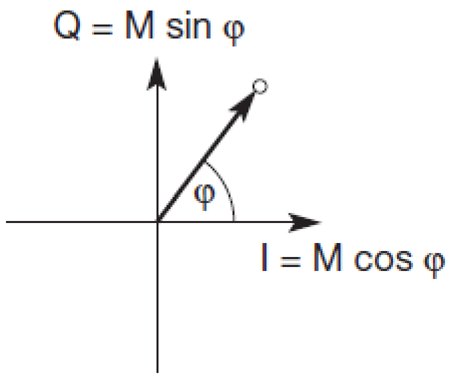
- Signals are the physical representation of data. Users of a communication system can only exchange data through the transmission of signals.
- Layer 1 of the ISO/OSI basic reference model is responsible for the conversion of data, i.e.,bits, into signals and vice versa
- Signals are functions of time and location. Signal parameters represent the data values. The most interesting types of signals for radio transmission are **periodic signals, especially sine waves as carriers.**
- The general function of a sine wave is:

  g(t) = At sin(2 π ft t + φt)
- Signal parameters are the **amplitude** ***A, the frequency f, and the phase shift φ.***
- The amplitude as a factor of the function *g may also change over time, thus At,*
- The frequency *f expresses the periodicity of the signal with* the period *T = 1/f.*
- Finally, the phase shift determines the shift of the signal relative to the same signal without a shift.
- Sine waves are of special interest, as it is possible to construct every periodic signal *g by using only sine and* cosine functions according to a fundamental equation of **Fourier:** ***g(t) = c + Σ an sin(2πnft) + Σ bn cos(2πnft)***

- In this equation the parameter *c determines the **Direct Current (DC) component*** of the signal, the coefficients *an and bn are the amplitudes of the nth* sine and cosine function.
- The equation shows that an infinite number of sine and cosine functions is needed to construct arbitrary periodic functions
- The frequencies of these functions (the so-called **harmonics) increase** with a growing parameter *n and are a multiple of the **fundamental frequency f.***
- The bandwidth of any medium, air, cable, transmitter etc. is limited and, there is an upper limit for the frequencies.
- A typical way to represent signals is the time domain
- Representations in the time domain are problematic if a signal consists of many different frequencies
- In this case, a better representation of a signal is the **frequency domain**
- Following fig shows the amplitude of a certain frequency part of the signal is shown versus the frequency



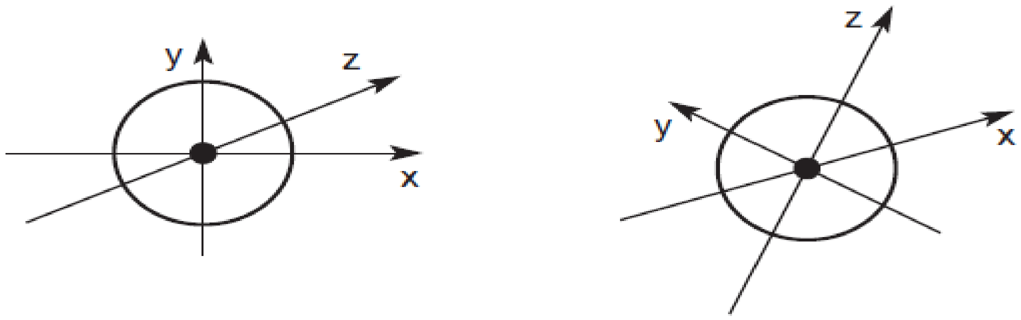
- Fig only shows one peak and the signal consists only of a single frequency part
- Fourier transformations are a mathematical tool for translating from the time domain into the frequency domain and vice versa
- Another way to represent signals is the **phase domain shown in shown in** Figure.



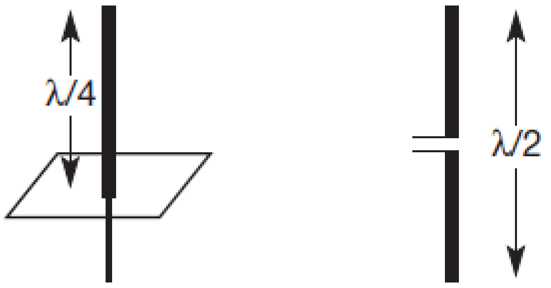
- This representation, also called phase state or signal constellation diagram, shows the amplitude M of a signal and its phase *in polar* coordinates.
- The x-axis represents a phase of 0 and is also called **In-Phase (I). A** phase shift of 90° or /2 would be a point on the y-axis, called **Quadrature.**

3.3 Antennas:

- As the name wireless already indicates, this communication mode involves 'getting rid' of wires and transmitting signals through space without guidance. We do not need any 'medium' (such as an ether) for the transport of electromagnetic waves.
- We have to couple the energy from the transmitter to the out-side world and, in reverse, from the outside world to the receiver. This is exactly what **antennas** do. Antennas couple electromagnetic energy to and from space to and from a wire or coaxial cable.
- A theoretical reference antenna is the **isotropic radiator**, a point in space radiating equal power in all directions.

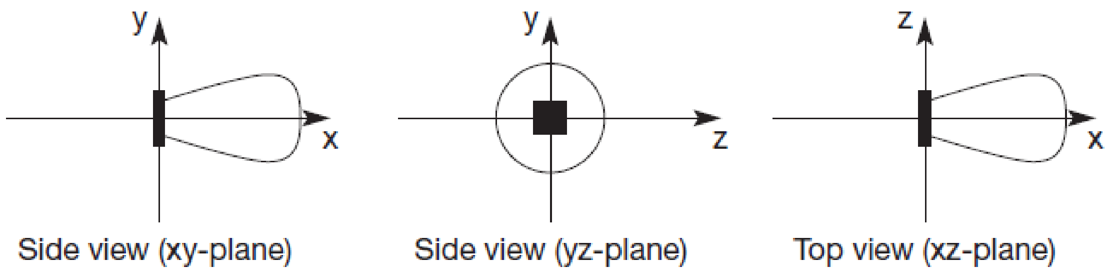


- The **radiation pattern** is symmetric in all direction's antenna does not exist in reality.
- Real antennas all exhibit **directive effects**, i.e., the intensity of radiation is not the same in all directions from the antenna.
- The simplest real antenna is a thin, center-fed **dipole**, also called Hertzian dipole, as shown in Figure.

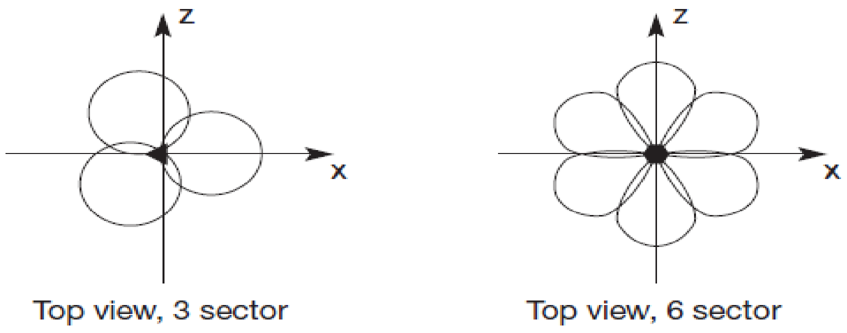


- The dipole consists of two collinear conductors of equal length, separated by a small feeding gap.

- The length of the dipole is not arbitrary.
- A $\lambda/2$ dipole has a uniform or **omni-directional** radiation pattern in one plane
- If an antenna is positioned, e.g., in a valley or between buildings, an omnidirectional radiation pattern is not very useful. In this case, **directional antennas** with certain fixed preferential transmission and reception directions can be used.
- Figure shows the radiation pattern of a directional antenna with the main lobe in the direction of the x-axis.



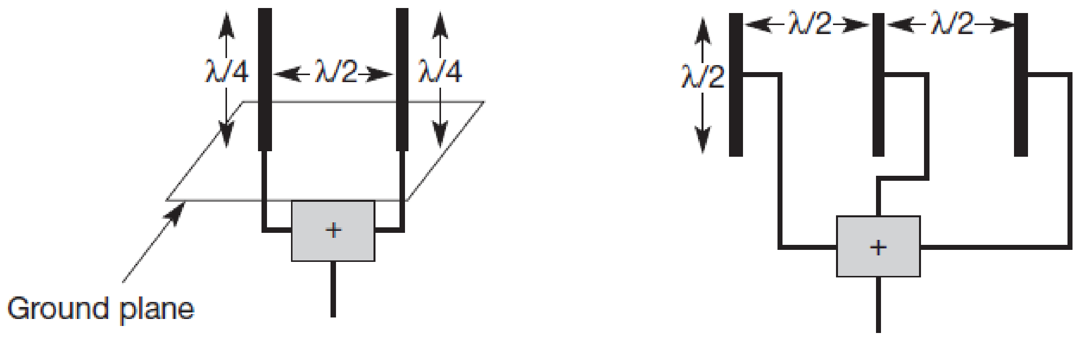
- Directed antennas are typically applied in cellular systems as presented in section
- Several directed antennas can be combined on a single pole to construct a **sectorized antenna**
- A cell can be sectorized into, for example, three or six sectors, thus enabling frequency reuse.



- Two or more antennas can also be combined to improve reception by counteracting the negative effects of multi-path propagation
- These antennas, also called **multi-element antenna arrays**, allow different diversity schemes.
- One such scheme is **switched diversity** or **selection diversity**, where the receiver always uses the antenna element with the largest output.
- **Diversity combining** constitutes a combination of the power of all signals to produce

gain.

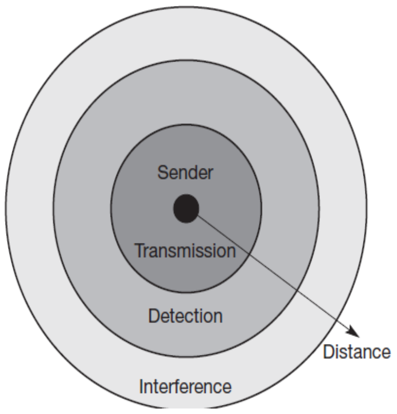
- The phase is first corrected to avoid cancellation.
- As shown in Figure different schemes are possible. On the left, two $\lambda/4$ antennas are combined with a distance of $\lambda/2$ between them on top of a ground plane.



- On the right, three standard $\lambda/2$ dipoles are combined with a distance of $\lambda/2$ between them. Spacing could also be in multiples of $\lambda/2$.
- A more advanced solution is provided by **smart antennas** which combine multiple antenna elements (also called antenna array) with signal processing to optimize the radiation/reception pattern in response to the signal environment.
- These antennas can adapt to changes in reception power, transmission conditions and many signal propagation effects.

3.4 Signal propagation:

- Like wired networks, wireless communication networks also have senders and receivers of signals.
- in connection with signal propagation, these two networks exhibit considerable differences.
- In wireless networks, the signal has no wire to determine the direction of propagation, whereas signals in wired networks only travel along the wire



- As long as the wire is not interrupted or damaged, it typically exhibits the same characteristics at each point.
- One can precisely determine the behavior of a signal travelling along this wire
- For wireless transmission, this predictable behavior is only valid in a Vacuum.
- The situation would be as follows
- **Transmission range**: Within a certain radius of the sender transmission is possible, i.e., a receiver receives the signals with an error rate low enough to be able to communicate and can also act as sender
- **Detection range**: Within a second radius, detection of the transmission is possible, i.e., the transmitted power is large enough to differ from background noise. However, the error rate is too high to establish communication.
- **Interference range**: Within a third even larger radius, the sender may interfere with other transmission by adding to the background noise. A receiver will not be able to detect the signals, but the signals may disturb other signals

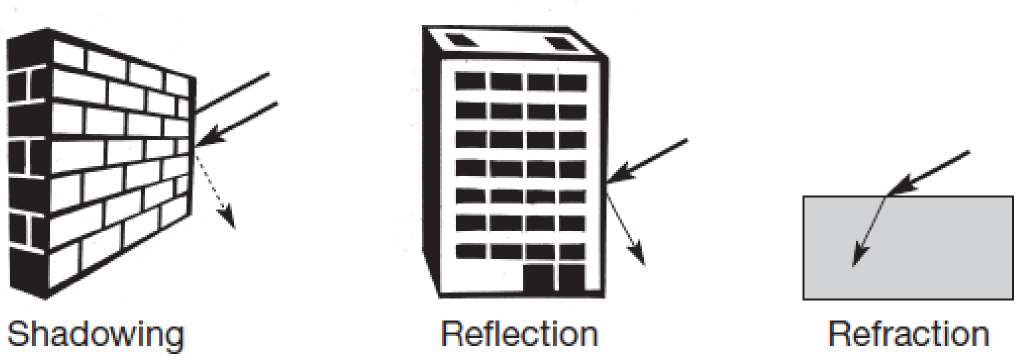
Path loss of radio signals

- In free space radio signals propagate as light does i.e. they follow a straight line
- If such a straight line exists between a sender and a receiver it is called **line-of-sight (LOS)**.
- Even if no matter exists between the sender and the receiver the signal still experiences the **free space loss**.
- The received power Pr is proportional to $1/d2$ with d being the distance between sender and receiver
- The sender now emits a signal with certain energy.
- This signal travels away from the sender at the speed of light as a wave with a spherical shape.
- If there is no delay, the sphere continuously grows
- with the sending energy equally distributed over the sphere’s surface.
- This surface area s grows with the increasing distance d from the center according to the equation $s = 4\pi d2$.
- Radio waves can exhibit three fundamental propagation behaviors depending on their frequency
- Ground wave (<2 MHz): Waves with low frequencies follow the earth’s surface

- Sky wave (2–30 MHz): Many international broadcasts and amateur radio use these short waves that are reflected at the ionosphere.
- Line-of-sight (>30 MHz): Mobile phone systems, satellite systems, cordless telephones etc. use even higher frequencies.

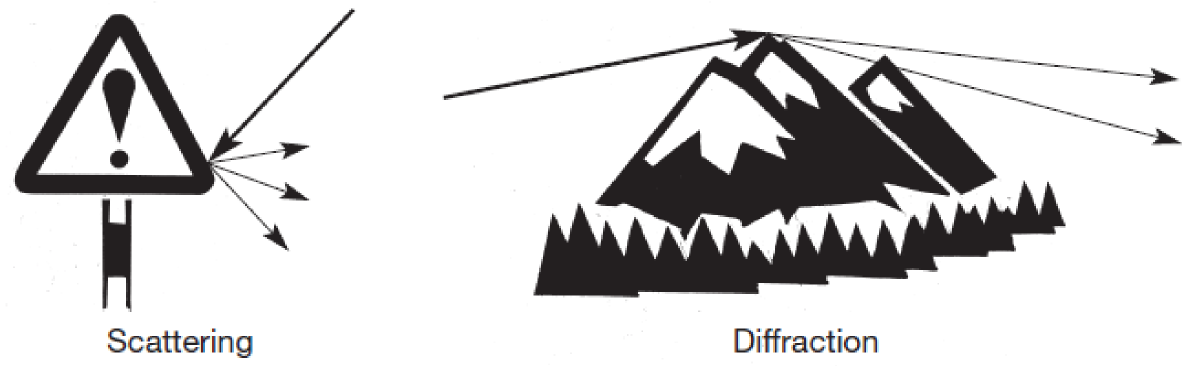
Additional signal propagation effects

- signal propagation in free space almost follows a straight line, like light. But in real life, we rarely have a line-of-sight between the sender and receiver of radio signals
- Mobile phones are typically used in big cities with skyscrapers, on mountains, inside buildings, while driving through an alley etc.
- Hare several effects occur in addition to the attenuation caused by the distance between sender and receiver, which are frequency dependent.
- An extreme form of attenuation is **blocking** or **shadowing** of radio signals due to large obstacles(stand in the middle)
- The higher the frequency of a signal, the more it behaves like light.
- Even small obstacles like a simple wall, a truck on the street, or trees in an alley may block the signal.
- Another effect is the **reflection** of signals as shown in the middle of Figure
- If an object is large compared to the wavelength of the signal, e.g., huge buildings, mountains or the surface of the earth, the signal is reflected.
- The reflected signal is not as strong as the original, as objects can absorb some of the signal's power



- Reflection helps transmitting signals as soon as no LOS exists.
- This is the standard case for radio transmission in cities or mountain areas.

- Signals transmitted from a sender may bounce off the walls of buildings several times before they reach the receiver
- The more often the signal is reflected, the weaker it becomes
- the following two effects exhibit the ‘wave’ character of radio signals.
- If the size of an obstacle is in the order of the wavelength or less, then waves can be **scattered**
- An incoming signal is scattered into several weaker outgoing signals. **diffraction** of waves Radio waves will be deflected at an edge and propagate in different directions
- As shown on the right side of Figure, this effect is very similar to scattering.
- Radio waves will be deflected at an edge and propagate in different directions.
- The result of scattering and diffraction are patterns with varying signal strengths depending on the location of the receiver.

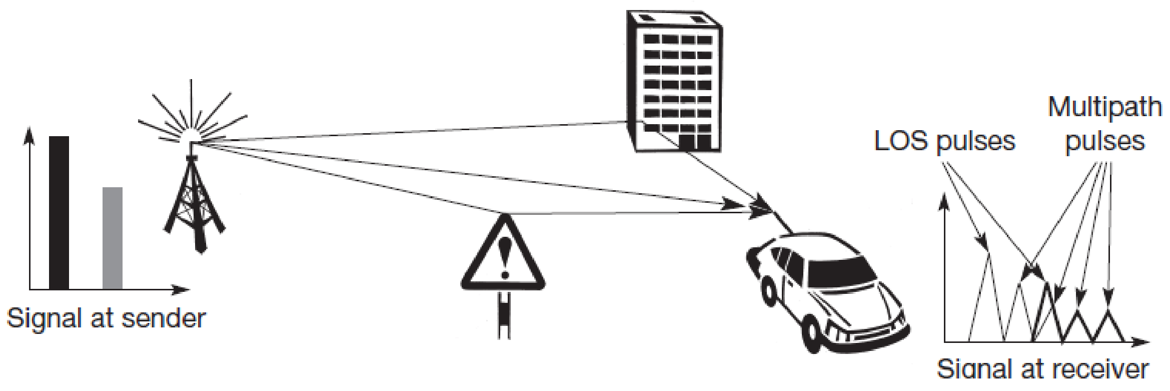


- Effects like attenuation, scattering, diffraction, and refraction all happen simultaneously and are frequency and time dependent

Multi-path propagation

- Together with the direct transmission from a sender to a receiver, the propagation effects mentioned in the previous section lead to one of the most severe radio channel impairments, called **multi-path propagation**.
- Figure shows a sender on the left and one possible receiver on the right
- Radio waves emitted by the sender can either travel along a straight line, or they may be reflected at a large building, or scattered at smaller obstacles.

- This simplified figure only shows **three** possible paths for the signal. In reality, many more paths are possible.



- Due to the finite speed of light, signals travelling along different paths with different lengths arrive at the receiver at different times.
- This effect is called **delay spread**: the original signal is spread due to different delays of parts of the signal.
- This delay spread is a typical effect of radio transmission, because no wire guides the waves along a single path as in the case of wired networks
- Notice that this effect has nothing to do with possible movements of the sender or receiver. Typical values for delay spread are approximately 3 microsec in cities, up to 12 microsec can be observed.

3.5 Medium access control (MAC):

The **Media Access Control** (**MAC**) data communication protocol sub-layer, also known as the Medium Access Control, is a sublayer of the Data Link Layer specified in the seven-layer OSI model (layer 2). The hardware that implements the MAC is referred to as a **Medium Access Controller**. The MAC sub-layer acts as an interface between the Logical Link Control (LLC) sublayer and the network's physical layer. The MAC layer emulates a full-duplex logical communication channel in a multi-point network. This channel may provide unicast, multicast or broadcast communication service.

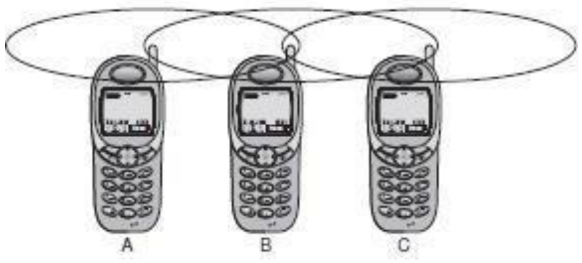
3.5.1 Motivation for a specialized MAC:

One of the most commonly used MAC schemes for wired networks is carrier sense multiple access with collision detection (CSMA/CD). In this scheme, a sender senses the medium (a wire or coaxial cable) to see if it is free. If the medium is busy, the sender waits until it is free. If the medium is free, the sender starts transmitting data and continues to listen into the medium. If the sender detects a collision while sending, it stops at once and sends a jamming signal. But this scheme doest work well with wireless networks. The problems are:

a) Signal strength decreases proportional to the square of the distance
b) The sender would apply CS and CD, but the collisions happen at the receiver
c) It might be a case that a sender cannot “hear” the collision, i.e., CD does not work
d) Furthermore, CS might not work, if for e.g., a terminal is “hidden”

3.5.1.1 Hidden and Exposed Terminals:

Consider the scenario with three mobile phones as shown below. The transmission range of A reaches B, but not C (the detection range does not reach C either). The transmission range of C reaches B, but not A. Finally, the transmission range of B reaches A and C, i.e., A cannot detect C and vice versa.



Hidden terminals

a) A sends to B, C cannot hear A
b) C wants to send to B, C senses a “free” medium (CS fails) and starts transmitting
c) Collision at B occurs, A cannot detect this collision (CD fails) and continues with its transmission to B
d) A is “hidden” from C and vice versa

Exposed terminals

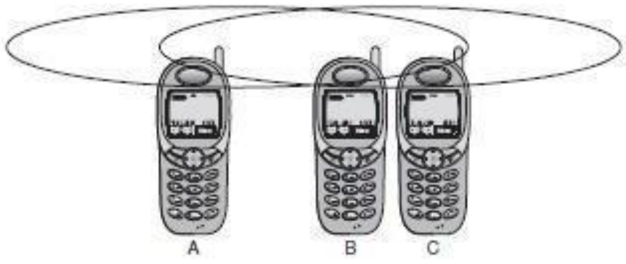
a) B sends to A, C wants to send to another terminal (not A or B) outside the range
b) C senses the carrier and detects that the carrier is busy.
c) C postpones its transmission until it detects the medium as being idle again but A is outside radio range of C, waiting is **not** necessary
d) C is “exposed” to B

Hidden terminals cause collisions, where as Exposed terminals causes unnecessary delay.

3.5.1.2 Near and far terminals:

Consider the situation shown below. A and B are both sending with the same transmission power.

a) Signal strength decreases proportional to the square of the distance
b) So, B’s signal drowns out A’s signal making C unable to receive A’s transmission
c) If C is an arbiter for sending rights, B drown out A’s signal on the physical layer making C unable to hear out A.



The **near/far effect** is a severe problem of wireless networks using CDM. All signals should arrive at the receiver with more or less the same strength for which Precise power control is to be implemented.

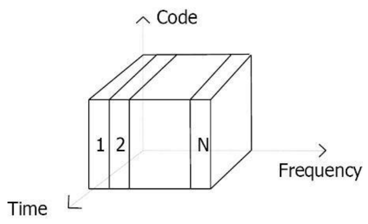
3.5.2 SDMA:

Space Division Multiple Access (SDMA) is used for allocating a separated space to users in wireless networks. A typical application involves assigning an optimal base station to a mobile phone user. The mobile phone may receive several base stations with different quality. A MAC algorithm could now decide which base station is best, taking into account which frequencies (FDM), time slots (TDM) or code (CDM) are still available. The basis for the SDMA algorithm is formed by cells and sectorized antennas which constitute the infrastructure implementing **space division multiplexing (SDM).** SDM has the unique advantage of not requiring any multiplexing equipment. It is usually combined with other multiplexing techniques to better utilize the individual physical channels.

3.5.3 FDMA:

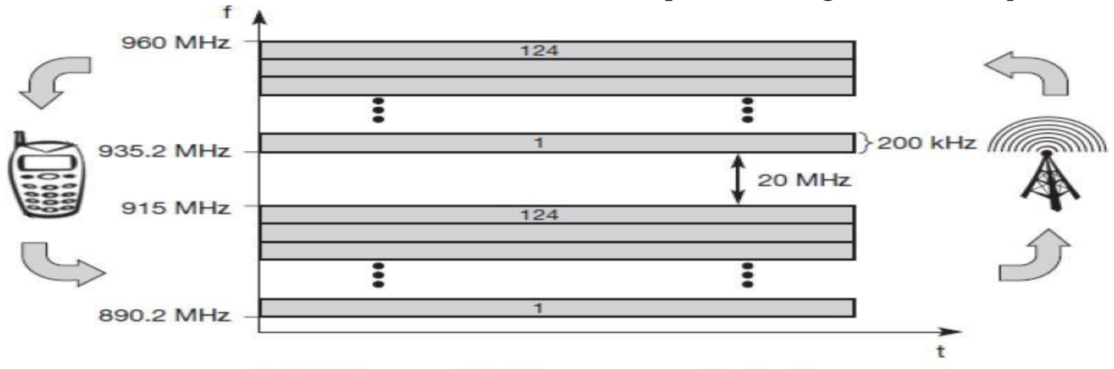
Frequency division multiplexing (FDM) describes schemes to subdivide the frequency dimension into several non-overlapping frequency bands.

Frequency Division Multiple Access is a method employed to permit several users to transmit simultaneously on one satellite transponder by assigning a specific frequency within the channel to each user. Each conversation gets its own, unique, radio channel.



The channels are relatively narrow, usually 30 KHz or less and are defined as either transmit or receive channels. A full duplex conversation requires a transmit & receive channel pair. FDM is often used for simultaneous access to the medium by base station and mobile station in cellular networks establishing a duplex channel.

A scheme called **frequency division duplexing (FDD)** in which the two directions, mobile station to base station and vice versa are now separated using different frequencies.



FDM for multiple access and duplex

FDM for multiple access and duplex

The two frequencies are also known as **uplink**, i.e., from mobile station to base station or from ground control to satellite, and as **downlink**, i.e., from base station to mobile station or from satellite to ground control.

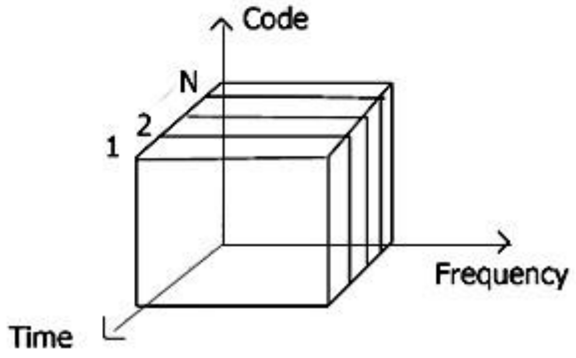
The basic frequency allocation scheme for GSM is fixed and regulated by national authorities. All uplinks use the band between 890.2 and 915 MHz, all downlinks use 935.2 to 960 MHz. According to FDMA, the base station, shown on the right side, allocates a certain frequency for up- and downlink to establish a duplex channel with a mobile phone. Up- and downlink have a fixed relation.

If the uplink frequency is fu = 890 MHz + n·0.2 MHz, the downlink frequency is fd = fu + 45 MHz, i.e**., fd = 935 MHz + n·0.2 MHz** for a certain channel n. The base station selects the channel. Each channel (uplink and downlink) has a bandwidth of 200 kHz.

This scheme also has disadvantages. While radio stations broadcast 24 hours a day, mobile communication typically takes place for only a few minutes at a time. Assigning a separate frequency for each possible communication scenario would be a tremendous waste of (scarce) frequency resources. Additionally, the fixed assignment of a frequency to a sender makes the scheme very inflexible and limits the number of senders.

3.5.4 TDMA:

A more flexible multiplexing scheme for typical mobile communications is time division multiplexing (TDM). Compared to FDMA, time division multiple access (TDMA) offers a much more flexible scheme, which comprises all technologies that allocate certain time slots for communication. Now synchronization between sender and receiver has to be achieved in the time domain. Again this can be done by using a fixed pattern similar to FDMA techniques, i.e., allocating a certain time slot for a channel, or by using a dynamic allocation scheme.

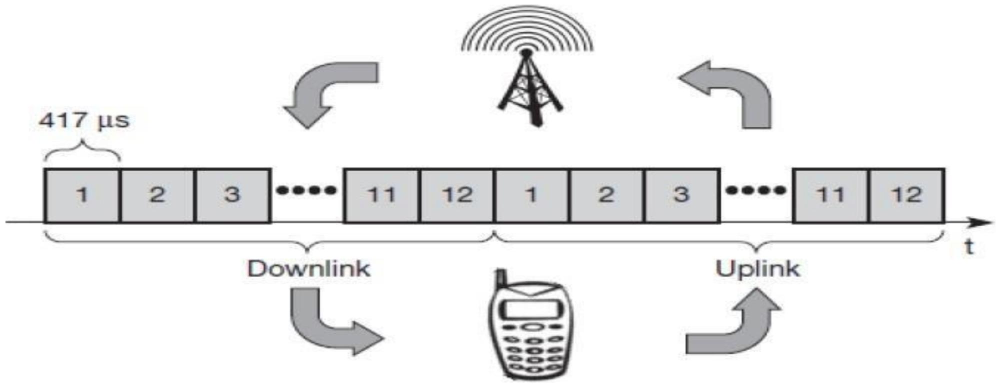


Listening to different frequencies at the same time is quite difficult, but listening to many channels separated in time at the same frequency is simple. Fixed schemes do not need identification, but are not as flexible considering varying bandwidth requirements.

3.5.4.1 Fixed TDM:

The simplest algorithm for using TDM is allocating time slots for channels in a fixed pattern. This results in a fixed bandwidth and is the typical solution for wireless phone systems. MAC is quite simple, as the only crucial factor is accessing the reserved time slot at the right moment. If this synchronization is assured, each mobile station knows its turn and no interference will happen.

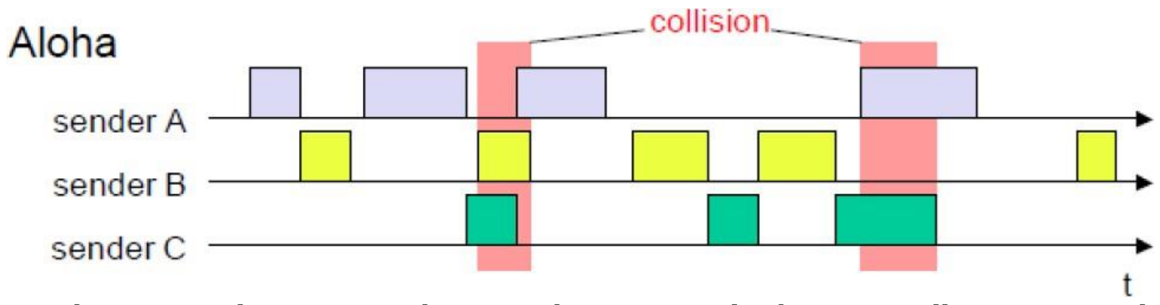
The fixed pattern can be assigned by the base station, where competition between different mobile stations that want to access the medium is solved.



The above figure shows how these fixed TDM patterns are used to implement multiple access and a duplex channel between a base station and mobile station. Assigning different slots for uplink and downlink using the same frequency is called **time division duplex (TDD)**. As shown in the figure, the base station uses one out of 12 slots for the downlink, whereas the mobile station uses one out of 12 different slots for the uplink. Uplink and downlink are separated in time. Up to 12 different mobile stations can use the same frequency without interference using this scheme. Each connection is allotted its own up- and downlink pair. This general scheme still wastes a lot of bandwidth. It is too static, too inflexible for data communication. In this case, connectionless, demand-oriented TDMA schemes can be used.

3.5.4.2 Classical Aloha:

In this scheme, TDM is applied without controlling medium access. Here each station can access the medium at any time as shown below:



This is a random access scheme, without a central arbiter controlling access and without coordination among the stations.

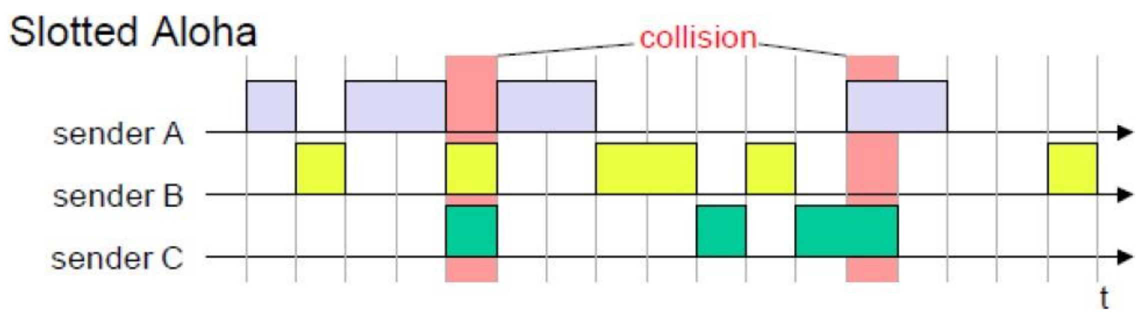
If two or more stations access the medium at the same time, a **collision** occurs and the transmitted data is destroyed.

Resolving this problem is left to higher layers (e.g., retransmission of data). The simple Aloha works fine for a light load and does not require any complicated access mechanisms.

3.5.4.3 Slotted Aloha:

The first refinement of the classical Aloha scheme is provided by the introduction of time slots (**slotted Aloha**). In this case, all senders have to be **synchronized**, transmission can only start at the beginning of a **time slot** as shown below.

The introduction of slots raises the throughput from 18 per cent to 36 per cent, i.e., slotting doubles the throughput. Both basic Aloha principles occur in many systems that implement distributed access to a medium. Aloha systems work perfectly well under a light load, but they cannot give any hard transmission guarantees, such as maximum delay before accessing the medium or minimum throughput.



3.5.4.4 Carrier sense multiple access:

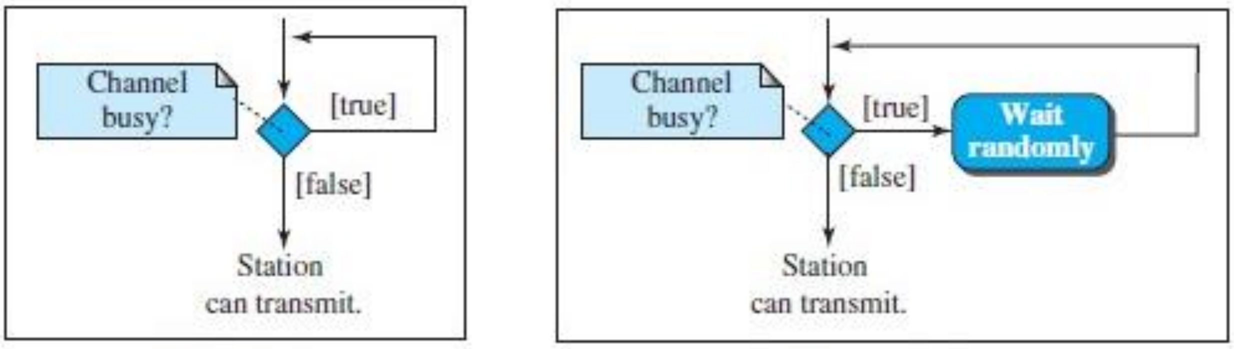
One improvement to the basic Aloha is sensing the carrier before accessing the medium. Sensing the carrier and accessing the medium only if the carrier is idle decreases the probability of a collision. But, as already mentioned in the introduction, hidden terminals cannot be detected, so, if a hidden terminal transmits at the same time as another sender, a collision might occur at the receiver. This basic scheme is still used in most wireless LANs. The different versions of CSMA are:

a) **1-persistent CSMA**: Stations sense the channel and listens if its busy and transmit immediately, when the channel becomes idle. It's called 1-persistent CSMA because the host transmits with a probability of 1 whenever it finds the channel idle.

