Platform for Control and Delivery of services in Next Generation Networks

Deliverable 1.2

Enabling Technologies

Questions about this document or the features it describes should be directed to:

CEFRIEL
Via Fucini, 2
20133 Milano (MI)
Italy

Politecnico di Torino
Corso Duca degli Abruzzi, 24
10129 Torino (To)
Italy

Abstract

The PICO project concentrates on application streaming and context awareness as topical techniques to build distributed applications in the domain of emergency situations (described in D1_1).

This document describes in detail the technologies: application streaming, context awareness, mobile devices and platforms.

These technologies will be used to develop prototypes of emergency applications.

CHANGE LOG

Version	Date	Description
1.0	26/03/2008	This is the first draft of the Deliverable 1.0
2.0	30 /10/2008	Final version

TABLE OF CONTENTS

Contents

Abstract **3**

TABLE OF CONTENTS **5**

Introduction **9**

1. Applications on Demand technologies **10**

1.1 Thin and fat client 10

1.1.1 Thin client Software 14

1.1.1.1 Xterm 14

1.1.1.2 OpenThinClient 15

1.1.2 Remote Desktop Software 16

1.1.2.1 VNC 17

1.1.2.2 ICA 18

1.1.2.3 RDP 19

1.1.2.4 Remote Desktop comparison 20

1.2 Virtual Machine 21

1.2.1 VMware 22

1.2.2 XEN 24

1.2.3 OpenVZ 26

1.3 Web 2.0 27

1.3.1 Ajax 28

1.3.2 Flash & Flex 29

1.3.3 Silverlight 31

1.3.4 JavaFX 32

1.3.5 Web 2.0 technologies comparison 34

1.4 Application Streaming 35

1.4.1 Application Deployment 35

1.4.2 SoftGrid 38

1.4.3 AppStream 41

1.4.4 Application streaming technologies comparison 45

1.5 Technologies Comparison 46

2. IP Multimedia Subsystem **51**

2.1 Introduction 51
2.2 IMS Architecture 53
2.2.1 HSS – Home Subscriber Server 54
2.2.2 CSCF - Call Session Control Function 55
2.2.2.1 P-CSCF 56
2.2.2.2 I-CSCF 56
2.2.2.3 S-CSCF 57
2.2.3 AS – Application Servers 57
2.2.4 MRF – Media Resource Function 58
2.2.4.1 MRFC – MRF Controller 58
2.2.4.2 MRFP – MRF Processor 58
2.2.5 BGCF – Breakout Gateway Control Function 58
2.2.6 MGCF – Media Gateway Controller Function 59
2.2.7 MGW – Media Gateway 59
2.2.8 SGW – Signalling Gateway 59
2.3 IMS implementation and services 59
2.3.1 SIP – Session Initiation Protocol 60
2.3.2 SDP – Session Description Protocol 62
2.3.3 RTP- Real time Transport Protocol 63
3. Mobile Devices 65
3.1 Development Environment 65
3.1.1 Java (J2ME) 65
3.1.2 Native code 66
3.1.3 Web 66
3.2 Mobile OS 67
3.2.1 iPhone 67
3.2.2 Symbian 68
3.2.3 Windows Mobile 69
3.2.4 Proprietary OS 69
3.2.5 Android 69
3.2.6 MOTOMAGX 71
3.2.7 Openmoko 71
3.3 Sample devices 72
3.3.1 Apple iPhone 72
3.3.2 Nokia N96 73
3.3.3 HTC TyTN II 73

4. Technologies support to Context-Aware 75
4.1 Context-Aware Mobility Solution 78
4.1.1 Active Location Tracking 79
4.1.2 Passive Location Tracking 80
4.1.3 Combining technologies 80
4.1.4 Mobility solution analysis 81
4.2 Common context-aware components 82
4.2.1 Web 2.0 83
4.2.2 Widget 84
4.2.3 REST (Representational State Transfer) 85
4.2.4 Social Networking 85
4.2.5 Context Ontology 86
4.2.6 Rule Engine 86
4.2.7 Recommendation Techniques 90
4.2.8 Data Mining 91
Appendix A Acronyms Table 92
Figures Index 94
Bibliography 95

Introduction

In the last years the telecommunication companies begin to develop new technologies based on the idea that in a server-client environment a lot of client resources result not necessary. The aim of this document is to describe and analyze the history of this development focusing on the current enabling technologies and devices. Indeed, Application on Demand, or Application streaming is an alternative to installing applications locally on individual PC's. Applications are streamed on demand from a central server; when the user has finished with the applications all components are completely removed - as if the application was never there. This solution leads to important advantages like: reduction of installation and software license costs, reduction of upgrading cost and consistent decrease of disk space requirements. Small storage capacity implies portability; as we saw, this is a fundamental feature in emergency scenarios. Moreover Public Protection Operators need flexible system able to be deployed in every situation and application on demand supply the solution of this problem with a quickly wireless communication.
In the document D1_1 we analyzed the possible emergency scenarios from an operator point of view, describing requirements, needs, and the importance of delivery application on demand.
In this document we will first present the state of the art regarding the available application on demand technologies. In the second section we will then introduce and describe the current mobile devices that can be used in order to stream applications focusing on features like space requirement, storage capacity, portability, etc. In the third section
The aim is in therefore to complete the emergency scenarios analysis proposed in D 1_1, with a technical overview on the available software and devices in order to enable a faster delivery of telecommunication services useful, for example, in emergency situations.

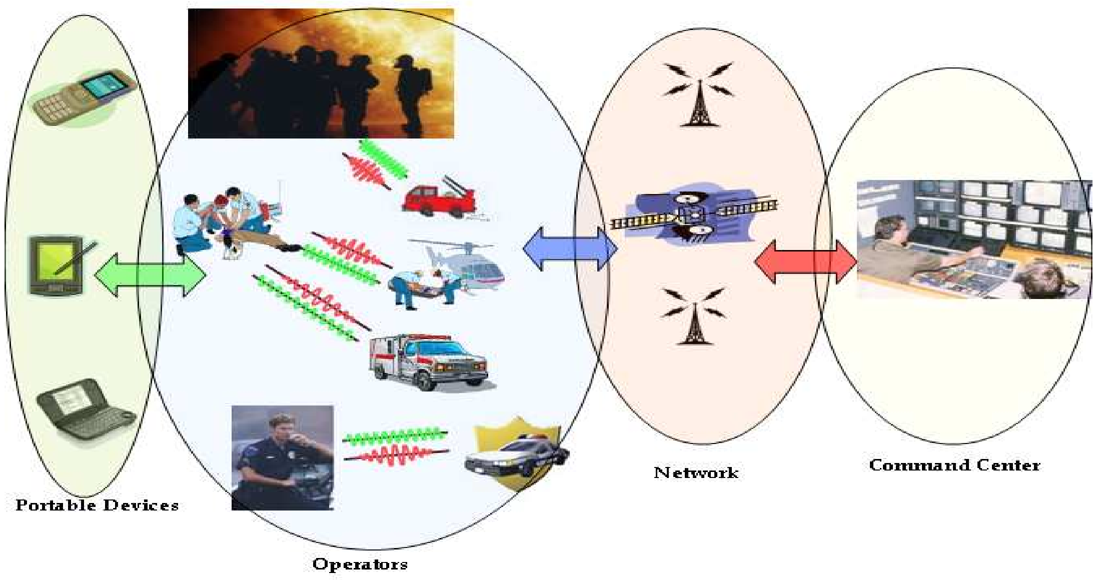


Figure 1: Communications in emergency situations.

1. Applications on Demand technologies

In this section we summarize the available technologies in order to delivery applications on demand and we propose a final comparison among the most important of them. In detail, in the first paragraph we are going to describe the starting point of this architecture, represented by the thin client and the remote desktop software; in the second paragraph we will focus on the virtualization technologies while in the third one we will present the new world wide web (Web 2.0) concepts and the available tools. Finally, before presenting a global comparison, we will describe the most famous proprietary application streaming software.

1.1 Thin and fat client

A *thin client* (sometimes also called a lean client) is a client computer or client software in client-server architecture networks which depends primarily on the central server for processing activities, and mainly focuses on conveying input and output between the user and the remote server. In contrast, a thick or *fat client* does as much processing as possible and passes only data for communications and storage to the server.
A ***thin client,*** as an application program, communicates with an application server and relies for most significant elements of its business logic on a separate piece of software, an application server, typically running on a host computer located nearby in a LAN or at a distance on a WAN or MAN. The term *thin client* is also sometimes used in an even broader sense which includes diskless nodes that does most of its processing on a central server with as little hardware and software as possible at the user's location, and as much as necessary at some centralized managed site. For example, the embedded operating system (OSE) in a thin client is stored in a "flash drive", in a Disk on Module (DOM), or is downloaded over the network at boot-up and usually uses some kind of write filter so that the OS and its configuration can only be changed by administrators.
In designing a client-server application, there is a decision to be made as to which parts of the task should be done on the client, and which on the server. This decision can crucially affect the cost of clients and servers, the robustness and security of the application as a whole, and the flexibility of the design for later modification or porting. One design question is how application-specific the client software should be. Using standardized client software such as a Web browser can save on development costs, since one does not need to develop a custom client, but one must accept the limitations of the standard client.

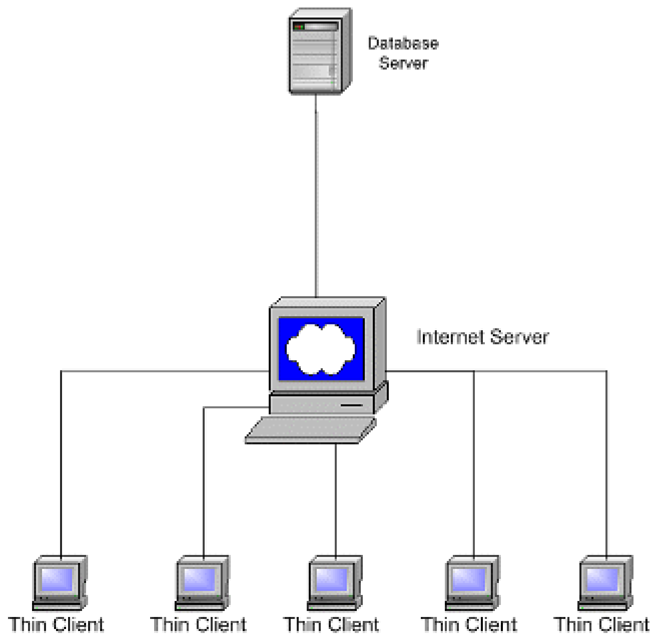


Figure 2: Thin clients in a client-server context.

The most important advantages using a thin client are:

- **Lower IT administration costs**. Thin clients are managed almost entirely at the server. The hardware has fewer points of failure and the client is simpler (and often lacks permanent storage), providing protection from malware.
- **Easier to secure**. Thin clients can be designed so that no application data ever resides on the client (just whatever is displayed), centralizing malware protection and reducing the risks of physical data theft.
- **Enhanced data security**. Should a thin client device suffer serious mishap or industrial accident, no data will be lost, as it resides on the terminal server and not the point-of-operation device.
- **Lower hardware costs**. Thin client hardware is generally cheaper because it does not contain a disk, application memory, or a powerful processor. They also generally have a longer period before requiring an upgrade or becoming obsolete. The total hardware requirements for a thin client system (including both servers and clients) are usually much lower compared to a system with fat clients. One reason for this is that the hardware is better utilized. A CPU in a fat workstation is idle most of the time. With thin clients, memory can be shared. If several users are running the same application, it only needs to be loaded into RAM once with a central server. With fat clients, each workstation must have its own copy of the program in memory.
- **Less Energy Consumption**. Dedicated thin client hardware has much lower energy consumption than thick client PCs. This not only reduces energy costs but may mean that in some cases air-conditioning systems are not required or need not be upgraded which can be a significant cost saving and contribute to achieving energy saving targets. However, more powerful servers and communications are required.

- **Easier hardware failure management**. If a thin client fails, a replacement can simply be swapped in while the client is repaired; the user is not inconvenienced because their data is not on the client.
- **Worthless to most thieves**. Thin client hardware, whether dedicated or simply older hardware that has been repurposed via cascading, is useless outside a client-server environment. Burglars interested in computer equipment have a much harder time fencing thin client hardware (and it is less valuable).
- **Hostile Environments**. Most thin clients have no moving parts so can be used in dusty environments without the worry of PC fans clogging up and overheating and burning out the PC.
- **Less network bandwidth**. Since terminal servers typically reside on the same high-speed network backbone as file servers, most network traffic is confined to the server room. In a fat client environment if you open a 10MB document that's 10MB transferred from the file server to your PC. When you save it that's another 10MB from your PC to the server. When you print it the same happens again – another 10MB over the network to your print server and then 10MB onward to the printer. This is highly inefficient. In a thin client environment only mouse movements, keystrokes and screen updates are transmitted from/to the end user. Over efficient protocols such as ICA or NX this can consume as little as 5 kbit/s bandwidth.
- **More efficient use of computing resources**. A typical thick-client will be specified to cope with the maximum load the user needs, which can be inefficient at times when it is not used. In contrast, thin clients only use the exact amount of computing resources required by the current task – in a large network, there is a high probability the load from each user will fluctuate in a different cycle to that of another user (i.e. the peaks of one will more than likely correspond, time-wise, to the troughs of another).
- **Simple hardware upgrade path**. If the peak resource usage is above a pre-defined limit, it is a relatively simple process to add another rack to a blade server (be it power, processing, storage), boosting resources to exactly the amount required. The existing units can continue to serve alongside the new, whereas a thick client model requires an entire desktop unit be replaced, resulting in down-time for the user, and the problem of disposing of the old unit.
- **Lower noise**. The aforementioned removal of fans reduces the noise produced by the unit. This can create a more pleasant and productive working environment.
- **Less Wasted Hardware**. Computer hardware is very environmentally damaging. Thin clients can remain in service longer and ultimately produce less surplus computer hardware than an equivalent thick client installation.

As we said, another possible solution in client-server architecture is the use of fat-clients. A ***fat client*** is a computer (client) in a network which typically provides rich functionality independently of the central server. A fat client still requires at least periodic connection to a network or central server, but is often characterized by the ability to perform many functions without that connection. In some cases fat client could be convenient. Important advantages are:

- **Fewer server requirements**. A thick client server does not require as high a level of performance as a thin client server (since the thick clients themselves do much of the application processing). This results in drastically cheaper servers.
- **Offline working**. Thick clients have advantages in that a constant connection to the central server is often not required.
- **Better multimedia performance**. Thick clients have advantages in multimedia-rich applications that would be bandwidth intensive if fully served. For example, thick clients are well suited for video gaming.

- **More flexibility**. On some operating systems software products are designed for personal computers that have their own local resources. Trying to run this software in a thin client environment can be difficult (such as Microsoft Windows). Linux and Unix-like systems in general have greater flexibility in this area.

Depending on the tradeoff between low development costs and the standard client limitations we might say that we use either a thin client, a thick/fat client, or a ***hybrid client model*** in which some applications (such as web browsers) are run locally, while other applications (such as critical business systems) are run on the terminal server. One way to implement this is simply by running remote desktop software on a standard desktop computer. This introduces us to the concept of *centralized computing*.
The centralized computing indicates a technology where the computing is done at a central location, using terminal that are attached to a central computer. The terminals may be text terminals or thin clients for example. It offers greater security over decentralized systems because all of the processing is controlled in a central location. In addition, if one terminal breaks down, the user can simply go to another terminal and log in again, and all of their files will still be accessible. This type of arrangement does have some disadvantages. The central computer performs the computing functions and controls the remote terminals. This type of system relies totally on the central computer. Should the central computer crash, the entire system will be unavailable. For these reasons a ***terminal server*** is commonly defined as a server used in centralized computing. There are two contemporary models of centralized computing:

- **Thin client model**: the terminal server provides a Windows or Linux desktop to multiple users.
- **Remote desktop model**: an ordinary computer acts temporarily as a terminal server, providing its desktop to a remote computer over a wide area network such as the Internet (software clients used in this architecture are known as remote desktop applications; however, these remote desktop applications are also used in the thin client model as well).

In the rest of this paragraph we describe the most important software and tools regarding these two models.

1.1.1 Thin client Software

Thin clients have been used for many years by businesses to reduce total cost of ownership, while web applications are becoming more popular because they can potentially be used on many types of computing device without any need for software installation. However, during the last years the structure is changing away from pure centralization, as thin client devices become more like diskless workstations due to increased computing power, and web applications start to do more processing on the client side, with technologies such as AJAX and rich clients. In addition, mainframes are still being used for some critical applications, such as payroll, or for processing day-to-day account transactions in banks. These mainframes will typically be accessed either using terminal emulators or via web applications.
Regarding thin client technologies, we report the most famous client that was the pioneer of the currents available technologies and the current open source solution:

- **Xterm** is the standard terminal emulator for the X Window System; a user can have many different invocations of xterm running at once on the same display, each of which provides independent input/output for the process running in it.
- **OpenThinClient** is an open source thin client Solution consisting of a Linux based operating system along with a comprehensive Java based management GUI and server component. It is intended for environments where a medium to large number of thin clients must be supported and managed efficiently.

1.1.1.1 Xterm

In computing, the X Window System is a system which implements the X display protocol and provides windowing on bitmap displays. It provides the standard toolkit and protocol with which to build graphical user interfaces (GUIs) on most Unix-like operating systems and OpenVMS, and has been ported to many other contemporary general purpose operating systems. X provides the basic framework, or primitives, for building GUI environments: drawing and moving windows on the screen and interacting with a mouse and/or keyboard. X does not mandate the user interface — individual client programs handle this. As such, the visual styling of X-based environments varies greatly; different programs may present radically different interfaces. X is not an integral part of the operating system; instead, it is built as an additional application layer on top of the operating system kernel.
An X terminal is a thin client that runs an X server. This architecture became popular for building inexpensive terminal parts for many users to simultaneously use the same large server (making programs being run on the server clients of the X terminal).
X terminals explore the network (the local broadcast domain) using the X Display Manager Control Protocol to generate a list of available hosts that they can run clients from. The initial host needs to run an X display manager. Dedicated (hardware) X terminals have become less common; a PC or modern thin client with an X server typically provides the same functionality at the same, or lower, cost.

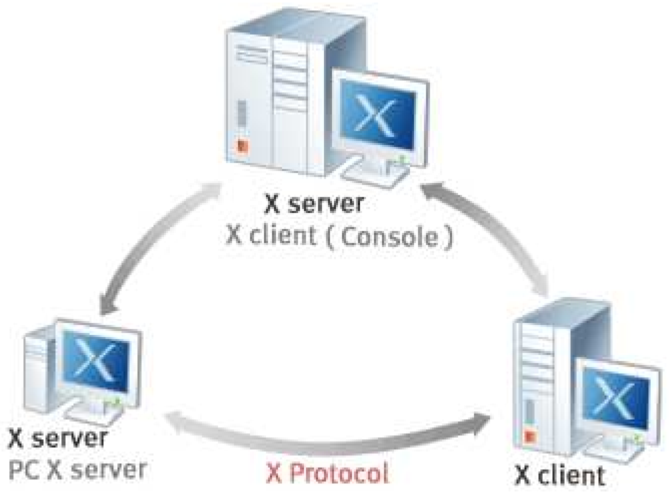


Figure 3: X Windows System.

1.1.1.2 OpenThinClient

The OpenThinClient operating system is based on a customized Ubuntu Linux distribution optimized for use in diskless devices. Booting and configuration of the thin clients is implemented using industry-standard technologies like LDAP, DHCP, PXE, TFTP and NFS. OpenThinClient provides a powerful, Java-based graphical user interface to manage all aspects of the thin clients under its control. Furthermore, it supports integration with enterprise-wide management environments like LDAP or MS ADS.

Openthinclient differs from other solutions in the following aspects:

- Based on industry-standard protocols and technologies-integrates smoothly with existing systems management solutions like LDAP and MS ADS.
- Features a powerful management GUI - supports a large range of thin client hardware.
- No specialized hardware is required. The thin client only needs a PXE-capable network interface and no local storage like flash or hard disk. (i.e. boots devices into thin client mode without flash drives thanks to its PXE boot support).
- Several thin client applications come pre-packaged, like a Web browser, RDP client etc.
- The OpenThinClient Manager and the OpenThinClient Server are written in Java so they will run on any OS that is supported by Sun Java 6.
- A complete open source thin client solution free of charge.

1.1.2 Remote Desktop Software

In computing, remote desktop software is remote access and remote administration software that allows graphical user interface applications to be run remotely on a server, while being displayed locally.
Remote desktop applications have varying features: some allow attaching to an existing user's session (i.e. a running desktop) and remote controlling it in front of the user's eyes. It can also be explained as remote control of a computer by using another device connected via the internet or another network (see figure below). This is widely used by many computer manufacturers for technical troubleshooting for their customers. The quality, speed and functions of any remote desktop protocol are based on the system layer where the graphical desktop is redirected. Software such as VNC uses the top software layer to extract and compress the graphic interface images for transmission. Other products such as Microsoft RDP and others use a kernel driver level to construct the remote desktop for transmission.
We report the features and characteristics of the most important solutions concerning this architecture.

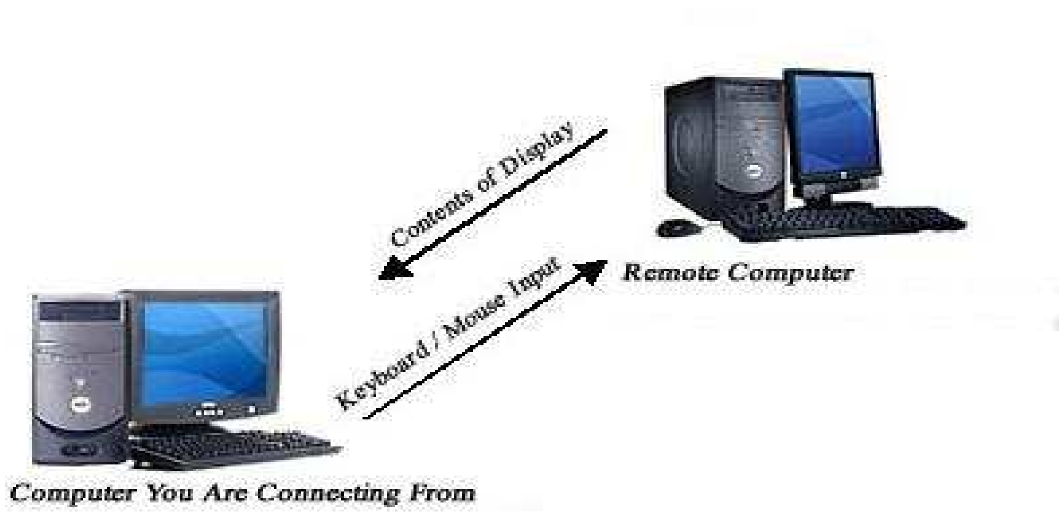


Figure 4: Remote Desktop interaction.

1.1.2.1 VNC

Virtual Network Computing (VNC) is a graphical desktop sharing system which is made up by three actors:

- a server
- a client
- a protocol

In detail, the VNC architecture uses the RFB protocol (remote framebuffer protocol). The server sends small rectangles of the framebuffer to the client to remotely control another computer. It transmits the keyboard and mouse events from one computer to another, relaying the graphical screen updates back in the other direction, over a network.