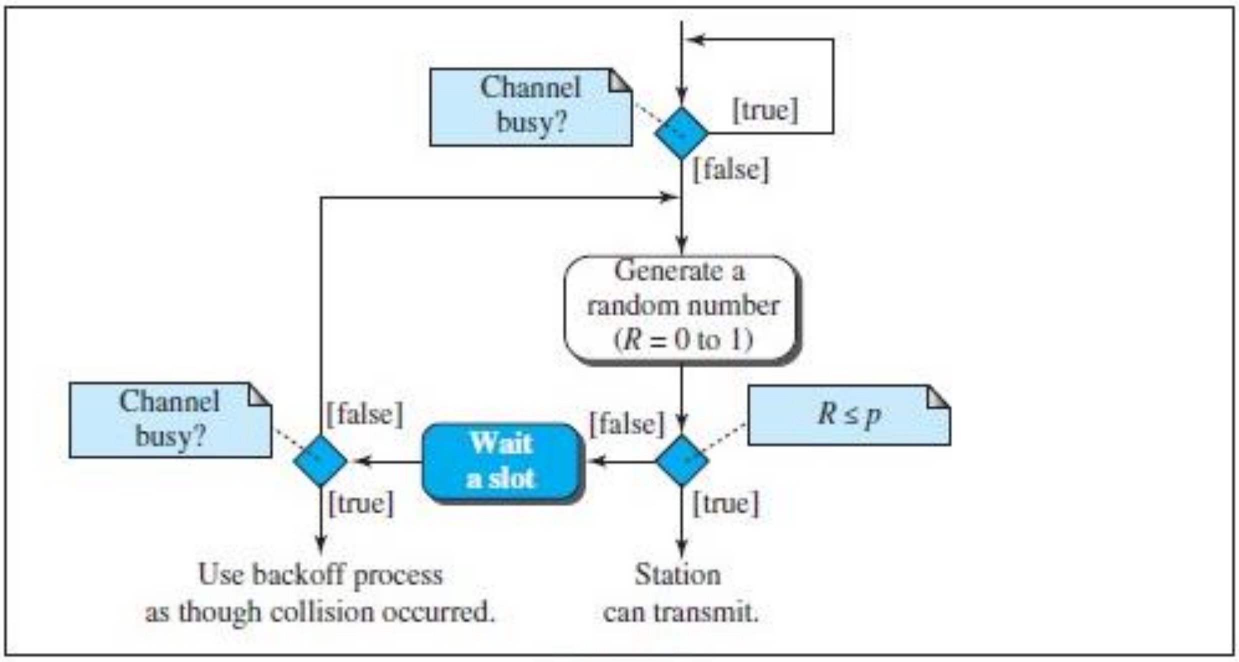
b) **non-persistent CSMA**: stations sense the carrier and start sending immediately if the medium is idle. If the medium is busy, the station pauses a random amount of time before sensing the medium again and repeating this pattern.

c) **p-persistent CSMA**: systems nodes also sense the medium, but only transmit with a probability of p, with the station deferring to the next slot with the probability 1-p, i.e., access is slotted in addition

CSMA with collision avoidance (**CSMA/CA**) is one of the access schemes used in wireless LANs following the standard IEEE 802.11. Here sensing the carrier is combined with a back-off scheme in case of a busy medium to achieve some fairness among competing stations.



a. 1-Persistent b. Nonpersistent



c. *p*-Persistent

3.5.4.5 Reservation TDMA:

In a fixed TDM scheme N mini-slots followed by N·k data-slots form a frame that is repeated. Each station is allotted its own mini-slot and can use it to reserve up to k data-slots.

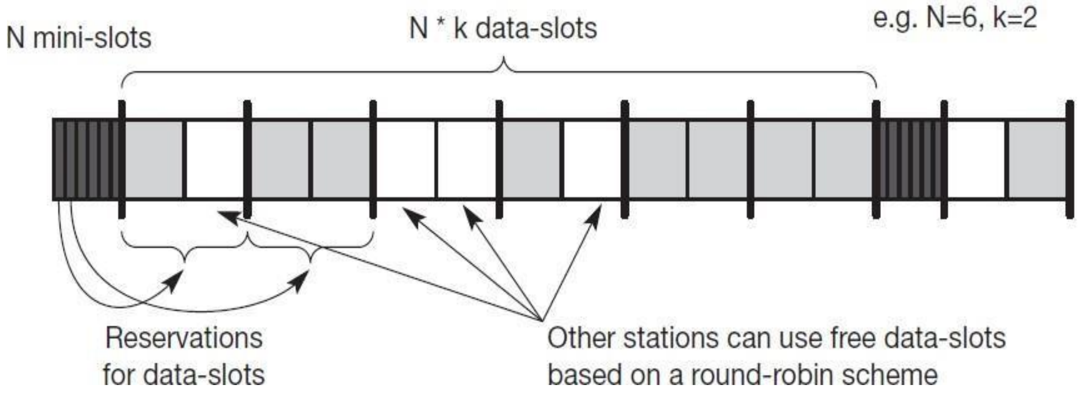

N mini-slots
N * k data-slots
e.g. N=6, k=2
Reservations for data-slots
Other stations can use free data-slots based on a round-robin scheme

Figure 3.9 Reservation TDMA access scheme

This guarantees each station a certain bandwidth and a fixed delay. Other stations can now send data in unused data-slots as shown. Using these free slots can be based on a simple roundrobin scheme or can be uncoordinated using an Aloha scheme.

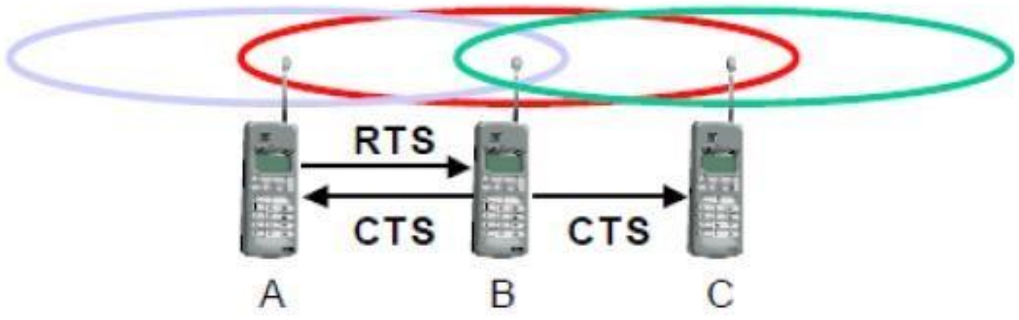
This scheme allows for the combination of, e.g., isochronous traffic with fixed bitrates and best-effort traffic without any guarantees.

3.5.4.6 Multiple access with collision avoidance:

Multiple access with collision avoidance (MACA) presents a simple scheme that solves the hidden terminal problem, does not need a base station, and is still a random access Aloha scheme – but with dynamic reservation. Consider the hidden terminal problem scenario.

A starts sending to B, C does not receive this transmission. C also wants to send something to B and senses the medium. The medium appears to be free, the carrier sense fails. C also starts sending causing a collision at B. But A cannot detect this collision at B and continues with its transmission. A is **hidden** for C and vice versa.

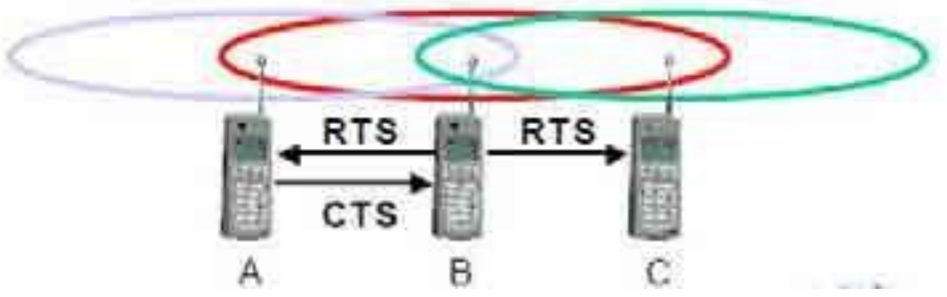
With MACA, A does not start its transmission at once, but sends a **request to send (RTS)** first. B receives the RTS that contains the name of sender and receiver, as well as the length of the future transmission. This RTS is not heard by C, but triggers an acknowledgement from B, called **clear to send (CTS)**. The CTS again contains the names of sender (A) and receiver (B) of the user data, and the length of the future transmission.



This CTS is now heard by C and the medium for future use by A is now reserved for the duration of the transmission. After receiving a CTS, C is not allowed to send anything for the duration indicated in the CTS toward B.

A collision cannot occur at B during data transmission, and the hidden terminal problem is solved. Still collisions might occur when A and C transmits a RTS at the same time. B resolves this contention and acknowledges only one station in the CTS.
No transmission is allowed without an appropriate CTS.

Now MACA tries to avoid the **exposed terminals** in the following way:



With MACA, B has to transmit an RTS first containing the name of the receiver (A) and the sender (B). C does not react to this message as it is not the receiver, but A acknowledges using a CTS which identifies B as the sender and A as the receiver of the following data transmission. C does not receive this CTS and concludes that A is outside the detection range. C can start its transmission assuming it will not cause a collision at A. The problem with exposed terminals is solved without fixed access patterns or a base station.

3.5.4.7 Polling:

Polling schemes are used when one station wants to be heard by others. Polling is a strictly centralized scheme with one master station and several slave stations. The master can poll the slaves according to many schemes: round robin (only efficient if traffic patterns are similar over all stations), randomly, according to reservations (the classroom example with polite students) etc. The master could also establish a list of stations wishing to transmit during a contention phase. After this phase, the station polls each station on the list.

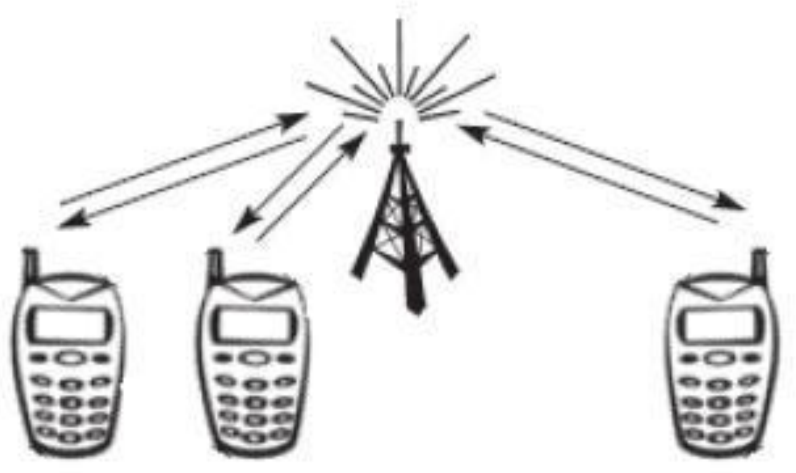
Example: Randomly Addressed Polling

- base station signals readiness to all mobile terminals
- terminals ready to send transmit random number without collision using CDMA or FDMA
- the base station chooses one address for polling from list of all random numbers (collision if two terminals choose the same address)
- the base station acknowledges correct packets and continues polling the next terminal
- this cycle starts again after polling all terminals of the list

3.5.4.8 Inhibit sense multiple access:

This scheme, which is used for the packet data transmission service Cellular Digital

Packet Data (CDPD) in the AMPS mobile phone system, is also known as **digital sense multiple access (DSMA)**. Here, the base station only signals a busy medium via a busy tone (called BUSY/IDLE indicator) on the downlink.



After the busy tone stops, accessing the uplink is not coordinated any further. The base station acknowledges successful transmissions; a mobile station detects a collision only via the missing positive acknowledgement. In case of collisions, additional back-off and retransmission mechanisms are implemented.

3.5.5 CDMA:

Code division multiple access systems apply codes with certain characteristics to the transmission to separate different users in code space and to enable access to a shared medium without interference.

All terminals send on the same frequency probably at the same time and can use the whole bandwidth of the transmission channel. Each sender has a unique random number, the sender XORs the signal with this random number. The receiver can "tune" into this signal if it knows the pseudo random number, tuning is done via a correlation function

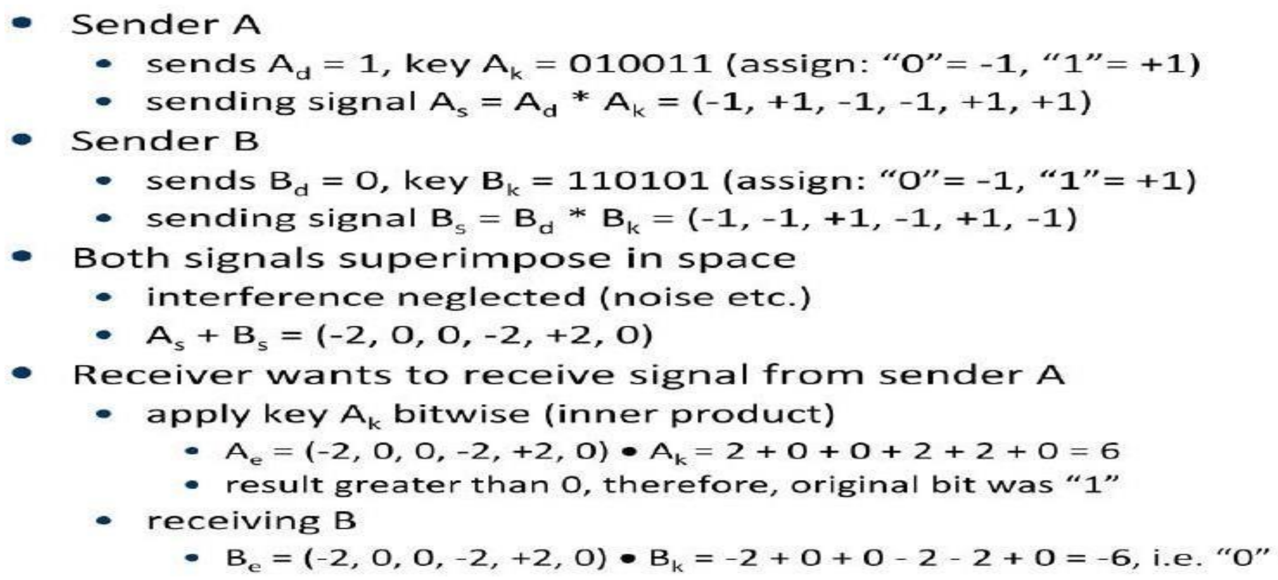
Disadvantages:

1. higher complexity of a receiver (receiver cannot just listen into the medium and start receiving if there is a signal)
2. all signals should have the same strength at a receiver

Advantages:

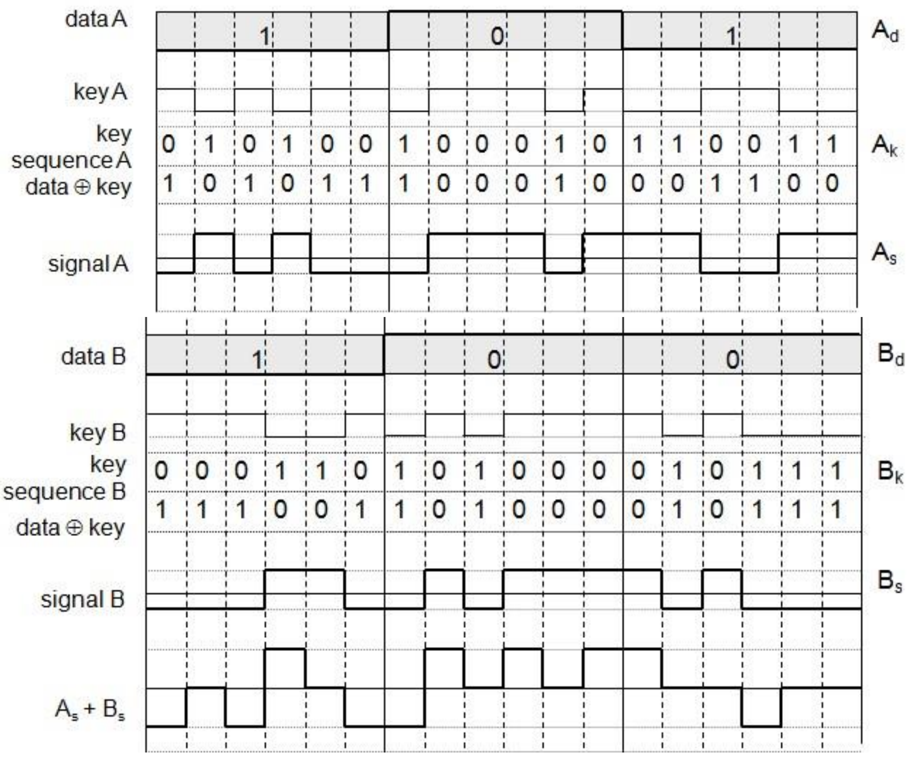
1. all terminals can use the same frequency, no planning needed

2. huge code space (e.g. 232) compared to frequency space
3. interferences (e.g. white noise) is not coded
4. forward error correction and encryption can be easily integrated



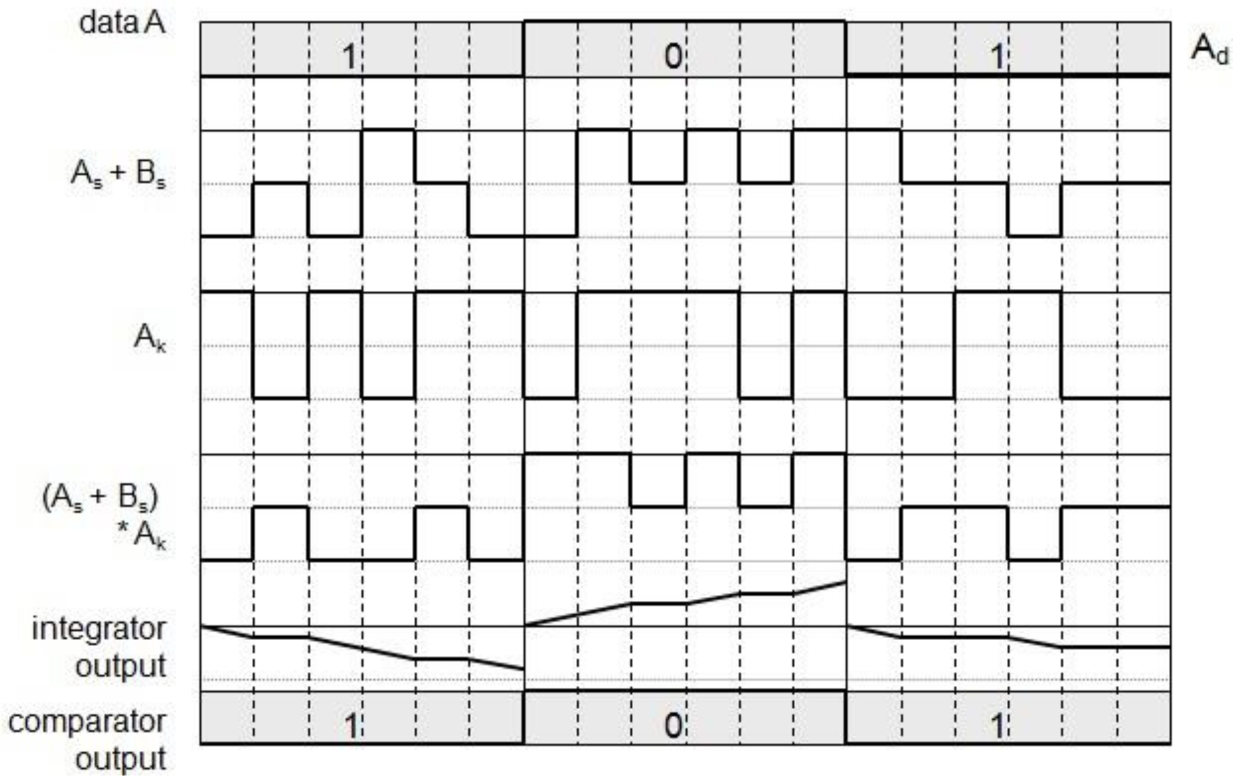
- Sender A
  - sends $A_d = 1$, key $A_k = 010011$ (assign: “0”= -1, “1”= +1)
  - sending signal $A_s = A_d * A_k = (-1, +1, -1, -1, +1, +1)$
- Sender B
  - sends $B_d = 0$, key $B_k = 110101$ (assign: “0”= -1, “1”= +1)
  - sending signal $B_s = B_d * B_k = (-1, -1, +1, -1, +1, -1)$
- Both signals superimpose in space
  - interference neglected (noise etc.)
  - $A_s + B_s = (-2, 0, 0, -2, +2, 0)$
- Receiver wants to receive signal from sender A
  - apply key A_k bitwise (inner product)
    - $A_e = (-2, 0, 0, -2, +2, 0) \bullet A_k = 2 + 0 + 0 + 2 + 2 + 0 = 6$
    - result greater than 0, therefore, original bit was “1”
  - receiving B
    - $B_e = (-2, 0, 0, -2, +2, 0) \bullet B_k = -2 + 0 + 0 - 2 - 2 + 0 = -6$, i.e. “0”

The following figure shows a sender A that wants to transmit the bits 101. The key of A is shown as signal and binary sequence Ak. The binary “0” is assigned a positive signal value, the binary “1” a negative signal value. After spreading, i.e., XORing Ad and Ak, the resulting signal is As.



Coding and spreading of data from sender A and sender B

The same happens with data from sender B with bits 100. The result is Bs. As and Bs now superimpose during transmission. The resulting signal is simply the sum As + Bs as shown above. A now tries to reconstruct the original data from Ad. The receiver applies A’s key, Ak, to the received signal and feeds the result into an integrator. The integrator adds the products, a comparator then has to decide if the result is a 0 or a 1 as shown below. As clearly seen, although the original signal form is distorted by B’s signal, the result is quite clear. The same happens if a receiver wants to receive B’s data.



Reconstruction of A’s data:

Soft handover or **soft handoff** refers to a feature used by the CDMA and WCDMA standards, where a cell phone is simultaneously connected to two or more cells (or cell sectors) during a call. If the sectors are from the same physical cell site (a sectorised site), it is referred to as **softer handoff**. This technique is a form of mobile-assisted handover, for IS-95/CDMA2000 CDMA cell phones continuously make power measurements of a list of neighboring cell sites, and determine whether or not to request or end soft handover with the cell sectors on the list.

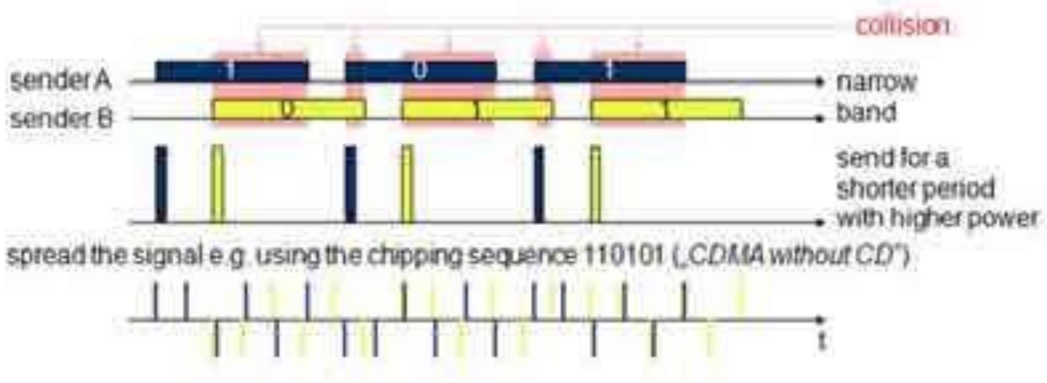
Soft handoff is different from the traditional hard-handoff process. With hard handoff, a definite decision is made on whether to hand off or not. The handoff is initiated and executed without the user attempting to have simultaneous traffic channel communications with the two base stations. With soft handoff, a c*onditional* decision is made on whether to hand off.

Depending on the changes in pilot signal strength from the two or more base stations involved, a hard decision will eventually be made to communicate with only one. This normally happens after it is evident that the signal from one base station is considerably stronger than those from the others. In the interim period, the user has simultaneous traffic channel communication with all candidate base stations. It is desirable to implement soft handoff in power-controlled CDMA systems because implementing hard handoff is potentially difficult in such systems.

3.5.5.1 Spread Aloha multiple access (SAMA):

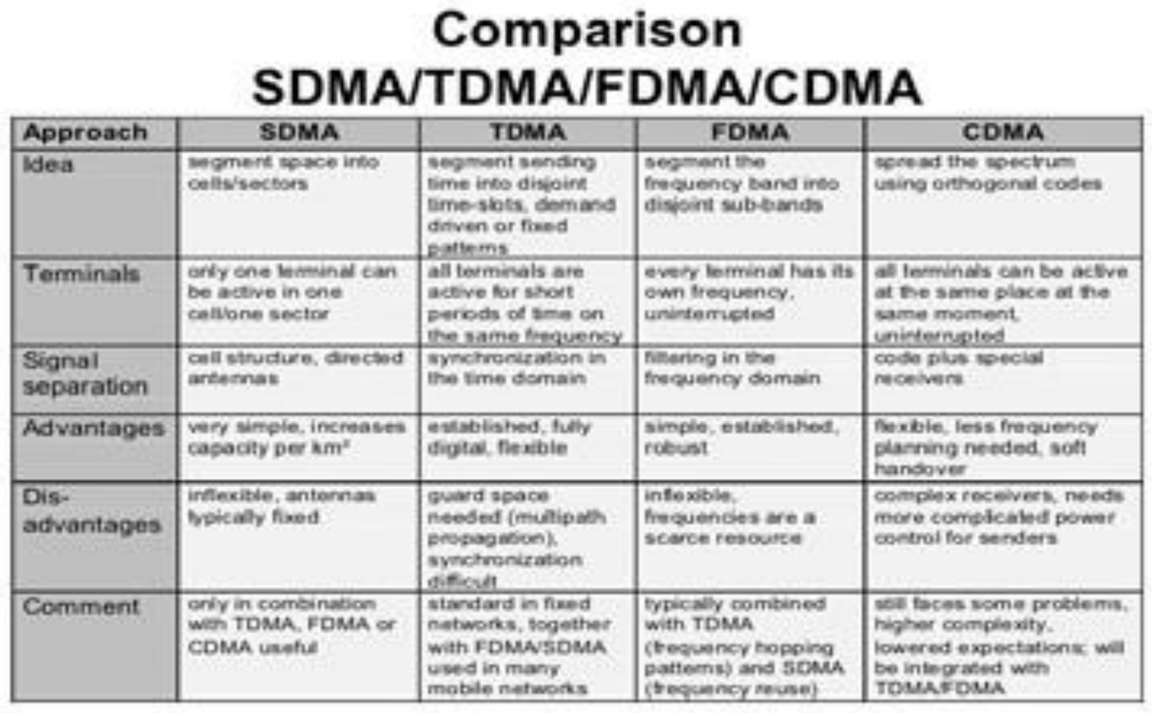
CDMA senders and receivers are not really simple devices. Communicating with *n* devices requires programming of the receiver to be able to decode *n* different codes. Aloha was a very simple scheme, but could only provide a relatively low bandwidth due to collisions. SAMA uses spread spectrum with only one single code (chipping sequence) for spreading for all senders accessing according to aloha.

In SAMA, each sender uses the same spreading code, for ex 110101 as shown below. Sender A and B access the medium at the same time in their narrowband spectrum, so that the three bits shown causes collisions. The same data could also be sent with higher power for shorter periods as show.



collision
sender A
sender B
1
0
1
0
1
1
narrow band
send for a shorter period with higher power
spread the signal e.g. using the chipping sequence 110101 („CDMA without CD")
t

The main problem in using this approach is finding good chipping sequences. The maximum throughput is about 18 per cent, which is very similar to Aloha, but the approach benefits from the advantages of spread spectrum techniques: robustness against narrowband interference and simple coexistence with other systems in the same frequency bands.


Comparison
SDMA/TDMA/FDMA/CDMA

Approach	SDMA	TDMA	FDMA	CDMA
Idea	segment space into cells/sectors	segment sending time into disjoint time-slots, demand driven or fixed patterns	segment the frequency band into disjoint sub-bands	spread the spectrum using orthogonal codes
Terminals	only one terminal can be active in one cell/one sector	all terminals are active for short periods of time on the same frequency	every terminal has its own frequency, uninterrupted	all terminals can be active at the same place at the same moment, uninterrupted
Signal separation	cell structure, directed antennas	synchronization in the time domain	filtering in the frequency domain	code plus special receivers
Advantages	very simple, increases capacity per km²	established, fully digital, flexible	simple, established, robust	flexible, less frequency planning needed, soft handover
Dis-advantages	inflexible, antennas typically fixed	guard space needed (multipath propagation), synchronization difficult	inflexible, frequencies are a scarce resource	complex receivers, needs more complicated power control for senders
Comment	only in combination with TDMA, FDMA or CDMA useful	standard in fixed networks, together with FDMA/SDMA used in many mobile networks	typically combined with TDMA (frequency hopping patterns) and SDMA (frequency reuse)	still faces some problems, higher complexity, lowered expectations; will be integrated with TDMA/FDMA

3.6 GSM:

INTRODUCTION

GSM was formally known as Groupe Speciale Mobile (found in1982) and now it is abbreviated as Global System for mobile communications. It is a standard set developed by the European Telecommunications Standards Institute (ETSI) to describe protocols for second generation (2G) digital cellular networks used by mobile phones. It became the de facto global standard for mobile communications with over 80% market share. The GSM standard was developed as a replacement for first generation (1G) analog cellular networks, and originally described a digital, circuit switched network optimized for full duplex voice telephony. Further improvements were made when the 3GPP developed third generation (3G) UMTS standards followed by fourth generation (4G) LTE Advanced standards.

The primary goal of GSM was to provide a mobile phone system that allows user to roam throughout Europe and provides voice services compatible to ISDN and other PSTN systems. GSM has initially been deployed in Europe using 890-915MHz for uplinks and 935-960 for downlinks.

- **GSM 1800:** Otherwise called as Digital cellular Systems DCS 1800
  Uplink: 1710 to 1785 MHz
  Downlink: 1805 to 1880 MHz
- **GSM 1900:** Otherwise called as Personal Communication Service PLS 1900

Uplink: 1850 to 1910 MHz

Downlink: 1930 to 1990 MHz

- **GSM 400:**

  Uplink: 450.4 to 478 MHz

  Downlink: 460 to 496 MHz

- GSM Rail is used in European Countries and railroad systems.

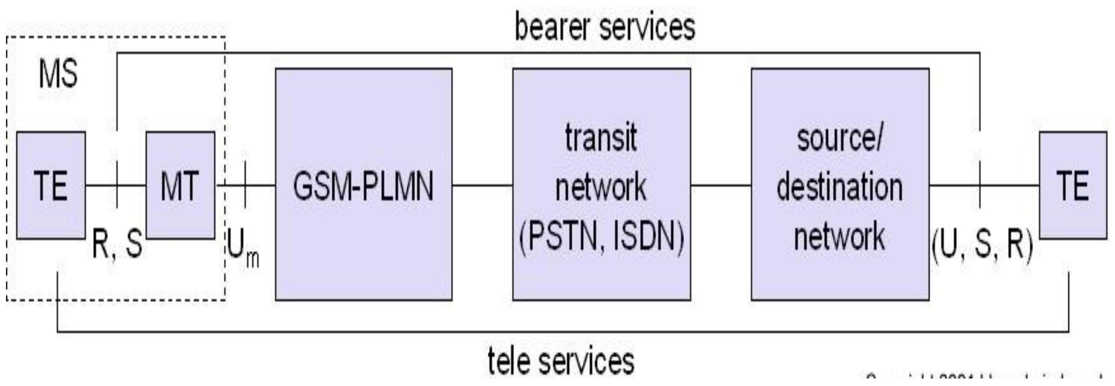
FEATURES OF GSM RAIL

It offers 19 exclusive channels for voice and data traffic. The special features like emergency calls, voice group call service etc. are available. Calls are prioritized. Mostly used to control the trains, switches, signals, gates.

GSM SERVICES

- GSM allows the integration of voice and data services and also the internetworking with the existing network.
- There are three types of services offered by GSM
  - ➢ Bearer Service
  - ➢ Tele Service
  - ➢ Supplementary Service

REFERENCE MODEL FOR GSM SERVICES



GSM services

EXPLANATION:

A mobile station MS is connected to the GSM public land mobile network PLMN via Um interface. PLMN is the infrastructure needed for the GSM networks. This network is connected to transit networks (eg.) PSTN, ISTN, etc. There will be additional network the source/ destinations network before another terminal TE is connected.

BEARER SERVICES:

Bearer Services comprises of all the services that enables the transparent transmission of data between the interface to the network. It permits transparent/non transparent, synchronous and asynchronous data transmission.

- **TRANSPARENT BEARER SERVICES:**

  This services uses the functions of physical layer to transmit data. Data transmission has a constant delays and throughout if no error occurs but not in real time. FEC is used to increase the transmission quality. It does not try to recover the lost data in case of handover.
- **NON TRANSPARENT BEARER SERVICES**:

  It uses the protocols of layers data link and network to transmit data. These services uses transparent bearer service radio link protocol (RLP). RLP has mechanisms of high level data link control HDLC. It allows retransmission of erroneous data by using elective reject mechanisms.

TELE SERVCIES:

Tele services are application specific and need all the 7 layers of ISO/OSI reference model. Services are specified end to end. There tele services are voice oriented tele service. They are encrypted voice transmission, message services and data communication with terminals from PSTN/ISDN.

There are some important services:

- **Telephony services:** It has high quality digital voice transmission.
- **Emergency Number:** Mandatory service for all service providers. Its of free of charge. This connection has the highest priority with pre-emption.
- **Short Message Service:** It is used from simple message transfer with the maximum of 160 characters. SMS does not use the standard data channels of GSM uses the signaling channels. Sending and g receiving SMS is possible during the data/ voice transmission.
- **Enhanced Message Service:** It is used for large message size with 760 characters, animated pictures, small images can be transmitted.

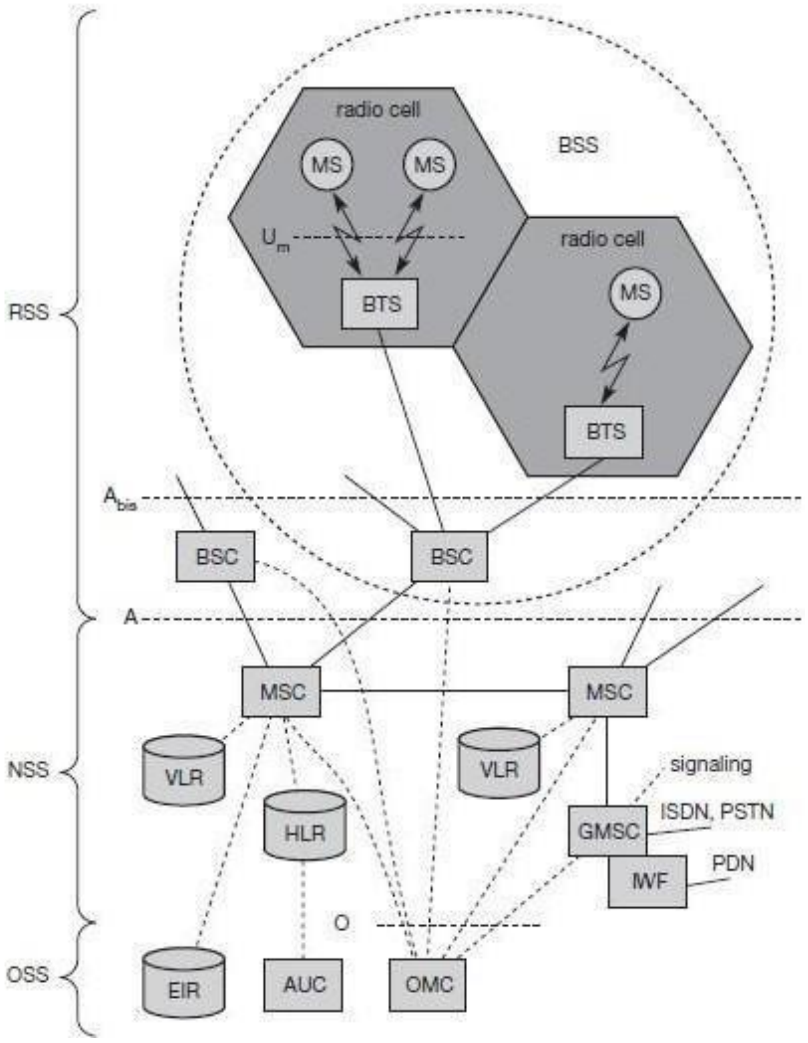
- **Multimedia Message Service:** It is used to transmit large pictures of GIF/ JPEG, video clips.
- **Group 3 fax:** Fax data is transmitted as digital data over analog telephone network using modem.

SUPPLEMENTARY SERVICES:

- User Identification
- Call Redirecting / Forwarding
- Closed User Group
- Multiparty communication

GSM ARCHITECTURE

The architecture of GSM comes in hierarchy, consisting of many entities, interfaces and subsystems.



GSM Architecture

The GSM system consist of three subsystems namely,

- The Radio Subsystems(RSS)
- Network and Switching Subsystems(NSS)
- Operation Subsystem(OSS)

The customer is able to notice few components of the network viz. Mobile Station and Antenna of the Base Transceiver Station(BTS). Remaining entities are not visible.

1. RADIO SUBSYSTEM:

As the name implies, the radio subsystem (RSS) comprises all radio specific entities. i.e. the mobile stations(MS) and the base station subsystem(BSS).

As they are in same frequency they form a cell. The components of RSS are

- Mobile station
- Base Transceiver Station
- Base Station Subsystem
- Base Station Controller

MOBILE STATION: (MS)

MS has all user equipment and software needed for mobile communications. It has user independent hardware and software. Subscriber Identity Module (SIM) stores all user specific data. Mobile Station can be identified as International Mobile equipment identity (IMEI). The sim card constitutes of Personal Identity number(PIN), PIN unlocking key(PUK), Authentication key k, International Mobile subscriber identity (IMSI). It also has Identifiers and tables. The current location of MS is found using Temporary
Mobile Subscriber Identity(TMSI) . With TMSI and Location Area Identification (LAI) the current location can be identified.

BASE TRNSCEIVER STATION: (BTS)

BTS contains the equipment for transmitting and receiving of radio signals,antennas and equipment for encrypting and decrypting communications with the Base station controller(BSC). A BTS is controlled by ap parent BSC via the base station control function(BSCF). The BCF is implemented as a discrete unit or even incorporated in a TRX in compact base stations. The BCF provides an operations and maintenance (O&M) connection to the Network Management System(NMS)and manage operational state of each TRX as well as soft handling and alarm collection.

The function of BTS vary depending on the cellular technology used and cellula telephone provider. There are vendors in which the BTS is a plain transceiver which receives information from MS through Um(Air Interface) and then it converts into TDM based interface, the Abis and it sends it towards the BSC. A GSM cell can measure between some 100m and 35km depending on the environment.

BASE STATION SUBSYSTEM: (BSS)

The base station subsystem is the section of traditional cellular telephone network which is responsible for handling traffic and signaling between a mobile phone and the network switching subsystem. The BSS carries out transcoding of speech channels, allocation of radio channels to mobile phones, paging, quality management of transmission and reception over the Air interface and many other tasks related to the radio network.

BASE STATION CONTROLLER : (BSC)

The BSC basically manages the BTSs. It reverses radio frequencies and handles the handover from one BTS to another within BSS, and performs paging of the MS. The BSC also multiplexes the radio channels onto the fixed network connections at A interface.

2. NETWORK AND SWITCHING SUBSYSTEM:

This network and switching subsystem is the heart of GSM. Their function are to connect wireless network with standard public network, performs handover between different BSS, localization (to locate the mobile station),Charging, Accounting and roaming of users.The NSS contains the following switches and databases.

MOBILE SERVIES SWITCHING CENTER(MSC):

MSC are digital ISDN switches. It establishes connection qith other MSC and BSC via A interfacet Gateway MSC connects to fixed networks(eg.) PSTN, ISDN. With the help of Internet Working Functions, MSC can connect to public data Network PDN. It handles all signaling needed for connection setup, connection release and handover.

HOME LOCATION REGISTER : (HLR)

It is an important database. It stores user relevant information and also has static information and dynamic information.

Static Information:

Mobile subscriber ISDN number is available. It has subscribed services for a particular number. It is also an international mobile subscriber identity.

Dynamic Information:

It is a current location area(LA) of MS. It consist of Mobile subscriber roaming number (MSRN), VLR and MSC in it. When MS leaves the current LA, then the information is updated in HLR. The Usage of the information is to locate the user.

VISITOR LOCATION REGISTER: (VLR)

The VLR is associated to each MSC. It is a dynamic database. It stores all the information needed for the MS currently in LA. If new MS comes to LA then the VLR is responsible for copying the information needed from HLR**.**

3. OPERATION SUBSYSTEM:

The third part of a GSM system is operation subsystem (OSS) which contains the necessary functions for network operation and maintenance. The OSS possesses network entities of its own and accesses other entities via SS7 signaling.
The following entities have been defined:

OPERATION AND MAINTENANCE CENTER (OMC):

The OMC monitors and controls all other network entities via the O interface (SS7 with X.25). Typical OMC management functions are traffic monitoring, status reports of network entities, subscriber and security management, or accounting and billing. OMCs use the concept of telecommunication management network (TMN) as standardized by the ITU-T. Authentication centre (AuC): As the radio interface and mobile stations are particularly vulnerable, a separate AuC has been defined to protect user identity and data transmission. The AuC contains the algorithms for authentication as well as the keys for encryption and generates the values needed for user authentication in the HLR.

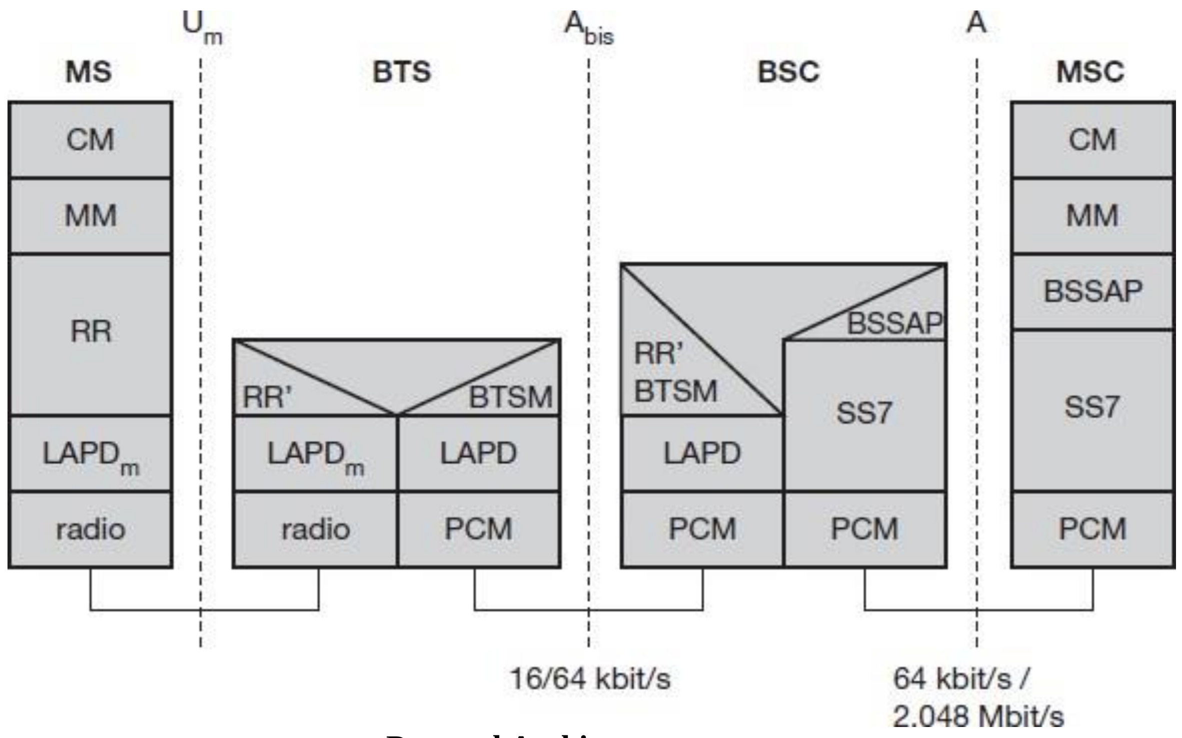
EQUIPMENT IDENTITY REGISTER (EIR):

The EIR is a database for all IMEIs, i.e., it stores all device identifications registered for this network. As MSs are mobile, they can be easily stolen. With a valid SIM, anyone could use the stolen MS. The EIR has a blacklist of stolen (or locked) devices. In theory an

MS is useless as soon as the owner has reported a theft. Unfortunately, the blacklists of different providers are not usually synchronized and the illegal use of a device in another operator‟s network is possible (the reader may speculate as to why this is the case). The EIR also contains a list of valid IMEIs (white list), and a list of malfunctioning devices (gray list).

GSM PROTOCOL SUITES:

The protocol architecture of GSM is shown below:



Protocol Architecture

The layers are

PHYSICAL LAYER:

The physical layer handles all radio specific functions.

Functions:

1. Creation of burst in any one of 5 format.
2. Multiplexing burst into a TDMA frame.
3. Synchronization with BTS.
4. Detection of idle channel.
5. Channel Quality measurement.
6. Channel coding and error detection and correction.

The Um interfaces use GMSK for modulation and perform encryption and decryption.

LAYER 2:

For signaling between entitites in GSM network this layer is used. The protocol used is LAPDM. LAPD stands for link access procedure for D channel. LAPDM has no buffers has to follow Um interface patterns. The functions of the layer are namely:

1. Reliable data transfer
2. Reseqeuncing of data
3. Flow control

LAYER 3: NETWORK LAYER

The network layer has sublayers. They are,

RADIO RESOURCE MANAGEMENT:

This is the lowest sub layer and it's a part of RR and RR' is implemented by BSC. The function of RR are Setup, Maintenance, Release of radio channels. RR directly access the physical layer. It supports BTS management. The function of RR' are supported by BSC via BSTM.

MOBILITY MANAGEMENT:

The main function of Mobility management are Registration, Authentication, Identification, Location Updating, Providing TMSI, IMSI.

LAYER 4: CALL MANAGEMENT:

This layer contains three entities. They are Call control, SMS, Supplementary services. Call control provides point to point connection between two terminals and also used for call clearance, change of call parameters. SMS allows messages transfer using control channels. The supplementary services discussed already is to be reproduced here.

CONNECTION ESTABLISHMENT:

The number dialed to reach a mobile subscriber (MSISDN) contains no information at all about the current location of the subscriber. In order to establish a complete connection to a mobile subscriber, however, one must determine the current location and the locally responsible switch (MSC). In order to be able to route the call to this switch, the routing address to this subscriber (MSRN) has to be obtained. This routing address is assigned temporarily to a subscriber by its currently associated VLR. At the arrival of a call at the GMSC, the HLR is the only entity in the GSM network which can supply this information, and therefore it must be interrogated for each connection setup to a mobile subscriber. An ISDN switch recognizes from the MSISDN that the called subscriber is a mobile subscriber, and therefore can forward the call to the GMSC of the subscriber's home PLMN based on the CC and NDC in the MSISDN. This GMSC can now request the current routing address (MSRN) for the mobile subscriber from the HLR using the MAP . By way of the MSRN the call is forwarded to the local MSC, which determines the TMSI of the subscriber and initiates the paging procedure in the relevant location area . After the MS has responded to the paging call, the connection can be switched through.Several variants for determining the route and interrogating the HLR exist, depending on how the MSRN was assigned and stored, whether the call is national or international and depending on the capabilities of the associated switching centers.

To locate a MS and to address the MS, several numbers are needed:

MOBILE STATION INTERNATIONAL ISDN NUMBER (MSISDN):

The only important number from the user of GSM is phone number. Remember that the phone number is not associated with certain device but with the SIM, which is personalized for user. The number consist of country code(CC) as eg.+49 179 1234567 with 49 for germany. National Destination Code (NDC) is used to locate the network provider and Subscriber number.

INTERNATIONAL MOBILE SUBSCRIBER IDENTITY(IMSI):

GSM uses the IMSI for internal unique identification of a subscriber. IMSI consist of a mobile country code (MCC) and mobile network code(MNC) and finally the mobile subscriber identification number(MSIN).

TEMPORARY MOBILE SUBSCRIBE IDENTITY(TMSI):

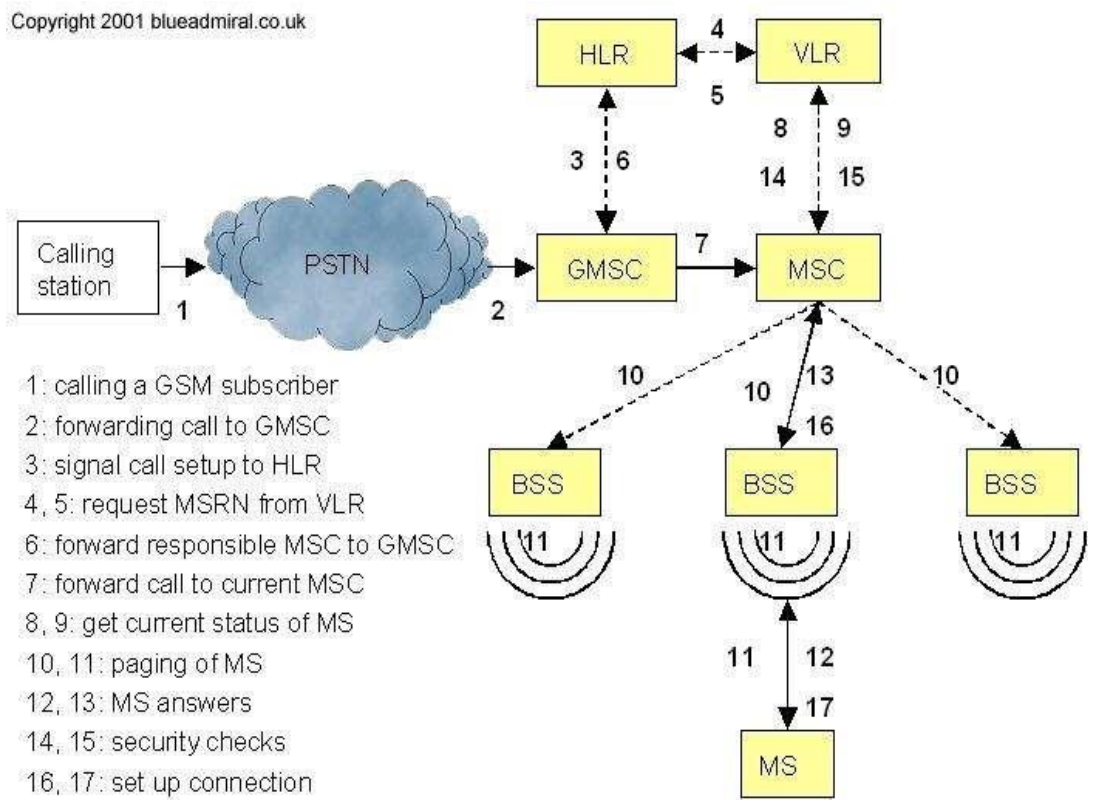
To hide the IMSI, which would give away the exact identity of user signaling over the air interface, GSM uses the 4 byte.TMSI is selected by current VLR and is only valid temporarily and within location area of VLR.

MOBILE STATION ROAMING NUMBER (MSRM):

Another temporary address which hides the identity and location of a subscriber is MSRN. The VLR generates this address on request from MSC, and the address is stored in the HLR. MSRN contains current visitory and visitor national destination code(VNDC).

MESSAGE TERMINATED CALL (MTC):

This figure shows the basic steps needed to connect the calling station with the mobile user.



Copyright 2001 blueadmiral.co.uk
HLR
VLR
Calling station
PSTN
GMSC
MSC
BSS
BSS
BSS
MS
1: calling a GSM subscriber
2: forwarding call to GMSC
3: signal call setup to HLR
4, 5: request MSRN from VLR
6: forward responsible MSC to GMSC
7: forward call to current MSC
8, 9: get current status of MS
10, 11: paging of MS
12, 13: MS answers
14, 15: security checks
16, 17: set up connection

. **Message Terminated Call (MTC)**

Step 1: A user dials the phone number of GSM subscriber. The fixed network PSTN notices that the number belongs to the user in the GSM network and forwards the call setup to the Gateway MSC.

Step 2: The GMSC identifies the HLR for the subscriber and signals the call setup to the HLR.

Step 3: The HLR now checks whether the number exists and whether the user has subscribed to the requested services, and the requests an MSRN from the current VLR.

Step 4: After receiving the request from MSRN

Step 5: HLR can determine the MSC responsible for the MS and forwards this information to GMSC

Step 6: GMSC can now forward the call setup request to MSC indicated.

Step 7: From this, MSC is responsible for all further steps. First it requests the current status of MS from VLR.

Step 8: If MS is available, then MSC initiates paging in all cells it is responsible for as searching for the right cell would be too time consuming.

Step 9: This approach puts some load on signalling channels so optimization exist.

Step 10: Location area (LA) can be determined.

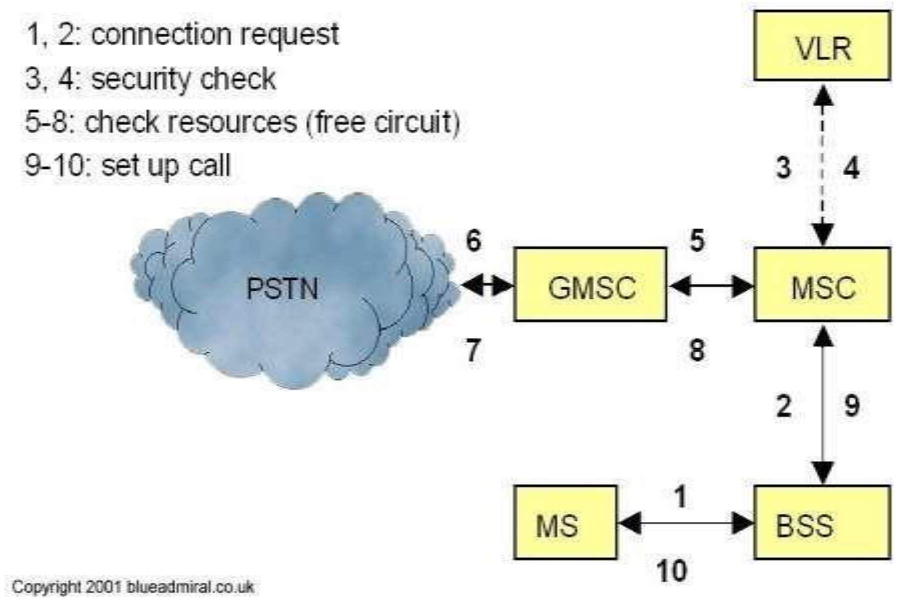
Step 11: The BTSs of all BSSs transmit this paging signal to MS.

Step 12 & 13: If MS answers (12 and 13) the VLR has to perform security checks set up be encryption techniques.

Steps 14 to 17: The VLR signals to MSC to setup a connection to MS.

MESSAGE ORIGINATED CALL (MOC):

It is simpler to perform message originated call(MOC) compared to MTC.



Message Originated Call (MOC)

The basic steps for MOC are,

Step 1: MS transmits a request for a new connection.

Step 2: BSS forwards the request to MSC.

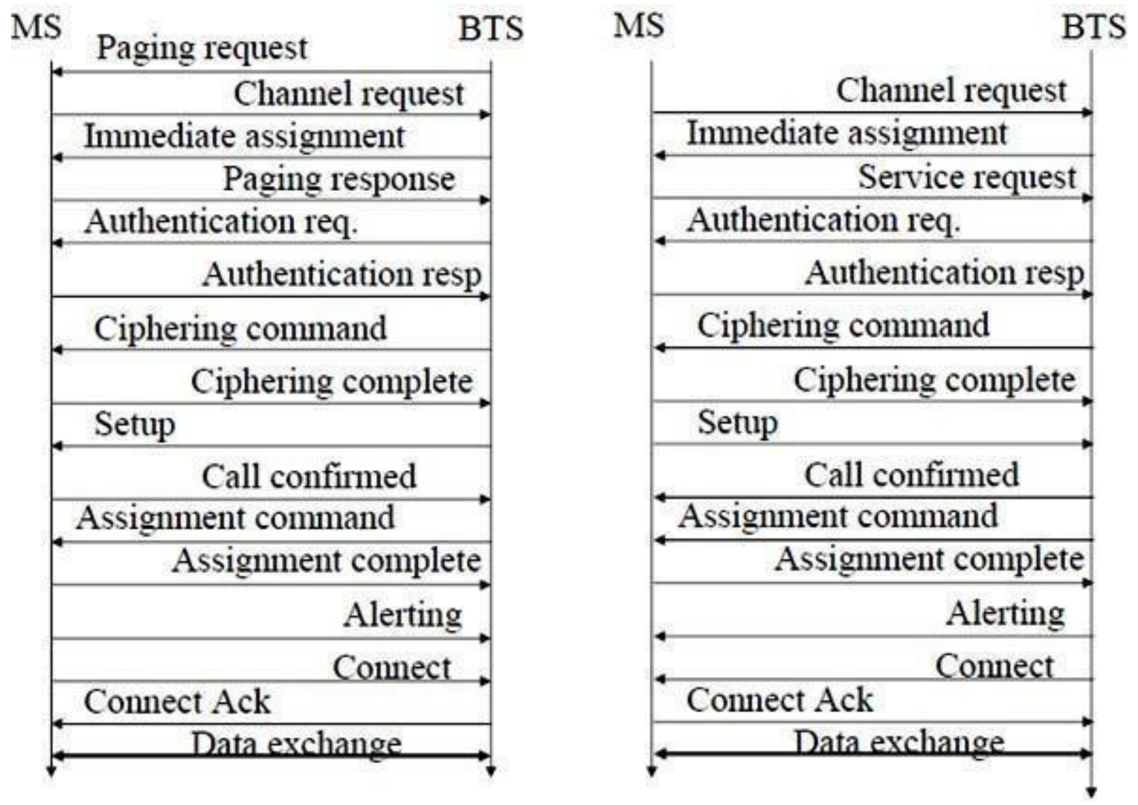
Steps 3 & 4: MSC then checks if the user is allowed to set up a call with the requested service (3 and 4) and checks the availability of resources through GSM network into PSTN.

Steps 5to 8: If all resources are available, MSC sets up a connection between MS and fixed network.

Steps 9 & 10: It’s set up a call with the help of BSS and MS.

MESSAGE FLOW FOR MTC AND MOC:

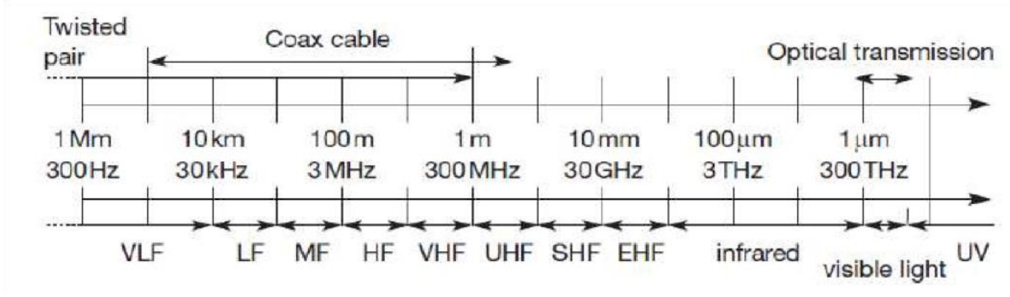
In addition to the above steps mentioned above, the other messages are exchanged between an MS and BTS during connection setup. These messages can be quite often heard in radios or badly shielded speakers as crackling noise before the phone rings. Figure shows the message for an MTC and MOC. Paging is only necessary for an MTC, then similar message exchanges follow. The next step which are needed for a communication security comprises the authentication of MS and switching to encrytpted communication The following steps which are mentioned in the figure denotes the use of MSC and MOC. If someone is calling the MS, it answers now with 'alerting' that MS is ringing and with 'connect' that the user has pressed the connect button. The same actions happen the other way round if MS has initiated the call. After connection acknowledgement both parties are exchanged.



Message flow for MTC and MOC

FREQUENCY ALLOCATION:

Radio transmission can take place using many different frequency bands. Each frequency band exhibits certain advantages and disadvantages. Figure gives a rough overview of the frequency spectrum that can be used for data transmission. The figure shows frequencies starting at 300 Hz and going up to over 300 THz. Directly coupled to the frequency is the wavelength λ via the equation: $\lambda = c/f$, where $c \cong 3 \cdot 108$ m/s (the speed of light in vacuum) and f the frequency. For traditional wired networks, frequencies of up to several hundred kHz are used for distances up to some km with twisted pair copper wires, while frequencies of several hundred MHz are used with coaxial cable (new coding schemes work with several hundred MHz even with twisted pair copper wires over distances of some 100 m).

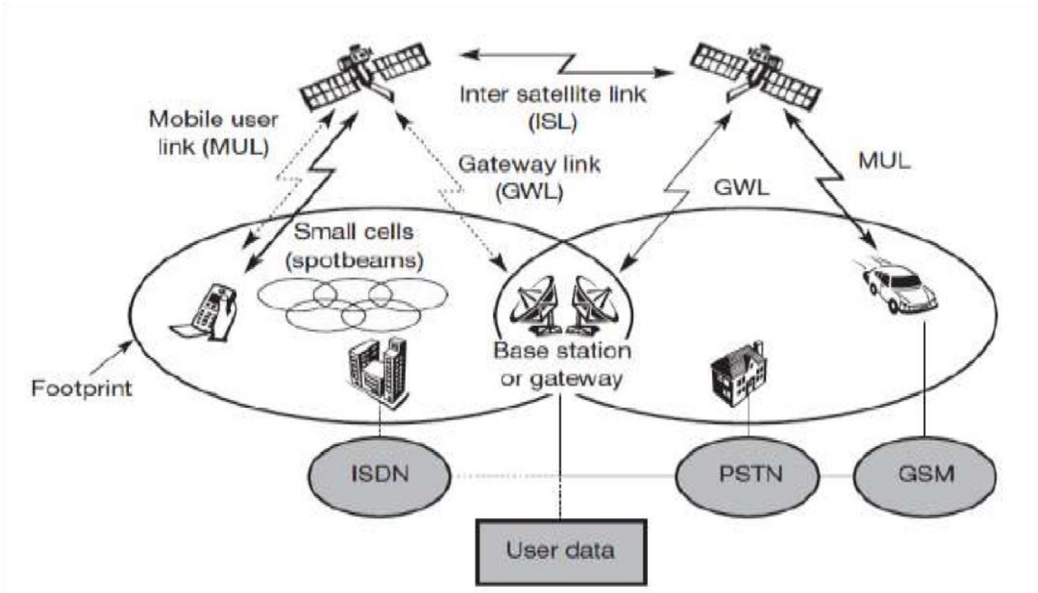


Frequency allocation

Fiber optics are used for frequency ranges of several hundred THz, but here one typically refers to the wavelength which is, e.g., 1500 nm, 1350 nm etc. (infra red). Radio transmission starts at several kHz, the **very low frequency (VLF)** range. These are very long waves. Waves in the **low frequency (LF)** range are used by submarines, because they can penetrate water and can follow the earth‟s surface. Some radio stations still use these frequencies, e.g., between 148.5 kHz and 283.5 kHz in Germany. The **medium frequency (MF)** and **high frequency (HF)** ranges are typical for transmission of hundreds of radio stations either as amplitude modulation **(AM)** between 520 kHz and 1605.5 kHz, as short wave **(SW)** between 5.9 MHz and 26.1 MHz, or as frequency modulation **(FM)** between 87.5 MHz and 108 MHz. The frequencies limiting these ranges are typically fixed by national regulation and, vary from country to country. Short waves are typically used for (amateur) radio transmission around the world, enabled by reflection at the ionosphere.

Transmit power is up to 500 kW – which is quite high compared to the 1 W of a mobile phone. As we move to higher frequencies, the TV stations follow. Conventional analog TV is transmitted in ranges of 174–230 MHz and 470–790 MHz using the very high frequency **(VHF)** and ultrahigh frequency **(UHF)** bands. In this range, digital audio broadcasting (DAB) takes place as well (223–230 MHz and 1452–1472 MHz) and digital TV is planned or currently being installed (470– 862 MHz), reusing some of the old frequencies for analog TV. UHF is also used for mobile phones with analog technology (450–465 MHz), the digital GSM (890–960 MHz, 1710–1880 MHz), digital cordless telephones following the DECT standard (1880–1900 MHz), 3G cellular systems following the UMTS standard (1900–1980 MHz, 2020–2025 MHz, 2110–2190 MHz) and many more. VHF and especially UHF allow for small antennas and relatively reliable connections for mobile telephony. **Super high frequencies (SHF)** are typically used for directed microwave links (approx. 2–40 GHz) and fixed satellite services in the C-band (4 and 6 GHz), Ku-band (11 and 14 GHz), or Ka-band (19 and 29 GHz). Some systems are planned in the **extremely high frequency (EHF)** range which comes close to infrared. All radio frequencies are regulated to avoid interference, e.g., the German regulation covers 9 kHz–275 GHz. The next step into higher frequencies involves optical transmission, which is not only used for fiber optical links but also for wireless communications. **Infrared (IR)** transmission is used for directed links, e.g., to connect different buildings via laser links. The most widespread IR technology, infrared data association (IrDA), uses wavelengths of approximately 850–900 nm to connect laptops, PDAs etc. Finally, visible light has been used for wireless transmission for thousands of years. While light is not very reliable due to interference, but it is nevertheless useful due to built-in human receivers.

ROUTING:



Routing

A satellite system together with gateways and fixed terrestrial networks as shown in Figure 5.1 has to route data transmissions from one user to another as any other network does. Routing in the fixed segment (on earth) is achieved as usual, while two different solutions exist for the satellite network in space. If satellites offer ISLs, traffic can be routed between the satellites. If not, all traffic is relayed to earth, routed there, and relayed back to a satellite. Assume two users of a satellite network exchange data. If the satellite system supports ISLs, one user sends data up to a satellite and the satellite forwards it to the one responsible for the receiver via other satellites. This last satellite now sends the data down to the earth. This means that only one uplink and one downlink per direction is needed. The ability of routing within the satellite network reduces the number of gateways needed on earth. If a satellite system does not offer ISLs, the user also sends data up to a satellite, but now this satellite forwards the data to a gateway on earth. Routing takes place in fixed networks as usual until another gateway is reached which is responsible for the satellite above the receiver. Again data is sent up to the satellite which forwards it down to the receiver. This solution requires two uplinks and two downlinks. Depending on the orbit and the speed of routing in the satellite network compared to the terrestrial network, the solution with ISLs might offer lower latency. The drawbacks of ISLs are higher system complexity due to additional antennas and routing hard- and software for the satellites.

MOBILITY MANAGEMENT:

Mobility management is one of the major functions of a GSM or a UMTS network that allows mobile phones to work. The aim of mobility management is to track where the subscribers are, allowing calls, SMS and other mobile phone services to be delivered to them. Location update procedure.

A GSM or UMTS network, like all cellular networks, is a radio network of individual cells, known as base stations. Each base station covers a small geographical area which is part of a uniquely identified location area. By integrating the coverage of each of these base stations, a cellular network provides a radio coverage over a much wider area. A group of base stations is named a location area, or a routing area.

The location update procedure allows a mobile device to inform the cellular network, whenever it moves from one location area to the next. Mobiles are responsible for detecting location area codes. When a mobile finds that the location area code is different from its last update, it performs another update by sending to the network, a location update request, together with its previous location, and its Temporary Mobile Subscriber Identity (**TMSI**).

There are several reasons why a mobile may provide updated location information to the network. Whenever a mobile is switched on or off, the network may require it to perform an IMSI attach or IMSI detach location update procedure. Also, each mobile is required to regularly report its location at a set time interval using a **periodic location update** procedure. Whenever a mobile moves from one location area to the next while not on a call, a **random location update** is required. This is also required of a stationary mobile that reselects coverage from a cell in a different location area, because of signal fade. Thus a subscriber has reliable access to the network and may be reached with a call, while enjoying the freedom of mobility within the whole coverage area.

When a subscriber is paged in an attempt to deliver a call or SMS and the subscriber does not reply to that page then the subscriber is marked as absent in both the Mobile Switching Center / Visitor Location Register (MSC/VLR) and the Home Location Register (HLR) (Mobile not reachable flag MNRF is set). The next time the mobile performs a location update the HLR is updated and the mobile not reachable flag is cleared.

TMSI

The Temporary Mobile Subscriber Identity (TMSI) is the identity that is most commonly sent between the mobile and the network. TMSI is randomly assigned by the VLR to every mobile in the area, the moment it is switched on. The number is local to a location area, and so it has to be updated each time the mobile moves to a new geographical area.

The network can also change the TMSI of the mobile at any time. And it normally does so, in order to avoid the subscriber from being identified, and tracked by eavesdroppers on the radio interface. This makes it difficult to trace which mobile is which, except briefly, when the mobile is just switched on, or when the data in the mobile becomes invalid for one reason or another. At that point, the global "international mobile subscriber identity" (IMSI) must be sent to the network. The IMSI is sent as rarely as possible, to avoid it being identified and tracked.

A key use of the TMSI is in paging a mobile. "Paging" is the one-to-one communication between the mobile and the base station. The most important use of broadcast information is to set up channels for "paging". Every cellular system has a broadcast mechanism to distribute such information to a plurality of mobiles. Size of TMSI is 4 octet with full hex digits and can't be all 1 because the SIM uses 4 octets with all bits equal to 1 to indicate that no valid TMSI is available.

ROAMING:

Roaming is one of the fundamental mobility management procedures of all cellular networks. Roaming is defined as the ability for a cellular customer to automatically make and receive voice calls, send and receive data, or access other services, including home data services, when travelling outside the geographical coverage area of the home network, by means of using a visited network. This can be done by using a communication terminal or else just by using the subscriber identity in the visited network. Roaming is technically supported by mobility management, authentication, authorization and billing procedures.

LOCATION AREA

A "location area" is a set of base stations that are grouped together to optimise signalling. Typically, tens or even hundreds of base stations share a single Base Station Controller (BSC) in GSM, or a Radio Network Controller (RNC) in UMTS, the intelligence behind the base stations. The BSC handles allocation of radio channels, receives measurements from the mobile phones, controls handovers from base station to base station.

To each location area, a unique number called a "location area code" is assigned. The location area code is broadcast by each base station, known as a "base transceiver station" BTS in GSM, or a Node B in UMTS, at regular intervals. In GSM, the mobiles cannot communicate directly with each other but, have to be channeled through the BTSs. In UMTS networks, if no Node B is accessible to a mobile, it will not be able to make any connections at all.

If the location areas are very large, there will be many mobiles operating simultaneously, resulting in very high paging traffic, as every paging request has to be broadcast to every base station in the location area. This wastes bandwidth and power on the mobile, by requiring it to listen for broadcast messages too much of the time. If on the other hand, there are too many small location areas, the mobile must contact the network very often for changes of location, which will also drain the mobile's battery. A balance has therefore to be struck.

ROUTING AREA:

The routing area is the PS domain equivalent of the location area. A "routing area" is normally a subdivision of a "location area". Routing areas are used by mobiles which are GPRS-attached. GPRS is optimized for "bursty" data communication services, such as wireless internet/intranet, and multimedia services. It is also known as GSM-IP ("Internet Protocol") because it will connect users directly to Internet Service Providers (ISP).

The bursty nature of packet traffic means that more paging messages are expected per mobile, and so it is worth knowing the location of the mobile more accurately than it would be with traditional circuit-switched traffic. A change from routing area to routing area (called a "Routing Area Update") is done in an almost identical way to a change from location area to location area. The main differences are that the "Serving GPRS Support Node" (SGSN) is the element involved.

TRACKING AREA:

The tracking area is the LTE counterpart of the location area and routing area. A tracking area is a set of cells. Tracking areas can be grouped into lists of tracking areas (TA lists), which can be configured on the User equipment. Tracking area updates are performed periodically or when the UE moves to a tracking area that is not included in its TA list.
Operators can allocate different TA lists to different UEs. This can avoid signaling peaks in some conditions: for instance, the UEs of passengers of a train may not perform tracking area updates simultaneously. On the network side, the involved element is the Mobility Management Entity.

HANDOVER:

Handover means handing over the mobile from one cell to another cell. There are two reasons for handover.
They are,

- When a mobile station moves out of the range of BTS the signal level decreases continuously and falls below the minimal requirements for communication.
- The error rate increases due to interference. The quality of radio link decrease.
- The traffic in one cell is too high, shifting of some MS to other cells with lower load. This is called load balancing.
- The number of handover will be more when the cell size is small.
- Due to handover the calls should not get to cutoff which is called as call drop.

TYPES OF HANDOVER:

(1) INTRA CELL HANDOVER:

- With in a cell, narrow band interference can cause transmission at a certain frequency impossible.

- The BSC decides to change the carrier frequency.