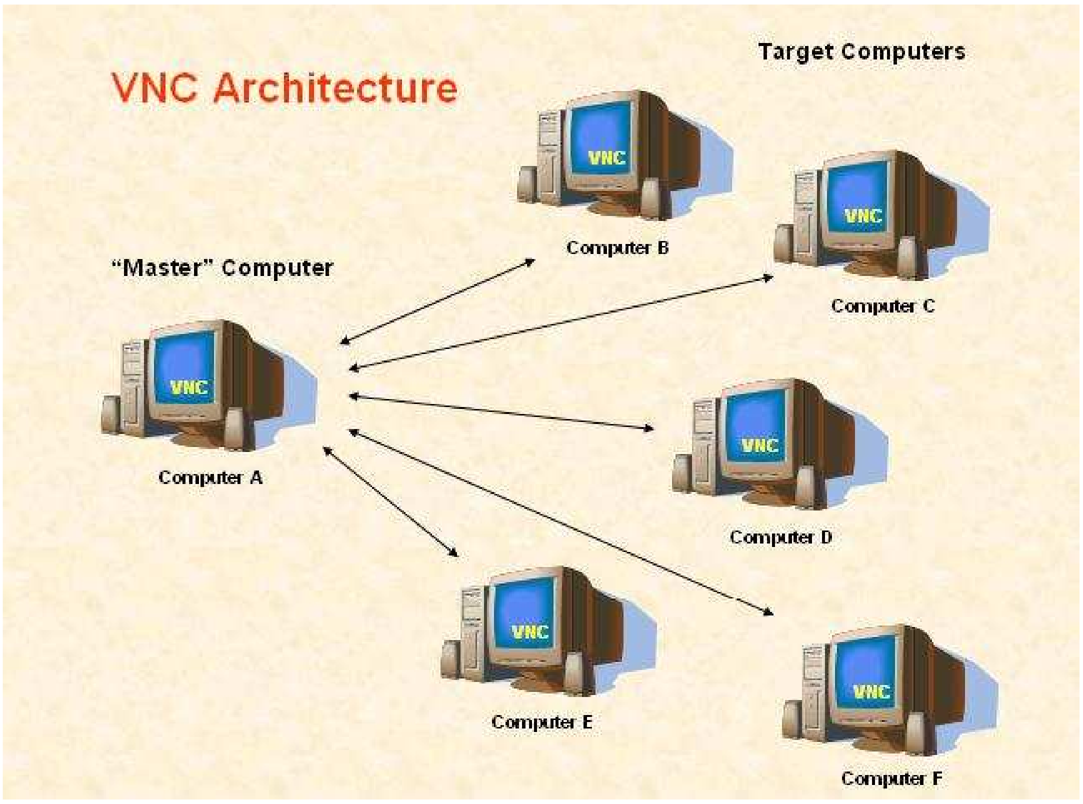


Figure 5: VNC architecture.

In its simplest form, the VNC protocol can use a lot of bandwidth, so various methods have been devised to reduce the communication overhead. For example, there are various *encodings* in order to determine the most efficient way to transfer the rectangles. The VNC protocol allows the client and server to negotiate which encoding will be used. The simplest encoding, which is supported by all clients and servers, is the *raw encoding* where pixel data is sent in left-to-right scan line order, and after the original full screen has been transmitted, only transfers rectangles that change. This encoding works very well if only a small portion of the screen changes from one frame to the next (like a mouse pointer moving across a desktop, or text being written at the cursor), but bandwidth demands get very high if a lot of

pixels change at the same time, such as when scrolling a window or viewing full-screen video.
The main features of VNC are:

- **VNC is platform-independent.** A VNC viewer on any operating system can usually connect to a VNC server on any other operating system. There are clients and servers for almost all GUI operating systems and for Java. Multiple clients may connect to a VNC server at the same time. Popular uses for this technology include remote technical support and accessing files on one's work computer from one's home computer, or vice versa.
- **VNC is not a secure protocol.** While passwords are not sent in plain-text (as in telnet), brute-force cracking could prove successful if both the encryption key and encoded password are sniffed from a network.
- **No state is stored at the viewer.** This means you can leave your desk, go to another machine, whether next door or several hundred miles away, reconnect to your desktop from there and finish the sentence you were typing. Even the cursor will be in the same place.
- **It is small and simple.** The Win32 viewer, for example, is about 150K in size and can be run directly from a floppy. There is no installation needed.
- **It is sharable.** One desktop can be displayed by several viewers at once even if only one is able to work on it.
- **It is available for download** and under the terms of the GNU Public License.

1.1.2.2 ICA

Independent Computing Architecture (ICA) is a proprietary protocol for an application server system, designed by Citrix Systems. A key challenge of the ICA architecture is performance; a graphically intensive application (as most are when presented using a GUI) served over a slow or bandwidth-restricted network connection requires considerable compression and optimization to render the application usable by the client. The client machine may be a different platform, and may not have the same GUI routines available locally: in this case the server may need to send the actual bitmap data over the connection. Depending on the client's capabilities, servers may also off-load part of the graphical processing to the client, e.g. to render multi-media content. The protocol lays down a specification for passing data between server and clients, but is not bound to any one platform.
Practical products conforming to ICA are Citrix's WinFrame and Citrix Presentation Server (formerly called Metaframe) products. These permit ordinary Windows applications to be run on a suitable Windows server, and for any supported client to gain access to those applications. Besides Windows, ICA is also supported on a number of Unix server platforms and can be used to deliver access to applications running on these platforms. The client platforms need not run Windows; for example, there are clients for Mac, Unix, Linux, and various Smartphones. ICA client software is also built into various thin client platforms.
ICA is broadly similar in purpose to window servers such as the X Window System. It also provides for the feedback of user input from the client to the server, and a variety of means for the server to send graphical output, as well as other media such as audio, from the running application to the client.

1.1.2.3 RDP

Terminal Services is one of the components of Microsoft Windows (both server and client versions) that allows a user to access applications and data on a remote computer over any type of network, although normally best used when dealing with either a Wide Area Network or Local Area Network, as ease and compatibility with other types of networks may differ. Terminal Services is Microsoft's implementation of thin-client terminal server computing, where Windows applications, or even the entire desktop of the computer running terminal services, are made accessible from a remote client machine.

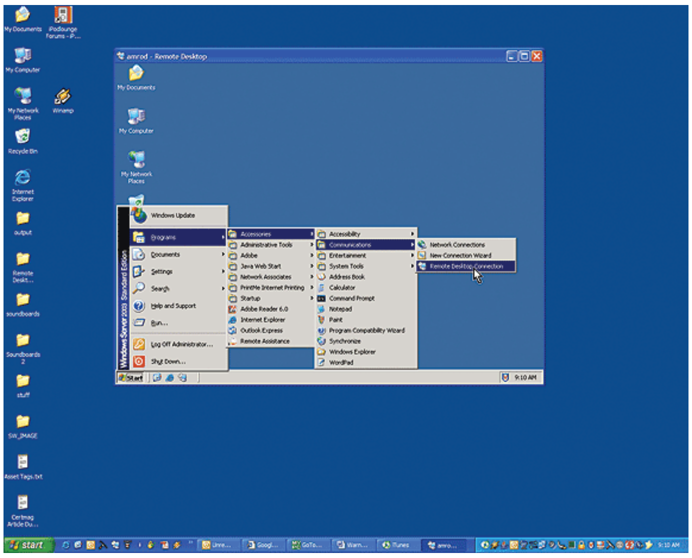


Figure 6: Remote Desktop Connection.

Remote Desktop Connection (RDC) is the client application for Terminal Services. It allows a user to remotely log in to a networked computer running the terminal services server. RDC presents the desktop interface of the remote system; as if it were accessed locally (i.e. it allows watching and controlling the desktop's session of another pc; see figure). RDC uses the Remote Desktop Protocol (RDP) that is a multi-channel protocol in order to allow a user to connect to a computer running Microsoft Terminal Services. Microsoft refers to the official RDP client software as either Remote Desktop Connection or Terminal Services Client. The server component of Terminal Services (Terminal Server) listens on TCP port 3389. On the server, RDP uses its own video driver to render display output by constructing the rendering information into network packets by using RDP protocol and sending them over the network to the client. On the client, RDP receives rendering data and interprets the packets into corresponding Microsoft Win32 graphics device interface (GDI) API calls. For the input path, client mouse and keyboard events are redirected from the client to the server. On the server, RDP uses its own on-screen keyboard and mouse driver to receive these keyboard and mouse events.

Regarding the encryption, RDP uses RSA Security's RC4 cipher, a stream cipher designed to efficiently encrypt small amounts of data. RC4 is designed for secure communications over networks. Beginning with Windows 2000, administrators can choose to encrypt data by using a 56-or 128-bit key.

For the bandwidth's problem RDP supports various mechanisms to reduce the amount of data transmitted over a network connection. Mechanisms include data compression, persistent caching of bitmaps, and caching of glyphs and fragments in RAM. The persistent bitmap cache can provide a substantial improvement in performance over low-bandwidth connections, especially when running applications that make extensive use of large bitmaps.

1.1.2.4 Remote Desktop comparison

In this section we propose a fast comparison among the systems presented; as we saw these protocols may differ in encoding method, caching, bandwidth usage, latency, video quality, etc. The idea is to underline these features evaluating in such a way the performances of these products.
Concerning the following results, the values are evaluated on tests over Wide Area Network and the PDA model used is AXIM V5.
Regarding the percentages on the audio/video quality they are related to a 100% available quality with a personal computer.

	VNC	**ICA**	**RDP**
Web Browsing (Page Download Latency) on PC	Medium ~0.7 Sec	Medium ~0.7 Sec	Medium ~1 Sec
Web Browsing (Page Download Latency) on PDA	High ~10 Sec	Medium ~1 Sec	Medium ~0.5 Sec
Audio/Video Quality on PC	Low ~ 5%	Medium ~20%	Low ~ 5%
Audio/Video Quality on PDA	Low ~ 2%	Low ~ 2%	Medium ~ 10%
Web Browsing Data Transfer (per page) on PC	Medium ~ 100 KB	Medium ~ 150 KB	Medium ~ 200 KB
Web Browsing Data Transfer (per page) on PDA	Medium ~ 60 KB	Low ~ 20 KB	Low ~ 10 KB
Audio Video Data Transfer on PC	Medium ~ 20 MB	Medium ~ 30 MB	Medium ~ 10 MB
Audio Video Data Transfer on PDA	Low ~ 2 MB	Low ~ 3 MB	Low ~ 3 MB
Bandwidth Usage	High	Medium	Medium

As it possible to see on the table RDP and ICA claim almost the similar performances. VNC has instead some drawbacks also because the display encoding is not well supported causing an increase in terms of bandwidth usage.

1.2 Virtual Machine

Virtualization is a broad term that refers to the abstraction of computer resources. One definition is the following: *Virtualization is a technique for hiding the physical characteristics of computing resources from the way in which other systems, applications, or end users interact with those resources. This includes making a single physical resource (such as a server, an operating system, an application, or storage device) appear to function as multiple logical resources; or it can include making multiple physical resources (such as storage devices or servers) appear as a single logical resource.*
As with other terms, such as abstraction and object orientation, *virtualization* is used in many different contexts, which can be grouped into two main types:

- **Platform virtualization** involves the simulation of whole computers.
- **Resource virtualization** involves the simulation of combined, fragmented, or simplified resources.

In practice, virtualization creates an external interface that hides an underlying implementation (e.g., by multiplexing access, by combining resources at different physical locations, or by simplifying a control system). Recent development of new virtualization platforms and technologies has refocused attention on this mature concept.
More in detail, a **hypervisor** (or **virtual machine monitor-VMM**) is indeed a virtualization platform that allows multiple operating systems to run on a host computer at the same time.
Hypervisors are currently classified in two types:

- A **native** (or **bare-metal**) hypervisor is a software that runs directly on a given hardware platform (as an operating system *control program*). A guest operating system thus runs at the second level above the hardware. The classic native hypervisor was CP/CMS, developed at IBM in the 1960s, ancestor of IBM's current z/VM. More recent examples are open source Xen, Citrix XenServer, Oracle VM, VMware's ESX Server, L4 microkernels, Green Hills Software's INTEGRITY Padded Cell, VirtualLogix's VLX, etc.

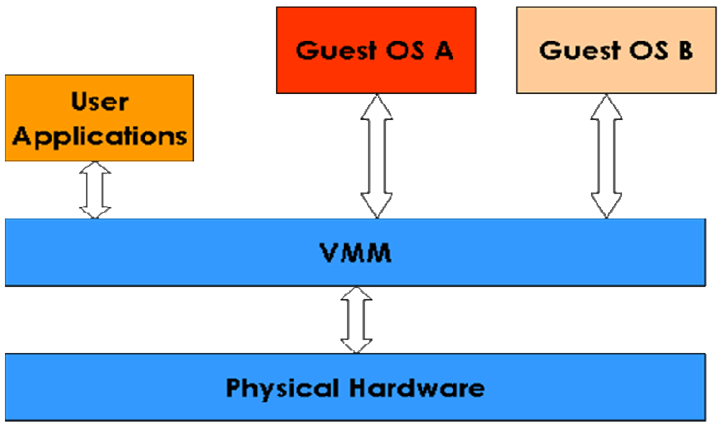


Figure 7: Native Hypervisor.

- A **hosted** hypervisor is software that runs within an operating system environment. A guest operating system thus runs at the third level above the hardware. Examples include VMware Server (formerly known as GSX), VMware Workstation, VMware Fusion, the open source QEMU, Microsoft's Virtual PC and Microsoft Virtual Server products, InnoTek's VirtualBox, as well as SWsoft's Parallels Workstation and Parallels Desktop.

There is seldom requirement for a guest OS to be the same as the host one. The guest system often requires access to specific peripheral devices to function, so the simulation must support the guest's interfaces to those devices.

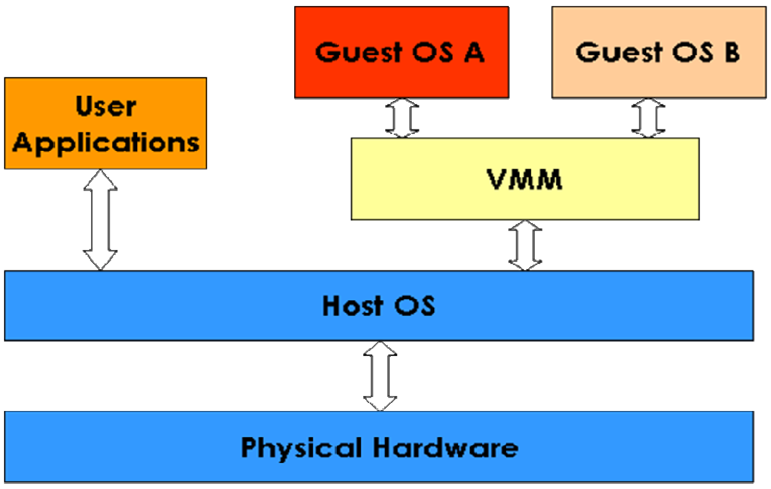


Figure 8: Hosted Hypervisor.

We present the most important solutions regarding platform virtualization.

1.2.1 VMware

VMware Inc, a publicly-listed company, develops proprietary virtualization software products for x86-compatible computers, including both commercially-available and freeware versions. The name VMware comes from the acronym VM, meaning *virtual machine* and ware comes from second part of *Software*.
The two main product categories produced from VMware are:

- **Desktop Software** like *VMware Workstation* that allows users to run multiple instances of x86 or x86-64 compatible operating systems on a single physical PC. *VMware Fusion* provides similar functionality for users of the MacIntel platform, along with full compatibility with virtual machines created by other VMware products. For users without a license to use VMware Workstation or VMware Fusion, VMware offers the freeware *VMware Player* product, which can run (but not create) virtual machines.

- **Server Software** like *VMware ESX Server* (formerly called "ESX Server") and *VMware Server* (formerly called "GSX Server"). VMware ESX, an enterprise-level product, can deliver greater performance than the freeware VMware Server, due to lower system overhead. In addition, VMware ESX integrates into VMware Virtual Infrastructure, which offers extra services to enhance the reliability and manageability of a server deployment. VMware Server is also provided as freeware, like VMware Player but it is possible to create virtual machines with it.

VMware's desktop software runs atop Microsoft Windows, Linux, and Mac OS X **(hosted)** instead VMware ESX Server, runs directly on server hardware without requiring an additional underlying operating system **(native-bare metal)**.

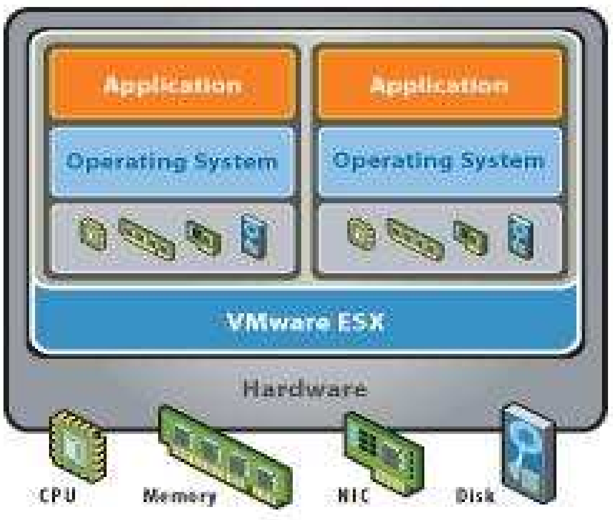


Figure 9: VMware ESX Architecture.

VMware refers to the physical hardware computer as the host machine, and identifies the operating system (or virtual appliance) running inside a virtual machine as the guest. This terminology applies to both personal and enterprise-wide VMware software. Like an emulator, VMware software provides a completely virtualized set of hardware to the guest operating system. VMware software virtualizes the hardware for a video adapter, a network adapter, and hard disk adapters. The host provides pass-through drivers for guest USB, serial, and parallel devices. In this way, VMware virtual machines become highly portable between computers, because every host looks nearly identical to the guest. In practice, a systems administrator can pause operations on a virtual machine guest, move or copy that guest to another physical computer, and there resume execution exactly at the point of suspension. Alternately, for enterprise servers, a feature called VMotion allows the migration of operational guest virtual machines between similar but separate hardware hosts sharing the same storage area network (SAN).
However, unlike an emulator, such as Virtual PC for PowerPC Macintosh computers, VMware software does not emulate an instruction set for different hardware not physically present. This significantly boosts performance, but can cause problems when moving virtual machine guests between hardware hosts using different instruction-sets (such as found in 64-bit Intel and AMD CPUs), or between hardware hosts with a differing number of CPUs. Stopping the virtual-machine guest before moving it to a different CPU type generally causes no issues.
The VMware Tools package adds drivers and utilities to improve the graphical performance for different guest operating systems, including mouse tracking. The package also enables some integration between the guest and host systems, including shared folders, plug-and-play devices, clock synchronization, and cutting-and-pasting across environments.

1.2.2 XEN

Xen is a free software virtual machine monitor for IA-32, x86-64, IA-64 and PowerPC 970 architectures. It allows several guest operating systems to be executed on the same computer hardware at the same time. Xen originated as a research project at the University of Cambridge, led by Ian Pratt, senior lecturer at Cambridge and founder of XenSource, Inc. This company now supports the development of the open source project and also sells enterprise versions of the software. The first public release of Xen was made available in 2003. XenSource, Inc was acquired by Citrix Systems in October 2007. XenSource's products have subsequently been renamed under the Citrix brand:

- *XenExpress* was renamed *XenServer Express Edition* and *XenServer OEM Edition* (embedded hypervisor);
- *XenServer* was renamed *XenServer Standard Edition*;
- *XenEnterprise* was renamed *XenServer Enterprise Edition*.

When Citrix Systems completed its acquisition of XenSource the Xen project moved to http://www.xen.org. This move had been under way for some time, and afforded the project an opportunity to make public the existence of the Xen Project Advisory Board (Xen AB), which currently has members from Citrix, IBM, Intel, Hewlett-Packard, Novell, Red Hat and Sun Microsystems. The Xen AB is chartered with oversight of the project's code management procedures, and with development of a new trademark policy for the Xen mark, which Citrix intends to freely license to all vendors and projects that implement the Xen hypervisor; the requirements for licensing will be solely the responsibility of the Xen AB.
Regarding the features system, a Xen system is structured with the Xen hypervisor as the lowest and most privileged layer. Above this layer are one or more guest operating systems, which the hypervisor schedules across the physical CPUs. The first guest operating system, called in Xen terminology "domain 0" (dom0), is booted automatically when the hypervisor boots and given special management privileges and direct access to the physical hardware. The system administrator logs into dom0 in order to start any further guest operating systems, called "domain U" (domU) in Xen terminology.
Modified versions of Linux, NetBSD and Solaris can be used as the dom0. Several modified Unix-like operating systems may be employed as guest operating systems (domU); on certain hardware, as of Xen version 3.0, unmodified versions of Microsoft Windows and other proprietary operating systems can also be used as guests if the CPU supports Intel VT or AMD V technologies.
As we know, the primary benefits of server virtualization are consolidation, increased utilization, and ability to rapidly provide and start a virtual machine, and increased ability to dynamically respond to faults by re-booting a virtual machine or moving a virtual machine to different hardware. Another benefit is the ability to securely separate virtual operating systems, and the ability to support legacy software as well as new OS instances on the same computer. Xen's support for virtual machine live migration from one host to another allows workload balancing and the avoidance of downtime. Besides, Xen virtual machines can be "live migrated" between physical hosts across a LAN without loss of availability. During this procedure, the memory of the virtual machine is iteratively copied to the destination without stopping its execution. Stoppage of around 60–300 ms is required to perform final synchronization before the virtual machine begins

executing at its final destination, providing an illusion of seamless migration. Similar technology is used to suspend running virtual machines to disk and switch to another virtual machine, and resume the first virtual machine at a later date.
Xen may also be used on personal computers that run Linux but also have Windows installed. Traditionally, such systems are used in a dual boot setup, but with Xen it is possible to start Windows "in a window" from within Linux, effectively running applications from both systems at the same time.

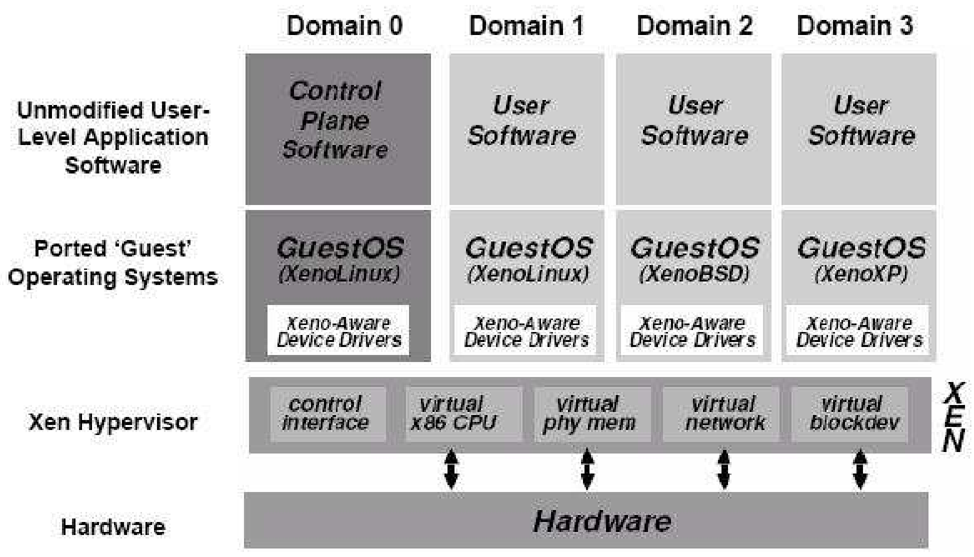


Figure 10: XEN Architecture.

1.2.3 OpenVZ

We conclude this section with a fast description of an open source based solution. OpenVZ is an operating system-level virtualization technology based on the Linux kernel and operating system. OpenVZ allows a physical server to run multiple isolated operating system instances, known as containers, Virtual Private Servers (VPSs), or Virtual Environments (VEs). Each container performs and executes exactly like a stand-alone server; containers can be rebooted independently and have root access, users, IP addresses, memory, processes, files, applications, system libraries and configuration files.
As compared to virtual machines such as VMware, OpenVZ is limited in that it requires both the host and guest OS to be Linux (although Linux distributions can be different in different containers). However, OpenVZ claims a performance advantage; according to its website, there is only a 1-3% performance penalty for OpenVZ as compared to using a standalone server. An independent performance evaluation confirms this.
The OpenVZ project is an open source community project supported by Parallels and is intended to provide access to the code and ultimately for the open source community to test, develop and further the OS virtualization effort. It is also a proving ground for new technology that may evolve into the Parallels Virtuozzo Containers product offering.

The most important features are:

- **Scalability**: As OpenVZ employs a single kernel model, it is as scalable as the 2.6 Linux kernel; that is, it supports up to 64 CPUs and up to 64 GiB of RAM.
- **Performance**: The virtualization overhead observed in OpenVZ is limited, and can be neglected in many scenarios.
- **Density**: OpenVZ is able to host hundreds of containers.
- **Mass-management**: An administrator (i.e. root) of an OpenVZ physical server (also known as a Hardware Node or host system) can see all the running processes and files of all the containers on the system. That makes mass management scenarios possible. Consider that VMware or Xen is used for server consolidation: in order to apply a security update to 10 virtual servers, an administrator is required to log in into each one and run an update procedure. With OpenVZ, a simple shell script can update all containers at once.