(2) INTER CELL, INTRA BSC HANDOVER:

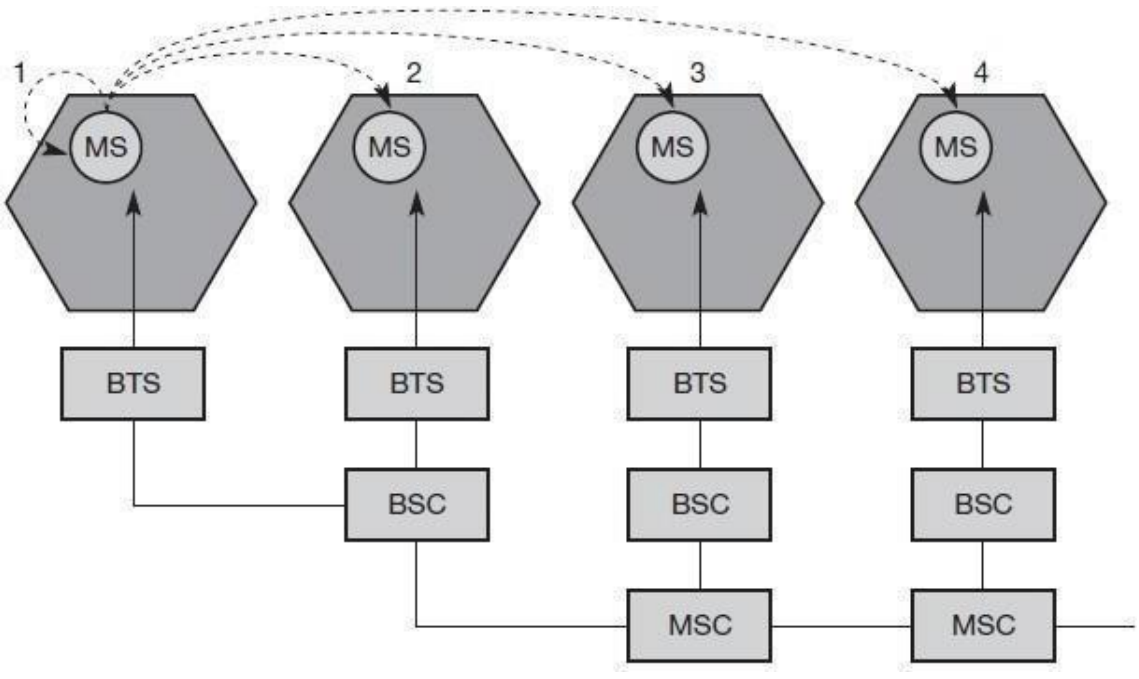
- The mobile station moves from one cell to another but remains with in the same BSC.
- The BSC performs a handover, assigns a new radio channel in the new cell **and** releases the old one.

(3) INTER BSC, INTRA MSC HANDOVER:

- The BSC controls only limited cells.
- Handover needs to be done between different BSC. This is controlled by MSC.

(4) INTER MSC HANDOVER:

- A handover is needed between 2 cells which belong to difference MSC.
- Both MSC performs the handover together.



SECURITY:

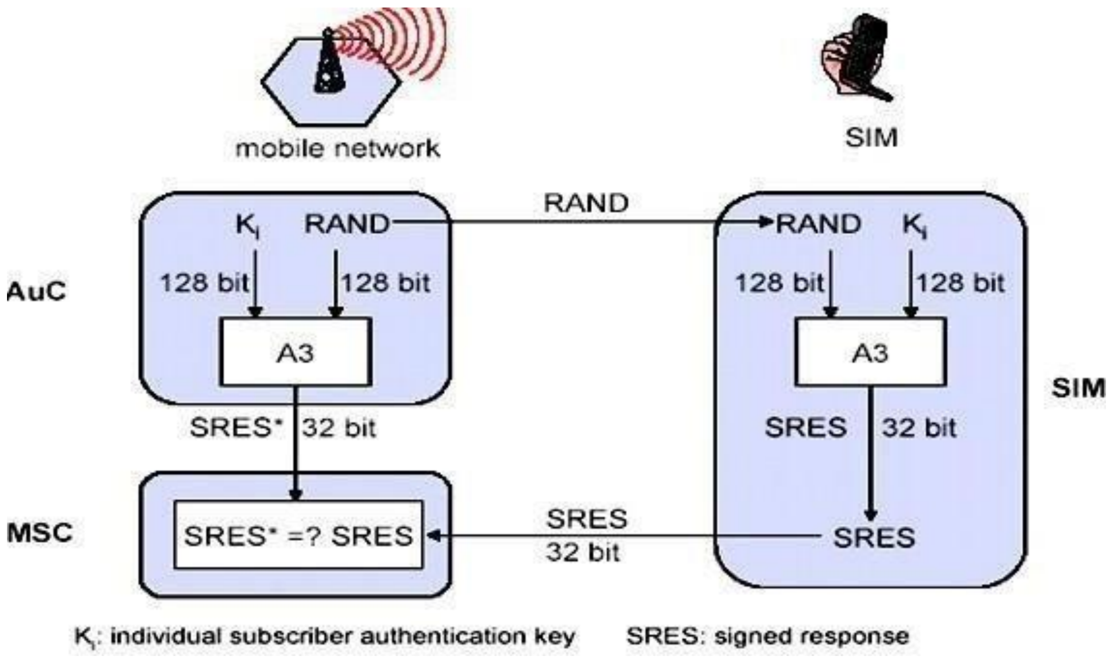
GSM offers several security services using confidential information stored in the AuC and in the individual SIM (which is plugged into an arbitrary MS). The SIM stores personal, secret data and is protected with a PIN against unauthorized use. (For example, the secret key Ki used for authentication and encryption procedures is stored in the SIM.) The security services offered by GSM are explained below:

- **ACCESS CONTROL AND AUTHENTICATION:** The first step includes the authentication of a valid user for the SIM. The user needs a secret PIN to access the SIM. The next step is the subscriber authentication.
- **CONFIDENTIALITY:** All user-related data is encrypted. After authentication, BTS and MS apply encryption to voice, data, and signaling. This confidentiality exists only between MS and BTS, but it does not exist end-to-end or within the whole fixed GSM/telephone network.
- **ANONYMITY:** To provide user anonymity, all data is encrypted before transmission, and user identifiers (which would reveal an identity) are not used over the air. Instead, GSM transmits a temporary identifier (TMSI), which is newly assigned by the VLR after each location update. Additionally, the VLR can change the TMSI at any time. Three algorithms have been specified to provide security services in GSM. Algorithm A3 is used for authentication, A5 for encryption, and A8 for the generation of a cipher key. In the GSM standard only algorithm A5 was publicly available, whereas A3 and A8 were secret, but standardized with open interfaces. Both A3 and A8 are no longer secret, but were published on the internet in 1998. This demonstrates that security by obscurity does not really work. As it turned out, the algorithms are not very strong. However, network providers can use stronger algorithms for authentication – or users can apply stronger end-to-end encryption. Algorithms A3 and A8 (or their replacements) are located on the SIM and in the AuC and can be proprietary. Only A5 which is implemented in the devices has to be identical for all providers.

AUTHENTICATION:

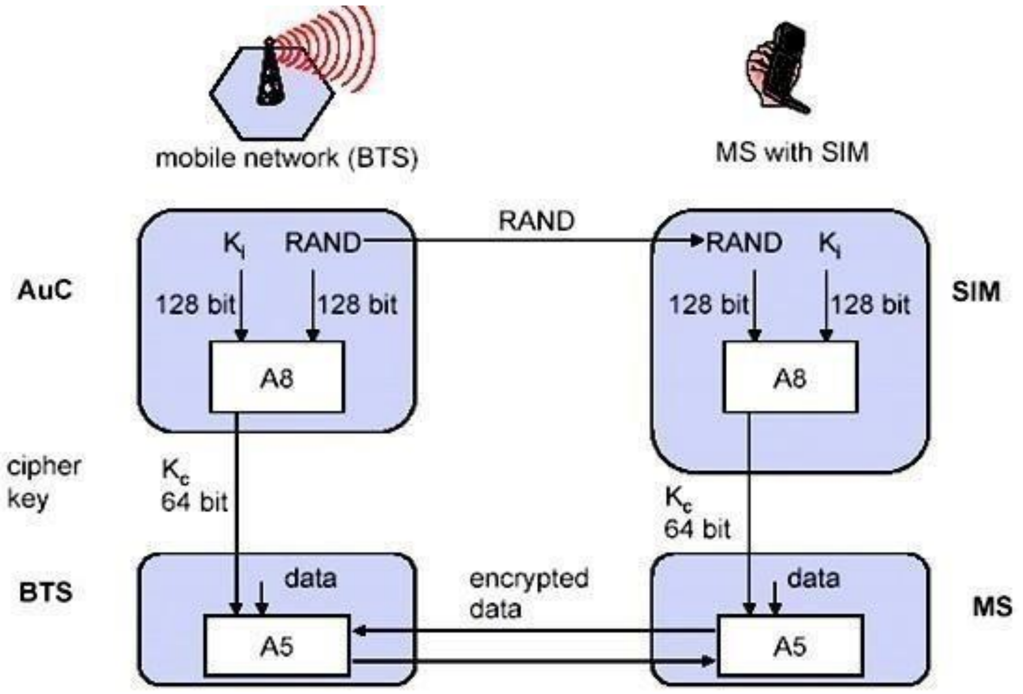
Before a subscriber can use any service from the GSM network, he or she must be authenticated. Authentication is based on the SIM, which stores the individual authentication key Ki, the user identification IMSI, and the algorithm used for authentication A3. Authentication uses a challenge-response method: the access control AC generates a random number RAND as challenge, and the SIM within the MS answers with SRES (signed response) as response. The AuC performs the basic generation of random values RAND, signed responses SRES, and cipher keys Kc for each IMSI, and then forwards this information to the HLR. The current VLR requests the appropriate values for RAND, SRES, and Kc from the HLR.

For authentication, For authentication, the VLR sends the random value RAND to the SIM. Both sides, network and subscriber module, perform the same operation with RAND and the key Ki, called A3. The MS sends back the SRES generated by the SIM; the VLR can now compare both values. If they are the same, the VLR accepts the subscriber, otherwise the subscriber is rejected.



ENCRYPTION:

To ensure privacy, all messages containing user-related information are encrypted in GSM over the air interface. After authentication, MS and BSS can start using encryption by applying the cipher key Kc (the precise location of security functions for encryption, BTS and/or BSC are vendor dependent).



Kc is generated using the individual key Ki and a random value by applying the algorithm A8. Note that the SIM in the MS and the network both calculate the same Kc based on the random value RAND. The key Kc itself is not transmitted over the air interface.

MS and BTS can now encrypt and decrypt data using the algorithm A5 and the cipher key Kc. As Figure shows, Kc should be a 64 bit key which is not very strong, but is at least a good protection against simple eavesdropping. However, the publication of A3 and A8 on the internet showed that in certain implementations 10 of the 64 bits are always set to 0, so that the real length of the key is thus only 54 consequently, the encryption is much weaker.

3.7 GPRS:

INTRODUCTION:

General packet radio service (GPRS) is a packet oriented mobile data service available to users of the 2G cellular communication systems, 3G systems and GSM. GPRS re-use the existing GSM infrastructure. It interworked with existing circuit-switched services. It is based on standardized open interfaces.

GPRS usage is typically charged based on volume of data transferred, contrasting with circuit switched data, which is usually billed per minute of connection time. 5 GB per month for a fixed fee or on a pay-as-you-use basis. Usage above the bundle cap is either charged per megabyte or disallowed.

GPRS is a best effort service, implying variable throughput and latency that depend on the number of other users sharing the service concurrently, as opposed to circuit switching, where a certain quality of service (QoS) is guaranteed during the connection. In 2G systems, GPRS provides data rates of 56–114 kbit/second. 2G cellular technology combined with GPRS is sometimes described as *2.5G*, that is, a technology between the second (2G) and third (3G) generations of mobile telephony. It provides moderate-speed data transfer, by using unused time division multiple access (TDMA) channels in, for example, the GSM system. GPRS is integrated into GSM Release 97 and newer releases.

GPRS provides two services:

1. Point-to-point **(PTP)**
2. Point-to-multipoint **(PTM)**

In point-to-point packet delivery service, in which packet is transfer between two users and in point-to-multipoint **(PTM)** service, in which packet is delivering to multiple destinations within one service request.

In PTP versions are PTP Connection oriented Network service **(PTPCONS)**, which establish a logical relation in between users. Multiple packets are sent between single source and a single destination. Other version is the PTP Connectionless Network Service **(PTP-CLNS)**, which does not require a logical link between users. Packets are sent between a single source and a single destination. Each packet is independent of its predecessor and successor.

QoS-profile can be specified by the users of the GPRS. It is maintained in the PDP context. **PDP Context** is nothing but which is created in each communication session. QoS-profile is used to indicate the network and radio resources required for data transmission. It has the attributes such as service precedence (high,normal,low),reliability class, delay class, peak throughput class, mean throughput class. GPRS must allocate radio resources to fulfill these user specifications. GPRS network is suffered by the following delays such as channel access delay, coding for error correction and transfer delay in the fixed part and wireless part of the network. GPRS also includes several security services namely authentication, user identity confidentiality, access control and user information confidentiality.

Main benefits:

Resources are reserved only when needed and charged accordingly. Connection setup times are reduced. It will enable new service opportunities. It has High Speed (Data Rate 14.4 – 115 kbps). It uses the efficient radio bandwidth (Statistical Multiplexing). Circuit switching & Packet Switching can be used in parallel. It has Constant connectivity.

Characteristics of GPRS:

1. GPRS uses packet switched resource allocation.
2. Flexible channel allocation.
3. Support for leading internet communication protocols.

GPRS Terminal Classes:

1. Class A:

It can be connected to GPRS service and GSM service (voice, SMS), using both at the same time. Such devices are known to be available today.

2. Class B:

It can be connected to GPRS service and GSM service (voice, SMS), but using only one or the other at a given time. During GSM service (voice call or SMS), GPRS service is suspended, and then resumed automatically after the GSM service (voice call or SMS) has concluded. Most GPRS mobile devices are Class B.

3. Class C:

They are connected to either GPRS service or GSM service (voice, SMS). Must be switched manually between one or the other service.

GPRS ARCHITECTURE:

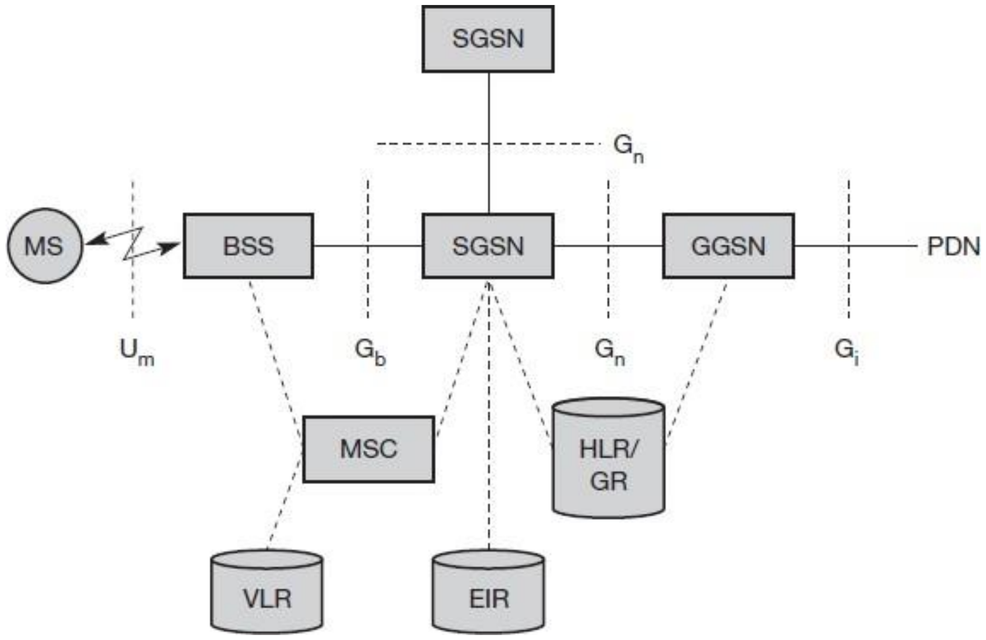
In order to understand the GPRS network architecture, some fundamental GSM terminology is necessary. This section describes some of the main components of the GSM network.

GPRS Networks

GPRS architecture has two network elements, which are called as **GPRS support nodes (GSN)**.

They are,

1. Gateway GPRS Support Node (GGSN)
2. Serving GPRS Support Node (SGSN)



GPRS Architecture Reference Model

All GSNs are integrated into the standard GSM architecture and many interfaces (see figure). The network elements are **gateway GPRS support node (GGSN)** is provisioned by router, which supports traditional gateway functionality. It is the interworking unit between the GPRS network and external **packet data networks (PDN).** This node contains routing information for GPRS users. It performs address conversion and tunnels data to a user via encapsulation. The GGSN is connected to external networks (e.g., IP or X.25) via the G_i and transfers packets to the SGSN via an IP-based GPRS backbone network (G_n interface).

The other element is the **serving GPRS support node (SGSN)** which connects BSS and GGSN. It supports the MS via the G_b interface. It requests the user address from the **GPRS register (GR).** It keeps track of the individual MSs' location, is in charge for collecting billing information. It performs many security functions.

Packet Control Unit (PCU)

The PCU separates the circuit switched and packet switched traffic from the user and sends them to the GSM and GPRS networks respectively. It also performs most of the radio resource management functions of the GPRS network. The PCU can be either located in the BTS, BSC, or some other point between the MS and the MSC. There will be at least one PCU that serves a cell in which GPRS services will be available. Frame Relay technology is being used at present to interconnect the PCU to the GPRS core.

GPRS interfaces

Um between an MS and the GPRS fixed network part. The Um is the access interface the MS uses to access the GPRS network. The radio interface to the BTS is the same interface used by the existing GSM network with some GPRS specific changes.

Gb between a SGSN and a BSS. The Gb interface carries the GPRS traffic and signaling between the GSM radio network (BSS) and the GPRS network. Frame Relay based network services is used for this interface.

Gn between two GSNs within the same PLMN. The Gn provides a data and signalling interface in the Intra-PLMN backbone. The GPRS Tunnelling Protocol (GTP) is used in the Gn (and in the Gp) interface over the IP based backbone network.

Mobile Station

A GSM subscriber needs a terminal called **Mobile Station (MS).** It is used to connect to the network using the radio interface $U_{m.}$ In idle mode an MS is not reachable and all contexts will be deleted. In the standby state there is only movement across routing areas which is updated to the SGSN. Before sending any data over the GPRS network, an MS must attach to it, following the procedures of the **mobility management**. The attachment procedure includes assigning a temporal identifier, called a **temporary logical link identity (TLLI)**, and a **ciphering key sequence number (CKSN)** for data encryption.

GPRS BSS

Base Station Subsystem (BSS) which performs radio-related functions. BSS contains Base Transceiver Stations (BTS) and Base Station Controllers (BSC).

BTS which provides new GPRS channel coding schemes through Channel Codec Unit (CCU). The BTS handles the radio interface to the MS. It consists of radio equipment (transceivers and antennas) required to service each cell in the network.

BSC forwards the Circuit-switched calls to MSC and the Packet-switched data to SGSN. The BSC provides the control functions and physical links between the MSC and the BTS. A number of BSCs are served by one MSC while several BTSs can be controlled by one BSC.

The Network Switching Subsystem

The NSS is responsible for call control, service control and subscriber mobility manage

a) Mobile Switching center (MSC):

MSC is in charge for telephony switching functions of the network. It also performs authentication to verify the user’s identity. It ensures the confidentiality of the calls. The Authentication Center (AuC) provides the necessary parameters to the MSC to perform the authentication procedure. The AuC is shown as a separate logical entity but is generally integrated with the HLR. The Equipment Identity Register (EIR) is on the other hand a database that contains information about the identity of the mobile equipment. It prevents calls from unauthorized or stolen MSs.

b) Home Location register (HLR):

HLR is a database used to store and manage permanent data of subscribers. HLR is used to map an MS to one or more GGSNs. It is used to update the SGSN of the MS. It is also used to store the fixed IP address and QoS profile for a transmission path.

c) Visitor location register (VLR):

VLR is a database used to store temporary information about the subscribers. It is needed by the MSC in order to service visiting subscribers. The MSC and VLR are commonly integrated into one single physical node and the term MSC/VLR is used instead. When a subscriber enters a new MSC area, a copy of all the ne) cessary information is downloaded from the HLR into the VLR. The VLR keeps this information so that calls of the subscriber can be processed without having to interrogate the HLR (which can be in another PLMN) each time. The temporary information is cleared when the mobile station roams out of the service area.

d)Equipment identity register (EIR):

EIR is also a database that encloses information about the identity of the mobile equipment. It prevents calls from unauthorized or stolen MSs.

GPRS Mobility Management States:

a) Idle State:

A MS in the idle state is not traceable and can only receive PTM-M transmissions such as general broadcast events destined to a specific geographical area. The MS needs to perform the attach procedure in order to connect to the GPRS network and become reachable.

b) Ready State:

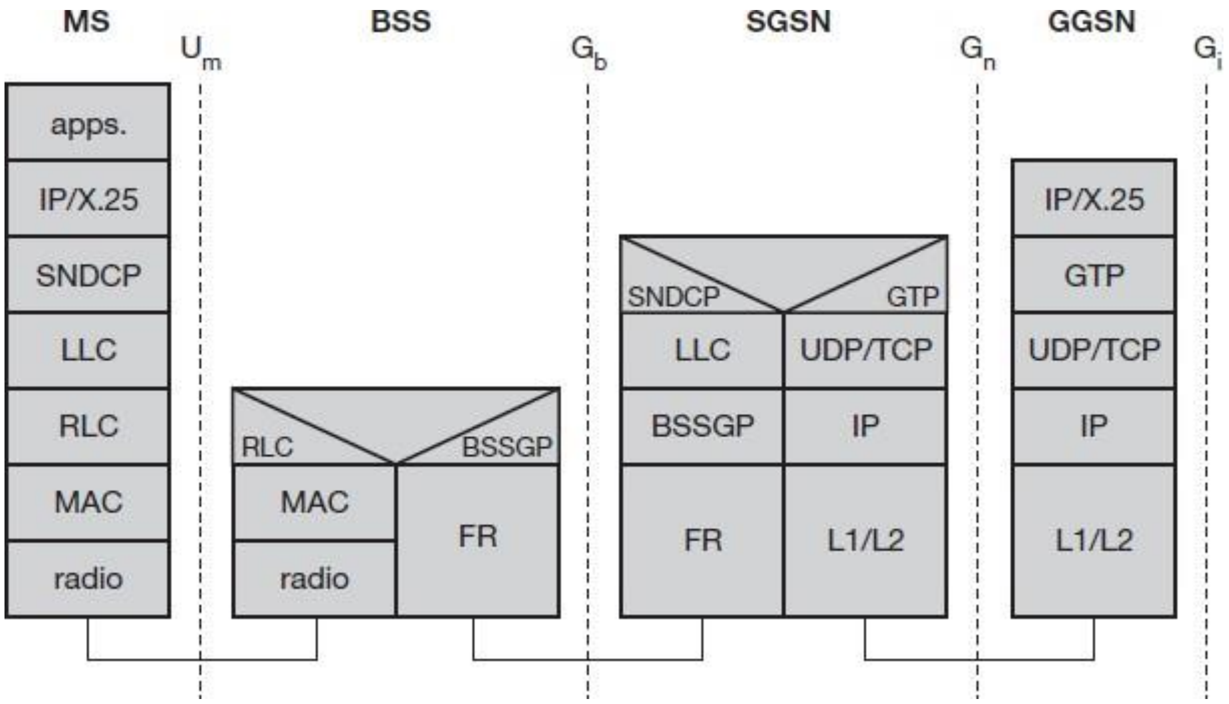
Data is sent or received in this state. The MS informs the SGSN when it changes cells. The MS may explicitly request (or can be forced by the network) to detach in which case it moves to Idle. A timer monitors the Ready state and upon its expiry, the MS is put on Standby. The timer ensures that resources are not wasted by an inactive MS.

c) Standby State:

A connected MS which is inactive is put in the Standby state. Moving back to Ready can be triggered by sending data or signalling information from the MS to the SGSN. Upon arrival of data destined to the MS, the SGSN pages the latter and a response to the page moves the MS back to the Ready state. The MS may wish (or can be forced by the network) to terminate the connection by requesting to detach in which case it returns to Idle. A timer is used by the SGSN to monitor the tracking of the MS, and when it expires, the MS is detached and is considered unreachable.

GPRS PROTOCOL ARCHITECTURE

A GPRS network introduces many new protocols designed to convey user data in a reliable and secure way. The protocol architecture is implemented for the transmission and signaling planes in GPRS. **Transmission plane protocols** are used for the transmission of user data and control functions. **Signaling plane protocols** are used to convey signaling information that controls and supports the transmission plane functions. (See figure 2)**.**



GPRS transmission plane protocol reference model

Transmission protocols in the Um interface

a) Physical layer:

The physical layer can be divided into the Radio Frequency (RF) layer and the Physical Link layer.

The **Radio Frequency (RF)** is the normal GSM physical radio layer. Among other things the RF layer specifies the carrier frequency characteristics and GSM radio channel structures. It uses the radio modulation scheme for the data. The GSM RF physical layer is used for GPRS with the possibility for future modifications.

The **Physical Link layer** supports multiple MSs sharing a single physical channel and provides communication between the MSs and the network. Network controlled handovers are not used in the GPRS service. Instead, routing area updates and cell updates are used.

b) Medium Access Control (MAC):

MAC protocol handles the channel allocation and the multiplexing. The RLC and the MAC together form the OSI Layer 2 protocol for the U_m interface. The radio interface at U_m need GPRS which does not require changes compared to GSM.

c) Radio Link Control (RLC):

RLC protocol establishes a reliable radio link to the upper layers. It also works either in acknowledged or unacknowledged modes.

d) Logical Link Control (LLC):

LLC layer establishes a secure and reliable logical link between the MS and the SGSN for upper layer protocols. It works either in acknowledged or unacknowledged modes. The data confidentiality is ensured by using ciphering functions.

Subnetwork dependent convergence protocol (SNDCP)

SNDCP is used to transfer data packets between SGSN and MS. It is used to provide multiplexing of several connections of network layer onto one logical connection of underlying LLC layer. It provides functions that help to improve channel efficiency. This is achieved by means of compression techniques. Data Link layer is divided into LLC layer and RLC/MAC Layer.

Transmission protocols in the Gb interface

a) Physical Layer Protocol:

Several physical layer configurations and protocols are possible at the Gb interface and the physical resources are allocated by Operation & Maintenance (O&M) procedures. Normally a G703/704 2Mbit/s connection is provided.

b) Network Services layer:

The Gb interface Network Services layer is based on Frame Relay. Frame Relay virtual circuits are established between the SGSN and BSS. LLC PDUs from many users are statistically multiplexed onto these virtual circuits. These virtual circuits may traverse a network of Frame Relay switching nodes, or may just be provided on a point to point link between the BSC and the SGSN.

Base station subsystem GPRS protocol (BSSGP)

BSSGP is used to deliver routing and QoS-related information between the BSS and SGSN. It is to enable two physically distinct nodes, the SGSN and BSS. It is to operate node management control functions. There is a one-to-one relationship between the BSSGP protocol in the SGSN and in the BSS. If one SGSN handles multiple BSSs, the SGSN has to have one BSSGP protocol device for each BSS. BSSGP does not perform error correction and works on top of a frame relay (FR) network.

Transmission protocols in the Gn interface

a) Layer 1 and Layer 2:

The **L1** and the **L2** protocols are vendor dependent OSI layer 1 and 2 protocols. It carries the IP datagrams for the GPRS backbone network between the SGSN and the GGSN.

b) Internet Protocol (IP):

The Internet Protocol (IP) datagram in the Gn interface is only used in the GPRS backbone network. The GPRS backbone (core) network and the GPRS subscribers use different IP addresses. This makes the GPRS backbone IP network invisible to the subscribers and vice versa. The GPRS backbone network carries the subscriber IP or X.25 traffic in a secure GPRS tunnel.

c) TCP or UDP:

TCP or UDP are used to carry the GPRS Tunnelling Protocol (GTP) PDUs across the GPRS backbone network. TCP is used for user X.25 data and UDP is used for user IP data and signalling in the Gn interface.

d) GPRS tunneling protocol (GTP):

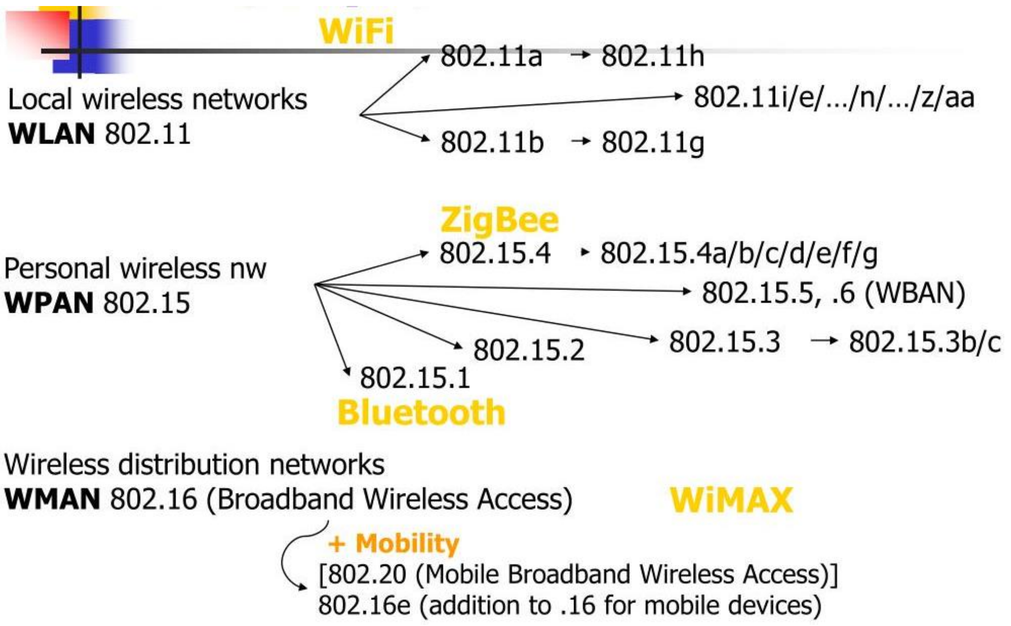
GTP is the basis for tunnel signaling. It uses two transport protocols such as reliable TCP and non-reliable UDP**.** The GPRS Tunnelling Protocol (GTP) allows multi-protocol packets to be tunnelled through the GPRS backbone between GPRS Support Nodes (GSNs).

UNIT IV WIRELESS NETWORKS

Infrared vs radio transmission – Infrastructure and ad hoc networks – WLAN, IEEE 802.11 standards protocols. Piconet- Bluetooth-architecture and services. Wireless Broadband networks and satellites networks.

4.1 Wireless LAN

- Wireless local area network (LAN) is a local area data network without wires.
- Wireless LAN is also known as WLAN in short.
- WLAN is a high-speed data networking technology that is being widely deployed in residential network, enterprises, and public areas around the world.
- A WLAN creates extensions to cabling LAN and reduces the need for wireless connections.
- It transmits and receives data over electromagnetic waves using radio frequency (RF).
- With it advantage and lack of wiring, mobile WLAN users can access real-time information and network resources easily.
- The WLAN ranges approximately 100 meters and is often used in building and on campuses. In short WLAN is a flexible data communication system providing wireless peer-to-peer and point-to-point connectivity within a building or campus.
- For that different types of technologies used for that which are show in below.



Advantages of WLAN

1. **Flexibility:**
    - Within radio coverage, node can communicate without further restriction.
    - Radio waves can penetrate walls, sender and receivers can be placed anywhere.
    - Sometimes wiring is difficult if firewalls separate building.
    - Penetration of a firewall is only permitted at certain points to prevent fire from spreading too fast.
2. **Planning:**
    - Only wireless ad-hoc networks allow for communication without previous planning any wired network needs wiring plans.
    - As long as devices follow the same standard they can communicate.
    - For wired network additional cabling with the right plugs and probably interworking unit have to be provided.
3. **Design:**
    - Wireless networks allow for the design of small, independent devices which can for example be put into a pocket.
    - Cables not only restrict users but also designers of small PDAs, notepads etc.
    - Wireless senders and receivers can be hidden in historic buildings i.e., current networking technology can be introduced without being visible.
4. **Robustness:**
    - Wireless network can survive disaster e.g., earthquakes or users pulling a plug.
    - If the wireless devices survive, people can still communicate. • Network requiring a wired infrastructure will usually break down completely.
5. **Cost:**
    - After providing wireless access to the infrastructure via an access point for the first user, adding additional users to a wireless network will not increase the cost.
    - This is, important for e.g. lecture halls, hotel lobbies or gate areas in airports where the numbers using the network may vary significantly.
    - Using fixed network, each seat in a lecture hall should have a plug for the network although many of them might not be used permanently.
    - Wireless connections do not wear out.

Disadvantages of WLAN

1. **Quality of Service:**
    - WLANs typically offer lower quality than their wired counterparts.
    - The main reasons for this are the lower bandwidth due to limitation in radio transmission, higher rates due to interference and higher delay/delay variation due to extensive error correction and detection mechanisms.
2. **Proprietary solutions:**
    - Due to slow standardization procedures many companies have come up with proprietary solution offering standardized functionality plus many enhanced features.
    - However, these additional features only work in a homogeneous environment i.e. when adapter from same vendors are used for all wireless nodes. • At least most components today here to the basic standard IEEE 802.11b or 8.2.11a.
3. **Restrictions:**
    - All wireless products have to comply with national regulations.
    - Several government and non-government institutions world wide regulate the operation and restrict frequencies to minimize interference.
    - WLAN are limited to low-power senders and certain license-free frequency bands, which are not the same worldwide.
4. **Safety and Security:**
    - Using radio waves for data transmission might interfere with other high-tech equipment in e.g. hospitals.
    - Senders and receivers are operated by laymen and radiation has to be low.
    - Special precautions have to be taken to prevent safety hazards.
    - The open radio interface makes eavesdropping much easier in WLANs than e.g. in the case of fiber optics.
    - All standard must offer encryption, privacy mechanisms , support for anonymity etc.
    - Otherwise more and more wireless networks will be hacked into as is the case already.

Design goals for wireless LANs

- Global, Faultless operation
- Low power for battery use
- No special permissions or licenses needed to use the LAN
- Robust transmission technology
- Simplified spontaneous cooperation at meetings
- Easy to use for everyone, simple management
- Protection of investment in wired networks
- Security (no one should be able to read my data), privacy (no one should be able to collect user profiles), safety (low radiation)
- Transparency concerning applications and higher layer protocols, but also location awareness if necessary

4.2 Types of Transmission Technologies

Today two different basic transmission technologies can be sued to set up WLANs.

1. **Infrared light:**
   - This technology use diffuse light reflected at walls, furniture etc. or directed light if an line-of-sight (LOS) exists between sender and receiver.
   - Infrared system are simple in design, therefore it is inexpensive.
   - InfraLAN is an example of wireless LANs using infrared technology.

2. **Radio way:**
   - It is more popular and uses radio transmission in the GHz range.
   - Almost all networks which are used any technology based on radio waves for data transmission.
   - RF systems must used spread spectrum technology in the united states.
   - This spread spectrum technology currently comes in two types : DSSS and FHSS.
   - E.g. GSM at 900,1,800,1,900 MHz etc.