1.3 Web 2.0

Web 2.0 is a trend in the use of World Wide Web technology and web design that aims to facilitate creativity, information sharing, and, most notably, collaboration among users. These concepts have led to the development and evolution of web-based communities and hosted services, such as social-networking sites, wikis, blogs, and folksonomies.
The term became notable after the first O'Reilly Media Web 2.0 conference in 2004. Although the term suggests a new version of the World Wide Web, it does not refer to an update to any technical specifications, but to changes in the ways software developers and end-users use webs.
Web 2.0 websites allow users to do more than just retrieve information. They can build on the interactive facilities of Web 1.0 to provide *Network as platform,* computing, allowing users to run software-applications entirely through a browser. Users can own the data on a Web 2.0 site and exercise control over that data. These sites may have an architecture of participation that encourages users to add value to the application as they use it. This stands in contrast to very old traditional websites, the sort which limited visitors to viewing and whose content only the site's owner could modify. Web 2.0 sites often feature a rich, user-friendly interface based on Ajax, Flex or similar rich media. The sites may also have social-networking aspects. The concept of Web-as-participation-platform captures many of these characteristics. Bart Decrem, a founder and former CEO of Flock, calls *Web 2.0 the participatory Web* and regards the Web as information-source as Web 1.0.
The impossibility of excluding group-members who do not contribute to the provision of goods from sharing profits gives rise to the possibility that rational members will prefer to withhold their contribution of effort and free-ride on the contribution of others.

We can summarize the main characteristics of Web 2.0:

- rich user experience
- user participation
- dynamic content
- metadata
- web standards
- scalability
- openness
- freedom
- collective intelligence by way of user participation

After this brief introduction on the meaning of Web 2.0, we are going to describe the most important available technologies used in the Web 2.0 world.

1.3.1 Ajax

Like DHTML and LAMP, AJAX (Asynchronous JavaScript and XML) is not a technology in itself, but a term that refers to the use of a group of technologies. It is a group of inter-related web development techniques used for creating interactive web applications. The main characteristics of AJAX are:

- **It increases responsiveness and interactivity of web pages** achieved by exchanging small amounts of data with the server "behind the scenes" so that entire web pages do not have to be reloaded each time there is a need to fetch data from the server. This is intended to increase the web page's interactivity, speed, functionality and usability.
- **AJAX is asynchronous**, in that extra data is requested from the server and loaded in the background without interfering with the display and behavior of the existing page. JavaScript is the scripting language in which AJAX function calls are usually made. Data is retrieved using the *XMLHttpRequest (the core of AJAX* which gives browsers the ability to make dynamic and asynchronous data requests without having to reload a page, eliminating the need for page refreshes*)* object that is available to scripting languages run in modern browsers, or, alternatively, through the use of Remote Scripting in browsers that do not support XMLHttpRequest. In any case, it is not required that the asynchronous content be formatted in XML.
- **Ajax is a cross-platform technique** usable on many different operating systems, computer architectures, and web browsers as it is based on open standards such as JavaScript and the document object model (DOM). There are free and open source implementations of suitable frameworks and libraries.
- **Ajax uses a combination of**:
  - *XHTML* (or *HTML*) and *CSS* for marking up and styling information.
  - The *DOM* accessed with a client-side scripting language, especially ECMAScript implementations such as JavaScript and JScript, to dynamically display and interact with the information presented.
  - The *XMLHttpRequest* object is used to exchange data asynchronously with the web server. In some Ajax frameworks and in certain situations, an IFrame object is used instead of the XMLHttpRequest object to exchange data with the web server, and in other implementations, dynamically added <script> tags may be used.
  - *XML* is sometimes used as the format for transferring data between the server and client, although any format will work, including preformatted HTML, plain text and JSON. These files may be created dynamically by some form of server-side scripting.

Finally, we conclude this brief description recalling that main advantages and disadvantages of this technology.

- **Bandwidth usage**: By generating the HTML locally within the browser, and only bringing down JavaScript calls and the actual data, Ajax web pages can appear to load relatively quickly since the payload coming down is much smaller in size, and the rest of the layout does not have to be redrawn on each update.
- **Separation of Data, Format, Style and Function**: A less specific benefit of the Ajax approach is that it tends to encourage programmers to clearly separate the

methods and formats used for the different aspects of information delivery via the web.

On the contrary the following problems arise:

- **Browser integration**: the dynamically created page does not register itself with the browser history engine, so triggering the "Back" function of the users' browser might not bring the desired result.
- **Response-time concerns**: Network latency (or the interval between user request and server response) needs to be considered carefully during Ajax development. Without clear feedback to the user, preloading of data and proper handling of the XMLHttpRequest object, users might experience delays in the interface of the web application, something which they might not expect or understand.
- **Search Engine Optimization**: Websites that use Ajax to load data which should be indexed by search engines must be careful to provide equivalent Sitemaps data at a public, linked URL that the search engine can read, as search engines do not generally execute the JavaScript code required for Ajax functionality.
- **JavaScript compliance**: not all browsers handle Javascript in the same way, and many users disable JavaScript in their browsers.

1.3.2 Flash & Flex

Adobe Flash is a set of multimedia technologies developed and distributed by Adobe Systems and earlier by Macromedia. Since its introduction in 1996, Flash technology has become a popular method for adding animation and interactivity to web pages; Flash is commonly used to create animation, advertisements, various web page components, to integrate video into web pages, and more recently, to develop Rich Internet applications (RIA).
Flash can manipulate vector and raster graphics and supports bi-directional streaming of audio and video. It contains a scripting language called *ActionScript*. It is available in most common web browsers and some mobile phones and other electronic devices (using Flash Lite). Several software products, systems, and devices are able to create or display Flash, including the Adobe Flash Player. The Adobe Flash Professional multimedia authoring program used to create content for the Adobe Engagement Platform, such as web applications, games and movies, and content for mobile phones and other embedded devices.
Files in the SWF format, traditionally called "Flash movies" or "Flash games", usually have a .swf file extension and may be an object of a web page, strictly "played" in a standalone Flash Player, or incorporated into a Projector, a self-executing Flash movie (with the .exe extension in Microsoft Windows). Flash Video (FLV) files have a .flv file extension and are used from within .swf files.
Flash is increasingly used as a way to display video clips on web pages, a feature available since Flash Player version 6.

- The key to this success has been the player's wide distribution in multiple browsers and operating systems, rather than any superior video quality or properties. It is available for many popular platforms, including Windows, Mac OS X and Linux. Flash is used as the basis for many popular video sites, including YouTube and Google Video.
- One major flaw with multimedia embedded through Flash, however, is the considerable performance penalty placed on playback hardware as compared with a purpose built multimedia playback system. Many files that drop frames

and skip audio when embedded within Flash play without any issues using other multimedia formats on the same hardware.

Flash Video (.flv files) is a container format, meaning that it is not a video format in itself, but can contain other formats. The video in Flash is encoded in H.263, and starting with Flash player 8, it may alternatively be encoded in VP6. The audio is in MP3. The use of VP6 is common in many companies, because of the large adoption rates of Flash Player 8 and Flash Player 9.

<u>**Adobe Flex** is a collection of technologies released by Adobe Systems for the development and deployment of cross platform, rich Internet applications based indeed on Adobe Flash platform.</u> Traditional application programmers found it challenging to adapt to the animation metaphor upon which the Flash Platform was originally designed. Flex seeks to minimize this problem by providing a workflow and programming model that is familiar to these developers. MXML, an XML-based markup language, offers a way to build and lay out graphic user interfaces. Interactivity is achieved through the use of ActionScript mentioned above. The Flex comes with a set of user interface components including buttons, list boxes, trees, data grids, several text controls, and various layout containers. Charts and graphs are available as an add-on. Other features like web services, drag and drop, modal dialogs, animation effects, application states, form validation, and other interactions round out the application framework. Unlike page-based HTML applications, Flex applications provide a stateful client where significant changes to the view don't require loading a new page. Similarly, <u>Flex and Flash Player provide many useful ways to send and load data to and from server-side components without requiring the client to reload the view</u>. Though this functionality offered advantages over HTML and JavaScript development in the past, the increased support for XMLHttpRequest in major browsers has made asynchronous data loading a common practice in HTML-based development too.

Currently, Flex has the largest market share of any other framework for rich Internet applications, with a penetration of around 90 percent (something that Microsoft Silverlight is challenging). When used properly, <u>Flex enables a website to behave like a thick client application</u> (one that exists solely on a user's computer rather than on the Internet).
<u>As with any client-side technology, there are drawbacks. Not all browsers start out with the Flash plug in installed, and Flash is also updated from time to time. In either case, the end user is required to download a new version if he or she reaches a page that requires it. Other frameworks for rich Internet applications have the same issue, which is seen by some as a drawback since not all users will (or are permitted to) download the plug in, and in many cases will navigate away from the page entirely.</u>

1.3.3 Silverlight

In response to the proliferation of other frameworks used to create rich Internet applications such as Flex from Adobe and AJAX-based frameworks, Microsoft Silverlight was recently introduced. Microsoft Silverlight is a browser plug-in that allows web applications to be developed with features like animation, vector graphics, and audio-video playback - features that characterize a rich internet application.
Silverlight competes with products such as Adobe Flash, Adobe Flex, Adobe Shockwave, Java FX, and Apple QuickTime. Version 2.0 brought improved interactivity and allows developers to use .NET languages and development tools when authoring Silverlight applications.
Silverlight was developed under the codename Windows Presentation Foundation/Everywhere (WPF/E). It is compatible with multiple web browser products used on Microsoft Windows and Mac OS X operating systems. A third-party free software implementation named Moonlight is under development to bring compatible functionality to GNU/Linux. Mobile devices, starting with Windows Mobile 6 and Symbian (Series 60) phones, will also be supported.

The main features of SilverLight are:

- **High Quality Video Experience**: silverlight enables very high quality videos, embedded in highly graphical websites. The same research and technology that was used for VC-1, the codec that powers BluRay and HD DVD, is used by Microsoft today with its streaming media technologies.
- **Cross-Platform, Cross-Browser:** web applications that work on most of the browser, and most of the operating systems.
- **Developers and Graphic Designers interaction:** Developers familiar with Visual Studio, Microsoft.net will be able to develop Silverlight applications very quickly, and they will work on Mac's and Windows. Developers will finally be able to strictly focus on the back end of the application core, while leaving the visuals to the Graphic Design team using the power of XAML.

1.3.4 JavaFX

JavaFX is a Sun's family of products for creating Rich Internet Applications with immersive media and content. The JavaFX products include a runtime and tools suite that web scripters, designers and developers can use to quickly build and deliver expressive rich interactive applications for desktop, mobile, TV and other platforms. JavaFX technology provides the presentation layer for the Java ecosystem that lays over the Java runtime environment.

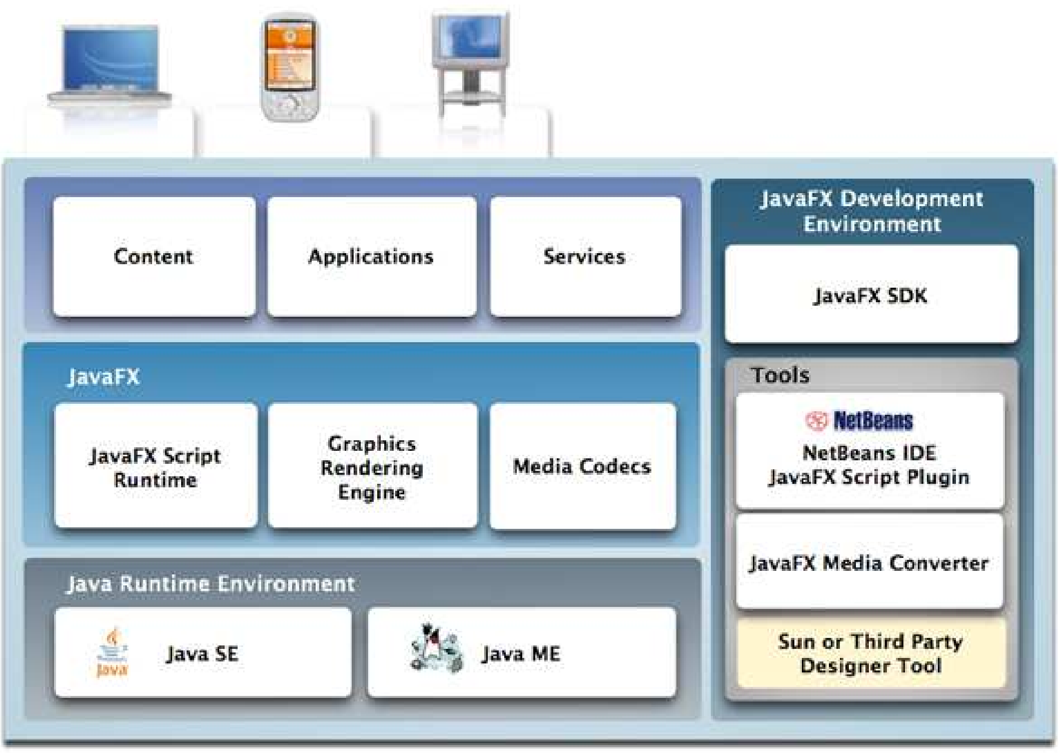


Figure 11: JavaFX Platform.

Sun Currently has an open source community project hosted for JavaFX, *OpenJFX*, where developers can sign up for a private preview of the JavaFX SDK, as well as download the JavaFX Script plugin for NetBeans 6.1. JavaFX is anticipated to compete on the desktop with Adobe AIR, OpenLaszlo, and Microsoft Silverlight. It may also target Blu-ray Disc's interactive BD-J platform, although as yet no plans for a Blu-ray release have been announced.
Currently JavaFX consists ***JavaFX Mobile*** and ***JavaFX Script***:

- JavaFX Mobile is a Java operating system for mobile devices initially developed by SavaJe Technologies and purchased by Sun Microsystems in April 2007. It is part of the JavaFX family of products. The JavaFX Mobile operating system provides a platform for PDAs, smartphones and feature phones. It features a Java SE and Java ME implementation running on top of a Linux kernel.

It is understood that Sun will distribute JavaFX Mobile as a binary operating system to device manufacturers who will brand the interface to differentiate their product.

- JavaFX Script, is a high-performance declarative scripting language for building and delivering the next generation of rich Internet applications for desktop, mobile, TV, and other platforms. It forms part of the JavaFX family of technologies on the Java Platform. JavaFX targets the Rich Internet Application domain (competing with Adobe Flex and Microsoft Silverlight), specializing in rapid development of visually rich applications for the desktop and mobile markets. JavaFX Script works with integrated development environments like NetBeans and Eclipse. The main features of this scripting language are:
  - JavaFX Script uses a declarative syntax for specifying GUI components, so a developer's code closely matches the actual layout of the GUI.
  - Through declarative databinding and incremental evaluation, JavaFX Script enables developers to easily create and configure individual components by automatically synchronizing application data and GUI components.
  - JavaFX Script will work with all major IDEs, including NetBeans, which is the reference implementation IDE for Java development.
  - Unlike many other Java scripting languages, JavaFX Script is statically typed and will have most of the same code structuring, reuse, and encapsulation features that make it possible to create and maintain very large programs in Java.
  - JavaFX Script is capable of supporting GUIs of any size or complexity.
  - JavaFX Script makes it easier to use Swing, one of the best GUI development toolkits of its kind.

1.3.5 Web 2.0 technologies comparison

The goal of all these frameworks is to be able to build Rich Internet Applications more easily and to make the user experience as rich as possible.

- Currently a lot of this kind of functionality is being built with AJAX (asynchronous Javascript, CSS, DOM manipulation and XML) and Flash/Flex. Manual construction of applications with the four AJAX components is though, and thus companies are trying to create the silver bullet for easy creation of RIA applications.
- Creating applications with Flash/Flex is relatively easy and makes the applications really rich, but it is vendor-specific (Adobe).
- Some of the newly announced frameworks are more open source (JavaFX) than others (Silverlight).

  To make the comparison easier, a table that lists different aspects of the frameworks is presented:

	Silverlight	**JavaFX**	**FLASH/FLEX**
Version	1.1 Alpha	1.0	3.0
Built-in UI Controls	Very limited to none	Via Swing	Yes
IDE	Visual Studio 2008 .NET Platform 3.5 Silverlight 1.1 Alpha	NetBean 6.01 JavaFX plugin	Flex Builder 3.0 (Eclipse platform)
IDE Visual Design	No	No	Yes
IDE Toolbar for Controls	No	No	Yes
Browser Client	Silverlight 1.1 Alpha	Java Plugin with JavaFX extension	Adobe Flash Player 9
Runtime	2-4MB	2-4MB	1.1MB.
Languages	XAMLJavaScript(C#, VB.Net, ASP.Net)	JavaFXScript,Java	MXMLActionScript
Platform	Windows+Mac	Windows+Linux+Mac	Windows+Linux+Mac

1.4 Application Streaming

1.4.1 Application Deployment

All Application virtualization software vendors have their own definition of Application virtualization. Basically it comes down to this: Application virtualization enables the deployment of software without modifying the local operating system or file system. It allows software to be delivered and updated in an isolated environment ensuring the integrity of the operating system and all applications. Application conflicts – and the need for regression testing - are significantly reduced. A single application can be bundled and deployed to multiple operating system versions. Applications are easier to provision, deploy, upgrade, and rollback.

In our opinion there are 3 approaches to application virtualization:

- **Standalone:** Applications are encapsulated in a single executable. These executables can run instantly from USB, CDROM or local disk. The applications can also be deployed using a management tool like Microsoft SMS.

- **Centrally controlled access:** Virtualized applications are "distributed" through a central deployment tool. A locally installed agent is required. The applications can be deployed (executables are copied locally) or shortcuts to the applications (located on a network-share) can be presented. When using shortcuts, streaming is usedto cache files locally.

- **Streaming:** Applications are encapsulated in a single file and are located on the network. When starting the application only the blocks needed to run the application are copied to a local drive (cache). When the more features of the application are used, more blocks are copied to the local cache. Streaming can be available for standalone virtualized applications accessed from the network or when the virtualized applications are presented with a locally installed agent.

  For what we are concerned we will present in detail the most important features regarding the two main software of the last branch.

Several years ago, surveys of data center activity attracted attention, because they indicated that few servers averaged more than 20 percent utilization of their CPU, memory and disk hardware – and some were far worse. The IT industry quickly responded by delivering virtualization technologies and many organizations with large data centers are now deploying these technologies successfully to improve hardware utilization. A similar trend is now emerging which focuses on PC software. Many organizations are beginning to recognize that the efficiency of their PC software utilization is also in the range of 20% and in some cases much worse.

This inefficient utilization of software assets has become very costly for companies. Software license fees are generally based on the number of copies of a specific application installed on each PC in the company regardless of how often each copy of the application is used. In order to insure that users have even rarely used

applications available when needed, companies tend to keep an overabundance of software on each PC and, therefore, pay far more in license fees than if the fees were tied to actual software usage. A typical license agreement also allows for software to be uninstalled and then re-installed on another PC as many times as desired, as long as it only resides on one PC at a time. While it would be highly impractical for a corporation to manually move applications from one PC to another on a regular basis to accommodate changing needs, some companies have implemented innovative **streaming technologies** which allow for the automatic provision of software assets based on user demand.
For example, AppStream Inc.'s dynamic license management software streams an application to the user's PC when they need to use it, effectively eliminating the need to pay application license fees for unused software. This technology has benefits relative to software license compliance and software change management as well.
Computer application streaming is indeed a form of on-demand software distrbution.
The basic concept of application streaming has its foundation in the way modern computer programming languages and operating systems produce and run application code. Indeed, only specific parts of a computer program need to be available at any instance for the end user to perform a particular function. This means that a program need not be fully installed on a client computer, but parts of it can be delivered over a low bandwidth network as and when they are required. Application streaming is usually combined with application virtualization, so that applications are not installed in the traditional sense.

We can therefore summarize the most important features and benefits:

- **Applications are streamed on demand from a central server**; when the user has finished with the applications all components are completely removed - as if the application was never there.
- **The cost of installation is sharply reduced**. Each staff member's requirements can be noted and the system set up to deliver what's needed when the staff member logs into the network.
- **The cost of upgrading software is also reduced**. When the organization decided to move to a new version of a software product, it's a simple matter to tell the application streaming software to deliver the new version when the user logs in.
- **The cost of software licenses and license administration can be reduced**. The organization only needs to acquire enough licenses to handle what's being done now, not enough for everyone to have a license. This means that organizations having worldwide operations should be able purchase much fewer licenses. Systems not in use don't need to be a "license prison."
- **Reduced disk space requirements:** No large downloads required.
- **Mobile users are able to access or updated applications from any location.**

Before going forward is necessary to underline that main difference among different Application Streaming software:

- ***The possibility to work with Application Streamed with or without a server connection.***
- ***The modification of existing environment or OS:*** *A growing number of vendors offer desktop streaming software that provisions the entire desktop*

environment from a server to a desktop PC (or thin client). Altiris, AppStream, and Microsoft (through its recent acquisition of Softricity) have pushed the concept to the next level, streaming applications rather then a complete desktop environment. This allows greater flexibility in what is provisioned, because IT can create a basic operating system image and then individual images for each application, and combine them as needed on the fly. You don't need a separate desktop image for each combination of applications.

On the figure below an example of Application Streaming utility is represented.

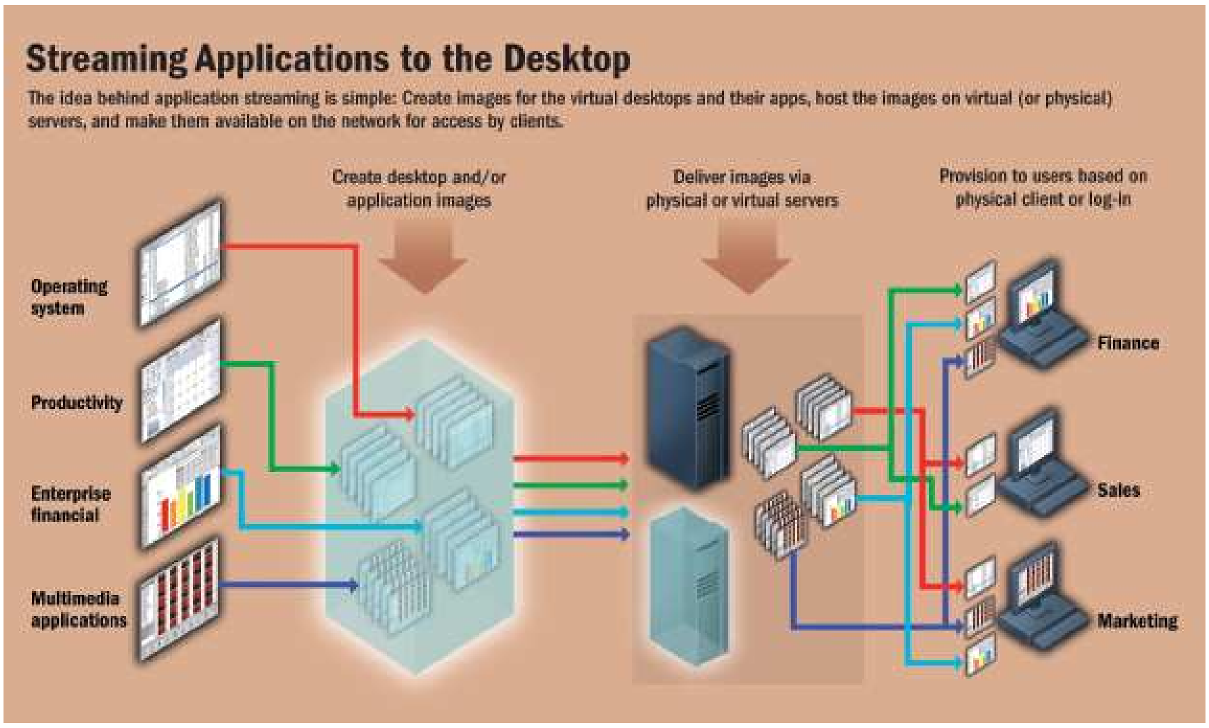

Streaming Applications to the Desktop
The idea behind application streaming is simple: Create images for the virtual desktops and their apps, host the images on virtual (or physical) servers, and make them available on the network for access by clients.
Create desktop and/or application images
Deliver images via physical or virtual servers
Provision to users based on physical client or log-in
Operating system
Productivity
Enterprise financial
Multimedia applications
Finance
Sales
Marketing

Figure 12: Application Streaming Example

In the rest of this section we will present the most important available technologies regarding Application Streaming proposing a final comparison between them.

1.4.2 SoftGrid

Microsoft Application Virtualization (formerly *Microsoft SoftGrid*) is an application virtualization and application streaming solution from Microsoft.
This platform allows applications to be deployed in real-time to any client from a virtual application server. It removes the need for local installation of the applications. Instead, only the SoftGrid runtime needs to be installed on the client machines. All application data is permanently stored on the virtual application server. Whichever software is needed is streamed from the application server on demand and run locally. The SoftGrid stack sandboxes the execution environment so that the application does not make changes to the client itself. Softgrid applications are also sandboxed from each other, so that different versions of the same application can be run under Softgrid concurrently. This approach enables any application to be streamed without making any changes to its code.
SoftGrid thus allows centralized installation and management of deployed applications. It supports policy based access control; administrators can define and restrict access to the applications by certain users by defining policies governing the usage. SoftGrid also allows replication of the applications across multiple application servers for better scalability and fault tolerance, and also features a tracking interface to track the usage of the virtualized application.
<u>The SoftGrid client runtime presents the user with a list of applications, to which the user has access</u>. The user can then launch a virtualized streamed instance of the application. Depending on the configuration, the systems administrator can be either notified of the action via email or it can require an explicit confirmation from the administrator for the application to start streaming and initialize or it can just simply check the active directory for the user's rights and stream the application to the user if it is authorized to run the application. The SoftGrid client can also install local shortcuts that bootstrap the process of launching individual virtualized software instances.
In detail SoftGrid is capable of packaging applications for on-demand, streamed delivery into virtualized end point runtime environments. The figure below depicts the first step in this process.

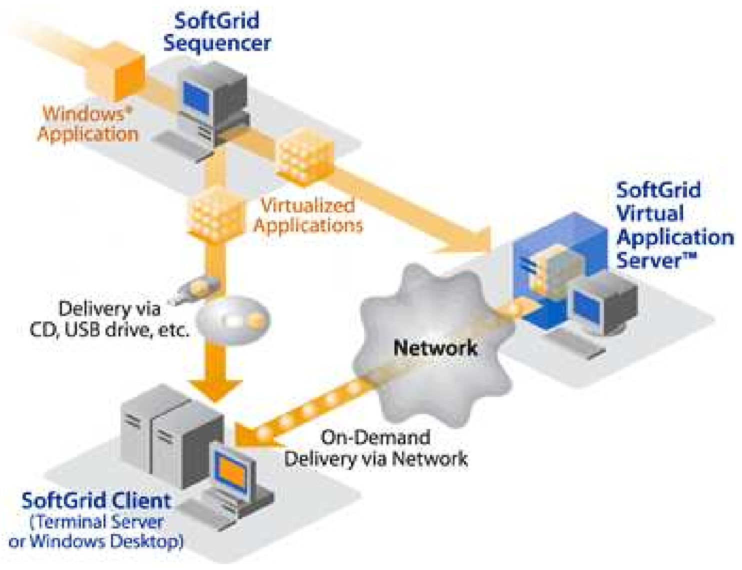

SoftGrid Sequencer
Windows® Application
Virtualized Applications
SoftGrid Virtual Application Server™
Delivery via CD, USB drive, etc.
Network
On-Demand Delivery via Network
SoftGrid Client (Terminal Server or Windows Desktop)

Figure 13: Softgrid Application Packaging

A workstation is configured with the SoftGrid Sequencer application. As the SoftGrid administrator installs the target application on the workstation, the sequencer monitors all installation steps, including changes to the registry. The administrator can also select specific components to be included in the virtualized application package, such as DLLs as well as Java and .NET components. Further, the application can be configured to store information in a centralized location (e.g. a secure data center).
The final outcome of the sequencer process is a set of four files that comprise the virtualized application, with an initial application load just big enough to load and initially execute the application. According to Microsoft, the load size is approximately 20 to 40 percent of the total application size.
The four files are placed on a SoftGrid application server for distribution. The administrator grants access to the application by adding users with approved access to a related AD group. Only members of the group will be able to see the application icon on their desktops or access the application files on the server. To reverse the process—to revoke a user's access—simply remove him or her from the group.
Once a user is added to the proper group, the application icon will appear on her desktop at next login. If the user is already logged in, she can force a refresh of her desktop by using a SoftGrid utility typically found in the system tray. The application is accessed by double-clicking the icon. The figure depicts what occurs when the user runs the application for the first time.

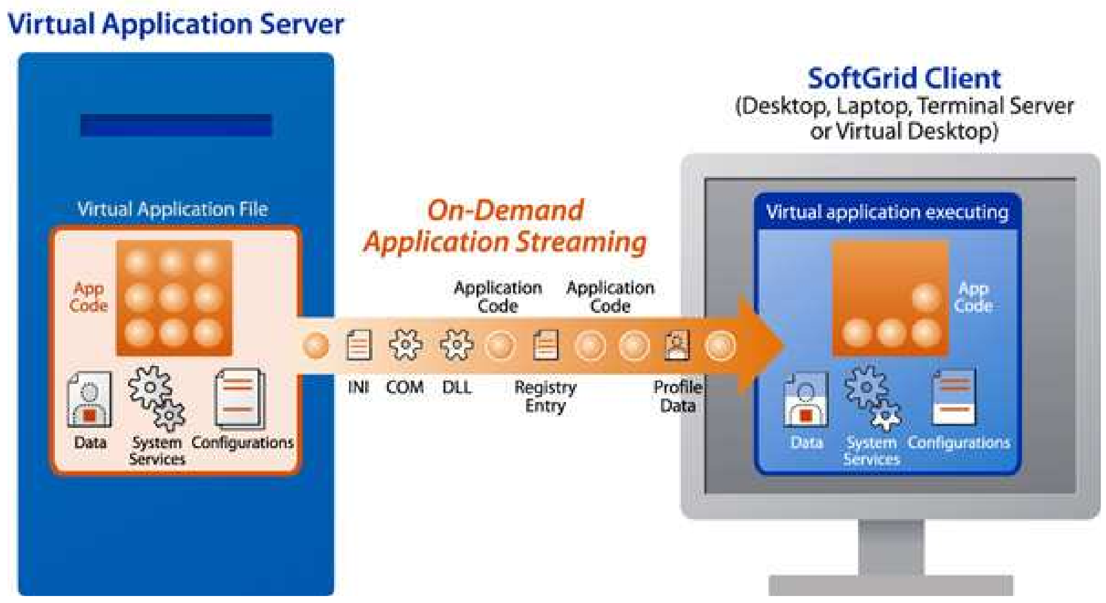


Figure 14: Softgrid Application Streaming

The four files created and installed on the SoftGrid Application Server are accessed by the desktop. The result is the creation of a virtual application environment on the user's machine with the bare minimum of application components streamed into it. The result is a self-contained application runtime space that virtualizes the following components :

- Registry – registry changes unique to the application are not made to the main OS on the desktop. Rather, they are virtualized within the isolated application runtime space.
- File system – calls from the application for local disk access can be redirected to access DLLs and other components from a virtual file system.

- COM/IPC
- INI files
- Process environment
- Fonts

We can therefore summarize the most important features of this solution:

- Applying patches in a virtualized environment is a simple rebuild of the appropriate SoftGrid package. The next time a user runs the application, the updated version is automatically streamed to the desktop.
- Help Desk costs associated with failed application installations, overwritten application components, corrupted registries, etc. are all but eliminated when files and settings unique to an application are virtualized (Customers have cut help-desk costs by up to 30% by reducing call volume for application-related problems, and reduced end-user downtime by up to 80% by easing challenges with business continuity of applications).
- Use of applications accessed via the SoftGrid server is tracked. Further, administrators can link an active instance of a running application to a license. This metering of applications helps organizations remain compliant with licensing agreements.
- If a virtualized application environment is infected with malware, the threat is contained—prevented from spreading to other applications or the base operating system.
- The threat of data leaks is mitigated due to the virtualization of local cache associated with application processing. Further, configuration of applications such as the Microsoft Office Suite can 'encourage' users to save documents in a secure, centralized environment.
- Application access is controlled by group membership. In addition, applications that run on laptops can be configured to stop running if the user doesn't authenticate to the enterprise network within a specified period. This prevents thieves from using laptop applications indefinitely.
- Mobile users are able to access patched or updated applications from any location.
- Minimize application conflicts and regression testing: By eliminating the requirement to install applications on desktops or laptops, and shielding the OS and applications from changes normally created when applications are installed and run, Microsoft SoftGrid prevents problems that hinder deployments. This also minimizes the need to perform regression testing and, as a result, speeds deployments.

1.4.3 AppStream

AppStream's on-demand application distribution and license management platform provides an optimal solution, developed by Symantec, to serve the application management needs of enterprise environments, providing high productivity with controlled, guaranteed access to necessary applications from any Windows applications from any location at any time, including remote and mobile users.
AppStream has a desktop management platform that is loaded on a network server and streams applications to the desktop. It is built to deliver PC applications on demand so that when the PC user initiates an application, the application is streamed directly to the PC and begins to run. AppStream can be ready to run as soon as it is installed and configured as it requires no special program coding or changes to any of the applications running on PCs. It works by loading a light-weight agent onto each client PC which then manages the loading of applications from the server to the PC. The AppStream software does not actually install the full application on the PC; it simply streams enough of the application to the PC for it to be able to run. Typically, PC users use only a small part of any PC application – often as little as 10 percent. AppStream takes advantage of this fact by dividing the PC application up into segments, initially only streaming the segments that are needed to launch the application and those features that the user normally uses. The benefits of this include:

- Quick access to new applications, particularly over a WAN.
- Low requirements for network bandwidth because the application is run locally.
- Reduced IT involvement because of user "self-service" environment.

AppStream keeps track of how users work with their applications and adjusts the usage profile dynamically if usage patterns change. Technically, AppStream communicates with PC's via a protocol that is HTTP compliant and can work through firewalls, proxy servers, and over VPNs. It is engineered to make efficient use of the network, optimizing network bandwidth and using local PC caches to smooth the network load.
The streaming process works in the same way both for local PC applications, such as Microsoft Office, and client/server applications that are distributed between the PC and network servers. It can also provision PC applications to laptops, but in that case it fully loads the application so that the laptop can work while disconnected.
The Platform runs on a Windows Server which is typically configured for high availability and fail-over, so that the service it provides runs 24 by 7. Hundreds of desktops can be managed from a suitably configured server. For very large desktop populations more servers can be added in a multi-tier deployment, with load balancing and load sharing capabilities configured between them. There is no obvious limit to how many desktops can be managed in this way. In the figure below it is possible to see an image representing AppStream architecture.

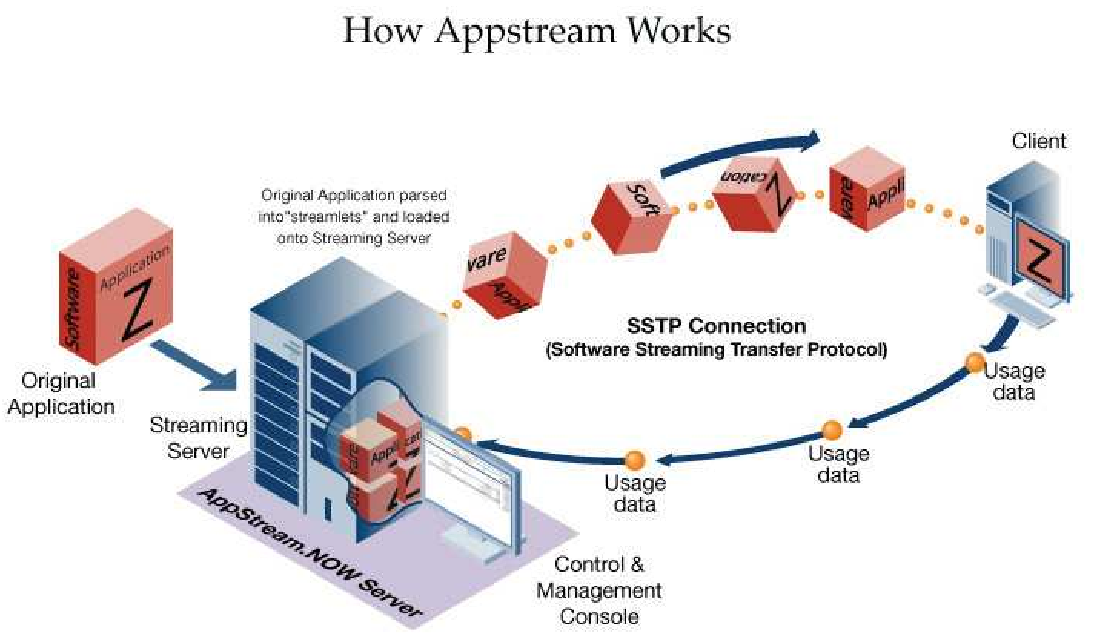


Figure 15: AppStream Application Streaming

Unlike more traditional models, the ability to distribute applications, monitor usage, and manage licenses is integrated in a single platform. IT retains control over application versioning and provisioning but gains flexibility to give off-line use of applications and the ability to use local graphics cards and other local resources.
The AppStream server holds its own database of usage information both to enable dynamic streaming and to accumulate usage statistics. Its management console allows the platform to be configured and managed, and for reports to be requested on desktop activity and license management. Some of the benefits of this information include:

- Management reporting of licenses installed.
- Software utilization information by department, user, or application.
- Contract renewal data to ensure that companies maintain software compliance.

The platform permits a fine level of definition of user/application access. User access to any given PC application can be time-based so that it is only available between specific days or "time-bombed" so that access rights cease on a given day (to cover the situation when staff leave). If desired, unlimited access to an application can be given to a user. Application access can be limited to specific groups of users or even to single individual and different versions of the same application can be assigned to different users if need be. Administrators can de-provide or re-provide applications at any time.
To ensure that license agreements are never violated the maximum number of installations of a particular application can be specified and this value will never be exceeded no matter who requests access. To ensure that the high water mark is efficiently implemented, AppStream cleans up any idle application packages that remain in the PC cache after a predefined period of time.

AppStream automatically reports on the efficiency of software utilization, identifying the usage frequency of all desktop applications and highlighting those applications that are under-utilized. It also provides a graphical global view of usage levels on an enterprise wide basis. A whole series of reports can be generated that provide details of users, user access rights, software package usage and license levels. Reports are customizable and the AppStream repository of information can be accessed directly via any reporting tool that is compliant with ODBC.
From a user perspective, AppStream is almost invisible, both in terms of how the Windows interface looks and how it performs. The AppStream agent is designed to load applications as fast as possible holding software components in local cache and only using the program components that are needed. The whole configuration is self-optimizing and it ensures that the minimum amount of network bandwidth is consumed so that communications bottlenecks do not arise.
As a consequence, PC applications behave as if they were fully installed on the PC, with very little difference in wait times when an application loads. An application may take slightly longer the very first time it loads, but from then on it is likely to be held in cache on the PC and it will load as fast as if the application were installed locally. In all other respects the users will probably be unaware of how applications load. Indeed it is possible to mix and match applications with some being local and some downloaded and the user is unlikely to guess which are which.
Furthermore AppStream is easy to administer. The web-based control and management console and web-based infrastructure lets IT staff perform management tasks from anywhere within the corporate intranet or extranet. The robust provisioning process ensures that on-demand access can be offered to employees without concern that end users will be using applications they shouldn't be. And compatibility with Microsoft's Active Directory and the Lightweight Directory Access Protocol (LDAP) standard ensures that AppStream integrates easily with any existing enterprise entitlement system.

The following points summarize all the advantages supplied by AppStream.NOW software

- Reduced hard disk storage space for streamed applications
- Flexible application distribution - via USB, CD, or network download
- Full support for Windows Vista
- WAN Optimizer option
- Enhanced LDAP/Active Directory support
- SDKs available for both client and server customization
- Altiris SVS support
- MSI Streaming
- Improved performance in package upload window, export utility, provisional behavior
- Firefox support (IE plug-in)
- Pre-popluation of icons
- Robust failover capabilities for high availability configurations
- Ability to create local groups for flexible user management and provisioning
- Improved, more-extensive reporting, including tracking and reporting on offline usage
- Ability to pre-stream applications without end-user intervention
- E-mail upon license-threshold-reached event

We finally propose a table with that summarize the most important capabilities and relative benefits of this software.

Capability	**Benefit**
Provides applications on-demand	Increases end user productivity through better, quicker access to needed applications
Software distribution and software deployment without IT intervention	Lowers administrative support costs, IT not required to push down all applications
Enables assigning of different package versions to different end users	Increases the flexibility of enterprise computing environment — end users can pull down the version they need
Reduces the steps required to deploy an application	Speeds software distribution of applications to end users
Provides sophisticated access provisioning and limiting	Ensures that end users are able to access the applications they need, while preventing access to unauthorized end users
Enables interoperability between streamed and traditionally installed applications	Lowers administrative requirements and allows for phased adoption
Provides a single point of access for end users, software packages, licensing and servers	Simple, intuitive administration

1.4.4 Application streaming technologies comparison

In order to conclude the presentation of these software packages we summarize the most important features with the following table.

	(Microsoft) Softgrid	**(Symantec) AppStream**
Stand-alone Virtualized applications can run on clients without agent locally installed.	**No**	**Yes**
Streaming Launch the application instantly from a remote location. The first blocks needed to start the application are locally cached on the client. When more features are used, more blocks are cached.	**Yes**	**Yes**
Centrally controlled access Management software is included that can manage authorization on application delivery. Agent locally installed on the client is required.	**Yes**	**Yes**
Off-Line Usage Applications can be launched even when a user is off-line (for example on a laptop). The streamed application is completely cached locally.	**Yes**	**Yes**
Application Interconnectivity / Binding Virtualized applications, which are isolated, can be connected to each other. For example, .NET 2.0 framework is packaged once. Applications that needs .NET framework connect to the virtualized .NET package.	**No**	**No**
Executes in user-mode only There is no interaction with the kernel of the OS. Therefore, applications cannot crash the OS.	**Yes**	**Yes**
License Management Can the usage of the applications be controlled? How many licenses do you have of an application and how many times is the application (concurrently) in use?	**Yes**	**Yes**
Tracking and reporting The usage of applications can be tracked and monitored. Reports can be created.	**Yes**	**Yes**
Support		
16-bit application supported (only run on 32-bit OS)	**Yes**	**No**
64-bit application supported	**No**	**No**
Windows 2000	**Yes**	**Yes**
Windows XP	**Yes**	**Yes**
Windows Vista 64-bit	**No**	**No**
Windows Server 2008 (TS) 64-bit	**No**	**No**
Citrix	**Yes**	**Yes**

1.5 Technologies Comparison

In this chapter we analyzed different methods in order to delivery applications on demand and fast deployment of services.
The basic idea that we have to account for is represented in the figure below.

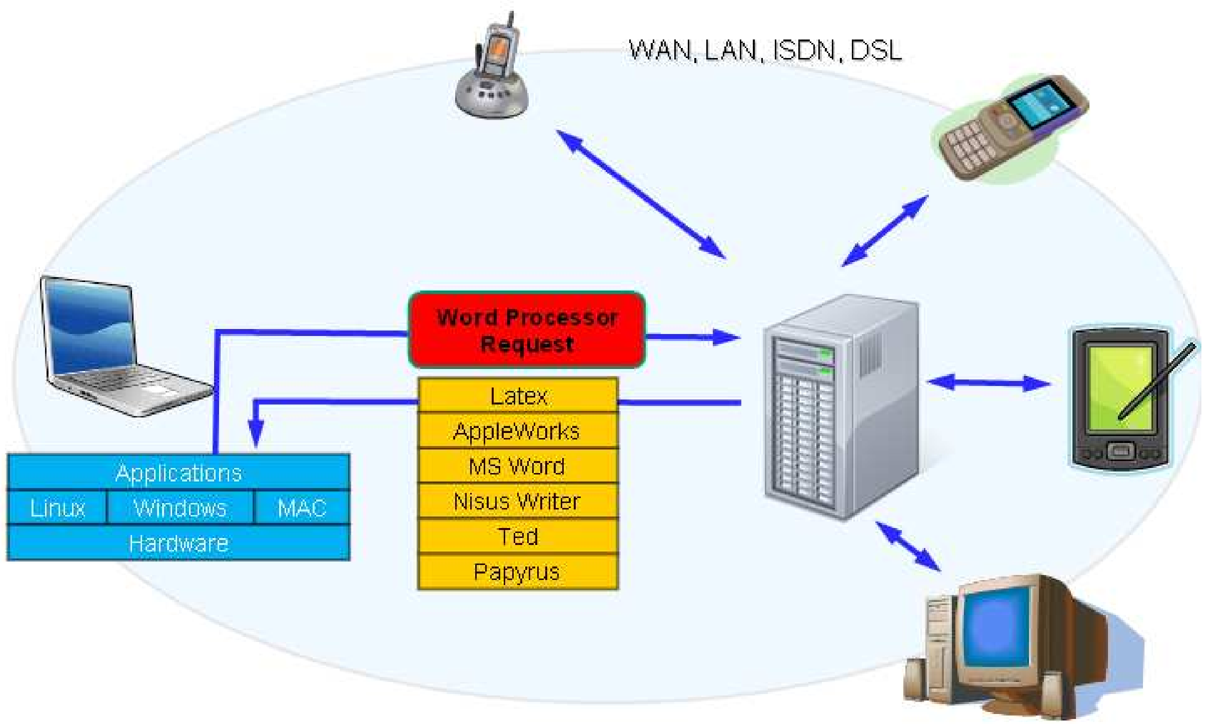


Figure 16: Devices Interaction in application streaming.

In this figure thee crucial points are shown:

- **Network:** the technologies have to support standards and protocols for different types of networks (WAN, LAN, PAN, etc;) (i.e. as we saw in D1_1 public operators involved in emergency situations can be far way or in the same building).
- **Devices:** the technologies have to allow full device interoperability; (i.e. public operators must be able to use different devices without any interaction issue).
- **Application on demand**: the technologies must delivery application on demand and services as fast as possible.

We already analyzed the features for every tool and the aim of this section is to summarize the crucial point sfor the selected technologies proposing a table as a comparison; we avoid including in this section the Virtualization systems not being strictly connected to the mobile and portable devices.

VNC

VNC is a nice graphical desktop sharing system which uses the RFB protocol to remotely control another computer. It transmits the keyboard and mouse events from one computer to another, relaying the graphical screen updates back in the other direction, over a network.
VNC is platform-independent; a VNC viewer on any operating system usually connects to a VNC server on any other operating system. There are clients and servers for almost all GUI operating systems and for Java. Multiple clients may connect to a VNC server at the same time.
It's improved a bit over the years, but still has several flaws.

- VNC server is required for an OS. Many operating systems have them, but some don't.
- RFB doesn't work well over high latency connections.
- All RFB clients and servers are only moderately adaptive to bandwidth.

ICA

ICA is a proprietary protocol for an application server system, designed by Citrix Systems. The protocol lays down a specification for passing data between server and clients, but is not bound to any one platform.
Besides Windows, ICA is also supported on a number of Unix server platforms and can be used to deliver access to applications running on these platforms. The client platforms need not run Windows; for example, there are clients for Mac, Unix, Linux, and various Smartphones. ICA client software is also built into various thin client platforms.
The ICA protocol is actually optimized for low bandwidth.

RDP

RDP is a multi-channel protocol that allows a user to connect to a computer running Microsoft Terminal Services. Clients exist for most versions of Windows (including handheld versions), and other operating systems such as Linux, FreeBSD, Solaris, Mac OS X, and PalmOS. The server listens by default on TCP port 3389. Microsoft refers to the official RDP client software as either Remote Desktop Connection (RDC) or Terminal Services Client (TSC).