	Infrared Transmission	**Radio Transmission**
Advantages	Simple, Cheap, Available in many mobile devices	Experience from wireless WLAN and mobile phones cam be used
	No licenses needed	Coverage of larger areas possible (radio can penetrate walls, furniture etc.)
	Simple shielding possible	
Disadvantages	Interference by sunlight, heat sources etc.	Very limited license free frequency bands
	Low bandwidth	Shielding more difficult, interference with other electrical devices
Example	IrDA (Infrared Data Association) interface available everywhere.	Many different products

4.3 Modes of Wireless LAN

Wireless works in two different modes:

1. Infrastructure
2. Ad-hoc Network:

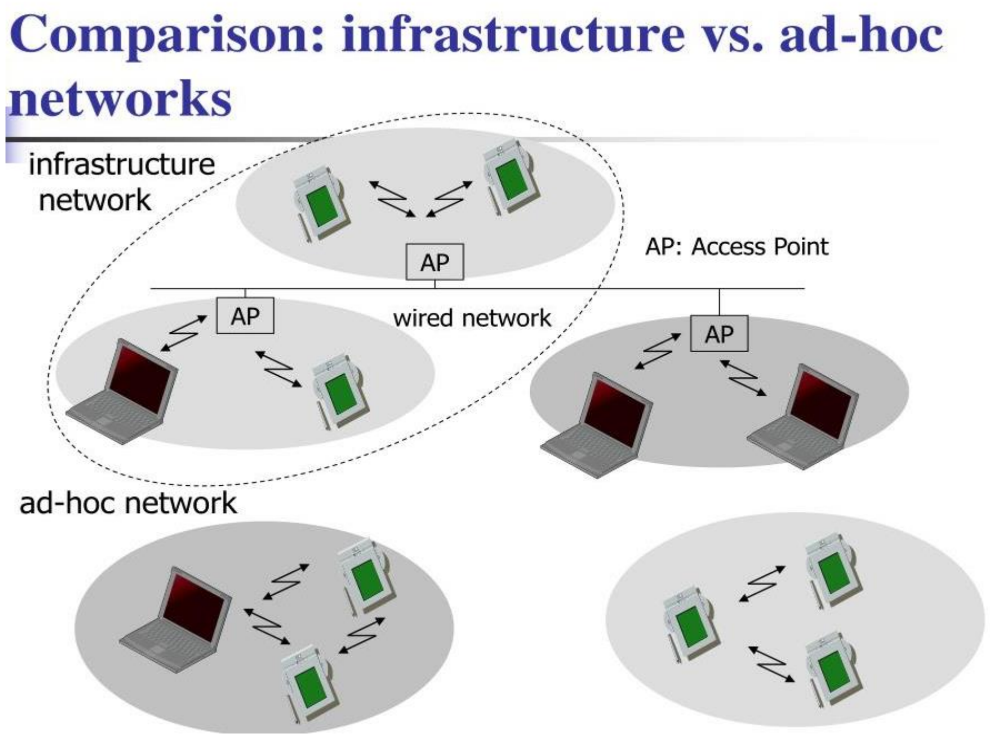
Infrastructure

- The infrastructure mode includes one or several interconnected WLAN-cell, which are connected to a fixed net through an access point.
- Wireless access points can be simply thought to function in a fashion analogous to Ethernet hub and switch are used to allow computers with wireless adapter to participate in a network.
- All device to device wireless communication goes through the WAP. • This is referred to as infrastructure mode.

Ad-hoc Networks

- The ad-hoc mode is WLAN-cell interacting without connection to wired networks, i.e. without connection to an access point.
- However the ad hoc networks work much like the Bluetooth.

- No access point is needed and the devices might connect to the internet through wired or other wireless techniques.
- Simple computing device to computing device wireless networking can be accomplished by installing a wireless network adapter (sometimes called wireless NICs) in each device. This is referred to as ad-hoc mode.



802.11 - Architecture of an infrastructure network

1. **Stations (STA):**

   Terminal with access mechanisms to the wireless medium and radio contact to the access point

2. **Basic Service Set (BSS):**

   Group of stations using same radio frequency

3. **Access Point:**

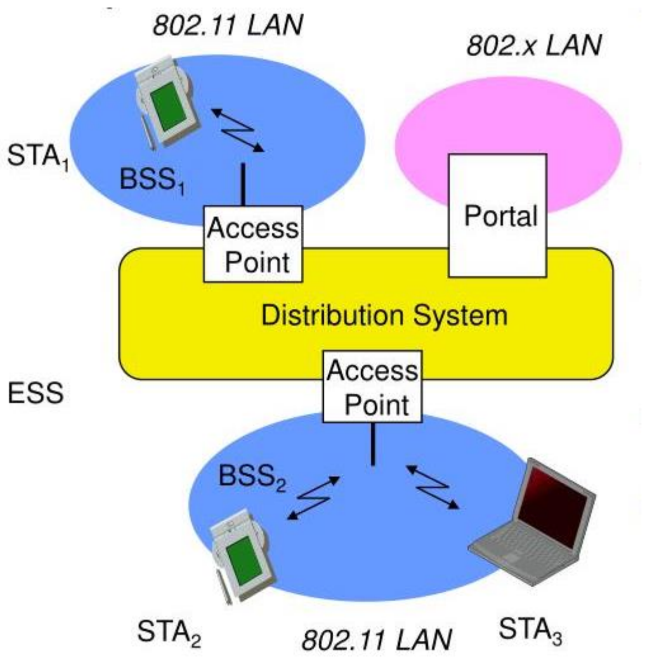
   Station integrated into the wireless LAN and the distribution system

4. **Portal:**

   Bridge to the (wired) networks

5. **Distribution System:**

   Interconnection network to form one logical network (EES: Extended Service Set) based on several BSS.



802.11 - Architecture of an ad-hoc network

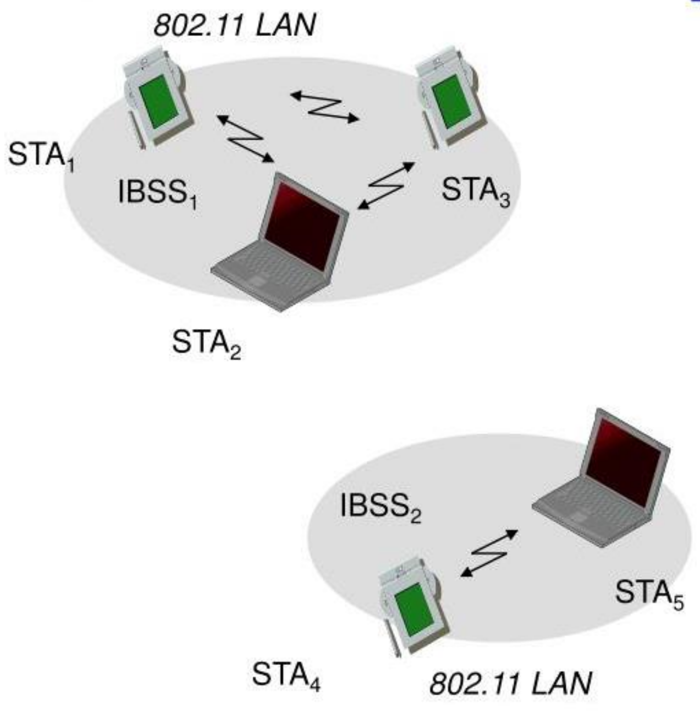
- Direct communication within a limited range

  1. **Station (STA):**

     Terminal with access mechanisms to the wireless medium

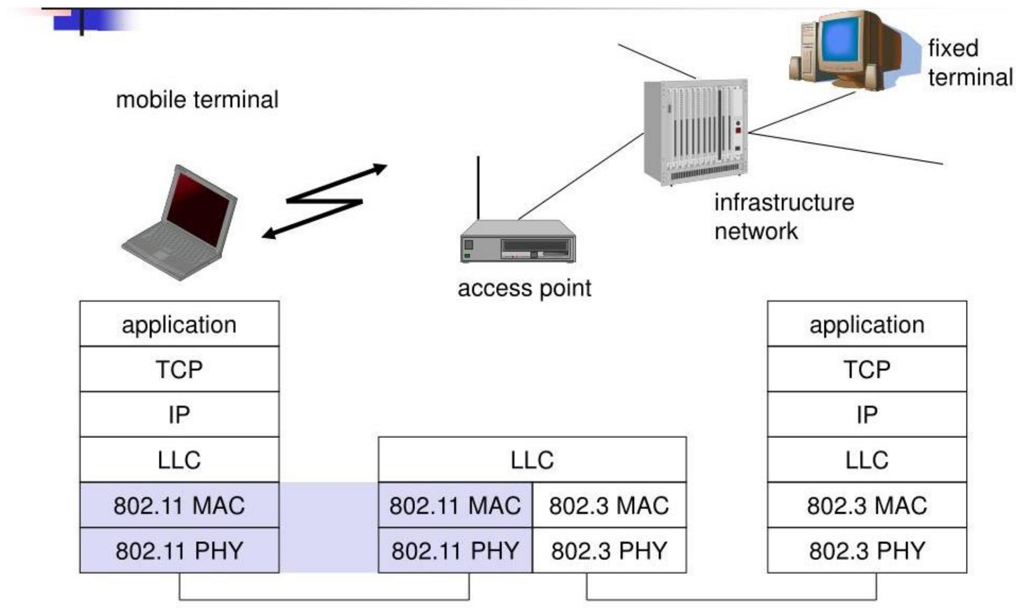
  2. **Independent Basic Service Set (IBSS):**

     Group of stations using the same radio frequency



4.4 802.11 Protocol Architecture

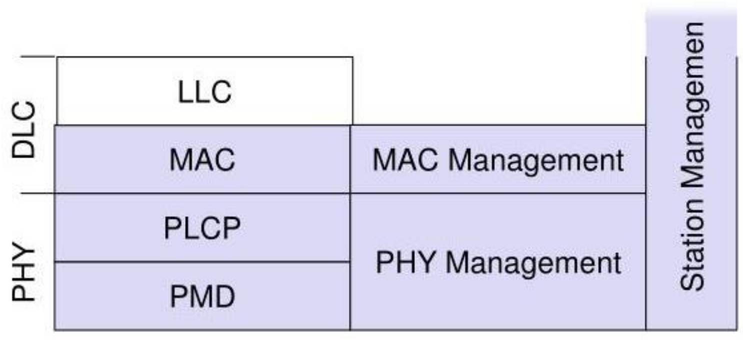
- The 802.11 standards cover definitions for both MAC (medium access control) and Physical Layer.
- The standard currently defines a single MAC while interacts with three PHYs as follow:
    1. Frequency Hopping Spread Spectrum
    2. Direct Sequence Spread Spectrum
    3. Infrared.
- PHY layer one based on infrared and two layers based on radio transmission.



IEEE 802.11 Sub layers Architecture and Management

• Physical Layer Convergence Procedure (PLCP)

• Physical Medium Dependent (PMD):



4.4.1 802.11 Protocol Architecture: Physical Layer Architecture

- The architecture of the Physical layer comprises of the two sub layers for each station:

1. Physical Layer Convergence Procedure (PLCP):

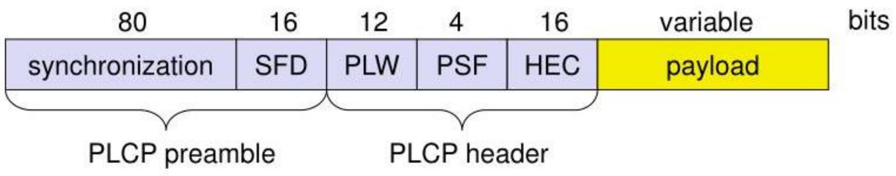
- PLCP sub layer is responsible for the Carrier Sense (CS) part of the Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) protocol.
- PLCP layer prepares the MAC Protocol Data Unit (MPDU) for transmission.
- The PLCP also delivers the incoming frames from the wireless medium to the MAC layer.
- PLCP appends fields to the MPDU that contains information needed by the physical layer transmitter and receiver.
- This frame is called PLCP Protocol Data Unit (PPDU).
- The structure of PLCP provides for asynchronous transfer of MPDU between stations.
- The PLCP header contains logical information that allows the receiving stations physical layer to synchronize with each individual incoming packet.

2. Physical Medium Dependent (PMD):

- The PMD provides the actual transmission and reception of physical layer entities between stations through the wireless media.
- This sub layer provides the modulation/demodulation of the transmission.

4.4.1.1 Frequency Hopping Spread Spectrum (FHSS):

- In FHSS mode this layer carries the clocking information to synchronize the receiver clock with the clock of the transmitted packet.
- Below figure depicts the FHSS PPDU packet.



the fields in the FHSS PLCP are as follows:

SYNC:

- This field is made up of alternate zeroes and ones.
- This bit pattern is to synchronize the clock of the receiver.

Start Frame Delimiter (SFD):

- This field indicates the beginning of the frame and the content of this field is fixed and always is 0000110010111101.

PLCP Signaling (PSF):

- This field contains information about the data rate of the fields form whitened PSDU.
- The PLCP preamble is always transmitted at 1Mbps irrespective of the data rate of the wireless LAN.
- This field contains information about the speed of the link.
- For example 0000 means 1 Mbps and 0111 signifies 4.5 Mbps bandwidth.

PSDU Length Word (PLW):

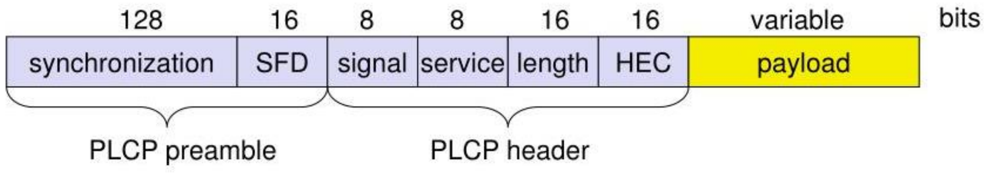
- This field specifies the length of the PSDU in octets.

Header Error Check (HEC):

- This field contains the CRC (Cyclic Redundancy Check) according to CCITT CRC-16 algorithm.
- FHSS PMD is responsible for converting the binary bit sequence into analog signal and transmit the PPDU frame into the air.
- FHSS PDM does this using the frequency hopping technique.
- The 802.11 standard defines a set of channels within the ISM band for frequency hopping.
- Once the hopping sequence is set in the access point, stations automatically synchronize to the correct hopping sequence.

4.4.1.2 Direct Sequence Spread Spectrum (DSSS):

- DSSS PLCP is responsible for synchronizing and receiving the data bits correctly.
- Below show a figure depicts the DSSS PPDU packet.



The fields in the DSSS PLCP are as following:

SYNC:

- This field is made up of alternate zeroes and ones.
- This bit pattern is to synchronize the clock of the receiver with the received frame.

Start Frame Delimiter (SFD):

- This field indicates the beginning of the frame and the content of this field is fixed and is always 1111001110100000.

Signal:

- This field defines the type of modulation the receiver must use to demodulate the signal.
- When the value of this field is multiplied by 100 Kbps, we get the bandwidth of the transmission,
- The PLCP preamble and the header are always transmitted at 1 Mbps.
- The bandwidth defined by this field applies to MPDU field.

Service:

- This field is not used and is usually 0.

Length:

- This field contains an unsigned 16-bit integer indicating the length of the frame.
- However, unlike the FHSS this is not in octets.
- It is rather in microseconds.
- The receiver will use this to synchronize with the clock to determine the end of frame.

Frame Check Sequence:

- This is a 16-bit checksum based on CCITT CRC-16 algorithm.
- **DSSS PMD** translates the binary digital sequence into analog radio signals and transmits the PPDU frame into the air.
- DSSS physical layer operates within the ISM band.
- If we take the 2.4 GHz band then it is between 2.4 GHz and 2.8435 GHz frequency band divided into multiple channels with 22 MHz width.

4.4.1.3 Infrared:

- The PHY layer, which is based on infrared (IR) transmission, uses near visible light at 850-950 nm.
- The standard to require a line-of-sight between sender and receiver.
- This allows for point-to-multipoint communication.
- The maximum range is about 10 m if no sunlight or hear source interfere with the transmission,

- Typically, such a network will only work in buildings, e.g. classrooms, meeting room etc.
- Today no products are available that offer infra red communication based on 802.11.

4.4.2 802.11 Protocol Architecture: MAC Layer Architecture

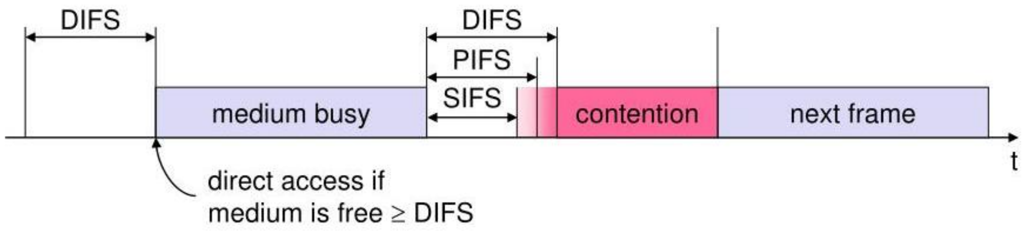
- The MAC layer has to fulfill several tasks.
- First of all it has to control medium access, but it can also offer support for roaming, authentication and power conservation.
- Two basic services provided by the MAC layer:
    1. **Asynchronous Data Service (mandatory)**
        - exchange of data packets based on “best-effort”
        - support of broadcast and multicast
    2. **Time-Bounded Service (optional)**
        - implemented using PCF (Point Coordination Function)

The following three basic access mechanisms have been defined for IEEE 802.11:

- **DFWMAC-DCF CSMA/CA (Distributed foundation wireless medium access control-Distributed coordination function)- (mandatory)**
    1. collision avoidance via randomized „back-off“ mechanism
    2. minimum distance between consecutive packets
    3. ACK packet for acknowledgements (not for broadcasts)
- **DFWMAC-DCF with RTS/CTS (optional)**
    1. Distributed Foundation Wireless MAC
    2. avoids hidden terminal problem
- **DFWMAC- PCF (Point coordination function) (optional)**
    1. access point polls terminals according to a list

Priorities:

- The standard defines 4 types of spacing intervals.
- These are called Inter Frame Spaces (IFS).
- Below show a figure Medium access and inter-frame spacing.



IFSs are used to defer a station's access to the medium and provide various levels of priorities:

Short Inter Frame Space (SIFS):

- it is the shortest Inter Frame Space with the highest priority.
- RTS, CTS & Acknowledge use SIFS intervals.
- SIFS value is a fixed value per PHY and is calculated in such a way that the transmitting station will be able to switch back to receive mode and be capable of decoding the incoming packet.

Point Coordination IFS (PIFS):

- It has medium priority to gain access to the medium.
- It is used for time-bounded service using PCF

Distributed IFS (DIFS):

- It has lowest priority to gain access to the medium.
- It is used for asynchronous data service using PCF

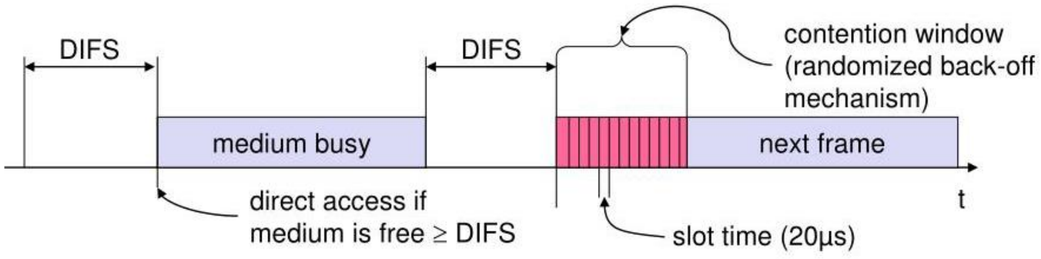
4.4.2.1 Basic DFWMAC-DCF (Distributed foundation wireless medium access control-Distributed coordination function) using CSMA/CA access method –I:

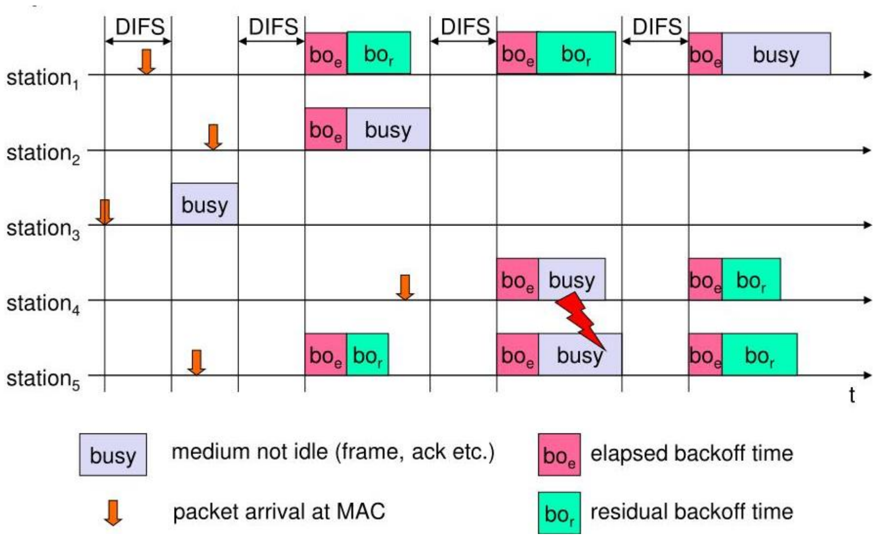
- station ready to send starts sensing the medium (Carrier Sense based on CCA, Clear Channel Assessment)
- if the medium is free for the duration of an Inter-Frame Space (IFS), the station can start sending (IFS depends on service type)

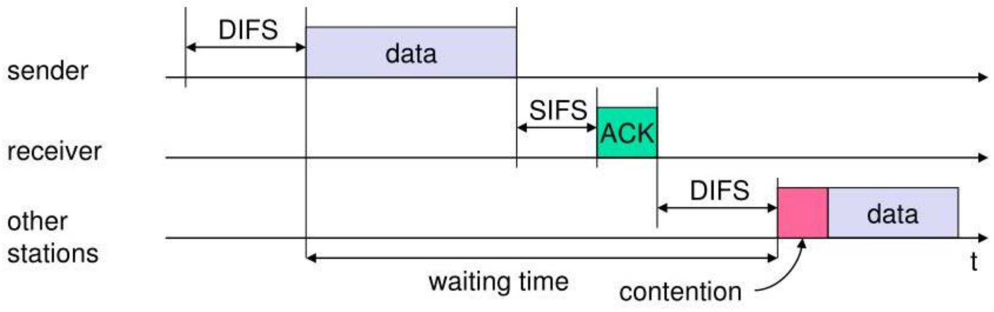
Basic DFWMAC-DCF using CSMA/CA access method –I:

- if the medium is busy, the station has to wait for a free IFS, then the station must additionally wait a random back-off time (collision avoidance, multiple of slot-time)
- if another station occupies the medium during the back-off time of the station, the back-off timer stops (fairness)

Basic DFWMAC-DCF using CSMA/CA access method –I:







4.4.2.2 Basic DFWMAC-DCF using CSMA/CD access method –II (Sending unicast packets):

- station has to wait for DIFS before sending data
- receivers acknowledge at once (after waiting for SIFS) if the packet was received correctly (CRC)
- automatic retransmission of data packets in case of transmission errors

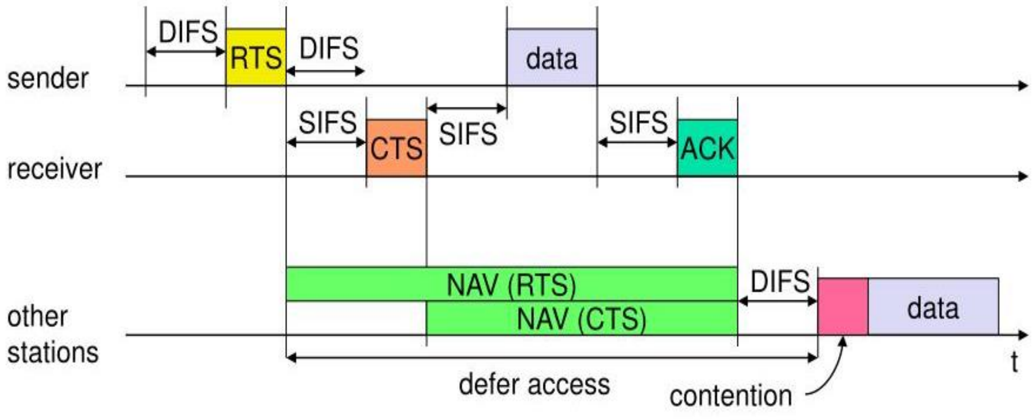
DFWMAC-DCF with RTS/CTS (Sending unicast packets):

- This access method to solve a hidden Terminal problem.
- Station can send RTS with reservation parameter after waiting for DIFS.

- acknowledgement via CTS after SIFS by receiver (if ready to receive). Sender can now send data at once, acknowledgement via ACK.
- other stations store medium reservations distributed via RTS and CTS.

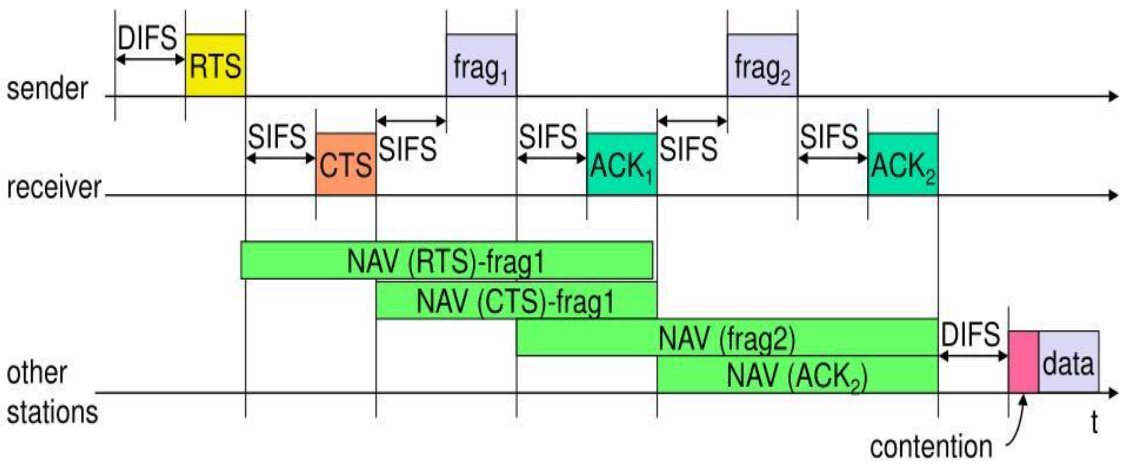
DFWMAC-DCF with RTS/CTS (Sending unicast packets):

- RTS-Request to send, CTS-Clear to send,
- NAV- net allocation vector



DFWMAC-DCF with RTS/CTS (Fragmentation):

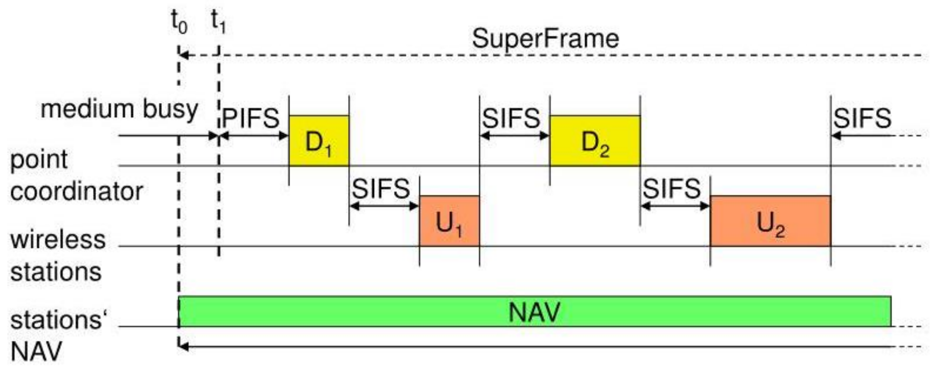
- the probability of an erroneous frame is much higher for wireless links assuming the same frame length.
- One way to decrease the error probability of frames is to use shorter frames.
- The solution is to use mechanism of fragmenting a user data packet into several smaller parts should be transparent for a user.
- the IEEE 802.11 standard specifies a fragmentation mode.
- This mechanism show in below.



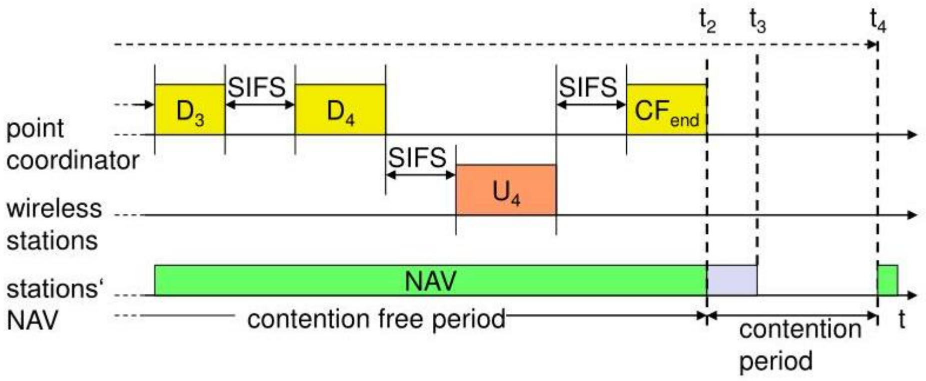
4.4.2.3 DFWMAC- PCF with polling:

- The two access mechanisms presented so far cannot guarantee a maximum access delay or minimum transmission bandwidth.
- To provide a time-bounded service the standard specifies a point coordination function (PCF) on top of the standard DCF (control-Distributed coordination function).
- The point coordinator in the access point splits the access time into super frame periods show in below.

DFWMAC- PCF (Point coordination function) -I with polling (optional)



DFWMAC- PCF-II with polling

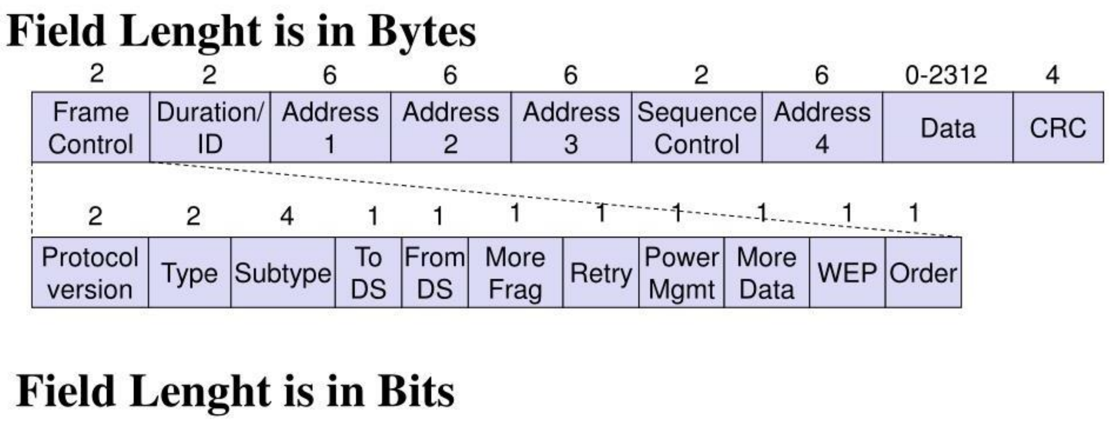


4.4.2.4 802.11 Protocol Architecture: MAC Frames

- In the next slide figure show the basic structure of an IEEE 802.11 MAC data frame together with content of the frame control field.
- The first figure fields in the figure refer to the following:

  Frame control:

  - The first 2 bytes serve several purpose.
  - Like: Protocol version, power management etc.



802.11 protocol Architecture: MAC Packet Structure

Duration/ID:

- If the field value is less than 32,768 the duration filed contains the value indicating the period of time in which the medium is occupied.
- It also used for reserved for identifiers.

Address 1 to 4:

- The four address fields contain standard IEEE 802 MAC address (48 bit each).
- The meaning of each address depends on the Distribution system bits in the frame control field.

Sequence control:

- Due to the acknowledgement mechanism frames may be duplicated.
- Therefore a sequence number is used to filter duplicates.

Data:

- The MAC frame may contain arbitrary data which is transferred from a sender to the receiver (s).

Checksum (CRC):

- Finally a 32, bit checksum is used to protect the frame.
- The frame control filed shown in the previous slide figure contains the following fields:

Protocol version:

- This 2 bits field indicates the current protocol version and is fixed to 0, If major revisions to the standard make it incompatible with the current version, this value will be increased.

Type:

- the type field determines the function of a frame: management (=00),control(=01),data(=10).
- Each type has several subtypes as indicated in the following field.

Subtype:

- Example, subtype for management frames are: 0000 for associated request, RTS us a control frame with subtype 1011, same as CTS coded as 1100.
- User data is transmitted as data frame with subtype 0000.

To DS/From DS:

- To indicate which Distribution system sender and receiver identification.

More fragments:

- This filed is set to 1 in all data or management frames that have another fragment of the current MAC service data unit (MSDU) to follow.

Retry:

- If the current frame is a retransmission of an earlier frame, this bit is set to 1.
- With the help of this bit it may be simpler for receivers to eliminate duplicate frames.

Power management:

- this field indicates the mode of a station after successful transmission of a frame.
- Set to 1 the field indicates that the station goes into power-save mode.
- If the field is set to 0 the satiation stay active.

More data:

- in general, this field is used to indicate a receiver that a receiver that a sender has more data to send that the current frame.
- This can be used by an access point to indicate to a station in power-save mode that more packets are buffered.

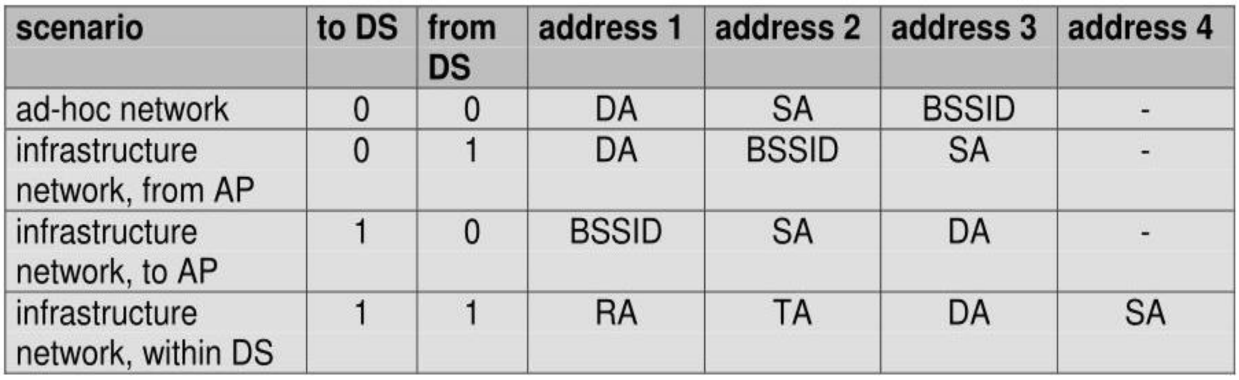
Wired equivalent privacy (WEP):

- This field indicates that the standard security mechanism of 802.11 is applied.

Order:

- If this bit is set to 1 the received frames must be processed in strict order.

802.11 Protocol Architecture: MAC address format



scenario	to DS	from DS	address 1	address 2	address 3	address 4
ad-hoc network	0	0	DA	SA	BSSID	-
infrastructure network, from AP	0	1	DA	BSSID	SA	-
infrastructure network, to AP	1	0	BSSID	SA	DA	-
infrastructure network, within DS	1	1	RA	TA	DA	SA

DS: Distribution System

AP: Access Point

DA: Destination Address

SA: Source Address

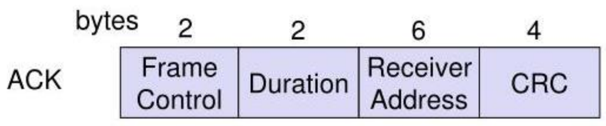
BSSID: Basic Service Set Identifier

RA: Receiver Address

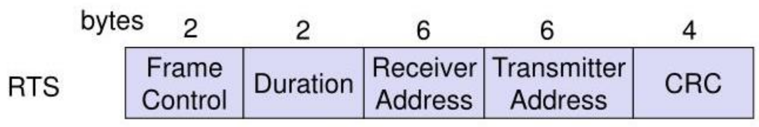
TA: Transmitter Address

802.11 protocol Architecture: Special Frames: ACK, RTS, CTS

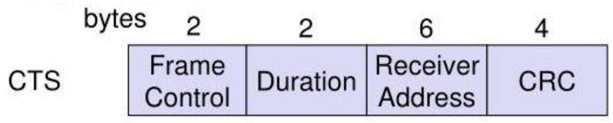
1. **Acknowledgement**



2. **Request To Send**



3. **Clear To Send**



4.4.2.5 MAC Management

- **Synchronization:**
  1. Try to find a LAN, try to stay within a LAN.
  2. Synchronization of inter clocks, generation of beacon signals etc.
- **Power management:**

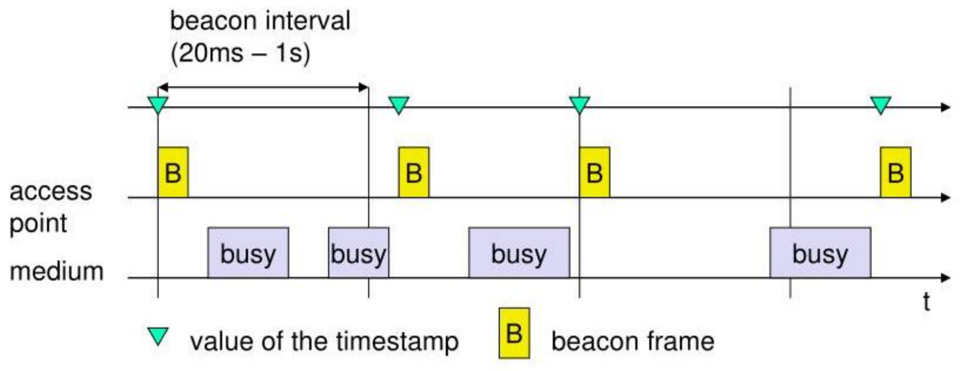
1. Functions to control transmitter activity for power conservation e.g. sleep-mode without missing a message, periodic sleep, frame buffering, traffic measurements.