AJAX

AJAX refers to the use of a group of technologies for building Rich Internet Applications. It's based on 100% open standards the most compact and high performance result can come out but it has some big drawbacks:

- It requires fairly good working knowledge of asynchronous Javascript, CSS, DOM manipulation and XML and the construction of applications with the these components is quite a pain.
- Not all browsers handle Javascript in the same way, and many users disable JavaScript in their browsers. Web developers are indeed looking for at least some assistance to elide browser differences.

Flash/Flex

Flash is a software for creating rich, interactive content for digital, web, and mobile platforms; it provides the tools you need to deliver an engaging user experience. Flex is an application development solution for creating and delivering cross-platform rich Internet applications (RIAs) within the enterprise and across the web. It has a client-runtime which is based upon the Flash Player. Flex and Flash have complementary strengths. Flash is the leading authoring tool for web developers, multimedia professionals, animators, and videographers who want to create rich interactive content. Flex 2 products enable more application developers to leverage the powerful Flash runtime to create data-driven RIAs. In addition, developers can use Flash and Flex Builder together to add rich interactive elements to a structured, Flex based application. Flash/Flex is relatively easy and makes the applications really rich, but it is vendor-specific (Adobe). Not all browsers start out with the Flash plug in installed, and Flash is also updated from time to time. In either case, the end user is required to download a new version if he or she reaches a page that requires it (in many cases users will navigate away from the page entirely).

Silverlight

Silverlight is a cross-browser, cross-platform plug-in for delivering the next generation of Microsoft .NET–based media experiences and rich interactive applications for the Web. Silverlight is meant to be running in the browser, it should be seen much more as a competitor to Flash/Flex than to the JavaFX, which are runtimes running *outside* the browser. Silverlight has the advantage it can easily be put on every Windows computer by letting it piggy-back with some automatic Windows update.
While Silverlight 1.0 only had basic capability in terms of simple animation and media support using Javascript as the primary scriptiong language, Silverlight 1.1/2.0 is a completely different animal. Silverlight 2.0 offers a complete .NET common language run-time in the browser including managed versions of Javascript and Python that will compile to binary on client and run extremely quickly. Obviously, by supporting Python (and Ruby as well, though not in the current alpha distribution) in the client, Silverlight's CLR in the browser now also support the Dynamic Language Runtime (DLR), making Silverlight have the richest support for RIA client-side languages currently available. Of course, sporting a

lightweight version of .NET and its libraries comes at some cost, particularly download and installation times.

JavaFX

Sun has created a very interesting new entry in the RIA space. Designed to leverage the full breadth and depth of the extremely mature and robust Java platform, JavaFX is a scripting language that doubles as a declarative programming model. With the express goal of making it significantly easier to create Rich Internet Applications than it is now with current Java technologies, JavaFX offers some serious productivity-oriented features including: A highly efficient Model-View-Controller (MVC) data binding construct in the scripting language itself, declarative event triggers for assertions and CRUD, and even some cutting edge features such as extents (a notation to let you see all class instances of a certain type) and other mechanisms that will give one some concerns about the sacrifice of long-term code maintenance to the altar of code efficiency, but it's a pretty well thought-out model. JavaFXScript takes advantage of the Java Runtime Environment's (JRE) ubiquity across devices and enables creative professionals to begin building applications based on their current knowledge base, but has also some drawbacks:

- It requires installation of at least one additional plugin.
- It’s a new technology not fully tested and ready to be compared with Flash for example

Softgrid

Softgrid is an application virtualization and application streaming solution from Microsoft. This platform allows applications to be deployed in real-time to any client from a virtual application server removing the need for local installation of the applications. All application data is permanently stored on the virtual application server. Whichever software is needed is streamed from the application server on demand and run locally. Unfortunately the interoperability between different applications environment is not implemented and the SoftGrid runtime needs to be installed on the client machines. It still suffers from usability quirks and an overly complex sequencing process, and it lacks support for headless services.

AppStream

AppStream has a desktop management platform that is loaded on a network server and streams applications to the desktop. It is built to deliver PC applications on demand so that when the PC user initiates an application, the application is streamed directly to the PC and begins to run. AppStream can be ready to run as soon as it is installed and configured as it requires no special program coding or changes to any of the applications running on PCs. Symantec gains acquiring AppStream a much needed streaming capability to support its already robust virtualization layer. The combined solution allows applications to be launched from a Web browser, and headless services are supported. However, the level of integration between the OEM components is imperfect and simple deployment tasks require too many steps, not to mention the slow initial response time for virtualized applications.

This table summarizes the most important features for the products presented:

	VNC	ICA	RDP	AJAX	Flash/ Flex	Silverlig	JavaFX	SoftGrid	AppStream
Bandwidth Usage	High	Medium	Medium	Low	Medium	Medium	Medium	Low	Low
Browser Compatib.	X	X	X	Medium	High	High	High	X	X
Platform Compatib.	High	High	Medium	Medium	High	Medium	High	Medium	Medium
Functions supported	Medium	Medium	High	High	High	Medium	Medium	High	High
Performance /Quality	Low	Medium	Medium	High	Medium	High	Medium	High	High
Proprietary	Free	Citrix	Micr.	Free	Adobe	Micr.	Sun	Micr.	Symantec

2. IP Multimedia Subsystem

2.1 Introduction

Over the last years, Next Generation Network (NGN) development has given to users the possibility to enter in to a large number of new applications based on multimedia contents transfer and sharing with high degree of versatility. The official definition of NGN, given by ITU-TI, is as follows: A packet-based network able to provide telecommunication services and make use of multiple broadband QoS-enabled transport technologies, in which service-related functions are independent from underlying transport-related technologies. To make this possible, NGN networks use the Internet Protocol Multimedia Subsystem, an architectural framework for delivering IP multimedia to mobile users.

In these few pages we're going to try to give a full vision of IMS: what is in detail, its history, why we need it, what are its basic principles, its architecture and how it can interfaces with the already existing communication technologies to achieve a real convergence of services and networks.

What is IMS?

IMS is a set of specifications hat describes the NGN network architecture for implementing IP based telephony and multimedia services. IMS defines a complete architecture and framework that enables the convergence of voice, data and mobile network technology over IP based infrastructure. It fills the gap between the two most successful communication paradigms, cellular and Internet technology. The vision for IMS is to provide cellular access to all the services that the internet provides.

History of IMS: a convergence of standards

As a result of the choice of IMS like convergence architecture, different standards bodies are involved in defining converged architecture in both fixed and wireless networks.

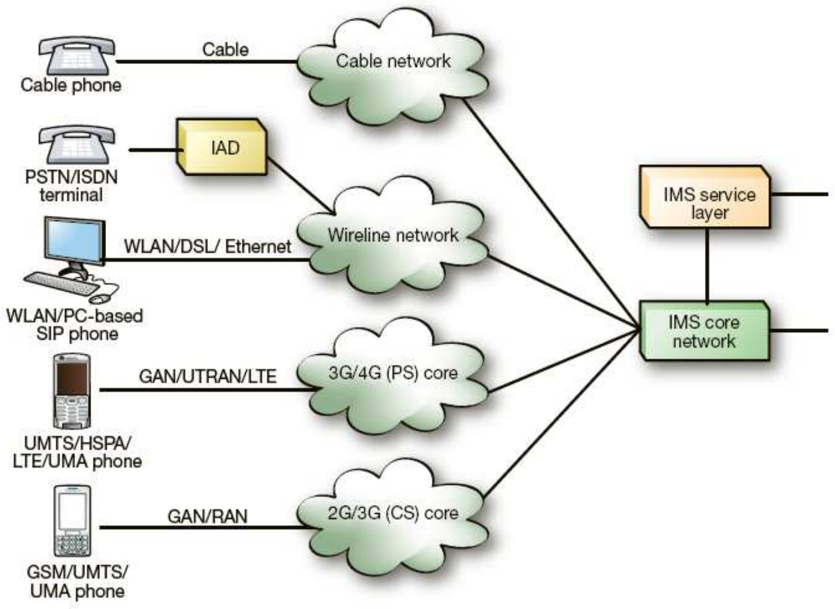
IMS was originally defined by 3G.IP (industrial form formed in 1999) that developed the initial architecture. This one was brought to the 3GPP (3rd Generation Partnership Project) as part of their standardization work for supporting GSM networks and radio technology evolution. IMS first appeared in 3GPP release 5, where SIP (Session Initiation Protocol, defined by IETF) was chosen as the main protocol for IMS. In releases 6 and 7 of 3GPP it was further enhanced to include additional features, like interworking with WLAN and support for fixed networks, by working together with TISPAN.

Benefits that make we need IMS

We can think to IMS as a way to offer Internet services everywhere using cellular technology. Today's cellular technology provides Internet services, but IMS can offer some important added values:

- IMS provides multimedia services with *Quality of Service* (QoS) enablement. In today's 3G technology, although the Internet access is much faster than before, there are no guarantees about QoS to the user. It's followed the principle of "best effort": the network will to its best to ensure a good level of service, but if during a multimedia transfer the bandwidth would have a down; it can't do anything to maintain the level of the service. Instead, IMS specifies enablement in QoS within the IP network and takes advantage of the QoS mechanism to improve and guarantee the transmission quality. To offer this service, IMS network service providers will have the capability to measure, adjust and deliver a certain level of transmission rate, gateway delay and error rates.
- IMS allows operators to make an *appropriate multimedia session charging*. In 3G systems, the users pay the number of bytes that is transferred. A videoconference needs a large amount of bytes transferred, so it is very expensive for the user. If the operator could provide a different charging in function of the type of services employed by the user, the user will take benefits. This is possible in IMS technology that provides information to the operator about the type of service required at the moment from the user, so the operator can apply the appropriate charging.
- IMS provides a *common platform for any type of service*. It's a cornerstone for efficient converged service offerings, making possible to introduce new converged services faster and easier. This integration feature of IMS will reduce time-to-market for rolling out new multimedia services. Service providers are very interested in this feature, because one of the biggest challenges in today's communication network is to improve the long and costly process to create a new service. IMS supports the development of new services and a rich business landscape: building on carefully designed interfaces, it allows operators to launch new services or to expand existing ones. Furthermore, to facilitate the development of new and creative services, the standardized mass-market[1] capabilities can be exposed through published interfaces to down able clients and network servers: the standardized interface and common features provided by IMS infrastructure enable service providers to easily adopt a service created by third parties and create a service that integrates with many services effectively. Therefore, the new services have the potential to rapidly come to market with a global reach and without affecting infrastructure. In IMS, multimedia session control and management is realized using SIP (Session Initiation Protocol) as the standard protocol. This protocol allows add/drop of media dynamically during a session, depending on the type of application in use. IMS enables to implement a common service control, execution and interaction platform for all services and subscribers accessing their network.

[1] Mass market: category of standardized services which are supported by a wide range of terminals and which interoperate within the global operator community. These services are characterized by scalability, availability and performance; functional growth is defined by standardization. This category of services is the alternative one from the "non standardized services", that individual operators can give to their customers with rapid time to market end flexibility.



•

Figure 17: Representation of IMS network with multiple access technology

- IMS allows the user to employ every type of service *irrespectively from his location*. IMS uses Internet technology in order to make possible for the users to roam in different countries and have everywhere the opportunity to gain access to any type of service. To make this possible, IMS provides subscriber registration and location information elements to enable mobility, in addiction to session set-up and management, and message forwarding to IMS and non-IMS networks.
- IMS architecture has the peculiarity to be *access-independent*. This is central in the move towards convergence. Each access type is being enabled to work with IMS core, be it DSL, WLAN, GPRS, WiMAX, and so on. This opportunity is very important because the majority of devices currently in the market are not IMS-capable. In due course, all terminals and access networks will be IMS-capable. But until that time, there will be a mix of IMS-capable and non-IMS-capable terminals. This skill of IMS, really valuable in convergence-oriented communication systems, facilitates the development of new service, but implicates the creation of specific standard interfaces. (see Figure 17)

2.2 IMS Architecture

Before starting to explain IMS architecture in detail, we're going to have a view of the IMS basic principles that determines several aspects of its structure.

- IMS enables *access independence*. This mean that all existing networks could work with IMS, through appropriate gateways and interfaces, in order of the layer considered.
- IMS works with *terminal and user mobility*.
- IMS allows operators and service providers to use *different underlying network architecture*.
- IMS offers extensive IP-based services, such as VOIP (Voice Over IP), POC (Push to talk Over Cellular), multiparty gaming, videoconferencing, presence information, instant messaging, content sharing, and so on.

We can imagine the target architecture of a communication system implementing IMS as follows viewed in Figure 18.

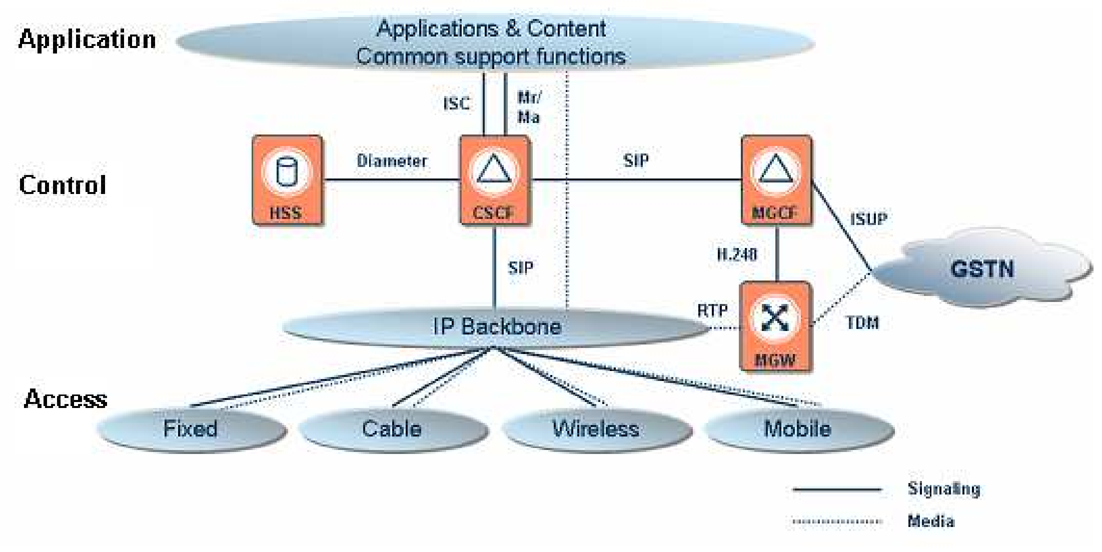


Figure 18: Target architecture

To satisfy the requirement imposed from IMS features, a layered architecture was selected for this architectural framework. It has been said that IMS will work over any type of underlying network architecture: this means that transport and bearer services must be separated from the IMS signaling network and session management services. An additional benefit of this type of architecture is that new access networks can be added later on. From bottom to up, in IMS layered architecture we have: Access plane, Control plane and Applications plane.

Access plane achieves the connection of all users to IMS core network. This is made directly if the user employs an IMS terminal, and through gateways if the device is not IMS. Gateways employ standard interfaces that make it possible communicating with all existing entities. This layer is hence directly responsible for carrying the traffic between endpoints.

Control plane has as core occupation the Call Session Control Function (CSCF). This function is reached by sharing the control between three different entities: Proxy, Serving and Interrogating. We’re going to analyze them in detail further on.

Applications plane hosts and executes services (IP applications), and delivers them using SIP to interface with the Control layer.

Now we can go beyond this trivial overview analyzing every part of IMS architecture in detail.

2.2.1 HSS - Home Subscriber Server

It’s the main database supporting IMS architecture entities responsible of call or session management. HSS contains all information about user profile, manage authentication and authorization at IMS level and can provide information about physical position of the user. Information about user profile includes user identity, allocated S-CSCF name (see the S-CSCF paragraph), roaming profile, authentication parameters and service information. User identities can be private or public. The private user identity is assigned by the home network operator and is used for such purpose as registration and authorization, while the public user can be used from other user for requesting communication with the end user. HSS also provides the traditional Home Location Register (HLR) and Authentication Centre (AUC) functions, required by the PS domain and the CS domain. Depending on several factors, like number of mobile subscribers, capacity of the equipment and network organization, there may be more than one HSS per home network. In case of multiple HSS, a Subscriber Location Function (SLF) is needed to map user addresses to enable I-CSCF, P-CSCF and AS to find the address of the HSS holding the required user-specific data. Both HSS and SLF communicate through Diameter protocol[2].

2.2.2 CSCF - Call Session Control Function

Processing SIP signaling packets in the IMS is achieved by using a group of entities that we could call "the session management and routing family". This family consists of three entities: Proxy CSCF (P-CSCF), Interrogating CSCF (I-CSCF), and Serving CSCF (S-CSCF). See Figure 19 for a schematic overview.

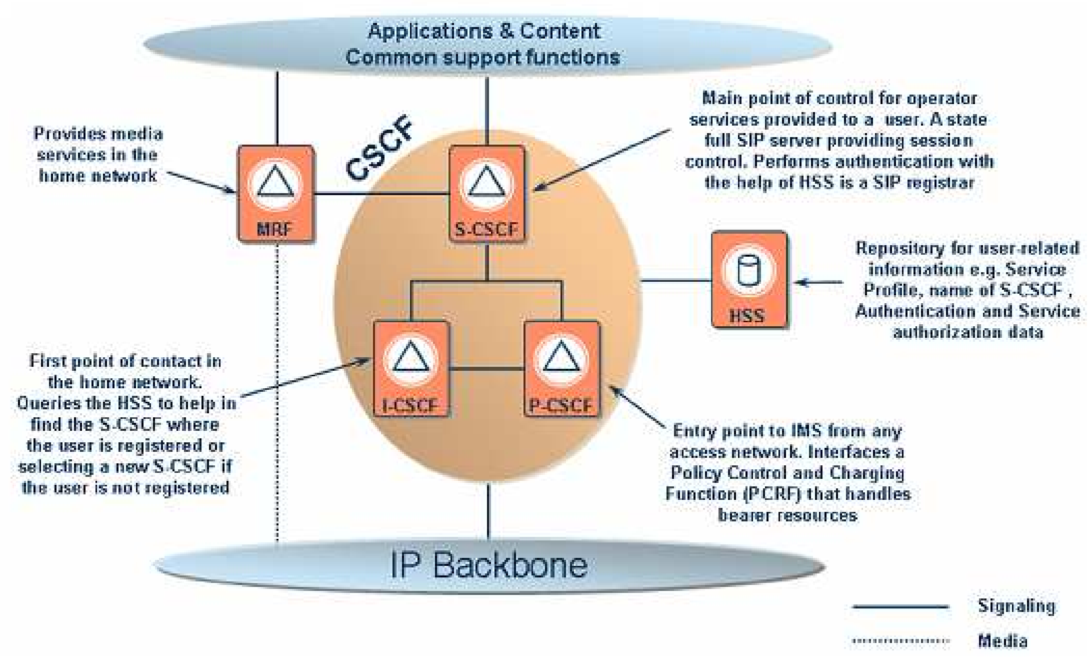

Applications & Content
Common support functions
CSCF
Provides media services in the home network
MRF
S-CSCF
Main point of control for operator services provided to a user. A state full SIP server providing session control. Performs authentication with the help of HSS is a SIP registrar
HSS
Repository for user-related information e.g. Service Profile, name of S-CSCF, Authentication and Service authorization data
I-CSCF
P-CSCF
First point of contact in the home network. Queries the HSS to help in find the S-CSCF where the user is registered or selecting a new S-CSCF if the user is not registered
Entry point to IMS from any access network. Interfaces a Policy Control and Charging Function (PCRF) that handles bearer resources
IP Backbone
Signaling
Media

Figure 19: IMS core overview

[2] Diameter is an AAA protocol (Authentication, Authorization and Accounting) subsequent to RADIUS, while not directly compatible. Diameter is a peer to peer protocol, and consists in a basic protocol extended with specific Diameter applications. The basic protocol alone is unusable, but can always be associated with any Diameter application service specific. It's supported by TCP or SCTP and requires imperatively IPsec. See below for more details.

2.2.2.1 P-CSCF

The first point of contact for users within IMS is P-CSCF. It performs a stateful SIP proxy, all the traffic from/to end users passes through this entity. The P-CSCF validates the request, forwards it to selected destinations and processes and forwards the response. It performs user authentication, can establish a secure session IPsec with IMS terminals and supports Resource Admission Control functionalities. In addition, it may behave as a User Agent, which role is needed for releasing session in abnormal conditions and for generating independent SIP transactions. In one operator network there can be one or many P-CSCF. The terminal discover its P-CSCF using DHCP[3] or, in GPRS, PDP[4] Context. When an IMS terminal is assigned to a P-CSCF this association does not change during the validity period of the registration. We're going to see each function in detail:

- Forwards SIP register requests to I-CSCF based on a home domain name provided by the user in the request, and other requests and responses received by the user to S-CSCF.
- Detects emergency session establishment requests.
- Sends accounting related information to the Charging Collection Function (CCF).
- Provides integrity protection of SIP signaling and maintain an IPsec security association with user.
- Can make compression of SIP messages to reduce the round trip time over critical links.
- Executes media policing, checking the content of the SDP payload (SDP: Session Description Protocol) to ensure the media is allowed for the user.
- Maintains session timers. It can detect a free resources used up by hanging sessions.
- Interacts with Policy Decision Function (PDF), which can also be included in P-CSCF. PDF is responsible of authorizations policy on media plane information obtained from P-CSCF. This function takes a service level policy request from the application layer and translates it into IP QoS parameters.
- Can provide defenses against SIP signaling attacks.

2.2.2.2 I-CSCF

It's a stateless SIP proxy placed on the edges of an administrative domain and it's a contact point for all connections destined to a user actually roaming in a different operator network. It's the entity that is able to determine the S-CSCF with which a user should register. This is achieved by querying the HSS. The I-CSCF can be removed from the signaling path once it has been used to establish which S-CSCF is in use. The I-CSCF IP address is published in the Domain Name System (DNS) of the domain, so that remote servers can find it and use it as a forwarding point. Up to release 6, I-CSCF can also be used to hide topology, capacity and configuration of the internal network from the outside world (function THIG), but from release 7 this function is part of the Interconnection Border Control Function

[3] Dynamic Host Configuration Protocol: protocol used from client to obtain parameters to operate in IP network. It reduces workload, sharing it between devices.

[4] Packet Data Protocol: contains user and session information while the client is working.

(IBCF). IBCF roles include the provision of NAT and Firewall functions for signaling, policing of signaling, topology hiding and conversion between IPv4 and IPv6. It also controls the media exchanged across the operator boundary.

2.2.2.3 S-CSCF

It's the brain of the IMS. This entity registers the users and provides services to them. It performs multimedia session setup, changing and release; routing, translation, provides billing information to media systems, interrogates HSS to retrieve authorization, service triggering information and user profile, using Diameter protocol. It has no local storage of the user. It's a SIP server, always located in the home network. During the session setup and it monitors SDP to ensure the session respects user profile edges. In detail, the functionalities performed by the S-CSCF are:

- Handling registration request; it knows the user IP address and which P-CSCF is using as IMS entry point.
- User registration and de-registration (with registration timer supervision) and authentication by means of the IMS Authentication and Key Agreement schema.
- Download of user information and service related data from HSS when necessary.
- Routing of mobile-terminating traffic to P-CSCF and of mobile-originated traffic to I-CSCF, BGCF or AS.
- Session control with capability to decide when a request must be further processed by routing to an AS, that is interaction with service platforms.
- Translation of E.164[5] numbers to SIP URI needed from SIP signalling routing.
- Media policing checking SDP payload.
- Supporting of emergency sessions.
- Sending of accounting-related information to the Charging Collection Function (CCF).

2.2.3 AS – Application Servers

It's a SIP server that hosts and executes services, interfacing with S-CSCF to obtain users information through SIP. As is usually employed to support different types of telephony services and to manage multi-client sessions involving multimedia resources. Depending on the services, the AS can work in different modes: SIP proxy, SIP UA or SIP B2BUA (back-to-back user agent, that is like user agent to both ends of a SIP call, responsible for handling all SIP signaling between both ends of the call). An AS can be located in the home network or in an external network. In the former case, it uses Diameter interface, in the latter it uses Mobile Application Part (MAP)[6] interface. "AS" is a term used to summarize the behaviors

[5] E.164 number is a number used in PSTN and some other network, defined from ITU-T recommendation which defines the international public telecommunication numbering plan.