- **Roaming:**
    1. roaming, i.e. change networks by changing access points.
    2. scanning, i.e. active search for a network.
- **MIB - Management Information Base**
    1. managing, read, write the current status of wireless station.

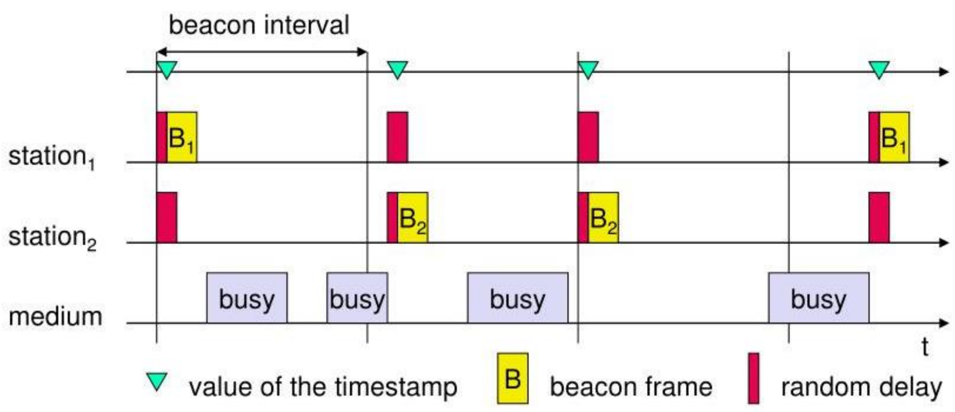
4.4.2.5.1 MAC Management: Synchronization

- Station need to maintain synchronization.
- This is necessary to keep hopping and other functions like power saving synchronized.
- On an infrastructure BSS, synchronization is achieved by all the stations updating their clocks according to the AP's clock.
- The AP periodically transmits frames called Beacon frames.
- These frames contain the value of the AP's clock at the moment of transmission. This is the time when physical transmission actually happens, and not when the packet was put in the queue for transmission.
- the receiving stations check the value of their clocks at the moment the signal is received, and correct it to keep in synchronization with the AP's clock.
- This prevents clock drifting which would cause loss of synch after a few hours of operations.

Synchronization using a Beacon (infrastructure)



Synchronization using a Beacon (ad-hoc)



4.4.2.5.2 MAC Management: Power Management

- Idea: switch the transceiver off if not needed
- States of a station: sleep and awake
- Timing Synchronization Function (TSF)
  - ➢ stations wake up at the same time

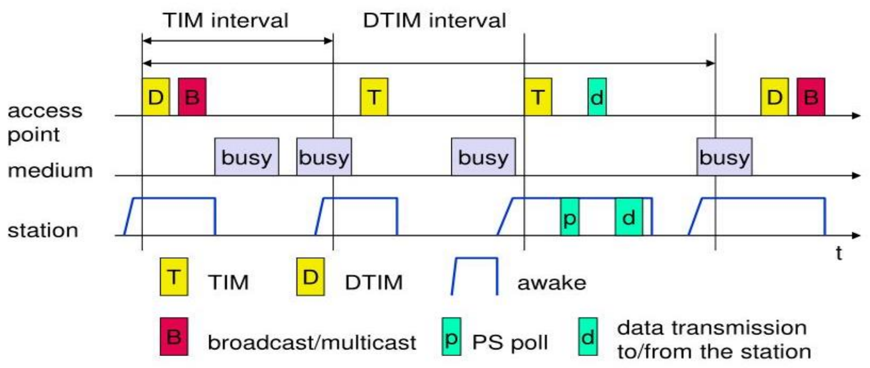
1. Infrastructure

- ➢ Traffic Indication Map (TIM)
  - list of unicast receivers transmitted by AP
- ➢ Delivery Traffic Indication Map (DTIM)
  - list of broadcast/multicast receivers transmitted by AP

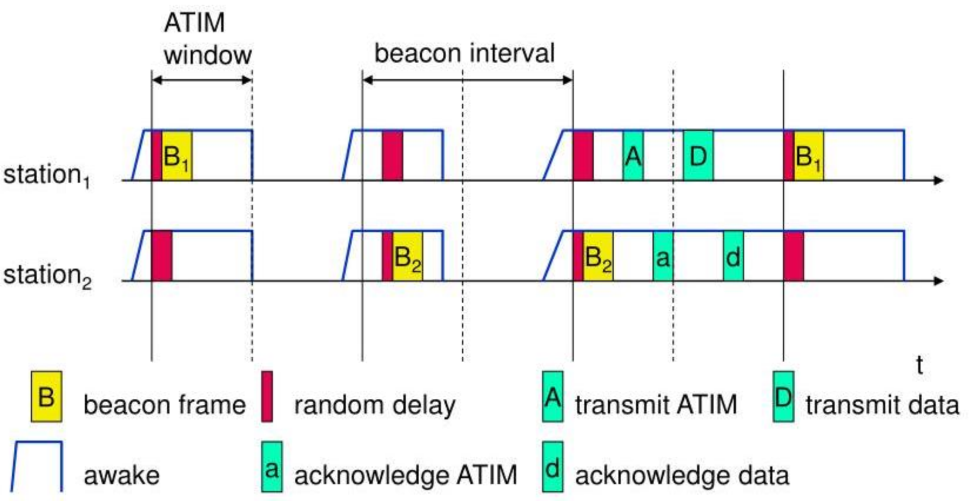
2. Ad-hoc

- ➢ Ad-hoc Traffic Indication Map (ATIM)
  - announcement of receivers by stations buffering frames
  - more complicated - no central AP
  - collision of ATIMs possible (scalability)
- ➢ APSD (Automatic Power Save Delivery)
  - new method in 802.11e replacing above schemes

Power saving with wake-up patterns (infrastructure)



Power saving with wake-up patterns (ad-hoc)



4.4.2.5.3 MAC Management: Roaming

- The characteristic of a mobile network that enables correct call routing when the subscriber move from one network to another or form one cell to another.
- Traveling from the range of one access point to another.
- The term "handover" or "handoff" as used in the context of mobile or cellular cell.
- However, for WLANs roaming is more common.
- The steps for roaming between access points are:

1. Scanning

- ➢ scan the environment, i.e., listen into the medium for beacon signals or send probes into the medium and wait for an answer.

2. Reassociation Request

- ➢ station sends a request to one or several AP(s).

3. Reassociation Response.

- ➢ success: AP has answered, station can now participate.
- ➢ failure: continue scanning.

4. AP accepts Reassociation Request

- ➢ signal the new station to the distribution system
- ➢ the distribution system updates its data base (i.e., location information).
- ➢ typically, the distribution system now informs the old AP so it can release resources.

5. Fast roaming – 802.11r

- ➢ e.g. for vehicle-to-roadside networks

4.5 Bluetooth

- Bluetooth is universal for short-range wireless voice and data communication. It is a Wireless Personal Area Network (WPAN) technology and is used for exchanging data over smaller distances.
- This technology was invented by Ericson in 1994. It operates in the unlicensed, industrial, scientific, and medical (ISM) band from 2.4 GHz to 2.485 GHz. Maximum devices that can be connected at the same time are 7.
- Bluetooth ranges up to 10 meters. It provides data rates up to 1 Mbps or 3 Mbps depending upon the version. The spreading technique that it uses is FHSS (Frequency-hopping spread spectrum). A Bluetooth network is called a **piconet** and a collection of interconnected piconets is called **scatternet**.

What is Bluetooth?

Bluetooth simply follows the principle of transmitting and receiving data using radio waves. It can be paired with the other device which has also Bluetooth but it should be within the estimated communication range to connect. When two devices start to share data, they form a network called piconet which can further accommodate more than five devices.

Points to remember for Bluetooth:

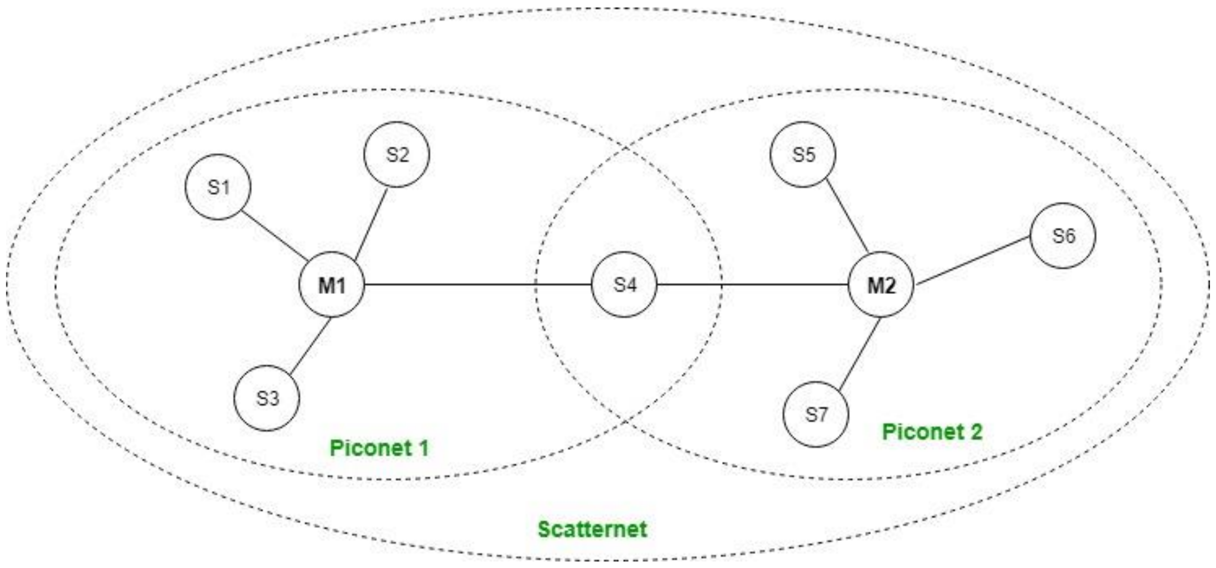
- Bluetooth Transmission capacity 720 kbps.
- Bluetooth is Wireless.
- Bluetooth is a Low-cost short-distance radio communications standard.
- Bluetooth is robust and flexible.
- Bluetooth is cable replacement technology that can be used to connect almost any device to any other device.
- The basic architecture unit of Bluetooth is a piconet.

Bluetooth Architecture:

The architecture of Bluetooth defines two types of networks:

1. Piconet

2. Scatternet



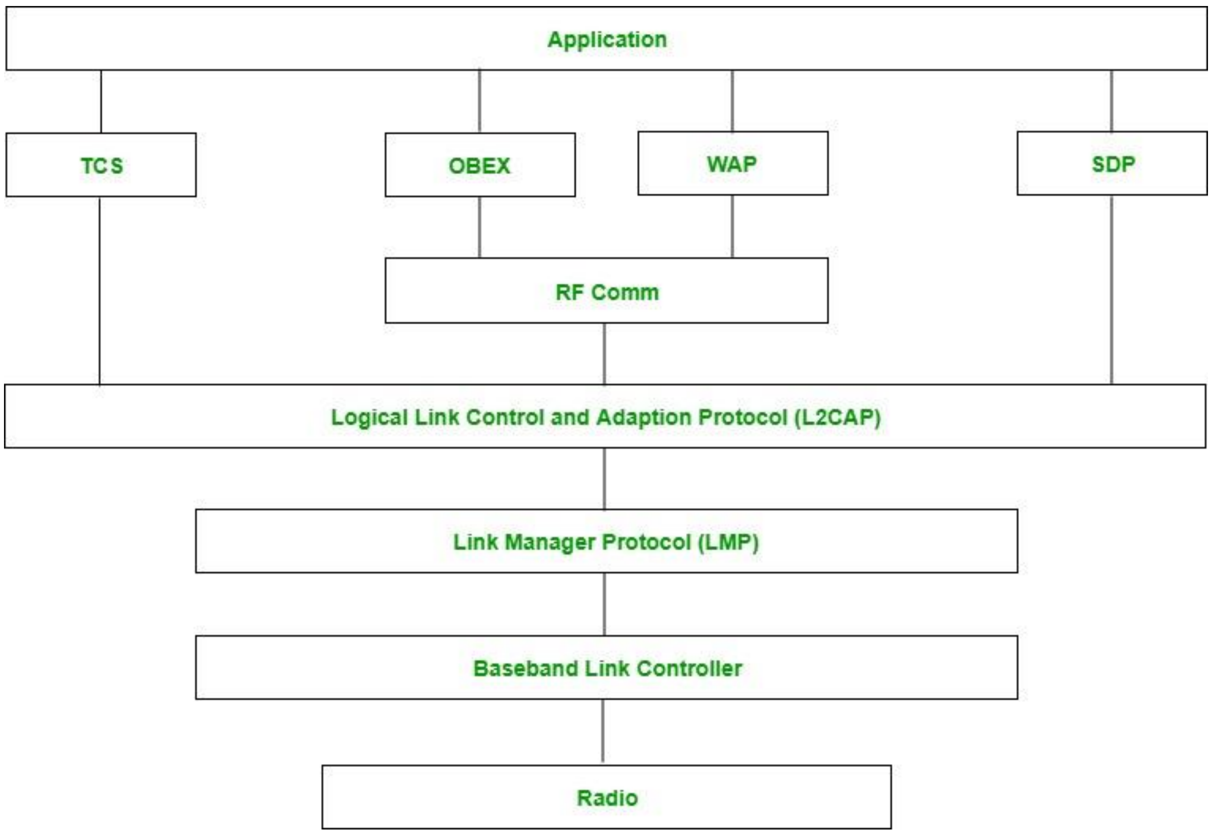
Piconet:

- Piconet is a type of Bluetooth network that contains **one primary node** called the master node and **seven active secondary nodes** called slave nodes.
- Thus, we can say that there is a total of 8 active nodes which are present at a distance of 10 meters.
- The communication between the primary and secondary nodes can be one-to-one or one-to-many.
- Possible communication is only between the master and slave; Slave-slave communication is not possible.
- It also has **255 parked nodes**, these are secondary nodes and cannot take participation in communication unless it gets converted to the active state.

Scatternet:

- It is formed by using **various piconets**. A slave that is present in one piconet can act as master or we can say primary in another piconet.
- This kind of node can receive a message from a master in one piconet and deliver the message to its slave in the other piconet where it is acting as a master.
- This type of node is referred to as a bridge node. A station cannot be mastered in two piconets.

Bluetooth protocol stack:


Application
TCS
OBEX
WAP
SDP
RF Comm
Logical Link Control and Adaption Protocol (L2CAP)
Link Manager Protocol (LMP)
Baseband Link Controller
Radio

1. **Radio (RF) layer:** It specifies the details of the air interface, including frequency, the use of frequency hopping and transmit power. It performs modulation/demodulation of the data into RF signals. It defines the physical characteristics of Bluetooth transceivers. It defines two types of physical links: connection-less and connection-oriented.
2. **Baseband Link layer:** The baseband is the digital engine of a Bluetooth system and is equivalent to the MAC sublayer in LANs. It performs the connection establishment within a piconet, addressing, packet format, timing and power control.
3. **Link Manager protocol layer:** It performs the management of the already established links which includes authentication and encryption processes. It is responsible for creating the links, monitoring their health, and terminating them gracefully upon command or failure.
4. **Logical Link Control and Adaption (L2CAP) Protocol layer:** It is also known as the heart of the Bluetooth protocol stack. It allows the communication between upper and lower layers of the Bluetooth protocol stack. It packages the data packets received from upper layers into the form expected by lower layers. It also performs segmentation and multiplexing.

5. **Service Discovery Protocol (SDP) layer:** It is short for Service Discovery Protocol. It allows discovering the services available on another Bluetooth-enabled device.
6. **RF comm layer:** It is a cabal replacement protocol. It is short for Radio Frontend Component. It provides a serial interface with WAP and OBEX. It also provides emulation of serial ports over the logical link control and adaption protocol(L2CAP). The protocol is based on the ETSI standard TS 07.10.
7. **OBEX:** It is short for Object Exchange. It is a communication protocol to exchange objects between 2 devices.
8. **WAP:** It is short for Wireless Access Protocol. It is used for internet access.
9. **TCS:** It is short for Telephony Control Protocol. It provides telephony service. The basic function of this layer is call control (setup & release) and group management for the gateway serving multiple devices.
10. **Application layer:** It enables the user to interact with the application.

Types of Bluetooth

Various types of Bluetooth are available in the market nowadays. Let us look at them.

- **In-Car Headset:** One can make calls from the car speaker system without the use of mobile phones.
- **Stereo Headset:** To listen to music in car or in music players at home.
- **Webcam:** One can link the camera with the help of Bluetooth with their laptop or phone.
- **Bluetooth-equipped Printer:** The printer can be used when connected via Bluetooth with mobile phone or laptop.
- **Bluetooth Global Positioning System (GPS):** To use GPS in cars, one can connect their phone with car system via Bluetooth to fetch the directions of the address.

Advantage:

- It is a low-cost and easy-to-use device.
- It can also penetrate through walls.
- It creates an Ad-hoc connection immediately without any wires.
- It is used for voice and data transfer.

Disadvantages:

- It can be hacked and hence, less secure.
- It has a slow data transfer rate: of 3 Mbps.
- It has a small range: 10 meters.

- Bluetooth communication does not support routing.
- The issues of handoffs have not been addressed.

Applications:

- It can be used in laptops, and in wireless PCs, printers.
- It can be used in wireless headsets, wireless PANs, and LANs.
- It can connect a digital camera wirelessly to a mobile phone.
- It can transfer data in terms of videos, songs, photographs, or files from one cell phone to another cell phone or computer.
- It is used in the sectors of Medical health care, sports and fitness, Military.

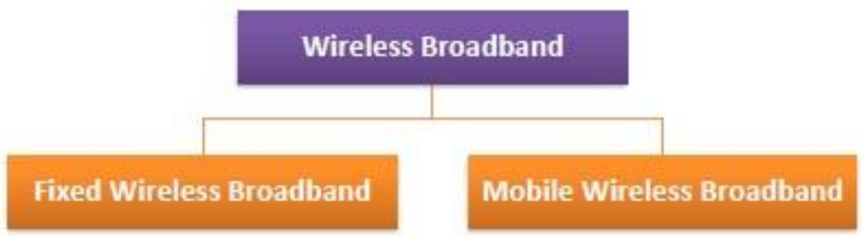
4.6 wireless broadband (WiBB)

Wireless broadband (WiBB) a networking technology designed to impart highspeed Internet and data service through wireless networks. Wireless broadband may be delivered through wireless local area networks (WLANs) or wide area networks (WWANs).
Similar to other wireless services, wireless broadband can be either fixed or mobile.

Features of WiBB

- WiBB is similar to wired broadband service since they connect to an internet backbone, with the difference that they use radio waves instead of cables to connect to the last mile of the network.
- The range of most broadband wireless access (BWA) services varies around 50 km from the transmitting tower.
- Download speeds provided by some wireless Internet service providers (WISPs) are over 100 Mbps.
- WiBB mostly provides asymmetrical data rates for downloads and uploads.
- WiBB may also be symmetrical, i.e. they have the same data rate in both downstream as well as upstream. This is most seen only in fixed wireless networks.
- Any device connected to WiBB needs to be equipped with a wireless adapter to translate data into radio signals which can be then transmitted using an antenna.

Types of WiBB



1. Fixed Wireless Broadband

Fixed WiBB provides wireless Internet services for devices located in more or less fixed locations, like homes and offices. The services are comparable to those provided through digital subscriber line (DSL) or cable modem, with the difference that it has wireless mode of transmission.

The two main technologies used in fixed WiBB are −

- LMDS (Local Multipoint Distribution System)
- MMDS (Multichannel Multipoint Distribution Service) systems

2. Mobile Wireless Broadband

Mobile WiBB, also called mobile broadband, provides high – speed broadband the connection from mobile phone service providers which is accessible from random locations.

The locations are within the coverage area of the phone towers of mobile service provider and the connections are subject to monthly service plan subscribed by the user.

Mobile broadband can be costlier due to its portability. Also, they generally have varying or limited speed except in urban areas.

4.7 Satellites networks

A satellite network has a ground-based station that uses a transceiver to control it. The network also has ground stations for users to send and receive communication through the satellite system.

How Does it Work

A satellite is a device in space that can receive signals from Earth and send them back to the same place or a different location.

Satellite systems send data at high speeds of the GigaHertz level, are costly, and are often used by large companies or institutions. It saves distance and time for a company with branches at many points.

Components of a Satellite Network

a. Transponders

Transponder devices receive and send signals. They boost the signals before returning them to the ground and change the frequency to avoid interference.

b. Earth Stations

Ground stations control the satellite reception and regulate interconnection between terminals. It also manages the output channels, encodes the data, and controls the transfer rate. There are three receiving stations, an antenna, and a broadcast station.

1. Receiving station

The receiving station gets all the info from the transmitting station and sends it to the satellite.

2. Antenna

The antenna picks up the signal from the satellite and sends it to the feeder at the focal point. A good antenna can reduce interference and noise. Antennas have devices for receiving and transmitting.

By adjusting the antennas' signal models, we can only cover one country's area globally.

3. Broadcast Station

A broadcast station has a transmitter and antenna. It has the reliable power to send signals well. The antenna gets the call and sends it to the satellite with the proper modulation and carrier.

- Air is the primary medium used for physical transmission without guidance. Microwave signals sent to satellites remain strong even when it's raining. It can also be low or high frequency between 100 MHz and 10 GHz.
- **Satellite broadcasts have two parts:** the bus and the payload. The bus controls how the payload works. The load carries the user's information.
- Satellite TV only sends signals one way. Sometimes it's essential to have one place that sends signals to a satellite. Additionally, many businesses receive calls from it.
- Ground stations can send and receive signals for different types of two-way services. Small antennas and low-power transmitters can serve many users in satellite services.

Satellite Rise Model

- The main component of a satellite system in the rise model is the ground station transmitter. A transmitter has four parts: an IF modulator, an IF-RF microwave converter. It also includes a high-power amplifier and a spectrum band limiter.

- The IF modulator changes the input signals, and the converter adjusts them for satellite transmission. The HPA makes sure the signal is strong enough to reach the transponder.

Satellite Transponder

- A transponder has input and output band-limiting devices, amplifiers, and a frequency converter.
- The transponder acts as a repeater that works like a microwave repeater. Other transponder setups can also repeat signals.
- The BPF input limits the total noise applied to the input of the LNA and is a tunnel diode. LNA's output goes to a frequency converter, an oscillator, and a BPF. The BPF changes from a high frequency to a low frequency.
- The TWT amplifier boosts the RF signal for better transmission to ground station receivers. SSP (Solid-State Amplifier) can also get a better consistent level than TWTs. Also, SSP devices can produce power at 50 watts at the greatest, while TWTs are 200 watts.

Drop Model

- A ground station receiver has an input BPF, an LNA, and an RF to IF converter device, and the BPF works by limiting the power of the input noise to the LNA.
- LNA, tunnel diode amplifier, or parametric amplifier, is a sensitive, low-noise device. The RF-to-IF converter acts as a combination scrambler filter that converts the RF signal into an IF frequency.

UNIT V MOBILE NETWORK AND TRANSPORT LAYERS

Mobile IP – DHCP – Routing in Mobile ad hoc networks – TCP improvements – TCP over 2.5/3G.

5.1 MOBILE IP:

- Mobile IP is an Internet Engineering Task Force (IETF) standard communications protocol that enhances the existing IP to accommodate mobility.
- Mobile IP allows mobile computers to stay connected to the internet regardless of their location & without having to continually change their IP address.
- Every mobile user needs continuous network connectivity irrespective of his physical location. The traditional IP does not support user mobility.
- Mobile IP was created by extending IP to enable users to keep the same IP address while travelling to different networks.

Advantages of using Mobile IP:

- It allows **fast, continuous low-cost access** to corporate networks in remote areas where there is no public telephone system or cellular coverage.
- It supports a **wide range of applications** from Internet access and e-mail to ecommerce.
- Users can be permanently **connected** to their Internet provider and **charged only** for the data packets that are sent and received
- It can move from one type of medium to another without losing **connectivity**

Disadvantage of Mobile IP:

- Routing inefficiency problem caused by the "**triangle routing**" formed by the home agent, correspondent host, and the foreign agent.
- Security risks are the most important problem facing Mobile IP.
- Problem in making Mobile IP coexist with the security features within the Internet.

5.1.1 GOALS, ASSUMPTIONS & REQUIREMENTS

Goal of a mobile IP:

Supporting end-system mobility while maintaining scalability, efficiency, and compatibility in all respects with existing applications and Internet protocols.

Requirements of Mobile IP:

- **Compatibility:** A new standard cannot require changes for applications or network protocols already in use.
- **Transparency:** Mobility should remain “invisible” for many higher layer protocols and applications.
- **Scalability and efficiency**: Introducing a new mechanism into the Internet must not degrade the efficiency of the network.
- **Security**: All messages used to transmit information to another node about the location of a mobile node must be authenticated to protect against remote redirection attacks.

5.1.2 MOBILE IP - ENTITIES AND TERMINOLOGIES

1. **Mobile Node (MN):**
    - Device that moves from home network to Foreign network
    - Node that can change the point of connection to the network without changing its IP address.
    - **Example:** laptops with antennas or mobile phones
2. **Home Address:**

    Permanent address of the MN in its original network i.e., IP address of MN
3. **Home Network**

    Original network the MN was associated with respect to its IP address before moving to new network. No mobile IP support is needed within the home network.
4. **Home Agent (HA)**
    - System in the home network of the MN, typically a router.
    - Provides several services for the MN : ü Tunnels IP datagrams to the COA. ü Maintains a location registry of MA.

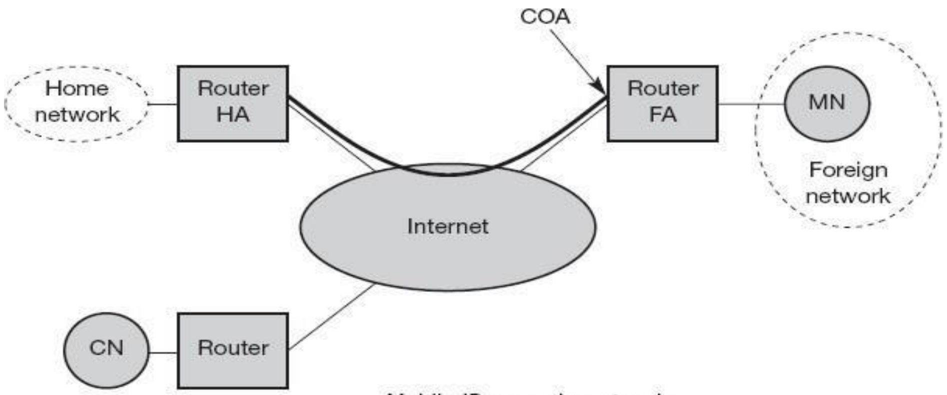
    Implementation of an HA:

    - Implemented on a router that is responsible for the home network.
    - Implemented on an arbitrary node in the subnet. Disadvantage: "Double crossing of the router" - A packet for the MN comes in via the router; the HA sends it through the tunnel which again crosses the router.

- The HA could be again on the ‘router’ but this time only acting as a manager for MNs belonging to a virtual home network.

5. **Foreign Agent (FA)**
   - System in the current foreign network of the MN, typically a router.
   - It can have COA
   - Provides several services to MN during its visit to the foreign network:
   - Forwards the tunnel datagrams to the MA.
   - Provides security services
6. **Foreign Network**

   A new network that MN visits and which is not the home network
7. **Correspondent Node (CN)**
   - Communication partner i.e., Node that wants to communicate with MN
   - At least one partner is needed for communication.
   - It can be a fixed or mobile node.



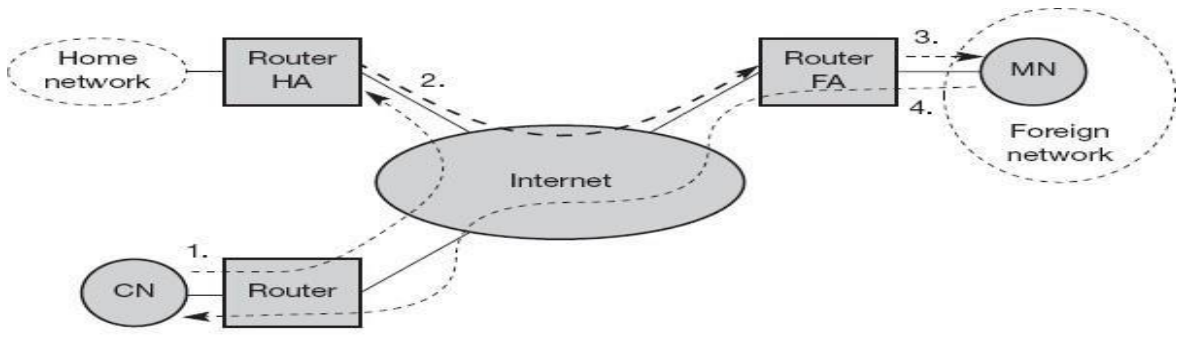
Mobile IP example network

8. **Care-of Address (COA)**
   - A new address of MN in the foreign network.
   - Two different possibilities for the location of the COA (Types of COA):

     (i) Foreign Agent COA - The static IP address of a foreign agent(FA) on a visited network

     (ii) Co-located COA - Temporary IP address assigned to the MN.
     - Represents the current position of the MN on the Foreign network & can be used by only one MN at a time.
     - A co-located care-of address can be obtained by Dynamic Host Configuration Protocol (DHCP).

5.1.3 IP PACKET DELIVERY (HOW MOBILE IP WORKS?)



Packet delivery to and from the mobile node

Steps used in the operation of mobile IP:

STEP 1:

- CN sends the Packet to the IP address home address) of the MN.

STEP 2:

- Internet Routes the Packet to the router of the MN’s home network.
- The HA examines the packet to find whether the MN is present in its current home network or not.
- If the MN is not present, then the HA encapsulates that datagram in a new packet.

STEP 3:

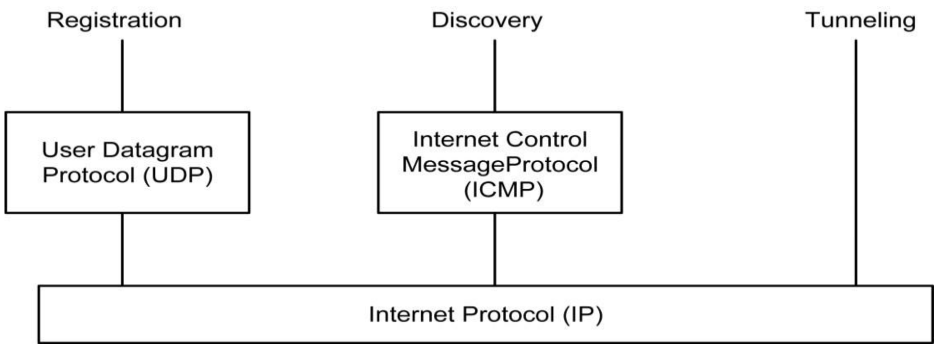
- The encapsulated packet is tunneled to the FA, which act as the new destination address.
- Then FA performs decapsulation to remove the additional header Then forwards the decapsulated packet to the MN.

STEP 4:

- MN after receiving the packet from CN forwards a reply packet to the CN by specifying its own IP address along with the address of the CN.

5.1.4 KEY MECHANISMS IN MOBILE IP (MOBILE IP OPERATION STAGES)

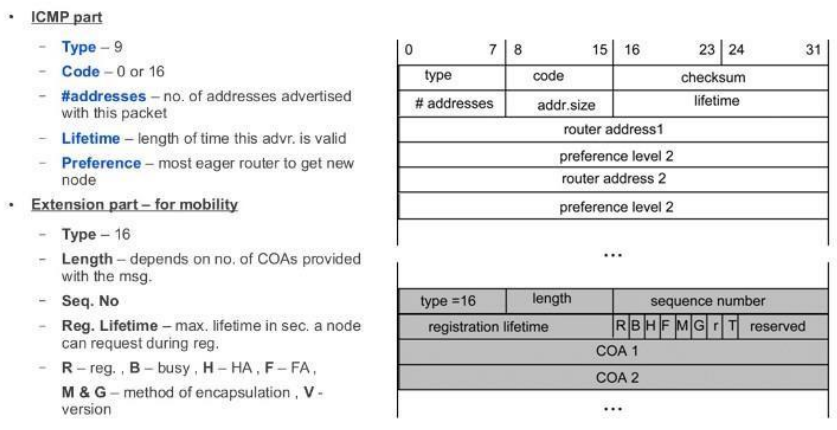
a) Agent Discovery
b) Registration
c) Tunneling & Encapsulation


Registration
Discovery
Tunneling
User Datagram Protocol (UDP)
Internet Control MessageProtocol (ICMP)
Internet Protocol (IP)

5.1.4.1 AGENT DISCOVERY

- A MN uses a discovery procedure to identify prospective home and foreign agents.
- Task of MN to determine its FA & HA:
    1. Both HA & FA periodically broadcast Agent Advertisement message.
    2. A MN must discover a HA before it leaves to a home network. iii) A MN must also discover a FA after it moved to a foreign network Ø Uses ICMP Router Discovery Protocol (IRDP).
- ICMP Router Discovery Protocol (IRDP) - Enables host to broadcast or multicast to discover the IP address (i.e., COA) of their neighbouring routers (i.e., FA)
- **Agent Discovery methods**: (i) Agent Advertisement (ii) Agent Solicitation.

a) Agent advertisement



- ICMP part
    - **Type** – 9
    - **Code** – 0 or 16
    - **#addresses** – no. of addresses advertised with this packet
    - **Lifetime** – length of time this advr. is valid
    - **Preference** – most eager router to get new node
- Extension part – for mobility
    - **Type** – 16
    - **Length** – depends on no. of COAs provided with the msg.
    - **Seq. No**
    - **Reg. Lifetime** – max. lifetime in sec. a node can request during reg.
    - **R** – reg. , **B** – busy , **H** – HA , **F** – FA , **M & G** – method of encapsulation , **V** - version

NOTE: Upper part represent ICMP while lower part represent extension neededfor

Agent advertisement packet Format

Functions:

1. It allows the MN to find whether an agent is its HA or a FA.
2. If it is FA then get the COA.
3. It allows the MN to know the type of services provided by the FA.

4. It allows the MN to know about the allowed registration lifetime or roaming period for visiting foreign network.

b) Agent solicitation:

- Rather than waiting for agent advertisements a MN can sen out an agent solicitation.
- This solicitation forces any agents on the link to immediately send an agent advertisement.
- If MN determines that it is connected to a foreign network, then it obtains a COA.
- **Types of COA:**
    - i. **Foreign Agent COA -** The static IP address of a foreign agent (FA) on a visited network.
    - ii. **Co-located COA -** Temporary IP address assigned to the MN.
        - Represents the current position of the MN on the Foreign network & can be used by only one MN at a time.
        - A co-located care-of address can be obtained by Dynamic Host Configuration Protocol (DHCP).

Steps:

1. MA (HA, FA) broadcast agent advertisement message at regular intervals.
2. The MN receiving the agent advertisement message observes whether the message is from its own HA & determine whether it is on the home network or on the foreign network.
3. If the MN does not wish to wait for the periodic advertisement, it can send out agent solicitation message that will be responded to by a MA.
4. After these steps of advertisements or solicitations the MN can now receive a COA, either one for an FA or a co-located COA. The MN knows its location (home network or foreign network) and the capabilities of the agent.
5. The next step for the MN is the registration with the HA if the MN is in a foreign network

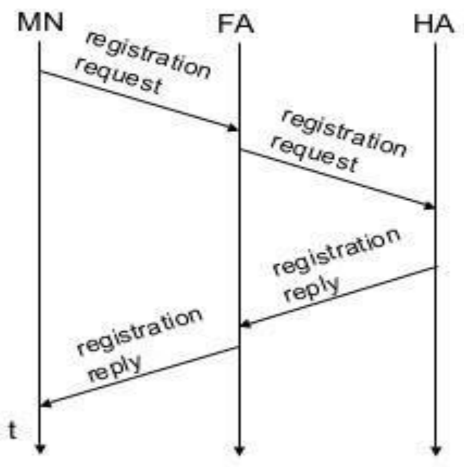
5.1.4.2 REGISTRATION

- If the MN discovers that it is on the home network, then it operates normally without Mobile IP

- If the MN has moved to a new network & obtain the COA from a FA, then this address should be registered with the HA.
- **Registration** – A MN uses an authenticated registration procedure to inform the HA of its COA.
- Registration messages uses UDP Protocol.

Registration can be done in two different ways:

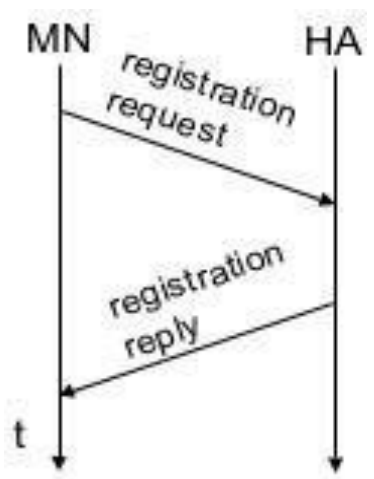
i. Registration of the MN through FA



If the COA is at the FA;

- MN sends its registration request containing the COA to the FA which then forward the request to the HA.
- Now HA will do the mobility binding containing the mobile node's home IP address and the current COA.
- Then finally the HA Acknowledges via FA to MN.

ii. Directly with HA

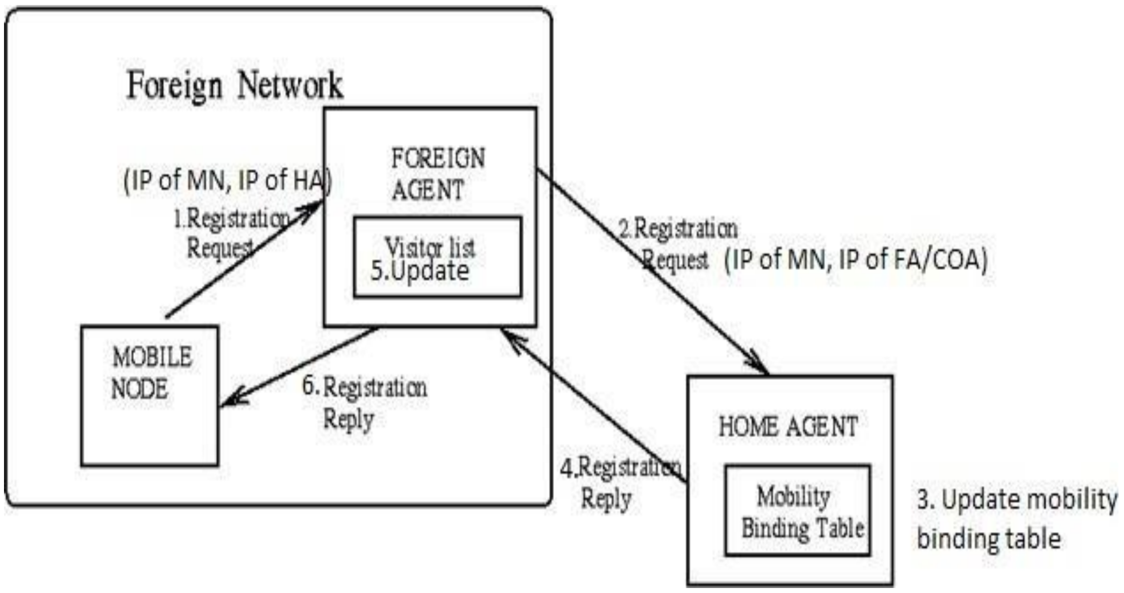


If the COA is co-located;

- MN sends the request directly to the HA and vice versa.
- Also, a registration procedure for MNs returning to their home network.

REGISTRATION PROCESS:

- The registration process involves the exchange of registration requests and registration reply messages.
- When the mobile node registers by using a foreign agent, the registration process takes the following steps, which is shown in the figure.



1. If MN travels to foreign network, it registers with the FA by sending a registration request message, which includes permanent IP address of the MN & IP address of HA.

2. The FA in turn performs the registration process on behalf of the MN by sending the registration request message to HA, which includes permanent IP address of the MN & IP address of FA(i.e., COA)

3. When the HA receives the registration request, it updates the “mobility Binding Table”.

4. Then HA sends an acknowledgement (registration reply) to the FA.

5-6. The FA in turn updates its “Visitor list” & relays the reply to the MN.

Registration Request Packet format:

0 7	8							15	16 23	24 31
type 1	S	B	D	M	G	r	T	X	lifetime	
home address										
home agent										
COA										
identification										
extensions ...										

- ***UDP packets are used for registration requests***.
- IP source address is the MN interface address and IP destination address is the FA or HA address.
- **Type** – 1 , **S** – an MN wants the HA to ***retain prior mobility binding***

  B – MN want to receive broadcast packets received by HA in home n/w
- **M & G** – ***minimal*** or ***generic*** routing encapsulation.
- ***Destnation port*** – 434
- ***UDP is used because of low overheads and better performance***.

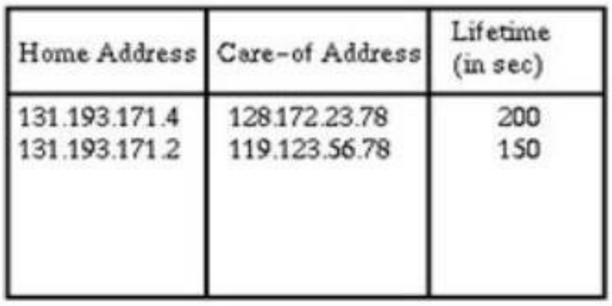
Registration Reply Packet format:

0 7	8 15	16 31
type = 3	code	lifetime
home address		
home agent		
identification		
extensions ...		

- **Type** – 3
- **code** – result of the registration request
- **lifetime** – validity of the registration ,
- **Home IP address**
- **Home Agent address**
- 64-bit **identification** used to match the registration request with reply

Mobility Binding Table:

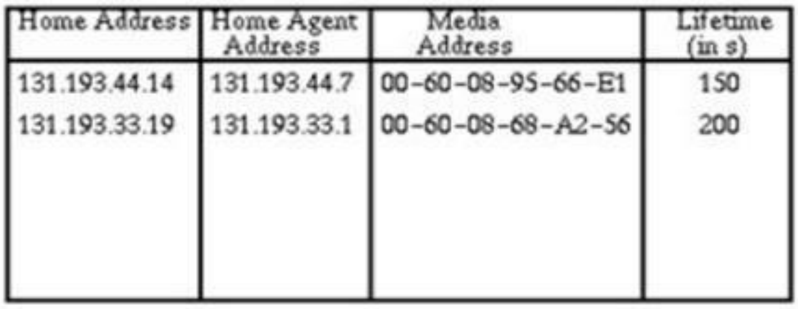
- Maintained on HA of MN.
- Maps MN's home address with its current COA



Home Address	Care-of Address	Lifetime (in sec)
131.193.171.4	128.172.23.78	200
131.193.171.2	119.123.56.78	150

Visitor List:

- Maintained on FA.
- Maps MN's home address with its MAC address (address of NIC) & HA's address.



Home Address	Home Agent Address	Media Address	Lifetime (in s)
131.193.44.14	131.193.44.7	00-60-08-95-66-E1	150
131.193.33.19	131.193.33.1	00-60-08-68-A2-56	200

5.1.4.3 TUNNELLING AND ENCAPSULATION

- **Tunneling (data transfer)** – Mechanism used to forward IP datagrams from a home address to a care-of address i.e., sending a packet through a tunnel
- A tunnel establishes a virtual pipe for data packets between a tunnel entry and a tunnel endpoint.

Two primary functions:

- **Encapsulations –** Mechanism of taking a packet consisting of packet header and data and putting it into the data part of a new packet.
  HA encapsulates all the packets addressed to MN & forward them to FA.
- **Decapsulation -** The reverse operation, taking a packet out of the data part of another packet
  FA decapsulates all the packets addressed to MN & forward them.

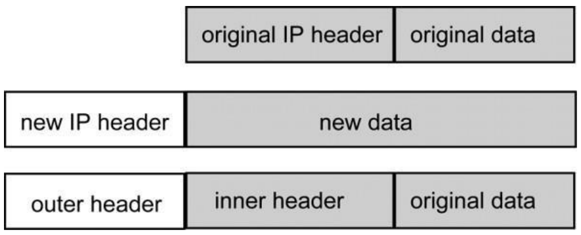
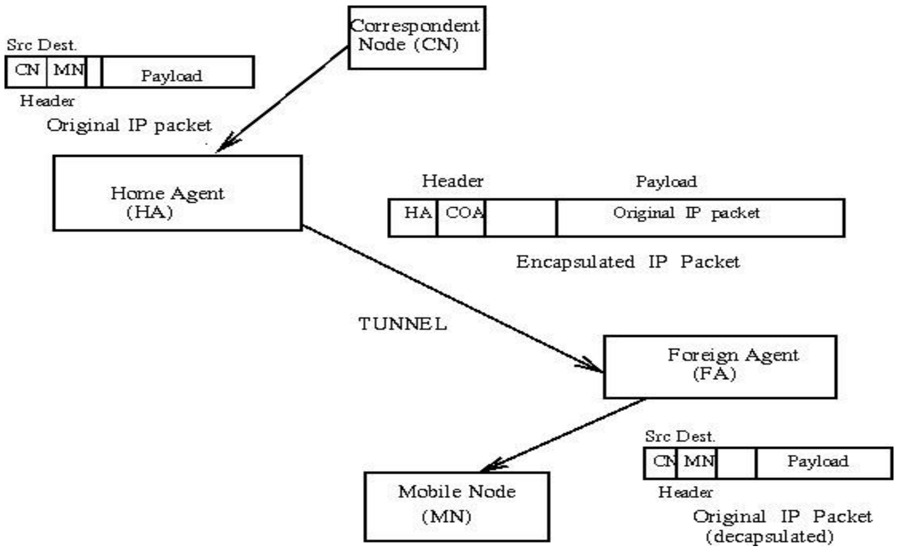


Fig. IP encapsulation



Steps in Encapsulation:

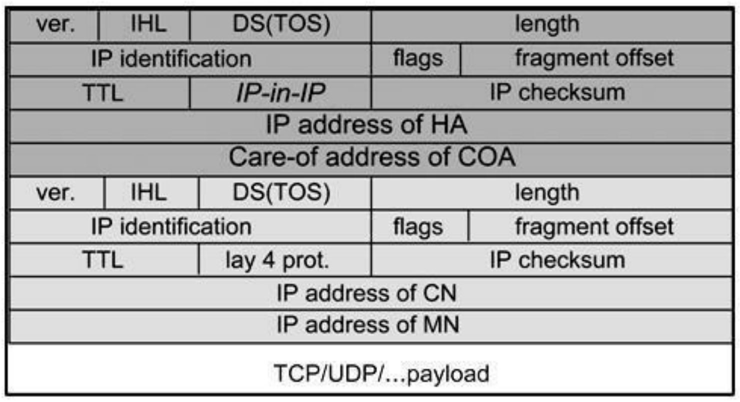
1. When a HA receives a packet addressed to a MN, it forwards the packet to the COA using IP -within -IP encapsulation
2. Using IP -within -IP , the HA inserts a new IP header in front of the IP header of any datagram.
3. Destination address is set to the COA.
4. Source address is set to the HA's address.
5. After stripping out the 1st header, IP processes the packet again. There are different ways of performing the encapsulation. They are:
   a. IP-in-IP Encapsulation
   b. Minimal Encapsulation
   c. Generic Routing Encapsulation

IP-in-IP Encapsulation:

- This is the mandatory method for Mobile IP.
- Full IP header added to the original IP packet.
- The inner IP header source and destination address identify the original sender and the receiver.

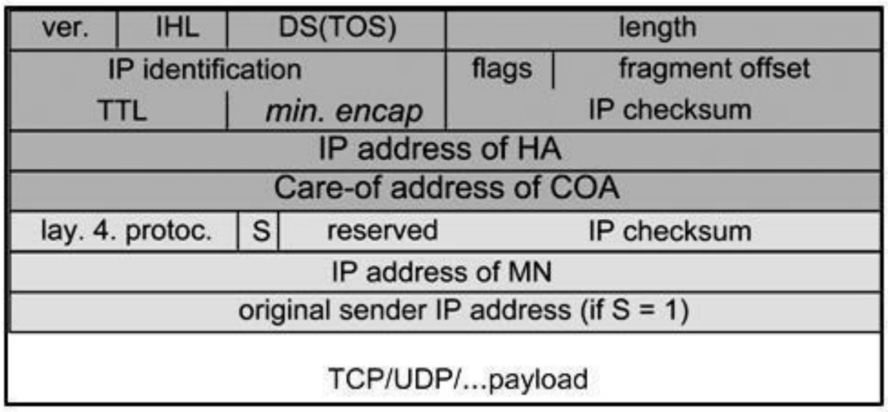
- The new(outer) header contains HA address as source & COA as destination.

- **Ver** – IP protocol version no.
- **IHL** – internet header length
- **TOS** – type of services (copied from inner header)
- **Length** – complete encapsulated packet length.
- **IP id.** , **flags** , **frag. offset** – used for fragments
- **TTL** -time to live
- **IP-in-IP** – upper layer protocol
- **IP checksum** – error detection



Minimal Encapsulation:

- It is an optional method for mobile IP
- In IP-in-IP several fields are redundant.
- Minimal Encapsulation will remove these redundancy.

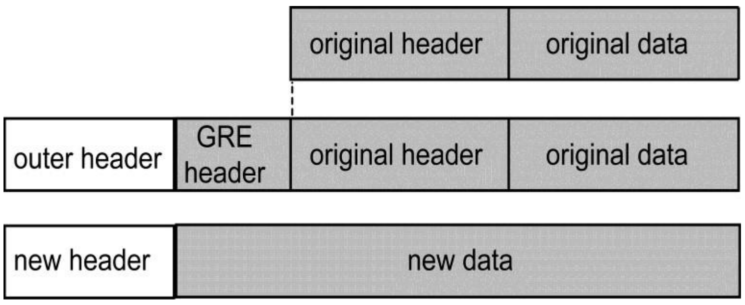


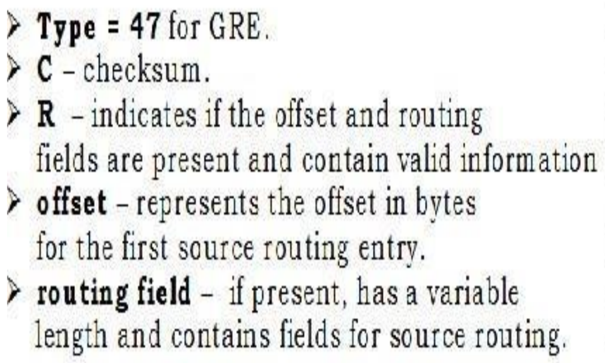
- Type – 55
- If S bit is set , ***the original sender address of the CN*** is included.

Generic Routing Encapsulation (GRE):

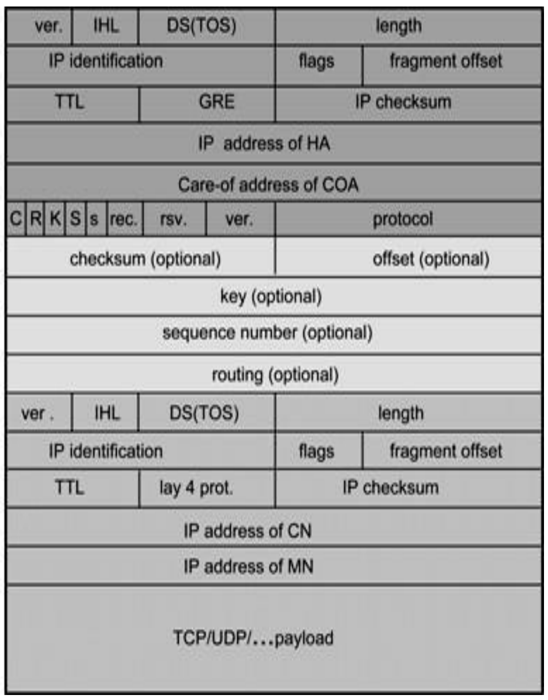
- Minimal Encapsulation & IP-in-IP only works for IP while GRE also supports other network layer protocols.
- Allows the encapsulation of packets of one protocol suite into the payload portion of a packet of another protocol suite.

- The packet of one protocol suite with the original packet header and data is taken and a new GRE header is prepended.
- Together this forms the new data part of the new packet.
- Finally, the header of the second protocol suite is put in front.
- The outer header is the standard IP header with HA as source address and COA as destination address.





- **Type = 47** for GRE.
- **C** – checksum.
- **R** – indicates if the offset and routing fields are present and contain valid information
- **offset** – represents the offset in bytes for the first source routing entry.
- **routing field** – if present, has a variable length and contains fields for source routing.



key - used for authentication.

K bit - if set indicates if authentication key is present.

S bit - if set indicates if the Sequence number field is present.

rec - recursion control field. This field represents a counter that shows the number of allowed recursive encapsulations.

rsv - reserved for future use. Must be zero.

ver - 0 for GRE version.

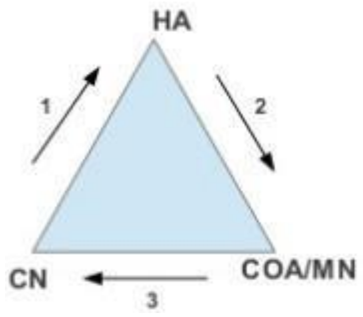
Lay 4 protocol specifies the protocol of the packet following the GRE header.

5.1.5 OPTIMIZATIONS (ROUTE OPTIMIZATOIN)

One of the problem with mobile IP – “Triangle Routing”

Triangle Routing:

- Involves two IP routes – Causes unnecessary network traffic overhead & higher latency
- An inefficient behavior of a non- optimized mobile IP



The triangle is made of the three segments,

- CN to HA
- HA to COA/MN
- MN back to CN.

To optimize the route:

- Enable direct notification of the CN.
- HA informs a sender about the location of MN
- Direct Tunnelling from the CN to MN.
- Binding cache maintained at the CN.
- **Binding cache** which is a part of the local routing table for the CN

The optimized mobile IP protocol needs four additional messages:

1. Binding request:

- Any node that wants to know the current location of an MN can send a binding request to the HA.
- The HA can check if the MN has allowed broadcasting of its current location.
- If the HA is allowed to reveal the location it sends back a binding update.

2. Binding update:

- This message sent by the HA to CNs reveals the current location of an MN.
- The message contains the fixed IP address of the MN and the COA.
- The binding update can request an acknowledgement.

3. Binding acknowledgement:

- If requested, a node returns this acknowledgement after receiving a binding update message.

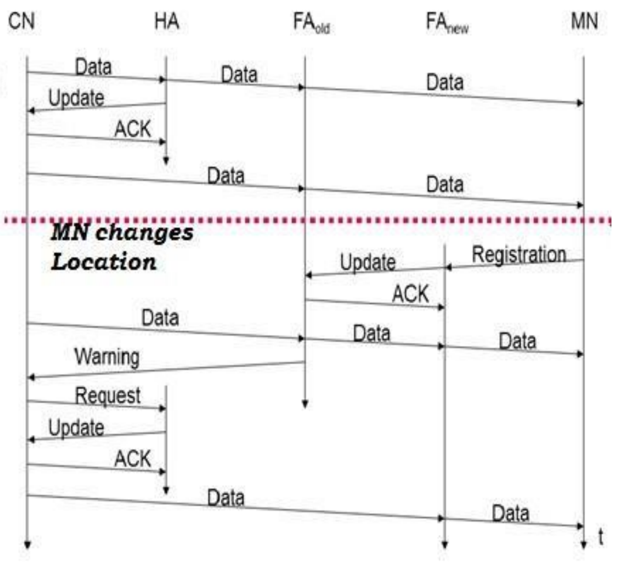
4. Binding warning:

- If a node decapsulates a packet for an MN, but it is not the current FA for this MN, this node sends a binding warning to the HA of the MN.

- CN request current location from HA using ***binding request***.
- HA return the COA address using ***binding update.***
- Now CN directly send to FA old.
- Tunnel is formed between CN nd FA old.

MN now change its location

- *Register with FA new*
- This info is forwarded to HA to update its location.
- **FA new** will inform **FA old** about the new registration.
- Still CN send data to **FA old** which forward the data to **FA new**.
- **FA old** will send the binding warning to CN.
- then , CN sends binding request to HA
- and get the updated FA of the MN.



Reverse tunneling is a tunneling from mobile host to home agent, and makes it possible for the mobile host from foreign network to communication in the network

5.2 DHCP - DYNAMIC HOST CONFIGURATION PROTOCOL

- DHCP stands for Dynamic Host Configuration Protocol. It is the critical feature on which the users of an enterprise network communicate. DHCP helps enterprises to smoothly manage the allocation of IP addresses to the end-user clients’ devices such as desktops, laptops, cellphones, etc.
- DHCP is used to merge the world of mobile phones with the internet and to support mobility.

- Automatically assigns a unique IP address to each device that connects to a network.
- Used to simplify the installation and maintenance of networked computers.
- If a new computer is connected to a network, DHCP can provide it with all the necessary information for full system integration into the network, e.g., addresses of a DNS server and the default router, the subnet mask, the domain name, and an IP address.

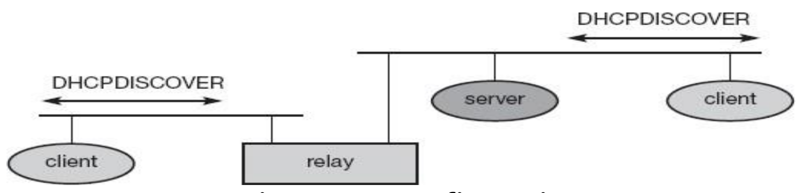
Components of DHCP

The main components of DHCP include:

1. **DHCP Server:** DHCP Server is basically a server that holds IP Addresses and other information related to configuration.
2. **DHCP Client:** It is basically a device that receives configuration information from the server. It can be a mobile, laptop, computer, or any other electronic device that requires a connection.
3. **DHCP Relay:** DHCP relays basically work as a communication channel between DHCP Client and Server.
4. **IP Address Pool:** It is the pool or container of IP Addresses possessed by the DHCP Server. It has a range of addresses that can be allocated to devices.
5. **Subnets:** Subnets are smaller portions of the IP network partitioned to keep networks under control.
6. **Lease:** It is simply the time that how long the information received from the server is valid, in case of expiration of the lease, the tenant must have to re-assign the lease.
7. **DNS Servers:** DHCP servers can also provide DNS (Domain Name System) server information to DHCP clients, allowing them to resolve domain names to IP addresses.
8. **Default Gateway:** DHCP servers can also provide information about the default gateway, which is the device that packets are sent to when the destination is outside the local network.
9. **Options:** DHCP servers can provide additional configuration options to clients, such as the subnet mask, domain name, and time server information.

10. **Renewal:** DHCP clients can request to renew their lease before it expires to ensure that they continue to have a valid IP address and configuration information.
11. **Failover:** DHCP servers can be configured for failover, where two servers work together to provide redundancy and ensure that clients can always obtain an IP address and configuration information, even if one server goes down.
12. **Dynamic Updates:** DHCP servers can also be configured to dynamically update DNS records with the IP address of DHCP clients, allowing for easier management of network resources.
13. **Audit Logging:** DHCP servers can keep audit logs of all DHCP transactions, providing administrators with visibility into which devices are using which IP addresses and when leases are being assigned or renewed.

DHCP is based on a client/server model.



Basic DHCP Configuration

1. DHCP clients send a request to a server (DHCPDISCOVER) to which the server responds.
2. A client sends requests using MAC broadcasts to reach all devices in the LAN.
3. A DHCP relay might be needed to forward requests across inter-working units to a DHCP server.

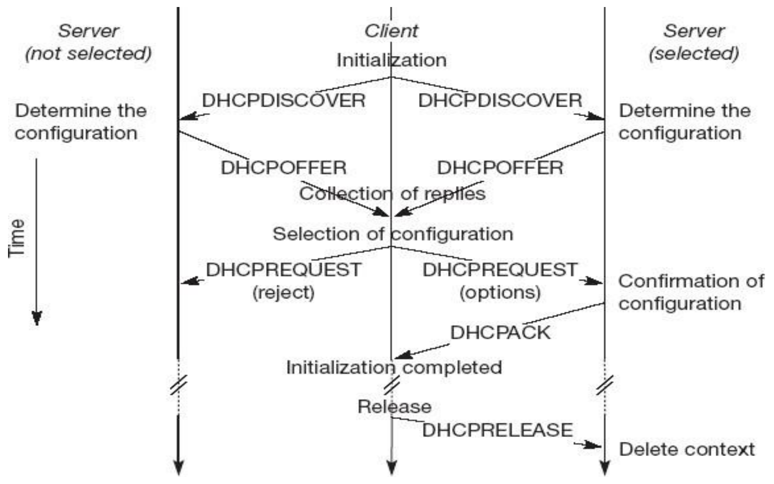


Fig. Client initialization via DHCP

The above figure shows one client and two servers.

1. The client broadcasts a **DHCP DISCOVER** into the subnet.
2. Two servers receive this broadcast and find the configuration they can offer to the client.
3. Servers reply to the client's request with **DHCP OFFER** and offer a list of configuration parameters.
4. Then the client can choose one of the configurations offered.
5. Then the client in turn replies to the servers, accepting one of the configurations and rejecting the others using **DHCP REQUEST**.
6. If a server receives a **DHCP REQUEST** with a rejection, it can free the reserved configuration for other possible clients.
7. The server with the configuration accepted by the client now confirms the configuration with **DHCP ACK**. This completes the initialization phase.
8. If a client leaves a subnet, it should release the configuration received by the server using **DHCP RELEASE**.
9. The configuration a client gets from a server is only leased for a certain amount of time, it has to be reconfirmed from time to time.

Advantages of DHCP

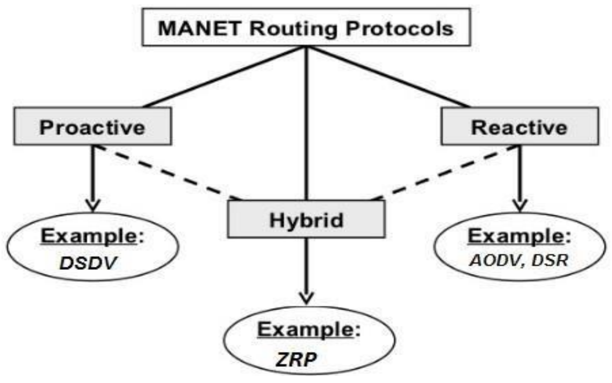
- Centralized management of IP addresses.
- Centralized and automated TCP/IP configuration.
- Ease of adding new clients to a network.
- Reuse of IP addresses reduces the total number of IP addresses that are required.
- The efficient handling of IP address changes for clients that must be updated frequently, such as those for portable devices that move to different locations on a wireless network.
- Simple reconfiguration of the IP address space on the DHCP server without needing to reconfigure each client.
- The DHCP protocol gives the network administrator a method to configure the network from a centralized area.
- With the help of DHCP, easy handling of new users and the reuse of IP addresses can be achieved.

Disadvantages of DHCP

- IP conflict can occur.
- The problem with DHCP is that clients accept any server. Accordingly, when another server is in the vicinity, the client may connect with this server, and this server may possibly send invalid data to the client.
- The client is not able to access the network in absence of a DHCP Server.
- The name of the machine will not be changed in a case when a new IP Address is assigned.

5.3 Routing in Mobile ad hoc networks

- Routing is a process of finding an efficient, reliable and secure path from a source node to a destination node via intermediate nodes in a network.
- Efficiency of the path is measured in various metrics like, Number of hops, traffic, security, etc.



5.3.1 PROACTIVE PROTOCOLS (Table-driven routing protocol)

- Maintain the global topology information in the form of tables at every node.
- These tables are updated frequently in order to maintain consistent and accurate network state information.
- **EX: DSDV, WRP, and STAR.**

5.3.1.1 DESTINATION-SEQUENCED DISTANCE-VECTOR ROUTING (DSDV)

- Based on Proactive method
- Enhanced version of the distributed Bellman-Ford algorithm or Distance Vector (DV) Routing Protocol

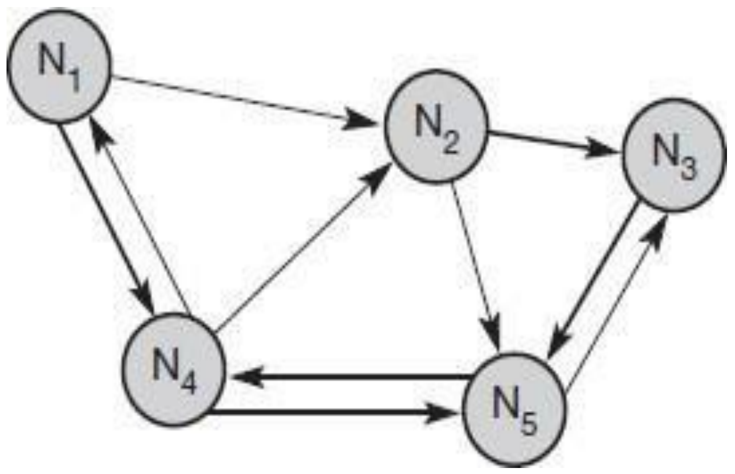
- DSDV adds two things to the distance vector algorithm

1 **Sequence Number:**

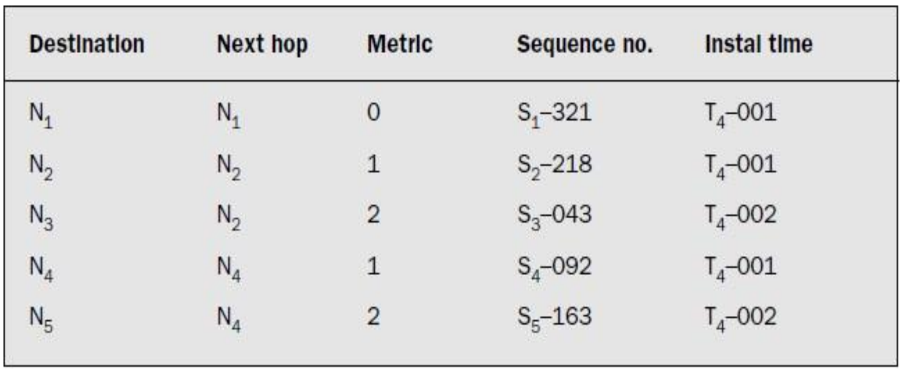
- Each routing advertisement comes with a sequence number.
- Within ad-hoc networks, advertisements may propagate along many paths.
- Sequence numbers help to apply the advertisements in correct order.
- This avoid the loops in the network.

2 **Damping:**

- Transient changes in topology that are of short duration should not weaken the routing mechanisms.
- Unstable changes in the topology are not forwarded



Example Ad-hoc network



Destination	Next hop	Metric	Sequence no.	Instal time
N_1	N_1	0	S_1–321	T_4–001
N_2	N_2	1	S_2–218	T_4–001
N_3	N_2	2	S_3–043	T_4–002
N_4	N_4	1	S_4–092	T_4–001
N_5	N_4	2	S_5–163	T_4–002

Part of a routing table for DSDV

For each node N1 maintain a table that contain;

- The next hop toward this node
- The metric (number of hops)

- The sequence number
- The time at which the path has been installed first.

Important steps in the operation of DSDV:

1. Each router(node) in the network collects route information from its neighbours.
2. After gathering information, the node determines the shortest path to the destination based on the gathered information.
3. Based on the gathered information, a new routing table is generated.
4. The router broadcasts this table to its neighbours. On receipt by neighbours, the neighbour nodes recompute their respective routing tables.
5. This process continues till the routing information becomes stable.

Advantages

- Simple
- Loop free through destination seq. numbers
- No latency caused by route discovery

Disadvantages

- No sleeping nodes
- Overhead: most routing information never used

5.3.2 REACTIVE PROTOCOLS (On-demanding routing protocol)

- They execute the path-finding process and exchange routing information only when a path is required by a node to communicate with a destination.
- i.e., a route is discovered only when it is necessary.
- Source initiates route discovery
- 2 step process
  - **i.** Route Discovery
  - **ii.** Route Maintenance
- Route discovery is expensive
- **Example: Dynamic Source Routing (DSR), Ad hoc On-demand Distance Vector (AODV)**

5.3.2.1 DYNAMIC SOURCE ROUTING PROTOCOL (DSR)

- DSR is a source initiated on-demand (or reactive) routing protocol for ad-hoc network
- Designed to restrict the bandwidth consumed by packets by eliminating the periodic table update messages i.e., the nodes do not need to exchange the routing information periodically, which helps to reduce the bandwidth overhead.
- Each mobile node participating in the protocol maintains a “routing cache” which contains the list of all routes that the node has learnt

DSR works in 2 phases:

(a) Route Discovery:

- Allows any host to dynamically discover the route to any destination in the ad-hoc network.
- Route Discovery Process takes place by:

  1. Broadcasting a route request (RREQ) packet to all its neighbours.

  ➢ The Route request (RREQ) packet contains the

  i) Source address

  ii) Request id

  iii) Route Record, in which the sequence of hops traversed by the request packet before reaching the destination is recorded.

  2. A node after receiving RREQ

  i. If the node is an intermediate node then,

  - If the message has the same ID i.e. has seen it before, then the node discards this message,
  - If not, the node appends its own address to the route record in the ROUTE REQUEST message then propagates the message to the next hop neighbours.

  ii. If the node is the Target (Destination) then,

  - Returns a Route Reply (RREP) message to the sender
  - Copies the accumulated route record from RREQ into RREP

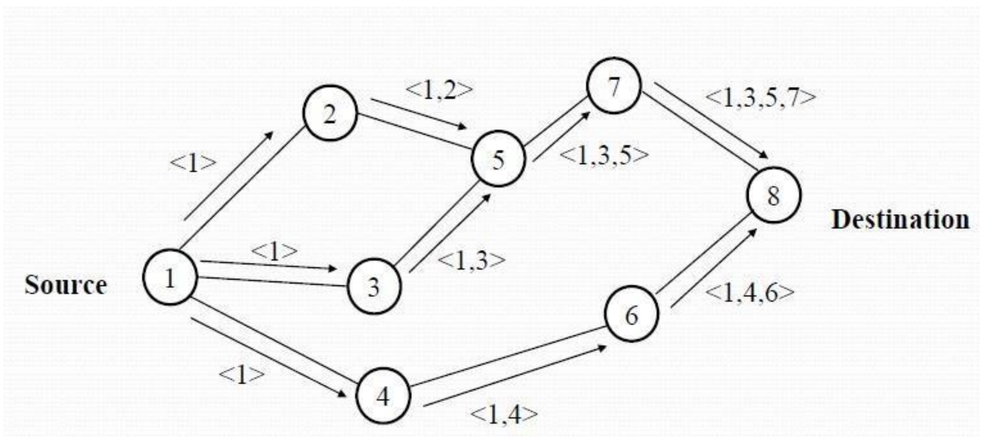


Fig. Broadcasting the RREQ packets

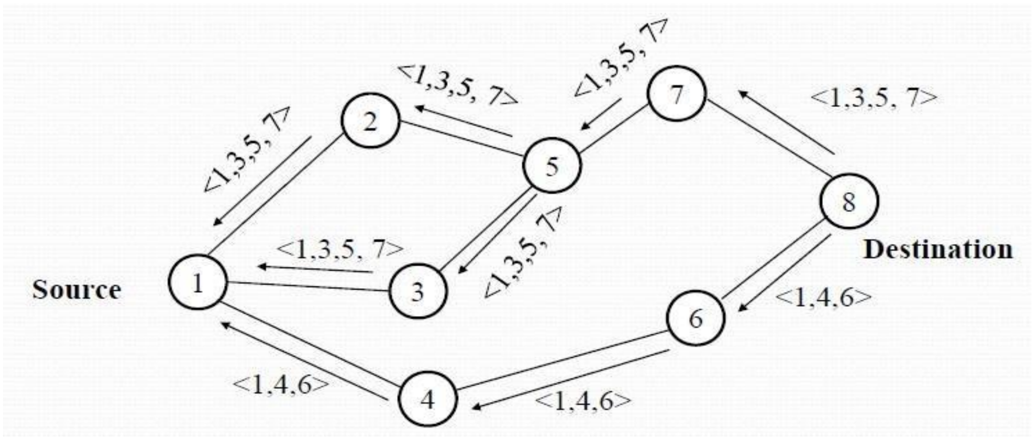


Fig. Propagation of RREP packets back to source

(b) Route Maintenance:

- A known route can get broken due to the movement of some node or the battery of a node getting exhausted.
- **Route maintenance**: The process of monitoring the correct operation of a route in use & taking corrective action when needed.