[6] MAP is a SS7 application layer protocol which provides an application layer for the various nodes in GSM and UMTS mobile core networks and GPRS core networks to communicate with each other in order to provide services to mobile phone users.

of SIP AS, OSA – SCS (Open Service Access – Service Capability Server) or IM-SSF (IP Multimedia Service Switching Function) to offer access to services based on the Customized Applications for Mobile network Enhanced Logic (CAMEL). There may be one or more AS per subscriber and one or more AS involved in the same session, in order to provide multiple services to the same user in the same time.

2.2.4 MRF - Media Resource Function

This entity provides for the media manage in the home network. It's used to process multimedia data, to make multimedia conferencing of chat with collaborative work sessions, Text-To-Speech (TTS) conversions and language recognition, real time conversion of multimedia data with different coding. Every MRF is shared in two functional units, explained later on.

2.2.4.1 MRFC - MRF Controller

It controls the resources of the multimedia flows made available from MRFP, supporting services on media like transcoding and conference. MRFC interprets SIP signaling received via S-CSCF, acting like a SIP User Agent for S-CSCF, and controls MRFP by a H.248 MEGACO interface or by Real Time Stream Protocol (RTSP). MRFC is also able to send information to CCF.

2.2.4.2 MRFP - MRF Processor

It's a node in the media plane and implements all the functions connected to the media. In MRFP are implemented set of codecs, transcoders and functions for mixing of different media to provide the chance to handle audio and video in connectivity layer. MRFP is shared in media bridges for real time applications (conference bridging and transcoding) and media player for streaming application (voice mail, network notifications and information services). This unit is requested and instructed by MRFC to connect media resources to media streams.

2.2.5 BGCF - Breakout Gateway Control Function

It's a SIP server that includes a routing function based on telephone numbers. BGCF is used only when performing a call from an IMS terminal to a telephone in a circuit switched network, such as PSTN or PLMN.

PSTN/CS Gateway

These entities connect IMS with Circuit Switched (CS) network PSTN, in order to make possible connecting the multimedia domain with the CS network. The IMS PSTN Gateway is formed from different parts, explained here on.

2.2.6 MGCF – Media Gateway Controller Function

It controls its slave gateways SGW and MGW by a H.248 interface. MGCF performs protocol conversion between the ISDN User Part (ISUP) and SIP protocols and forwards the session to IMS or to CS domain. MGCF also manage setup, changes and termination of the multimedia session using SIP in multimedia domain and ISUP in PSTN. Again, it provides addressing and routing of the multimedia sessions from and to CSCFs and PSTN nodes. MGCF also controls media channels in the associated user plane entity, the MGW.

2.2.7 MGW – Media Gateway

It connects IMS to the media layer in CS network, making the conversion of Real-time Transport Protocol (RTP) in Pulse Code Modulation (PCM) and vice versa, adapting the payload from PSTN circuit in IP packets. MGW can also transcode when codecs are different. In addition, it's able to provide tones and announcements to CS users.

2.2.8 SGW – Signalling Gateway

It is used to interconnect different signaling networks, connecting IMS to the CS signalling plane. It converts lower layers protocols such as Stream Control Transmission Protocol (SCTP), an IP protocol, in Message Transfer Part (MTP), a Signalling System 7 (SS7) protocol, to pass ISUP from MGCF to CS network.

2.3 IMS implementation and services

We're going now to analyze other aspects of IMS. To start, we'll see the protocols employed, including identification and charging praxis, and then an outline of the main services provided by IMS.

The protocols employed in IMS can be clustered in three main groups: Signaling protocols, Security protocols, Media management protocols. For each one of these we're going to analyze the most significant parts: SIP and SDP for Signaling, Diameter for Security and RTP and RTCP for Media management.

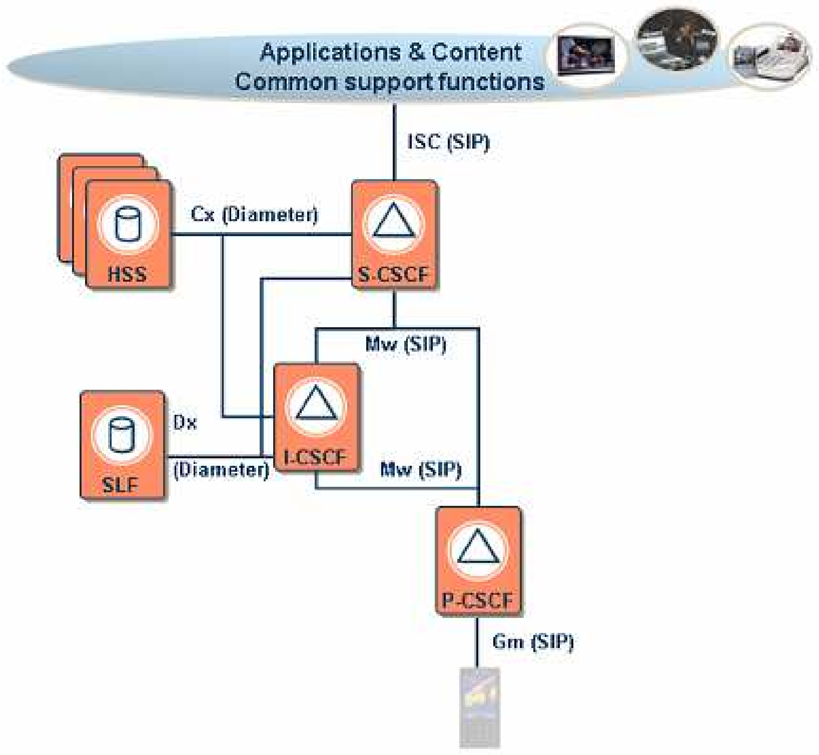


Figure 20: Use of SIP and Diameter protocol in IMS

2.3.1 SIP - Session Initiation Protocol

SIP is an IETF standard and it is employed in conjunction with other protocols such as the Session Description Protocol (SDP) and the Real Time Streaming Protocol (RTSP). SIP is a signalling protocol and manages multimedia sessions setup, tear down and supervision. In theory, SIP may be used with any transport protocol, but in practice the one that's always used is RTP. SIP defines two entity categories: client (User Agent client, an application that sends SIP requests) and server (an application that answers to SIP requests). It is hence based on a classic distributed logic architecture, client/server.

In the same physical unit there may be client and server. In SIP, there are four types of servers: proxy server (receives requests and manage them, redirecting them to the correct server or endpoint, allowing functions such as call forwarding, time-of-day routing, follow me..), redirect server (receives SIP requests and makes address translation), user agent server (typically in users terminals, receives call requests and answers to them) and registrar (a server that accepts registration requests).

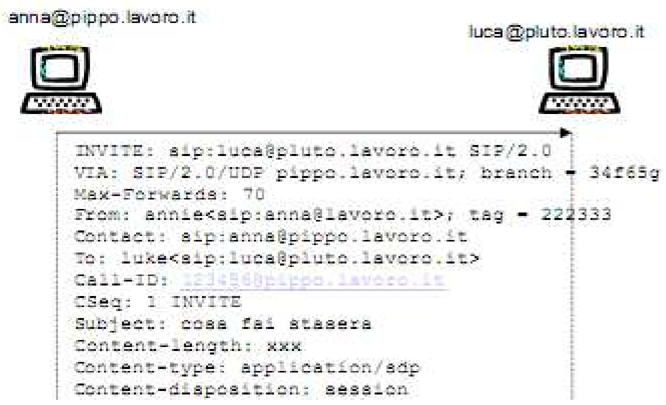
Sip is very simple and flexible. It doesn't direct manage media; users do it using proper customizable fields. SIP messages are in text format: this makes it clearer, but using more bandwidth. SIP messages have a field, called message body that is left free for SDP or other protocol. For example, it can be used to transport ISUP messages to interwork with PSTN networks.

Basic SIP methods are:

- INVITE to open a session, specifying user identity and media type;

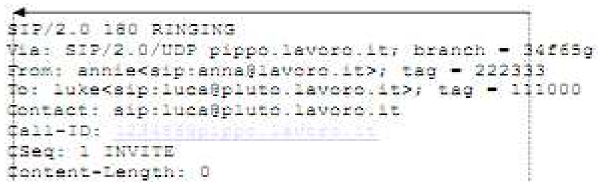
- ACK to answer to a request;
- OPTIONS to know server capabilities, which type of functionalities and media supports..;
- BYE to close a session;
- CANCEL to terminate an occurring transaction;
- REGISTER to make the registration of a US client in a SIP server
- INFO to transfer information such as multi-frequency tones, billing information, tunneling of non-SIP signaling during a session

We can have look to an example of session initiation in SIP:



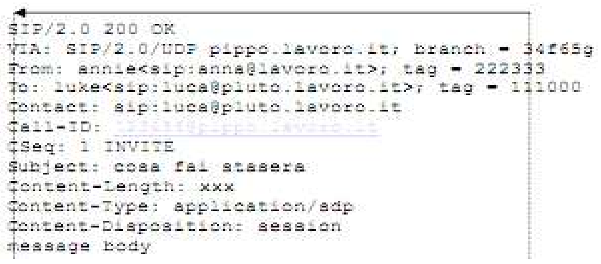
```
INVITE: sip:luca@pluto.lavoro.it SIP/2.0
VIA: SIP/2.0/UDP pippo.lavoro.it; branch = 34f65g
Max-Forwards: 70
From: annie<sip:anna@lavoro.it>; tag = 222333
Contact: sip:anna@pippo.lavoro.it
To: luke<sip:luca@pluto.lavoro.it>
Call-ID: 123456@pippo.lavoro.it
CSeq: 1 INVITE
Subject: cosa fai stasera
Content-length: xxx
Content-type: application/sdp
Content-disposition: session
```

Figure 21: Example SIP INVITE



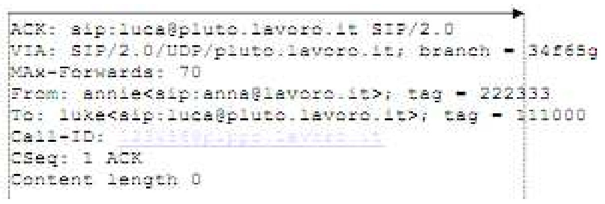
```
SIP/2.0 180 RINGING
Via: SIP/2.0/UDP pippo.lavoro.it; branch = 34f65g
From: annie<sip:anna@lavoro.it>; tag = 222333
To: luke<sip:luca@pluto.lavoro.it>; tag = 111000
Contact: sip:luca@pluto.lavoro.it
Call-ID: 123456@pippo.lavoro.it
CSeq: 1 INVITE
Content-Length: 0
```

Figure 22: Example SIP RINGING



```
SIP/2.0 200 OK
VIA: SIP/2.0/UDP pippo.lavoro.it; branch = 34f65g
From: annie<sip:anna@lavoro.it>; tag = 222333
To: luke<sip:luca@pluto.lavoro.it>; tag = 111000
Contact: sip:luca@pluto.lavoro.it
Call-ID: 123456@pippo.lavoro.it
CSeq: 1 INVITE
Subject: cosa fai stasera
Content-Length: xxx
Content-Type: application/sdp
Content-Disposition: session
message body
```

Figure 23: Example SIP 200 OK



```
ACK: sip:luca@pluto.lavoro.it SIP/2.0
VIA: SIP/2.0/UDP/pluto.lavoro.it; branch = 34f65g
MAx-Forwards: 70
From: annie<sip:anna@lavoro.it>; tag = 222333
To: luke<sip:luca@pluto.lavoro.it>; tag = 111000
Call-ID: 123456@pippo.lavoro.it
CSeq: 1 ACK
Content length 0
```

Figure 24: Example SIP ACK

After the session initiation, that could be a little more complex if the call passes through one or more proxies, in SIP generally we have the delivery of the session description (using SDP), the session management (once the connection is established, endpoints directly exchange media streams), and the session termination.

2.3.2 SDP - Session Description Protocol

SDP is a media description protocol, it doesn't manage media transport, so it must be used with a protocol adapt to transport media over user plane, for example SIP. For SDP a session is formed of one or more media streams. The description provided regards each of the session streams. SDP, like SIP, uses text format messages.

Security protocols

Security protocols are also defined as AAA protocols. The triple A stands for Authentication Authorization and Accounting.

Authentication refers to the process of establishing the identity of one entity to another entity. Commonly one entity is a client (a user, a client computer, etc.) and the other entity is a server (computer). Authentication is accomplished via the presentation of an identity and its corresponding credentials, such as passwords, digital certificates, and phone numbers (calling/called).

Authorization refers to the request of specific types of privileges (including "no privilege") to attribute to an entity or a user, based on their authentication. Most of the time the request of a privilege constitutes the ability to use a certain type of service. Examples of types of service include, but are not limited to: IP address filtering, address assignment, route assignment, QoS/differential services, bandwidth control/traffic management.

Accounting refers to the tracking of the consumption of network resources by users. This information may be used for management, planning, billing, or other purposes. Typical information that is gathered in accounting is the identity of the user, the nature of the service delivered, when the service began, and when it ended. IMS architecture employs Diameter as authentication protocol.

Diameter

Diameter is loosely based on the Remote Authentication Dial In User Service (RADIUS). Diameter is divided in two parts: Diameter base protocol and Diameter applications. The former is needed for delivering Diameter data units, negotiating capabilities, handling errors and providing for extensibility. The latter defines application-specific functions. Currently, a few applications have been defined and some are in the process of being defined. Diameter uses both Transmission Control Protocol (TCP) and Stream Control Transmission Protocol (SCTP), but SCTP is preferred. Diameter base protocol provides the following services: capability negotiation, error notification, transporting of user authentication information and of service specific authorization information, exchanging of resource usage information, relaying, proxying and redirecting of Diameter messages.

Media management protocols

To deliver media, IMS employs the Real time Transport Protocol and the Real time Transport Control Protocol (RTCP) that is a part of RTP.

2.3.3 RTP- Real time Transport Protocol

RTP provides end-to-end network transport functions suitable for applications transmitting real-time data, such as audio and video over multicast or unicast network services. RTP provides end-to-end delivery services for data with real-time characteristic. These services include: payload type identification, sequence numbering, time stamping and delivery monitoring. RTP does not assume that the underlying network is reliable and delivers packets in sequence. The sequence numbers included in RTP allow the receiver to reconstruct the sender's packet sequence. RTP consists of two closely-linked parts: the Real time Transport Protocol, to carry data with real-time properties, and the Real time Transport Control Protocol, to monitor the quality of service and to convey information about the participants in an on-going session. RTCP is based on the periodic transmission of control packets to all participants in the session, using the same distribution mechanism as the data packets. RTCP performs three main functions: provides feedback on the quality of the data distribution, controls the number of participants and the rate, and provides session control information through the user interface.

Identification

To make use of IMS services, the user must have the IP Multimedia Services Identity Module (ISIM). This unity contains parameters for identifying and authenticating the user to the IMS. The ISIM contains a private user identity (a NAI address, `username@operator.com`), one or more public user identities (a SIP, `sip:user@operator.com`, or TEL, `tel:+1-212-555-12345`, address) and a long-term secret used to authenticate and calculate cipher keys.

Charging

The IMS architecture allows different charging models to be used. It is possible to use different charging schemes at the transport and IMS level. The IMS architecture supports both online and offline charging capabilities. Online charging is a charging process in which the charging can influence in real time the service provided and therefore directly interacts with session and service control. In this approach, an operator could check the user's account before allowing the user to engage a session and to stop a session when the credit is over. This method is needed in prepaid services. In the Offline charging instead the charging information does not influence the service provided in real time. The charging information is collected from the operator (in the CCF that builds the Charging Data Record) during a certain period and then the bill is sent to the customer.

Services

IMS has the tools and functions to handle numerous of non-standardized services in a standardized way: interoperability, access-awareness, policy support, security, quality of service, interworking with existing networks; the properties necessary to meet ever-increasing consumer demand for attractive and convenient offerings. The end user services offered by IMS are the result of the combination of elementary services such as presence (involving person and terminal availability, communication preferences, terminal capabilities, current activity, location…), messaging (instant messaging, session-based messaging, deferred delivery messaging, MMS messaging, voice-video messaging), conferencing (with multimedia contents addition) and streaming multimedia. By the combination of these elementary services, IMS can provide different complex services, like Push to talk Over Cellular (POC) for make use of traditional walkie-talkie service over mobile networks with mobile handsets, or all the possible application derived from the seamless mobility offered by IMS: you can start a call on the fixed home phone and then transfer it on the mobile without loose it; start looking a film on TV, stop it momentarily, and go on the vision on a PDA in the garden. You could also exploit a mix of services, like sending a MMS from your car and making it appear on the home TV screen; during a conversation with some friends you could share the match video you're watching, and so on. Exploiting the possibility of make a remote access, you can also control the devices in you home: to program a registration, to send an image on the TV screen or to a digital photo frame, and so on…

In conclusion, we can consider that in this new emerging communication culture that revolves around the sharing of everyday life experience, anywhere, anytime, on any device, IMS is the natural choice to address this transformation. IMS combines the quality and interoperability of telecom with the quick and innovative development of the internet, making the unique values of the telecom industry readily available to the application development community.

3. Mobile Devices

In the D1_1 we analyzed emergency scenarios and we saw the importance of application delivery and the operators' equipments during such a situation. The application and services deployment needs to be as fast as possible and the operators' devices need to be small and portable.
In the first chapter we presented the available technologies regarding the application on demand service; the aim of this chapter is instead to provide a full overview of the available mobile devices including operating systems.
In the first paragraph we present indeed the most important environment regarding the development of software for small devices; in the second paragraph we will present the operating systems and in the third one we will give a brief overview concerning the most important current devices.

3.1 Development Environment

There are several ways to add software to a mobile device, depending on the device capabilities. The latest devices can have all the capabilities described below, making the device more versatile.

3.1.1 Java (J2ME)

J2ME (Java 2 Platform Micro Edition) is a specification of a subset of the Java platform aimed at providing a certified collection of Java APIs for the development of software for small, resource-constrained devices such as cell phones, PDAs and set-top boxes.
Most of the mobile phones can run J2ME applications, but there can be a lot of difference between the JVM capabilities from one mobile phone to another. The access to the net, gps, bluetooth etc. is still regulated by the phone operating system.
The JVM in the mobile phone use the Connected Limited Device Configuration (CLDC) that contains a strict subset of the Java-class libraries, and is the minimal amount needed for a Java virtual machine to operate. CLDC is basically used to classify myriad devices into a fixed configuration.
Using this common configuration it's possible to built applications using the upper layer of the J2ME stack, the Mobile Information Device Profile (MIDP).
Designed for cell phones, the Mobile Information Device Profile boasts GUI API, and MIDP 2.0 includes a basic 2D gaming API. Applications written for this profile are called MIDlets and can be downloaded directly from the web, installed and run just after the download.

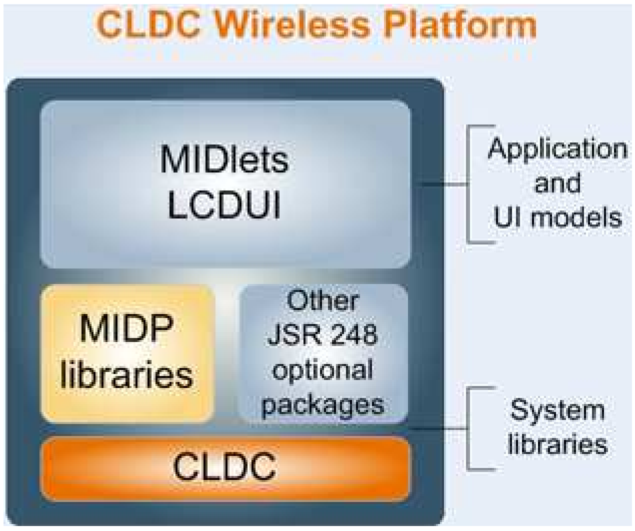


Figure 25: J2ME architecure.

3.1.2 Native code

Building application in native code, means build an application that can run only in similar devices (typically with the same OS) but that can use all the features of the mobile phones without any limitations.
The developer can optimize the resource used by his own application, optimizing also the performance and the memory usage. He could access the storage and the hardware directly using the OS I/O.
Some device's manufacturer has distributed the SDK for develop native application on their mobile phone and often an IDE with simulator's device.
As we can see, there are a lot of difference between a J2ME MIDlet and a native application. In general, a J2ME MIDlet can be more deployable and compatible in a lot of devices, but it can't reach the performance and the integration that a native application can reach in the OS that was made for.

3.1.3 Web

As we saw in the previous chapter the Web is changing its structure facilitating the sharing and collaboration among users. For this reason the latest devices have new versions of their internet browsers, that support al lot of the new features used in the Web 2.0. It's therefore possible to create internet based applications, using css, javascript and ajax interaction.
Mobile browsers are optimized so as to display Web content most effectively for small screens on portable devices. Mobile browser software must be small and efficient to accommodate the low memory capacity and low-bandwidth of wireless handheld devices.
Obviously the user will interact only with the browser and the applications will not use any of the advanced features like gps, accelerometers, bluetooth etc.
But these applications can be run everywhere (if we have an internet connection) and in every device with an advanced web browser.

3.2 Mobile OS

After this introduction on the development environments for mobile devices and before presenting the most famous devices with the related capabilities, we give an overview on the current available operating systems.

3.2.1 iPhone

iPhone OS is the operating system developed by Apple Inc. for the iPhone and iPod touch. Like Mac OS X, from which it was derived, it uses the Darwin foundation. iPhone OS has three abstraction layers: a Core Services layer, a Media layer, and a Cocoa Touch layer.
The iPhone OS's user interface is based on the concept of direct manipulation, using multi-touch gestures. Interface control elements consist of sliders, switches, and buttons. The response to user input is supposed to be immediate to provide a fluid interface. Interaction with the OS includes gestures such as swiping, tapping, and pinching. Additionally, turning the device alters orientation in some applications.
Mac OS X applications cannot be copied to and run on an iPhone OS device. They need to be written and compiled specifically for the iPhone OS and the ARM architecture. However, the Safari web browser supports "web applications," as noted below.
The iPhone support third-party "applications" via the Safari web browser, referred to as web applications. The applications can be created using web technologies such as AJAX. Many third party iPhone web applications are now available.
With the new generation, the iPhone support also the deployment of third party applications.
These applications can be developed with the free SDK available on the developer apple web site, and it's complete of an IDE and simulator.
The SDK itself is a free download, but in order to release software, one must enroll in the iPhone Developer Program, a step requiring payment and Apple's approval.
Signed keys are given to upload the application to Apple's App Store which is the sole method of distributing the software to an iPhone.
At the moment there is no Java support.

Figure 26: Apple iPhone.

3.2.2 Symbian

Symbian OS is a proprietary operating system, designed for mobile devices, with associated libraries, user interface frameworks and reference implementations of common tools, produced by Symbian Ltd. It is a descendant of Psion's EPOC and runs exclusively on ARM processors.

The Symbian OS System Model contains the following layers, from top to bottom:

- UI Framework Layer
- Application Services Layer
    - Java ME
- OS Services Layer
    - generic OS services
    - communications services
    - multimedia and graphics services
    - connectivity services
- Base Services Layer
- Kernel Services & Hardware Interface Layer

Symbian is not Open Source software. However, phone manufacturers and other partners are provided with parts of its source code. The APIs are publicly documented and up to Symbian 8.1 anyone could develop software for Symbian OS.

Symbian 9.1 introduced capabilities and Platform Security framework. To access certain capabilities, the developer has to digitally sign their application. Basic capabilities are user-grantable and developer can self-sign them, more advanced require certification and signing via the Symbian Signed program.

The native language of the Symbian OS is C++, although it is not a standard implementation. There are multiple platforms based upon Symbian OS that provide an SDK for application developers wishing to target a Symbian OS device.

Symbian C++ programming is commonly done with an IDE like Carbide.c++, an Eclipse-based IDE developed by Nokia.

Symbian OS's C++ is very specialized. However, many Symbian OS devices can also be programmed in OPL, Python, Visual Basic, Simkin, Perl and with the Java ME.

Once developed, Symbian OS applications need to find a route to customers' mobile phones. They are packaged in SIS files which may be installed over-the-air, via PC connect or in some cases via Bluetooth or memory cards. An alternative is to partner with a phone manufacturer to have the software included on the phone itself. The SIS file route is more difficult for Symbian OS 9.x, because any application wishing to have any capabilities beyond the bare minimum must be signed via the Symbian Signed program.