Steps:

1. When a node detects that one of its next hop neighbour node is not responding, it sends back a route error (RERR) packet containing its own address and the address of the hop that is not working
2. As Soon as source node receives the RERR message it deletes the broken link route from its cache.

3. If it has another route to the destination, it starts to retransmit the packet using the alternative route.
4. Otherwise, it initiates the route discovery process again. The basic message set consists of:
    i. RREQ – Route request
    ii. RREP – Route reply
    iii. RERR – Route error
    iv. HELLO – For link status monitoring

Advantages:

- A perfect route is discovered always.
- Highly efficient.
- Low bandwidth Consumption.

Drawback:

- Packet header size (Non Uniform Packet Size) grows when intermediate node increases.
- Flood of route requests may potentially reach all nodes in the network

5.3.2.2 AD HOC ON-DEMAND DISTANCE VECTOR ROUTING (AODV)

- Based on Reactive method

DSR vs AODV:

- Major problem of DSR is its non-uniform packet size because it includes source routes in its packet header which degrades the performance. If a packet is large, it has to be split into smaller packets.
- The packet size in AODV is uniform unlike DSR.
- AODV attempts to improve on DSR by maintaining routing tables at the nodes, so that data packets do not have to contain routes.
- AODV holds the desirable feature of DSR that routes are maintained only between nodes which need to communicate.
- Route is established only when it is required by a source node for transmitting data packets
- Make use of hop-by-hop routing, sequence numbers and beacons.

Steps:

1. The node that needs a route to a specific destination generates a route request(RREQ).

2. The route request(RREQ) is forwarded by intermediate nodes which also learn a reverse route from the source to themselves.

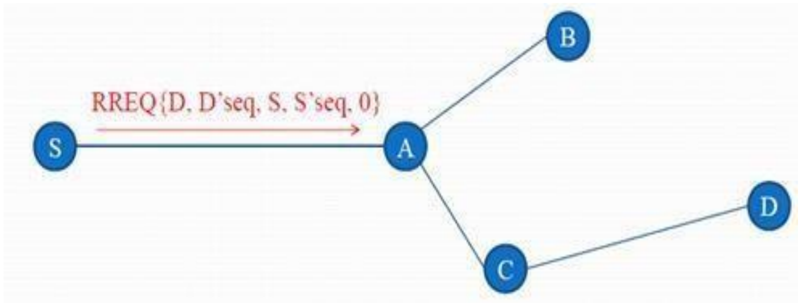
3. When the request reaches a node with route to destination, it generates a route reply(RREP) containing the number of hops required to reach the destination.

4. All nodes that participate in forwarding this reply to the source node create a forward route to destination.

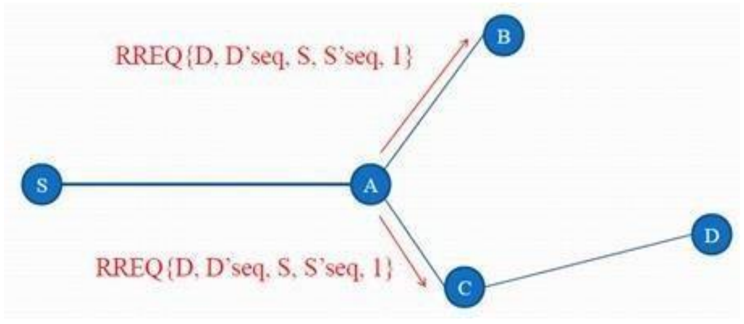
5. This route created from each node from source to destination is a hop-by-hop route.

Example: Suppose Node S needs a routing path to Node D

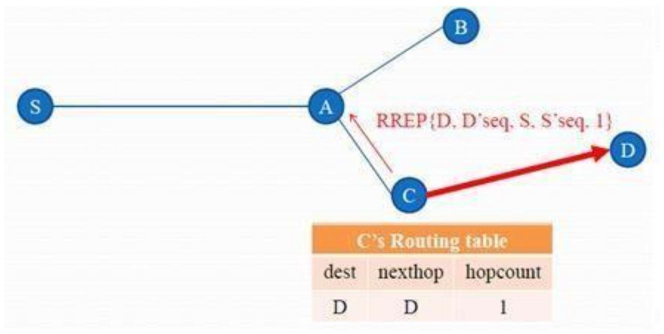
1. Node S creates a RREQ packet & broadcasts to its neighbours. RREQ [D's IP addr, Seq#, S's IP addr, hopcount]



2. Node A rebroadcasts RREQ to all its neighbours.

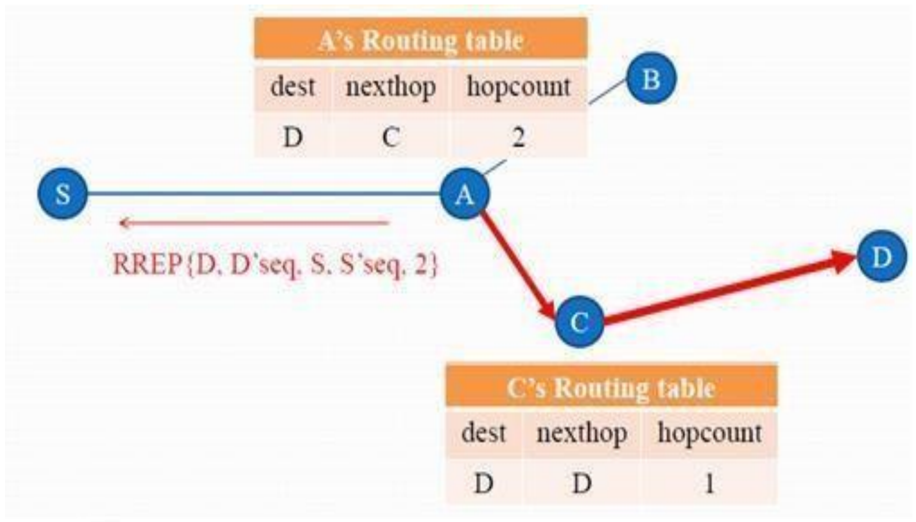


3. Since, Node C known a route to Node D, Node C creates a RREP & unicasts RREP to A. Set forward path in C's routing table.



C's Routing table		
dest	nexthop	hopcount
D	D	1

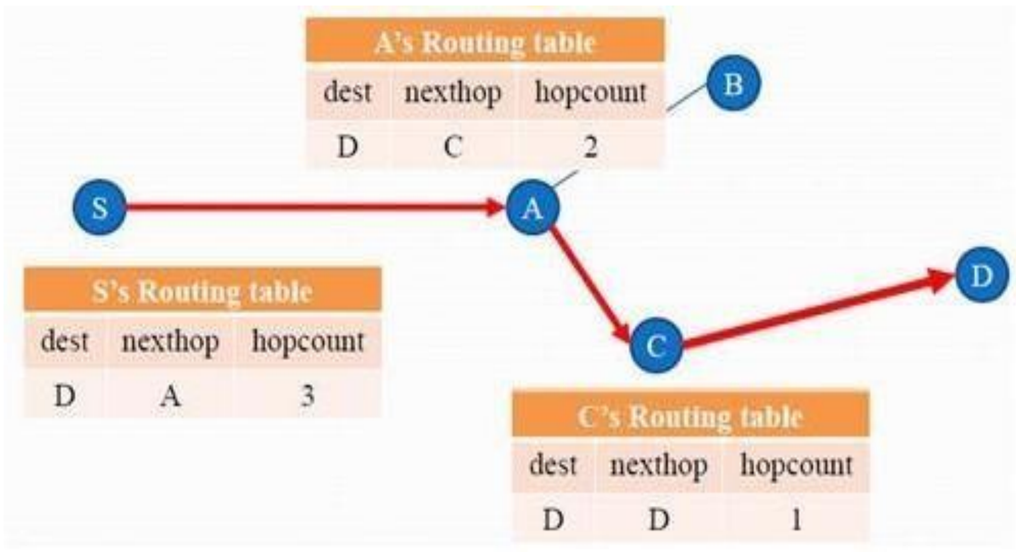
4. Node A creates a RREP & unicasts RREP to S
5. Set forward path in A's routing table



A's Routing table		
dest	nexthop	hopcount
D	C	2

C's Routing table		
dest	nexthop	hopcount
D	D	1

6. Set forward path in S's routing table



A's Routing table		
dest	nexthop	hopcount
D	C	2

S's Routing table		
dest	nexthop	hopcount
D	A	3

C's Routing table		
dest	nexthop	hopcount
D	D	1

Difference between DSR, DSDV & AODV

Property	**DSR**	**DSDV**	**AODV**
Loop Free	Yes	Yes	Yes
Multicast Routes	Yes	No	No
Unidirectional Link	Yes	No	No
Periodic Broadcast	No	Yes	Yes
Routes maintained	Route Cache	Route Table	Route Table
Reactive	Yes	No	Yes

5.3.3 HYBRID PROTOCOLS

- Combines the best features of both proactive & reactive routing protocols.
- Eg: ZONE ROUTING PROTOCOL (ZRP)

5.3.3.1 ZONE ROUTING PROTOCOL (ZRP)

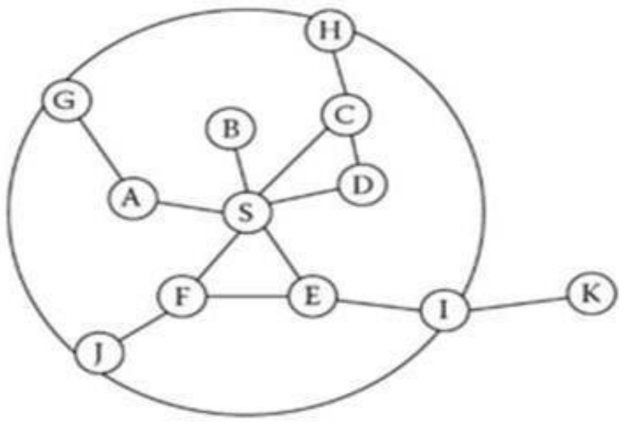
- It is Hybrid Protocol
- Based on the concept of zones.
- A routing zone is defined for each node separately and zones of neighbouring nodes overlap.
- The routing zone has a radius expressed in hops. i.e., **Zone radius:** Number of hops

Key concept in ZRP to:

- Use a proactive routing scheme within a limited zone
- Use a reactive routing scheme for nodes beyond this zone.

Routing is divided into two parts:

- **Intrazone routing:** 1st the packet is sent within the routing zone of the source node to reach the peripheral nodes
- **Interzone routing:** The packet is sent from the peripheral nodes towards the destination node



In the diagram the routing zone of S includes the nodes A-I, but not K.

- The nodes are divided into peripheral nodes and interior nodes.
- Peripheral nodes: Nodes whose minimum distance is less than the radius.
- Interior nodes - Nodes A-F
- Peripheral nodes - Nodes G-J
- Node K is outside the routing zone
- Within the zone table driven is used
- Outside the zone on demand Route Discovery is used

Procedure:

1. The source sends a Route Request packet (RREQ) to the border nodes of its zone, containing its own address, destination address and the unique sequence no.
2. Each border nodes checks its local zone for the destination.
3. If the destination is not a member of local zone, then the border node adds its own address to the route request packet and forwards the packet to its own border nodes.
4. When the destination node is reached in this process, a route reply (RREP) is sent on the reverse path back to the source.
5. The source saves the path which is mentioned in Route Reply to send data packets to the destination.

5.4 TCP improvements

There are several mechanisms of the Transmission Control Protocol (TCP) that influence the efficiency of TCP in a mobile environment.

1. Traditional TCP improvements - Wired

2. Classical TCP improvements – Wireless

5.4.1 Traditional TCP improvements

Improvement in TCP: TCP was initially designed for wired (traditional) networks

1. Slow start

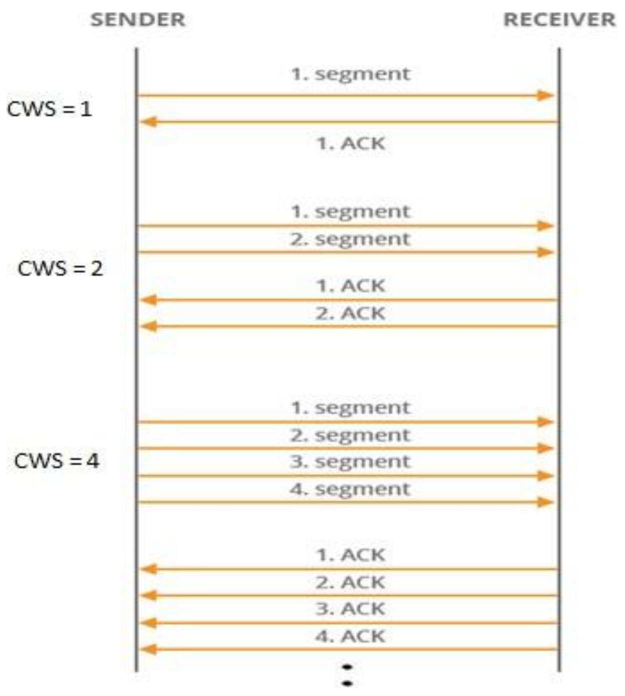
2. Congestion Avoidance

3. Fast retransmit/fast recovery

5.4.1.1 Slow start

- The behaviour TCP shows after the detection of congestion is called **slow start.**
- Instead of starting transmission at a fixed transmission window size, the transmission is started at the lowest window size and then doubled after each successful transmission.
- If congestion is detected, the transmission window size is reduced to half of its current size.
- The sender always calculates a **congestion window** for a receiver.

1. The start size of the congestion window is one segment.
2. The sender sends one packet and waits for acknowledgement.
3. If this acknowledgement arrives, the sender increases the congestion window by one, now sending two packets.
4. After arrival of the two corresponding acknowledgements, the sender again adds 2 to the congestion window, one for each of the acknowledgements.
5. Now the congestion window equals 4.
6. This scheme doubles the congestion window every time the acknowledgements come back, which takes one round trip time (RTT). This is called the exponential growth of the congestion window in the slow start mechanism.

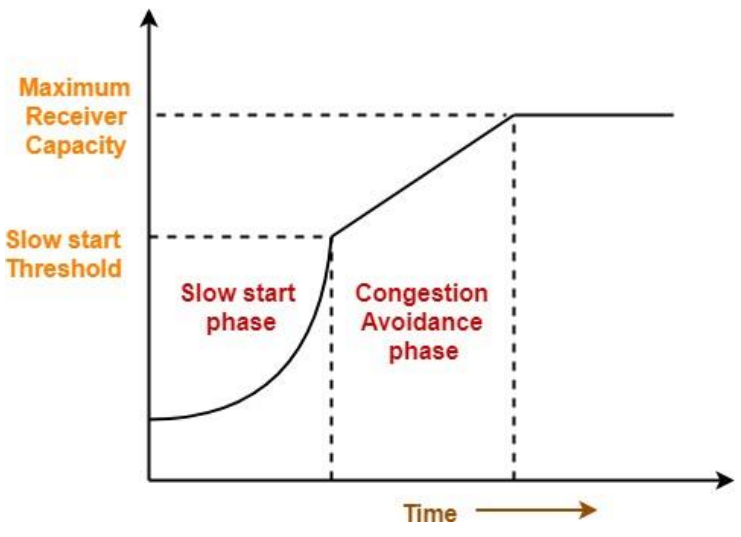


5.4.1.2 Congestion Avoidance

Drawback in slow start: It is too dangerous to double the congestion window each time because the steps might become too large

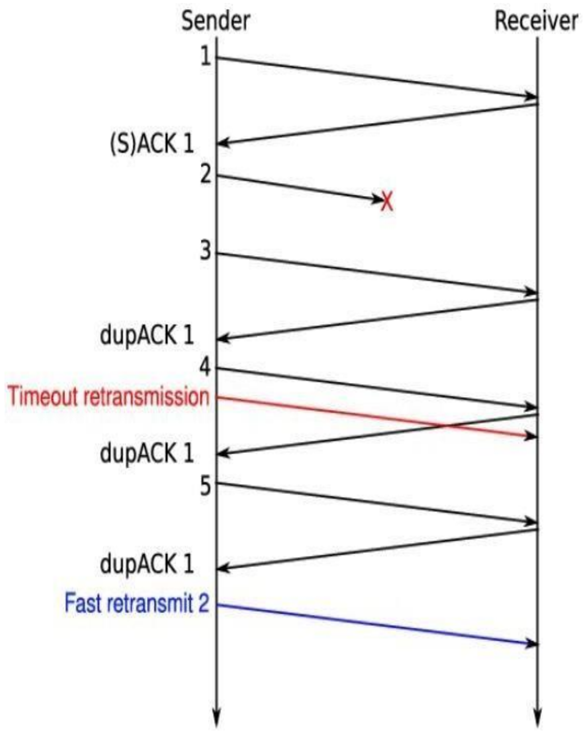
Solution:

- Window size is doubled until it reaches a threshold level.
- When it reaches a threshold level, then window size is increased linearly.
- If congestion is occurred, then the window size is reduced to half of its size.
- If it reaches zero then again slow start begins
- Congestion avoidance start when slow start stops



5.4.2.3 Fast retransmit/fast recovery

- The sender can retransmit the missing packet(s) before the timer expires.
- It does not wait until the timer expires it retransmit a packet whenever sender is getting 3 duplicate acknowledgements.
- After retransmitting a packet, it sets the window size is reduced to its half



5.4.2 Classical TCP improvements

Mechanisms to increase TCP's performance in wireless and mobile environments:

1. Indirect TCP (I-TCP)

2. Snooping TCP (S-TCP)

3. Mobile TCP (M-TCP)

4. Fast retransmit/fast recovery

5. Transmission/time-out freezing

6. Selective retransmission

7. Transaction-oriented TCP (T-TCP)

5.4.2.1 Indirect TCP (I-TCP)

- Two competing insights led to the development of indirect TCP:

  1) TCP performs poorly together with wireless links

  2) TCP within the fixed network cannot be changed

Working:

- I-TCP segments a TCP connection into a fixed part and a wireless part.
- Mobile host connected via a wireless link and an access point to the 'wired' internet where the correspondent host resides. The correspondent node could also use wireless access.
- Standard TCP is used between the fixed computer and the access point.
- The foreign agent (access point) acts as a proxy and relays all data in both directions.

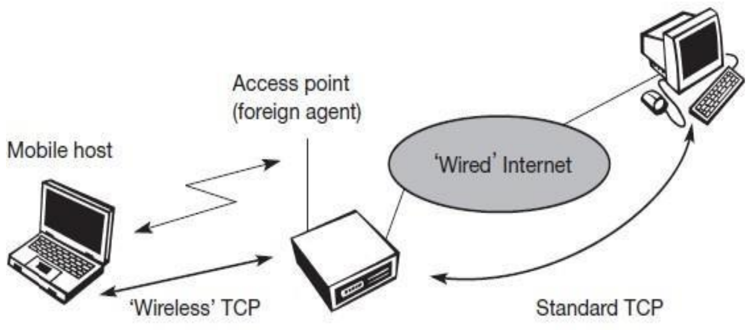
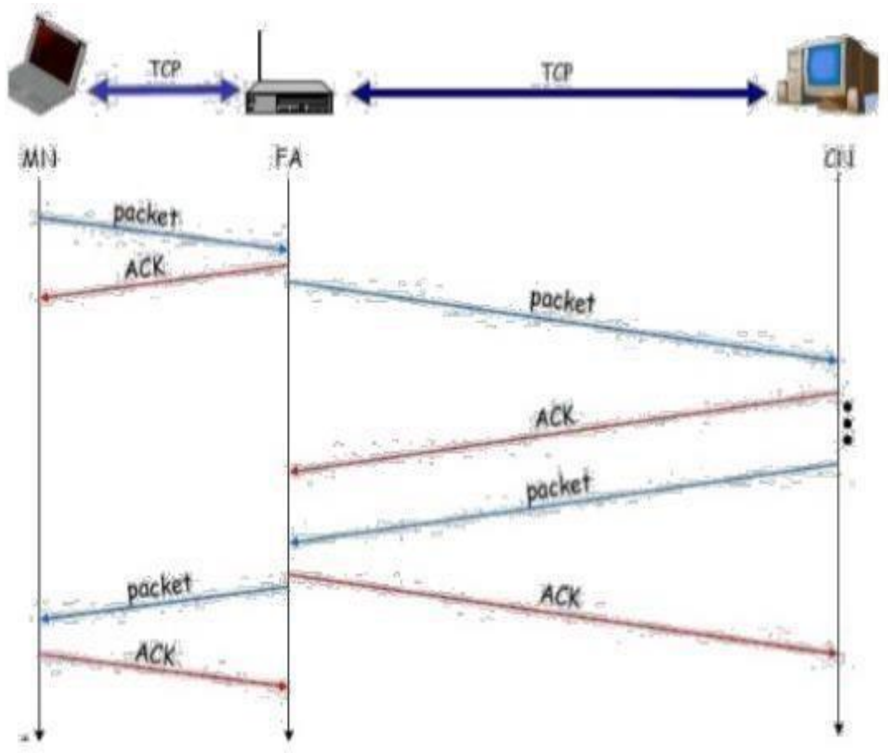


Fig. I-TCP segments a TCP connection into two parts

Packet delivery:

- If CN sends packet, FA acknowledges packet and forwards packet to MN
- If MN receives packet, it acknowledges
- This acknowledgement only used by CN
- Similarly, if MN sends packet, FA acknowledges packet and forwards it to CN



Packet Loss:

Case 1: If a packet is lost on the wireless link due to a transmission error:

- Then the CN would not notice this.
- The FA tries to retransmit this packet locally to maintain reliable data transport.

Case 2: If the packet is lost on the wireless link:

- The MN notice this much faster due to the lower round trip time and can directly retransmit the packet.
- Packet loss in the wired network is now handled by the FA.

Advantages with I-TCP:

1. TCP does not require any changes in the TCP protocol as used by the hosts in the fixed network.
2. Due to the strict partitioning into two connections, transmission error cannot propagate into the fixed network.
3. Partitioning into two connections allows the use of a different transport layer protocol between the FA and the MN.
4. Different solutions can be tested or used at the same time without disturbing the stability of the Internet.

Disadvantages of I-TCP:

1. The loss of the end-to-end semantics of TCP might cause problems if the FA partitioning the TCP connection crashes:
2. Increased handover latency may be much more problematic
3. The FA must be integrated into all security mechanisms.

5.4.2.2 Snooping TCP (S-TCP)

- The segmentation drawback of I-TCP is eliminated by Snooping TCP.
- "The FA buffers all packets with destination MN and additionally ‘snoops’ the packet flow in both directions to recognize acknowledgements"
- Reason for buffering: To enable the FA to perform a local retransmission in case of packet loss on the wireless link.

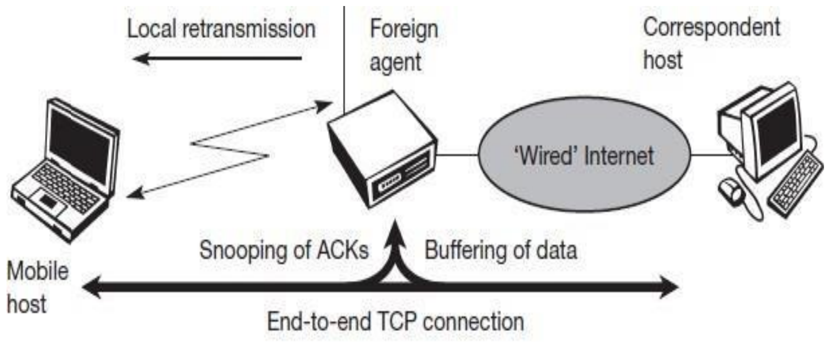


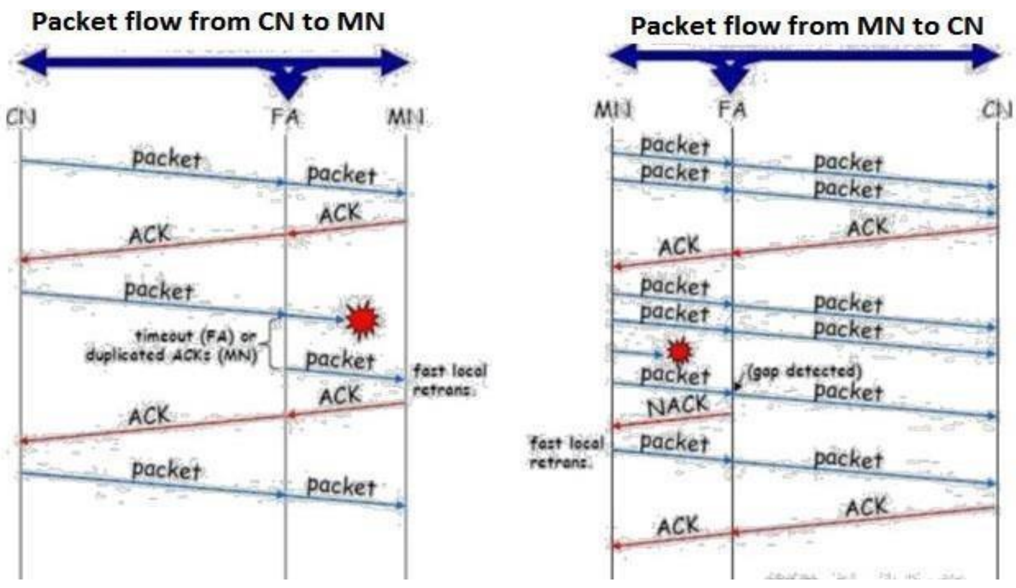
Fig. Snooping TCP

Data transfer to the MH (Mobile Host)

- FA buffers data until it receives ACK of the MH
- FA detects packet loss via duplicated ACKs or time-out

Data transfer from the MH (Mobile Host)

- FA detects packet loss on the wireless link via sequence numbers, FA answers directly with a negative acknowledgement (NACK) to the MH
- MH can now retransmit data with only a very short delay



Advantages:

1. The approach automatically falls back to standard TCP if the enhancements stop working.
2. The CN does not need to be changed since most of the enhancements are in the FA.
3. It does not need a handover of state as soon as the MH moves to another

FA.

4. It does not matter if the next FA uses the enhancement or not. If not, the approach automatically falls back to the standard solution.

Disadvantages:

1. Snooping TCP does not isolate the behaviour of the wireless link as good as I-TCP.
2. Additional mechanism for negative acknowledgements (NACK) between FA and MH.
3. Snooping and buffering data may be useless if certain encryption schemes are applied end-to-end between the correspondent host and mobile host.

5.4.2.3 Mobile TCP (M-TCP)

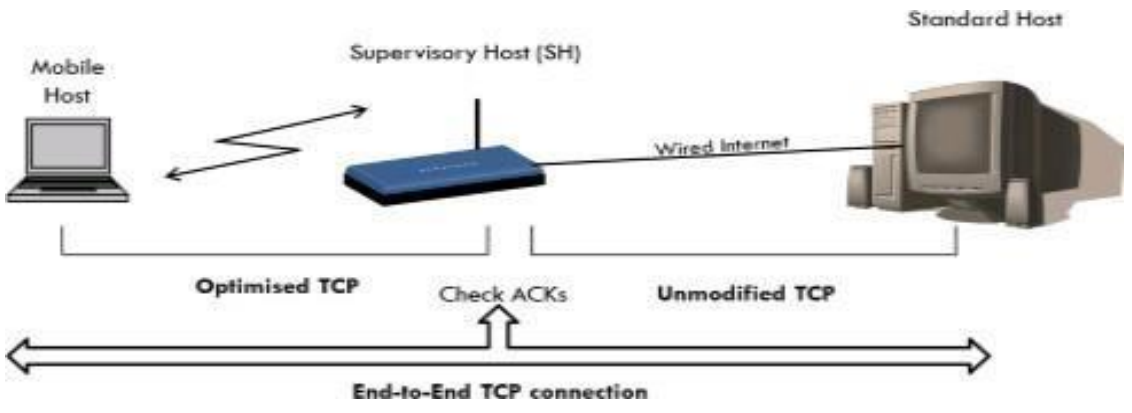
- I-TCP and S-TCP does not work well, if a MH is disconnected.
- The M-TCP has the same goals as I-TCP and snooping TCP

Goals of M-TCP:

- Prevent the sender window from shrinking if bit errors or disconnection.
- Improve overall throughput
- Lower the delay
- Maintain end-to-end semantics of TCP
- Provide a more efficient handover
- Adapted to the problems arising from lengthy or frequent disconnections

The M-TCP splits up the connection into two parts:

- An unmodified TCP is used on the Standard host-Supervisory Host section
- An optimised TCP is used on the Supervisory Host- Mobile Host section.



- The SH is responsible for exchanging data to both the Standard host and the Mobile host.
- In this approach, we assume that the error bit rate is less as compared to other wireless links.
- So if any packet is lost, the retransmission has to occur from the original sender and not by the SH.
    1. The SH monitors the ACKs being sent by the MH.
    2. If for a long period ACKs have not been received, then the SH assumes that the MH has been disconnected.
    3. If so the SH blocks the sender by setting its window size to 0.
    4. Then the sender goes into persistent mode i.e. the sender will not try to retransmit the data.
    5. Now when the SH detects a connectivity established again with the MH, the window of the sender is restored to original value.

Advantages of Mobile TCP:

1. M-TCP maintains the TCP end-to-end semantics.
2. If the MH is disconnected, it avoids useless retransmissions, slow starts or breaking connections by simply shrinking the sender's window to 0.
3. M-TCP does not buffer data so, no forwarding.

Disadvantages of Mobile TCP:

1. The SH does not act as proxy
2. M-TCP assumes low bit error rates, which is not always a valid assumption.
3. Requires new network elements like the bandwidth manager.

5.4.2.4 Fast retransmit/fast recovery

- Change of FA often results in a packet loss. TCP reacts with slow start although there is no congestion.

Solution: Fast retransmit method.

- Fast retransmit method: When a MH moves to a new FA, it transmits the ACK of the last packet was received.
- It is indication for the CN to continue transmission at the same rate it did before MH moves to another FA.
- This approach puts the CN to fast retransmission mode.

Advantages:

1. It is simple.
2. Only minor changes in the MN software results in performance increase.
3. No FA or CN host has to be changed.

Disadvantages:

1. Increased time delay in the retransmitted packets to move from CN to MH.

5.4.2.5 Transmission/time-out freezing

- In normal TCP, a disconnection takes place when the connection is lost for a longer time.
- Example: When a MN moving through a tunnel or passing black out areas, the connection is lost and it needs to make connection once again, when it comes back.

TCP freezing:

- MAC layer is often able to detect interruption in advance
- MAC can inform TCP layer of upcoming loss of connection
- TCP stops sending, but does not assume a congested link.
- MAC layer signals again if reconnected.

Advantages:

1. Offers a way to resume TCP connection even after longer interruptions of the connection.
2. Independent of any other TCP mechanism, such as ACKs, sequence numbers etc.

Disadvantages:

1. The software on the MN and CN needs to be changed.
2. Depends on MAC layer

5.4.2.6 Selective retransmission

- TCP acknowledgements are cumulative.
- ACK n acknowledges correct & in-sequence receipt of packet up to n.
- If a single packet is lost quite often a whole packet sequence beginning at the gap has to be retransmitted.
- Bandwidth wastage.

Solution: Selective Retransmission

- Allows the receiver acknowledge a single packets.

- Now the sender can retransmit only the missing packet.

Advantage:

1. The sender retransmits only the lost packets.
2. Much higher efficiency. Lowers bandwidth requirement

Disadvantage:

1. More complex software on the MH.

5.4.2.7 Transaction-oriented TCP (T-TCP)

- TCP requires several trans reception of packets for:
  1. Connection setup
  2. Data transmission
  3. Connection release

(-) Even a short message needs minimum of 7 packets leads to connection overhead.

Solution: T-TCP

- Connection setup, Data transmission, Connection release can be combined, thus only 2 or 3 packets are needed.
- Reduces the total overhead.

Advantage: Reduction in overhead.

Disadvantage: Requires changed TCP, Mobility no longer transparent.

COMPARISON OF VARIOUS TCP

Approach	Mechanism	Advantages	Disadvantages
Indirect TCP	Splits TCP connection into two connections	Isolation of wireless link, simple.	Loss of TCP semantics. Higher latency at handover, security problems.
Snooping TCP	Snoops data and acknowledgements, local retransmission	Transparent for end to end connection, MAC integration possible	Insufficient isolation of Wireless link, security problems

M-TCP	Splits TCP connection, chokes sender via window size	Maintains end to end semantics, handles long term and frequent disconnections	Bad isolation of wireless link, processing overhead due to bandwidth management, security problems
Fast Retransmission /Fast Recovery	Avoids slow start ate roaming	Simple and efficient	Mixed layers, not transparent.
Transmission / Time out freezing	Freezes TCP state at disconnection, resumes after reconnection	Independent of content, works for longer interruptions	Changes in TCP required, MAC dependent
Selective retransmission	Retransmits only lost data.	Very efficient	Slightly more complex receiver software, more buffer space needed
Transaction oriented TCP	Combines connection setup-/ release and data retransmission	Efficient for certain applications	Changes in TCP required not transparent, security problems.

5.5 TCP over 2.5/3G

Introduction:

TCP over 2.5G/3G wireless networks describes a profile for optimizing TCP over wireless WANs such as GSM/GPRS, UMTS, or cdma2000. The focus on 2.5G/3G for transport of internet data is important as already more than 1 billion people use mobile phones and it is obvious that the mobile phone systems will also be used to transport arbitrary internet data.

Characteristics:

The following characteristics have to be considered when deploying applications over 2.5G/3G wireless links:

- **Data rates:** Typically, data rates are asymmetric as it is expected that users will download more data compared to uploading. Typical data rates of today's 2.5G systems are 10–20 Kbit/s uplink and 20–50 Kbit/s downlink, 3G and future 2.5G systems will initially offer data rates around 64 Kbit/s uplink and 115–384 Kbit/s downlink. Asymmetry does not exceed 3–6 times, however, considering broadcast systems as additional distribution media (Digital radio, satellite systems), asymmetry may reach a factor of 1,000.
- **Latency:** All wireless systems comprise elaborated algorithms for error correction and protection, such as forward error correction (FEC), check summing, and interleaving. FEC and interleaving let the round trip time (RTT) grow to several hundred milliseconds up to some seconds.
- **Jitter:** Wireless systems suffer from large delay variations or 'delay spikes'. Reasons for sudden increase in the latency are: link outages due to temporal loss of radio coverage, blocking due to high-priority traffic, or handovers.
- **Packet loss:** Packets might be lost during handovers or due to corruption. Due to link-level retransmissions the loss rates of 2.5G/3G systems due to corruption are relatively low

Configuration parameters to adapt TCP to wireless environments:

Based on these characteristics following configuration parameters are suggested to adapt TCP to wireless environments:

- **Large windows:** TCP should support large enough window sizes; A larger initial window may increase performance particularly for short transmissions. With the help of the windows scale option (RFC 1323) and larger buffer sizes this can be accomplished
- **Limited transmit:** It is an extension of Fast Retransmission/Fast Recovery and is particularly useful when small amounts of data are to be transmitted

- **Large MTU:** The larger the MTU (Maximum Transfer Unit) the faster TCP increases the congestion window. large MTUs may be used to increase performance. MTU path discovery should be used to employ larger segment sizes instead of assuming the small default MTU.
- **Selective Acknowledgement (SACK):** SACK allows the selective retransmission of packets and is almost always beneficial compared to the standard cumulative scheme.
- **Explicit Congestion Notification (ECN):** allows a receiver to inform a sender of congestion in the network by setting the ECN-Echo flag on receiving an IP packet that has experienced congestion. This mechanism makes it easier to distinguish packet loss due to transmission errors from packet loss due to congestion
- **Timestamp:** With the help of timestamps higher delay spikes can be tolerated by TCP without experiencing a spurious timeout. The effect of bandwidth oscillation is also reduced.
- **No header compression**: Header compression is not compatible with TCP options such as SACK or timestamps**.**