Java ME applications for Symbian OS are developed using standard techniques and tools such as the Sun Java Wireless Toolkit.

Nokia S60 phones can also run python scripts when the interpreter is installed, with a custom made API that allows for Bluetooth support and such. There is also an interactive console to allow the user to write python scripts directly from the phone.

3.2.3 Windows Mobile

Windows Mobile is the OS developed by Microsoft as evolution of Windows CE. It's used by different brands, like HTC, Samsung, HP etc. and it provide a common environment in different devices.

The main features of the new version are:

- High screen resolution supported (from 320x320 to 800x480)
- Integrated with Office and Exchange
- VoIP support
- Remote Access (RDP)
- Browser with Ajax, JavaScript and XML DOM capabilities
- .NET Compact Framework 2
- SQL Server compact edition

Additional software can be installed directly after a download from the web; typically the executables are .cab files and are installed automatically by the system.
Third-party software development is available for the Windows Mobile operating system. There are several options for developers to use when deploying a mobile application. This includes writing native code with Visual C++, writing Managed code that works with the .NET Compact Framework, or Server-side code that can be deployed using Internet Explorer Mobile or a mobile client on the user's device. The .NET Compact Framework is actually a subset of the .NET Framework and hence shares many components with software development on desktop clients, application servers, and web servers which have the .NET Framework installed, thus integrating networked computing space. Support also J2ME applications.

3.2.4 Proprietary OS

Many manufacturers prefers to build their own OS and don't release any SDK for develop additional software.
Typically, the only way to expand and to install new applications it's with the J2ME Midlets or, if there's an advanced web browser, with the web-based application.
Samsung, LG, Sony Ericsson and RIM are an example of these manufacturers.

3.2.5 Android

Android is a Linux-based OS for mobile devices.
Is developed by Google and Open Handset Alliance and will be available in open-source license.
Google provides the SDK from their website (http://code.google.com/android) with a plug-in for the Eclipse IDE, for an easy development and deployment.

The SDK allow the developer to create application writing code in Java language, but the result it's a lot different than a J2ME Midlets.
The OS has a complete and complex architecture (see figure below). Built in the OS there is an optimized JVM (called Dalvik) that exposes all the features of the device at the applications.

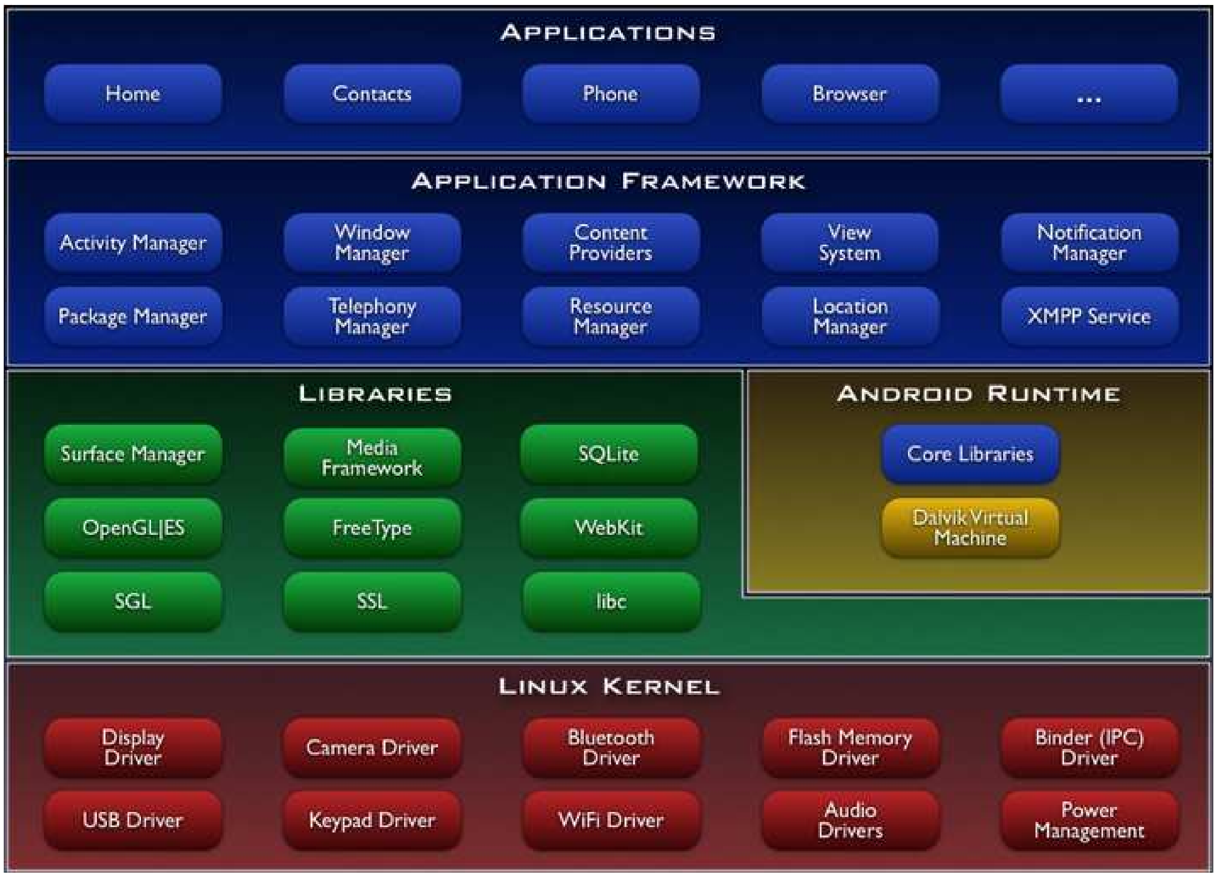


Figure 27: Android architecture.

It's possible therefore to create applications, which act like a native application, using the popular java language. The applications will run and will be managed in the Android Runtime, which includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language.
Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently with minimal memory footprint.

The main features of the Android device will be:

- *Optimized graphics:* The platform is adaptable to both larger, VGA, 2D graphics library, 3D graphics library based on OpenGL ES 1.0 specifications, traditional smartphone layouts.
- *Storage:* SQLite for structured data storage.
- *Connectivity:* Android supports a wide variety of connectivity technologies including GSM, CDMA, Bluetooth, EDGE, EV-DO, 3G, and Wi-Fi.
- *Messaging:* SMS, MMS, and XMPP are available forms of messaging including threaded text messaging.
- *Web browser:* The web browser available in Android is based on the open-source WebKit application framework.

- *Java virtual machine:* Software written in Java can be compiled into Dalvik bytecodes and executed in the Dalvik virtual machine, which is a specialized VM implementation designed for mobile device use, although not technically a standard Java Virtual Machine.
- *Media support:* Android will support advanced audio/video/still media formats such as MPEG-4, H.264, MP3, and AAC, AMR, JPEG, PNG, GIF.
- *Additional hardware support:* Android is fully capable of utilizing video/still cameras, touch screens, GPS, compasses, accelerometers, and accelerated 3D graphics.
- *Development environment:* Includes a device emulator, tools for debugging, memory and performance profiling, a plugin for the Eclipse IDE.

At the moment there is only the software simulator of the platform and only few experimental devices not for sale.

3.2.6 MOTOMAGX

MOTOMAGX is Motorola's next-generation Mobile Linux platform and will support three different application environments:

- Java
- WebUI: Web application that need Ajax and JavaScript capabilities, the browser will be based on WebKit browser engine.
- Native Linux: Developed in C/C++

3.2.7 Openmoko

Openmoko is an open source operating system for mobile phone Linux-based.
Native applications can be developed and compiled using the C or C++ programming languages and it use ipkg (Itsy Package Management System) for the package installation and management (similar to dpkg in Debian).
At the moment there is only one device model available at the web site of Openmoko.

3.3 Sample devices

We complete this chapter with a selection of devices that use the most diffuse operating systems already treated.

3.3.1 Apple iPhone

Apple iPhone is the innovative device released by Apple the past year and updated with the new version in June 2008.
Has a lot of interesting features and a unique user interaction.
It has a multi-touch screen with virtual keyboard and buttons, but a minimal amount of hardware input. The iPhone's functions include those of a camera phone and portable media player (iPod) in addition to text messaging and visual voicemail. It also offers Internet services including e-mail, web browsing, and local Wi-Fi connectivity. The first generation phone hardware was quad-band GSM with EDGE; the second generation uses UMTS and HSDPA and a GPS receiver.

Specifications:

- Size: 110 mm(h) × 61 mm(w) × 12 mm(d)
- Screen size: 3.5 in (89 mm)
- Screen resolution: 480×320 pixels at 163 ppi
- Input devices: Multi-touch screen interface plus a "Home" button
- Built-in rechargeable, non-removable battery
- 2 megapixel camera
- 412 MHz ARM 1176 processor
- PowerVR MBX 3D graphics co-processor
- Memory: 128 MB DRAM
- Storage: 8 GB or 16 GB flash memory
- Operating System: iPhone OS
- Quad band GSM (GSM 850, GSM 900, GSM 1800, GSM 1900)
- GPRS and EDGE data
- Tri band UMTS/HSDPA (850, 1900, 2100 MHz)
- Wi-Fi (802.11b/g)
- Bluetooth 2.0 with EDR
- Weight: 133 g (4.7 oz)
- Headphone jack (non-recessed)
- Camera features geotagging (producing geocoded photograph)
- Battery has up to 10 hours of 2G talk, 5 hours of 3G talk, 5 (3G) or 6 (Wi-Fi) hours of Internet use, 7 hours of video playback, and up to 24 hours of audio playback, lasting over 300 hours on standby.
- Assisted GPS, with fallback to location based on Wi-Fi or cell towers
- Accelerometers

3.3.2 Nokia N96

The Nokia N96 it's the most evolutes smart phone equipped with a Symbian OS.

Specifications:

- Quad band GSM / GPRS / EDGE: GSM 850, GSM 900, GSM 1800, GSM 1900
- Dual band UMTS / HSDPA: UMTS 900, UMTS 2100
- 3G and WLAN access.
- Mobile TV (network-dependent feature).
- GPS Navigation.
- Access to Ovi (Nokia service web based)
- Instant upload to Flickr, Vox, Yahoo! and Google.
- Full-HTML browser.
- Symbian OS v9.3 S60 3.2 Edition, user interface.
- Up to 16 GB of internal flash memory.
- 2-way slide, as in Nokia N95.
- Expandable memory currently up to 24 GB courtesy of MicroSD cards.
- 5-megapixel camera, Carl Zeiss optics.
- High quality VGA camera in front of the phone, for video calling and self-portrait use.
- Double LED flash for the camera.
- Plays music files, and lets you download easily via Nokia Web.
- Allows high-quality video calling using 3G
- A built-in motion sensor that automatically rotates the screen when tilted.

3.3.3 HTC TyTN II

The HTC TyTN II is a Microsoft Windows Mobile 6.0 Pocket PC phone manufactured by HTC.

Specification:

- Quad band UMTS/HSDPA/HSUPA: UMTS 800, UMTS 850, UMTS 1900, UMTS 2100, UMTS 1700
- Quad band GSM/GPRS/EDGE: GSM 850, GSM 900, GSM 1800, GSM 1900
- Connectivity: Wi-Fi 802.11b/g, Bluetooth 2.0 + EDR with A2DP, A-GPS and GPS, USB 2.0
- GPS: Qualcomm
- CPU: 400 MHz Qualcomm 7200 (dual CPU, and integrates Imageon hardware 2D/3D graphics accelerator) Although the graphics hardware is without a Windows Mobile driver, so all graphics are done using the main CPU, therefore unaccelerated.
- Operating System: Windows Mobile 6
- Camera(s): 3.0(2048x1536) MP still/video camera with autofocus. VGA Video conferencing camera.
- Memory: 128 MB RAM, 256 MB ROM
- Memory card: SDIO, microSD, microSDHC 4GB and up, TransFlash
- Screen: 240x320, 2.8" (42 x 57 mm) TFT-LCD
- Weight: 190g

- Size: 112mm (L) x 59mm (W) x 19mm (T)
- Battery: Li-Ion 1350 mAH

4. Technologies support to Context-Aware

Context-aware applications could be defined as computing applications that use context information in order to automatically adapt their behaviour to match the situation. Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between an user and an application. Context information can be gathered from a variety of sources, such as, sensors, profiles (capabilities of hardware devices or preferences of users) and applications that report their current state and data interpretation services (considers context information to derive higher-level information).

Context-aware applications can intelligently support users in a variety of tasks, including tasks that support disabled people. Context-aware applications can create smart home environments, provide health-care services and also support user tasks. Some common applications related to that kind of situations are:

- **Flexible communication**

  Such applications provide instant communication with a family member, friend or health worker, achieved using context-aware communication channel (telephone, SMS, etc.) according to the current activity, available communication devices, and people preferences.

- **Support for social interactions and virtual communities**

  Such applications have diverse goals in order to support independent communities:

  - assistance with everyday tasks by remote family or community members;
  - dynamic organization of groups that are interested in activities based on their preferences and availability;
  - awareness of people activities in homes in order to minimize isolation.

- **Smart spaces**

  Such spaces can provide automatic configuration and reconfiguration of devices, in order to adapt their functionality/behaviour to user needs and preferences.

- **Multi-modal technologies and home appliances**

  Context adaptive interfaces can assist individuals by providing alternate modes of interaction with common devices such as phones, microwaves, and televisions.

- **Health care applications**

  These applications can provide health monitoring, accident monitoring, behavioural trend monitoring and cognitive health monitoring. Other health care applications provide reminders for eating, taking medications, and a variety of other tasks and activities.

Services are able to modify their behaviour and the information provided, based on the emergency context in which the user is. Users can be notified in case of an emergency (e.g. fires, accidents, robberies…) and be informed on how to proceed in those situations. The user context is especially dynamic for mobile systems such as mobile phones, laptop, etc. When using such devices, it is important to adapt the application's behaviour to the ever changing situation. Some important aspects related to context are:

- **Environmental Factors**

  There are different approaches to choose the relevant aspects in environmental context. For example, it is distinguished between comprising network variations (bandwidth, latency, etc.), hardware variation (screen size, buttons, etc.), and memory and software variations (memory capacity, installed applications, etc.).

  The information about the infrastructure and the location information (physical and logical, at home versus at work, e.g.) is also depicted as very relevant for the user context. The following kinds of environmental context can be considered:

  - **User Context**

    This aspect takes into account personal information about the user and user's classes as user's identity, characteristics, capabilities, universal preferences, the state of the user, information about his or her main activity, etc.

  - **Resource Context**

    The resource context refers to the demand of multi-delivery and thus includes information about the relevant devices, device classes, documents, network, available services, etc.

  - **Location Context**

    This context describes the geographical coordinates, identity and the state of the location (the people present at that location, etc).

    The location context includes also aspects of the perception of the physical characteristics of the location, e.g. temperature, brightness, noise levels, etc.

  - **Temporal Context**

    The temporal context (in diverse forms, e.g. the absolute time, hour, am / pm, etc.) allows to adapt the application with regard to certain timing constraints such as the hour to go to work.

General context categories	**Features**

User	Tasks of the user. User's profile (experience, etc.). People nearby. Characters, date and time formats.
Resources	Size of a display. Type of the display (colour, etc.). Input method (touch panels, buttons). Network connectivity Communication cost and bandwidth. Nearby resources (printers, displays).
Location	Position (latitude and longitude). Lighting, temperature, weather conditions, noise levels. Surrounding landscape. User's direction and movement.
Temporal	Time of day. Week, month. Season of the year

Table 1: Categorization of environmental context

- **Application State Factor**

  This factor has the information about the state of the application itself and complements the environmental factors.

- **History Factor**

  This is necessary to identify if the context values change over time. For this, it is needed to consider the historical circumstances and the current ones.

- **Static and Dynamic Context**

  User information can be static or dynamic. Some context information is considered static if the context is defined and it does not change during the application execution. The context information is considered as dynamic if it is determined during run-time.

- **Availability of Context Information**

  Not all context factors are always available, e.g. if one user has not interacted with the system yet, there may be no data about him/her available.

Context Aware Computing relies on several independent enabling technologies. For input it relies on sensors, and these have their own limitations. For processing it relies on hardware, it depends heavily on artificial intelligence principles and those depend on specialized software for implementation, which in turn, relies on data and rules extracted from knowledge of the world, which probably includes knowing specific user's preferences. Hence, a key-enabling feature is the ability of computer systems to learn from past experience, and use that knowledge in future computations. One area that needs to be formally developed is the concept of similarity. A key feature to the realization of Context Aware Computing is the ability for a system to have “common sense”, at least to a degree that the user would consider it as such.

The integration of communication and computation technologies, the availability of pocket mobile computers, and the widespread penetration of mobile radio access networks will enable a range of new context-aware mobile services to be offered to the users. Research on methods and technologies for the development of such services requires the involvement of domains such as software engineering. In order to enable the development of integrated solutions, the work within these domains should be co-ordinated. Working across technological areas is often difficult. With the recent advances in mobile computer technology and the penetration of wireless networks, the nature of the services proposed to the users is moving towards mobile and context-aware services.

4.1 Context-Aware Mobility Solution

Context-aware mobility solutions have evolved and adapted to the requirements of complex business processes, integrating information beyond the location of assets such as telemetry and motion. Such solutions provide the ability to dynamically capture and use contextual information about mobile assets to optimize, change, or create communications flow and business processes. Contextual information can be collected for any mobile asset involved in a business process, and this includes not just devices and products but also people. For instance, a mobile asset can be a worker, a customer, or a patient, or it can be a pallet of finished goods.

Typically, several wireless networks have to be used during execution of the same business process. The most widely adopted wireless technology is the wireless LAN (WLAN), but Wi-Fi mesh, WiMAX, cellular, or GPS networks can also be used when devices, tags, and sensors with the proper radios are available. A variety of context-aware technical solutions using Wi-Fi or other wireless networks can be deployed running standalone or simultaneously. Context-aware mobility solutions are most often Wi-Fi based, since the majority of enterprises today have a WLAN deployed, which they use for corporate communications. Also, the majority of mobile devices include Wi-Fi radios.

In a context-aware mobility solution, wireless devices, tags, or sensors send the contextual information they collect via the WLAN. The network, in turn, uses these signals to calculate the location of the assets and correlate it to additional sensor information, if available. The algorithms used to determine the location vary depending on the RF environment and the accuracy needed for a specific application. For some business applications, it is important to track an asset continuously throughout an entire facility. On the other hand, some business processes only require information that captures whether an asset is in or out of a zone (room, parking place, and so on).

Sensor capabilities can be directly embedded into tags in order to link the data captured (for instance, motion or temperature data) with the location of the mobile asset. Many industries are already using context-aware technologies to manage the mobility of their assets. Healthcare, manufacturing, retail, and education are integrating context-aware information into their business practices and processes for innovation. Typically, in a context-aware mobility solution there are two kinds of location-tracking systems used: active and passive tracking. In an active tracking system, the locatable device has a battery and uses it to send messages to the readers continuously. In a passive system, the locatable device has no battery and sends messages only after it receives energy coming from a chokepoint.

4.1.1 Active Location Tracking

Active location tracking uses two different technologies: received signal strength indication (RSSI) and time difference of arrival (TDoA).

Received Signal Strength Indication (RSSI)

Active location tracking can be performed on wireless assets or assets such as wheelchairs, which carry active RFID tags. It provides a versatile option for enterprises that like to take advantage of context-aware mobility solutions to increase the productivity of their business. Active tracking using RSSI is a technique in which a measuring device detects the signal strength of a transmitter's packets and determines its own location or the location of the transmitter based on those measurements.

One advantage of active tracking is that it typically uses a wireless network infrastructure that is already in place and at the same time functions as a network for data, voice, and video communications. All wireless devices that are working on the network can be directly tracked. Most context-aware mobility solutions are based on Wi-Fi networks for a lower total cost of ownership. Indeed, the network serves multiple purposes, including connectivity and the collection of contextual information, and all these network services can be centrally managed.

When the network infrastructure provides tracking data, the location information can be easily integrated with other relevant network data, such as security credentials, statistical information, and user information, to enable a complete collection of contextual information on the mobile device in the network. In an active tracking technology, the device being tracked sends signals to the wireless network so its location can be calculated.

For environments with many obstacles and obstructions, such as in the outdoors or in warehouses, active location tracking based on RSSI performs better than other techniques. RSSI is most suitable to collect context-aware information of high value or high utility items indoors. Indoor offices, hospitals, banks, smaller warehouses, are suitable for active tracking with RSSI. In these environments, it is particularly useful and outperforms other location determination techniques.

Time Difference of Arrival (TDoA)

All receivers are synchronized, using either over-the-air signaling or a common clock distributed via separate cabling to each receiver. When a client transmits, all receivers record the time of arrival (ToA) of the packet and send it to the device that performs the location calculation (typically a location server). Even though the location calculation engine doesn't know when the packet was transmitted, it can compute TDoAs.

The location calculation becomes more accurate with higher bandwidth and greater transmit power. In line-of-sight environments, TDoA systems can achieve greater accuracy than RSSI systems, especially when the clients are at greater distances from the receivers or the receivers are mounted at great heights. Network infrastructure TDoA systems can be used to track assets in large physical open spaces such as university campuses, car lots, ports, or in RF-challenging environments, such as large, open warehouses and factories. In schools that have a pervasive WLAN, attendance can be tracked using the students' Wi-Fi laptops as locatable devices. If a laptop is lost, it can also be located throughout the campus using the WLAN.

4.1.2 Passive Location Tracking

Passive tracking is a technique where wireless network infrastructure devices, such as passive radio frequency ID (RFID) readers, or chokepoints are deployed to track devices that have no battery. These locatable devices can send messages only after they receive energy coming from a chokepoint or RFID reader.

Passive tracking can be deployed for specific applications such as tracking items that are being moved in and out of a specific limited zone or for error checking (for example, when associating two objects such as patient and medication). This is usually accomplished by protecting entrances and exits in a facility with readers or chokepoints, which capture the movement of assets in and out of the facility.

The limitations of passive tracking include the fact that the location of a device tracked by passive technology cannot be determined when it is not within range of a measurement device. Furthermore, in passive tracking, additional information such as telemetry data cannot be captured. In general, passive RFID tags are smaller and lower in cost than active tags, thus they are useful for tracking low-cost items in retail, distribution, or for keeping track of paperwork in hospitals.

4.1.3 Combining technologies

The choice of location-tracking technologies in a context-aware mobility solution depends on the environment of a facility or outdoor area, the accuracy needed for a specific application, and the type of information that should be recorded. For indoor applications, location tracking is often based on received signal strength indication (RSSI). For outdoor or high-ceiling environments such as warehouses, time difference of arrival (TDoA) is most appropriate. Passive location tracking is best used when it is important to note the close proximity to a specific point that has a reader attached. Complex business processes commonly require the use of multiple location-tracking techniques based on the existing network, the accuracy, or range needed by the applications, the type of locatable devices, and the continuous tracking of assets indoor and outdoor.

In healthcare, several hospitals have used their context-aware solution based on a WLAN (already used for voice, data, and video applications) to continuously track physicians and make sure they are present in the emergency room when needed. A chokepoint can be attached to the doorway of the emergency room so that personnel entering and exiting can be tracked and the system can record whether there is a physician present in the emergency room at any certain time. When a medical specialists needs to be called in, the closest medical specialist can be located using RSSI technology.

Some business processes profit most when a combination of active and passive tracking technologies is combined into a single context-aware solution. A network device that has both active and passive capability can be tracked almost continuously using the network. If the device is not within range of a passive network infrastructure, the device is tracked by the active network infrastructure. When it is in range of the passive network infrastructure, the device location is determined by the passive components of the network tracking solution. Typically, both elements of the network tracking solution can provide their data to a central location server, where aggregation of the data can provide the optimal location determination.

GPS is another technology that is highly mature and shows great promise for being integrated into context-aware mobility solutions. GPS tracking functions similar to the TDoA system already described, except that with GPS, satellites transmit signals to the client instead of vice versa. Satellite locations are embedded in the signalling, so that the client has all the information it needs to calculate its location autonomously. GPS is very accurate in the outdoors where line-of-sight links to the satellites are possible, but does not penetrate well into large buildings. Hybrid solutions like GPS with TDoA and Wi-Fi outdoor mesh networks with RSSI provide a combination that can avoid this problem.

4.1.4 Mobility solution analysis

In a context-aware mobility solution, wireless devices, tags, or sensors send the contextual information they collected via a Wi-Fi, cellular, or other wireless network, which in turn uses these signals to calculate the location of locatable assets. The location-tracking technologies used vary depending on the RF environment and the accuracy needed for a specific application. For typical indoor facilities such as business offices, a combination of active and passive location tracking provides the best context-aware mobility solution. RSSI works well as location-tracking technique in these environments. For outdoor areas, such as university campuses and challenging environments such as buildings with high ceilings or warehouses, TDoA is more appropriate. When the mobile asset does not have a radio capability, RFID tags can be attached to it to collect the parameters needed.

When selecting a context-aware mobility solution, it is crucial to consider several primary requirements:

• **Supports a broad range of enterprise environments:** For a context-aware mobility solution to support complex business processes, it has to integrate different location technologies and techniques. Active and passive location-tracking technologies combined with RFID tags and chokepoint technology provide the best solution for diverse environments and application requirements.

• **Helps innovate business processes and operations:** A context-aware mobility solution should directly support your current business processes and lead to innovations throughout the entire business operation. To optimize business processes, it is necessary to track more information about assets than just their location. It is crucial to be able to collect and process additional information about an asset, such as motion and telemetry information (temperature, humidity, pressure, and so on). To enable an enterprise to expand the solution seamlessly according to its growing needs, choose a context-aware mobility solution that supports the tracking of thousands of devices and clients.

• **Supports tight integration with business applications for vertical solutions:** Any context-aware mobility solution requires tightly integration with the specific business applications of an enterprise. Solutions that offer an open API support the integration of specific and vertical applications with best-in-class partners. Furthermore, the solution needs to be compatible with a broad range of clients.

• **Offers investment protection:** We get the best investment protection from a context-aware mobility solution based on a Wi-Fi network that is already in use for business communications and applications and is easily expandable and extendable to future technologies. Easy deployment and central management of all network services reduces total cost of ownership.

4.2 Common context-aware components

Nowadays, a user is usually accompanied anywhere at anytime by a constantly more capable mobile device that can act as intermediary by sensing (GPS, RFID, etc) and communicating (Wi-Fi, Bluetooth, GPRS/UMTS) with the environment, and so enabling rich interactions with it. The user controls through the device services discovered in the environment (explicit interaction) or the environment triggers autonomously services based on users' context, profile and preferences published by the device (implicit interaction). In all these situations, technologies related to context-aware development, are an important way to implement functionalities regarded to context-aware applications.

Context-aware systems can be implemented in many ways. The approach depends on special requirements and conditions such as the location of sensors (local or remote), the amount of possible users (one user or many), the available resources of the used devices (high-end-PCs or small mobile devices) or the facility of a further extension of the system. Furthermore, the method of context-data acquisition is very important when designing context-aware systems because it predefines the architectural style of the system based on available technologies.

4.2.1 Web 2.0

Web 2.0 was defined as "a trend economic, social and technologic that is building the base of new internet generation", such trend is becoming more mature and characterized by user participation. *Web 2.0* has been presented mainly as a cultural revolution: User is not only a costumer anymore, he is also an information provider.

In the last two years the Web 2.0 approach has revolutionised the way we use the web. On one hand, it enables the active participation of users with new content such as wiki pages, blogs or online multimedia tagged repositories such as Flickr or YouTube. On the other hand, Web 2.0 transforms the Web into an application enabling platform. Currently, many different organisations, publicise diverse types of functionality (such as maps, advertisement, weather info or photo repositories) accessible through REST or SOAP APIs. Thus, a popular trend is being to define new web applications by mixing and integrating the functionality offered by others.

Web 2.0 use network as a platform as it deliver or receive applications thoroughly via a browser. Users gets, manipulates and controlled the data on the site. Participatory architecture in which user can add or edit value to the application according to their requirement. A rich, interactive, user-friendly interface based on Ajax or similar frameworks.

After emerging of Web 2.0, it is being vastly used because of its wide range of variety and very attractive features. Descriptive list of Web 2.0 tools are endless even though we can say that the new generation of Internet approximately uses its tools. Web 2.0 tools include Weblogging, Wikis, Social networking, Podcasts, Feeds, Social bookmarking, and Cascading Style Sheet.

Web 2.0 has a complex and growing technology that includes server-software, content-syndication, messaging- protocols, standards-based browsers with plugins and extensions, and various client-applications. All these differ in functions and approaches but provide all the requirements beyond the expectation such as information- storage, creation, and dissemination capabilities.

A web 2.0 website may usually feature a number of following techniques:

- Rich Internet application techniques, optionally Ajax based.
- Cascading Style Sheet, CSS.
- Semantically valid XHTML markup and the use of Microformats.
- Organization and collection of data in RSS/Atom.
- Clean and meaningful URLs.
- Excessive use of folksonomies (in the form of tags or tag clouds).
- Use of wiki software either completely or partially (where partial use may grow to become the complete platform for the site) partially, e.g. the LAMP solution stack.
- XACML over SOAP for access control between organizations and domains.
- Blog publishing

- Mashups (A mix up of content and Audio usually from different musical style).
- REST or XML Webservice APIs.

In conclusion,

- A web2.0 website should be completely interactive and dynamic with a friendly user-interface based on the latest web2.0 technologies like AJAX.
- Web2.0 websites should deliver web based applications to Internet users and allowing them to make use of these applications through a web browser.
- A web2.0 website should implement social networking capabilities allowing users to interact with each other and create friend lists.
- A web2.0 website should be a democratic website where users will be able to add value by interacting with the web based application.
- Web2.0 websites should allow it's users to exercise various controls over the website data and content (adding/deleting/editing content).
- Web2.0 websites are build on participatory web based applications focusing basically on user experience and collaboration.

4.2.2 Widget

Widgets, also known as *gadgets,* are small software components easy integrated on more than one platform. Many names known in informatics (Google, Yahoo!, Microsoft, Apple, Opera) support their own framework for widget execution, that actually are incompatible to each other. A *widget* typically provides by a friendly graphic interface a simple way to access the services more used, for example the photo album on internet, email box or weather forecasting.

Widgets hide low-level details of sensing and ease application development due to their reusability. Because of the encapsulation in widgets it is possible to exchange widgets which provide the same kind of context data (e.g., exchange a radio frequency widget by a camera widget to collect location data). Widgets are usually controlled by some kind of a widget manager. The tightly coupled widget approach increases efficiency but is not robust to component failures.

Such software components provide applications with access to context information from their operating environment; they hide the complexity of the actual sensors used from the application; they abstract context information to suit the expected needs of applications; they provide reusable and customizable building blocks of context sensing. From the application's point of view, the widgets encapsulate context information and provide uniform operations to access context. For instance, a widget could be responsible for translating latitude and longitude to a city and a street.

4.2.3 REST (Representational State Transfer)

REST is a programming style for hypermedia distributed systems, as World Wide Web. Each resource should be reached by an universal syntax used to define a specific *hyperlink*, and also sharing a standard interface to transfer its own state to the client. Separation between client and server, message auto-description, and the stateless interaction included in *REST*, make easier the component implementation and also reduce the semantic complexity of container. All this improves the performance and scalability of the server.

4.2.4 Social Networking

Internet version regarded to the social nets is one of the most evolved ways related to net communication: Social relationship that each one has everyday, in a way more or less casual, it is possible by organizing an information “map”, and increasing the number of contacts. *Social network* was born in US and developed around three main topics: professional, friendship and love relations. Further evolution has been related to *Semantic Social Network*, that interconnect both people and *weblog*.

Applications related to social networks exploit available knowledge about the situation of the user (i.e., the context) adapting their functionalities in order to better serve the user's needs, e.g. to produce more focused and useful recommendations. Most considered context dimensions include physical location, time, mobile device capabilities, user preferences and network availability and capabilities. Social context information, i.e. the relationships existing among a group of tourists, is seldom taken into account.

The social dimension of tourism, however, is very important, because people often enjoy tourist activities in groups. Moreover, some tourist activities, like sharing photos or buying a souvenir for someone, involve both people actually being at the tourist location and people not being there, but sharing a social relationship with the tourists. Mobile applications are a very good source of social context data, coming in the form of address books, buddy lists and agendas; mobile devices such as smartphones and PDAs and their software are designed for fast and easy user interaction with these data. Moreover, location technologies increasingly available in mobile platforms and devices (e.g. GPS, GSM cell-based location, Wi-Fi, Bluetooth, RFIDs) can detect the peers in the vicinity of a tourist and match/filter them on the social network data available to the mobile environment, allowing on-the-fly group recognition.

For example, in one of many possible scenarios, a member of a group can ask his mobile context-aware guide for recommendations about nearby points of interests, telling it that the group context is to be considered. The guide initially brings together a list of the members of the group, either detecting them via location services, or by address book entry selection. Then the guide provides a set of recommended points of interest, taking into account the group context: the recommendation could be tailored to the recognized overall group type (e.g. “family”, “friends”, “couple”, “total strangers”, etc.), and/or tailored to the common interests of the group’s members. In another scenario, a person can ask his mobile guide for suggestions about a gift to buy in a location, selecting from a buddy list the person who will receive the gift. The guide gives then recommendations of gifts and shops where to buy them, taking into account the relationship between the person and the gift receiver (e.g. “girlfriend/boyfriend”, “parent”, “friend”, etc.).

4.2.5 Context Ontology

An ontology suggests a complete conceptual schema related to one specific domain; it is often regarded to a hierarchic data structure that contains all relevant entities, the relationship between them, rules, and the specific links of the domain. O*ntologies* are applied commonly in both the artificial intelligence field and the knowledge representation. Programs in PCs can use *ontology* for many purposes, for example inductive reasoning, classification and *problem solving* techniques, in order to facilitate communication and information exchange between different systems.

Web ontology based context model is defined to facilitate explicit context representation, semantic context sharing, and logic reasoning. The context ontology provides a shared vocabulary for describing basic concepts including users, locations, activities as well as a set of computational entities (i.e., services, applications and devices). It also defines a set of attributes (e.g., user’s status) and relationships (e.g., spatial containment between different locations) hold between these contextual entities.

Context-aware services are often interested in more than the current state of context, as temporal context plays an important role in analyzing historical information and predicting future trends. Capturing the temporal information requires the annotation of temporal information including both instantaneous time point of events, duration of activities and ordering between events, among other things.

In the recent years, research efforts regarded to service platforms, have provided architectural and programming support for context-awareness. This provision is usually done by hard-coding semantics into the underlying system implementation, obviously in this way platforms cannot easily evolve and interoperate. Actually, many contributions are mainly related to the use of *ontologies* and *Semantic Web* concepts to explicitly formalize the properties and structure of contextual information in order to guarantee common semantic understanding among different architectural components.

4.2.6 Rule Engine

Rule Engine is a software component of a rule-based system that assesses individual rules and determines their applicability in a specific case or situation. The purpose of such component is related to manage business logic, it can be defined as an advanced interpreter of “*rules*”. Each rule is composed of two parts, one condition and one action: when the condition (for example: “operation_cost > 1000 euros”) is verified, the action (for example: give discount 5%) is executed. *Rule engine* input is composed of both a set of rules and also a set of data. The output is defined from the information received as input, and also can execute modifications on the received data, new data and also side-effects (for example: sending an e-mail to the client).

Rules engines are most applicable to those situations where the required logic (e.g., rules) is sufficiently dynamic to cause the inclusion of such logic in the source code of software to be impractical, thus necessitating the need for an externalized rules engine facility. At the most basic level, a rules engine contains, a knowledge-base (consisting of rules represented in a computational format), and an inference or execution engine which can reason about incoming data based upon the contents of the knowledge-base in order to generate some form of output such as an instruction set or alert. An additional critical component of a rules engine is the knowledge engineering facility, which provides the capability to curate the contents of the knowledge-base on an automated, semi-automated, or manual basis.

Figure 20 contains basic rules engine architecture, including a knowledge-base and execution or inference engine. In this scenario, incoming data (1) is reasoned upon by the execution or inference engine using the rules encoded in the knowledge-base (2), in order to generate some form of output (3), such as instructions or alerts. An addition critical component of this architectural model is the knowledge engineering facility, which allows for the automated, semi-automated, or manual curation of the contents of the knowledge-base.

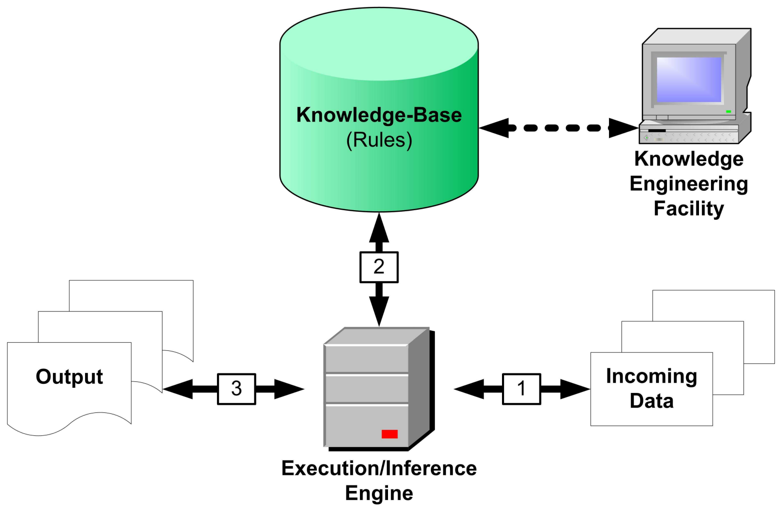


Figure 28: Rule engine architecture

The execution engine is the rules engine component that actually acts upon specific incoming data via the application of rules embodied in the knowledge-base, in order to generate some type of output. At the most basic level, the execution engine can be defined as a software component that takes as its input one or more elements of data, applies a set of rules to that data, and generates some form of output.

Rules engine technologies have been demonstrated in numerous biomedical and computer science domains. Examples of these varied types of applications include:

- Workflow automation for grid services integration
- Physical and conceptual workflow automation in distributed organizations
- Ontology anchored biological network discovery
- Critical value alerting in clinical environments
- Clinical guideline execution and delivery
- Semi-automated critiquing of clinical trials protocols.

A common thread throughout all of these example cases is the externalization of highly dynamic and/or complex business logic from application-level code, in order to ensure the efficient use and maintenance of such rules. Such an approach is consistent with prevailing best practices within contemporary software engineering, thus leading to the conclusion that the use of rules engine technologies in these and many other cases is both warranted and highly desirable.

Execution engines platforms commonly utilized for rule-based systems can be based on the type of rule representation:

- Production rules

  The most important component of a production rules system is the rule engine. The rules engines interpret the rules in the light of the facts that are available to them and select which of the rules to fire and in which order. Rule engines are needed anywhere it is not desirable for the logic to be hard coded into the application and has to be amenable for modification and customization for different needs and situations.

  Rule engines allow separation of the business logic from the application code and data, which reduces the software development and maintenance costs. Frequently, the rules change by the time the software developers understand the requirements, codify and develop the application. With the use of rule engines this premature obsolescence may be avoided.

  The use of rule engines facilitates declarative programming approach that allows separation of concerns while making possible a great flexibility at runtime. The need for keeping the business logic separate from the more static parts of the applications has already been identified in many places.

  The rule engines primarily have two components, Pattern Matching Algorithm and Agenda. The pattern matcher identifies if the facts match the patterns of any of the rules' LHS. Depending upon the facts and the patterns of the rules, several rules may be considered eligible for execution. These rules are aggregated by the agenda which after applying the conflict resolution strategy of the rule engine decides which, if any, of the rules are to be actually executed. The Rete Algorithm or some variant of it are the most common pattern matching algorithm commonly used, although other algorithms have also been described.

The execution of rules engines can occur in one of two ways: forward chaining and backward chaining. In forward chaining, the execution proceeds with assertion of facts and propagates until the conclusion is determined. In backward chaining, the conclusion is provided to the engine which then tries to identify sets of facts that will fulfill the conclusion. Thus forward chaining is data driven while backward chaining is goal driven. Most systems have forward chaining capabilities, while some systems also provide backward chaining, making them Hybrid Production Rules Systems. Backward chaining is an infrequently required feature and in implementations it is possible to create backward chaining kind of functionality even with a purely forward chaining engine.

Another characteristic of rule engines to consider is whether they maintain states or not. As designed, Rete is a stateful algorithm. That is, it can keep accumulating facts, discarding them or modifying them in response to the changes in the external world, over the life of an instance. However, for many situations, e.g., in web applications, it may be desirable to have a stateless rule engine instance where all the facts required for an execution of the engine are provided to it in one go.

A robust system that is capable of supporting needs of enterprises and network of organizations for production rules will also need rules servers and rules management systems besides rules engines, rule languages and editing facilities.

- Logical or axiomatic rules

  The domain of currently execution engine components employed by rules engine technologies can be largely divided into sub-domains based upon the type of rule representation utilized by their associated models.

- Hybrid rules

  In contrast to the preceding execution or inference engine components which are used to execute logical or axiomatic rule-bases, the current element is related to three execution engine environments that are used to support the delivery of higher-level rules represented using hybrid models: *BPEL Execution Engines, GLEE* and *SAGE.*

4.2.7 Recommendation Techniques

Recommendation systems are programs that try to forecast objects (film, songs, books, news, web pages...) to which an user could be interested in, based on some profile information. Often such techniques are implemented using *collaborative filtering* algorithms. Recommendation systems that belong to this family, combine information provided from user (for example: marks assigned to a set of objects), with data gathered implicitly (for example: objects acquired in past). Such information is compared to similar information that belongs to other users, with the purpose of generating a list of recommended objects.

This element provides data analyzing from the user profile in order to intelligently determine interesting services for him not only taking into account explicit interests stated by the user, but also implicit interest inferred from the analysis of all the data available. Through data accessed by some kind of User Information Management element, it will be possible performing correlations and collaborative filtering techniques, which aim to improve end user experience through recommendation of interesting services.

A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task. The key goal of context-aware systems is to provide *a* user with relevant information and/or services based on his current context. This goal matches with the goal of recommender systems. Recommender systems are systems that use opinions of a community of users to help individuals in that community more effectively identify content of interest from a potentially overwhelming set of choices.

Recommender systems do not only have to incorporate the opinions of other users, but may also use other methods, such as content-based reasoning. For this reason, recommender systems can be interpreted as systems capable of helping people to quickly and easily find their way through large amounts of information by determining what is of interest to a user.

Both context-aware systems and recommender systems are used to provide users with relevant information and/or services; the first based on the user's context; the second based on the user's interests. Therefore, the logical step is to combine these two systems. Context and interests can be used as hard or soft criteria in the selection of relevant services.

Hard criteria are used to limit the set of available information and/or services; those services that do not match a hard criterion are discarded from the result set. Soft criteria are used to order the set of selected services or to present a relevance score to the user for each selected service. For example, location, by far the most exploited context factor, can be used to select only the services within a certain distance from the user (hard criterion); location can also be used to decrease the predicted relevance of a service the further away that service is located from the user (soft criterion).

In recommender systems, the interests of a user are mostly used as soft criteria where the predicted level of interest is presented as a score, using for example a number of stars. However, interests can also be used as hard criteria by only selecting services that match the users' interests. In some applications, location is used as a hard criterion to select relevant services that are close to the user; the predicted interest of the user is used as a soft criterion, just like some other contextual factors.

4.2.8 Data Mining

Data Mining consists of gathering useful information executed in automatic or semi-automatic way from big amounts of data. This kind of activity is very important in many fields of scientific research, but also in other sectors (for example: marketing research). Techniques and algorithms of *Data Mining* have as main purpose analyzing many set of data in order to identify interesting regularities, known as patterns. Patterns can be the starting point for verifying new casual relations between phenomenon's; in general, it can be useful in statistic way in order to propose predictions about new data sets. A concept correlated to *Data Mining* is related to machine learning; pattern identification can be compared to casual learning previously unknown, such thing is applied to heuristic algorithms and artificial intelligence.

Due to dynamic nature of environment, data must be interpreted differently depending on situation (context). Context is a powerful concept, in computer-human interaction it can be mostly captured via explicit models of communication. However, implicit context factors (e.g., physical environmental conditions, location, time etc.) are normally ignored due to absence of knowledge base or appropriate model. Implicit context-aware factors could be used to interpret and enhance explicit user inputs and thereby affecting data mining results to deliver accurate and precise prediction results.

Data mining is a process that discovers patterns in data that may be used for valid predictions. Such process can filter useful and interesting context factors, produce accurate and precise prediction using those factors. Nowadays, huge volume of data is available, but this data-rich environment does not guaranty for information-rich environment. Different behaviors and functionalities of data mining are highly useful and required in generating information in dynamic, uncertain, and distributed environments. It is because such behaviors and capabilities can help to increase the various degrees of effectiveness and flexibility of data mining process.

Appendix A Acronyms Table

WAN	Wild Area Network
LAN	Local Area Network
SAN	Storage Area Network
DOM	Disk On Module
RDP	Remote Desktop Protocol
VNC	Virtual Network Computing
RFB	Remote Frame Buffer
ICA	Independent Computing Architecture
RDC	Remote Desktop Connection
VMWare	Virtual Machine softWare
OpenVZ	Operating system-level VirtualiZation
AJAX	Asynchronous JavaScript and XML
XML	Extensible Markup Language
HTML	HyperText Markup Language
XHTML	Extensible HyperText Markup Language
JSON	JavaScript Object Notation
LAMP	Linux Apache MySQL Perl Python PHP
MySQL	Multi-user Structured Query Language
DHTML	Dynamic HyperText Markup Language
GPL	General Public License
IA	Intel Architecture
CSS	Cascading Style Sheets
LDAP	Lightweight Directory Access Protocol
DHCP	Dynamic Host Configuration Protocol
PXE	Preboot eXecution Environment
TFTP	Trivial File Transfer Protocol
NFS	Network File System

VMM	Virtual Machine Monitor
RIA	Rich Internet Application
J2ME	Java 2 Platform Micro Edition
CLDC	Connected Limited Device Configuration
GUI	Graphic User Interface

Figures Index

Figure 1: Communications in emergency situations. 9
Figure 2: Thin clients in a client-server context. 11
Figure 3: X Windows System. 15
Figure 4: Remote Desktop interaction. 16
Figure 5: VNC architecture. 17
Figure 6: Remote Desktop Connection. 19
Figure 7: Native Hypervisor. 21
Figure 8: Hosted Hypervisor. 22
Figure 9: VMware ESX Architecture 23
Figure 10: XEN Architecture. 25
Figure 11: JavaFX Platform. 32
Figure 12: Application Streaming Example 37
Figure 13: Softgrid Application Packaging 38
Figure 14: Softgrid Application Streaming 39
Figure 15: AppStream Application Streaming 42
Figure 16: Devices Interaction in application streaming. 46
Figure 17: Representation of IMS network with multiple access technology 53
Figure 18: Target architecture 54
Figure 19: IMS core overview 55
Figure 20: Use of SIP and Diameter protocol in IMS 60
Figure 21: Example SIP INVITE 61
Figure 22: Example SIP RINGING 61
Figure 23: Example SIP 200 OK 61
Figure 24: Example SIP ACK 61
Figure 17: J2ME architecure. 66
Figure 18: Apple iPhone. 67
Figure 19: Android architecture. 70
Figure 20: Rule engine architecture 87

Bibliography

[1] CEFRIEL, D1.1 *"Application Scenarios and User-Provider Requirements"*, PICO project

[2] CEFRIEL, D1.2 *"Enabling Technologies"*, PICO project

[3] www.wikipedia.com

[4] www.appstream.com

[5] www.endeavors.com

[6] http://ieeexplore.ieee.org

[7] http://portal.acm.org

[8] http://springerlink.metapress.com

[9] www.microsoft.com/systemcenter/softgrid/default.mspx

[10] http://openthinclient.org/home

[11] http://www.realvnc.com/index.html

[12] http://www.citrix.com/lang/English/home.asp

[13] www.vmware.com

[14] http://www.xen.org/

[15] http://openvz.org/

[16] http://www.adobe.com/products/flash/

[17] http://silverlight.net/

[18] http://www.sun.com/software/javafx/index.jsp