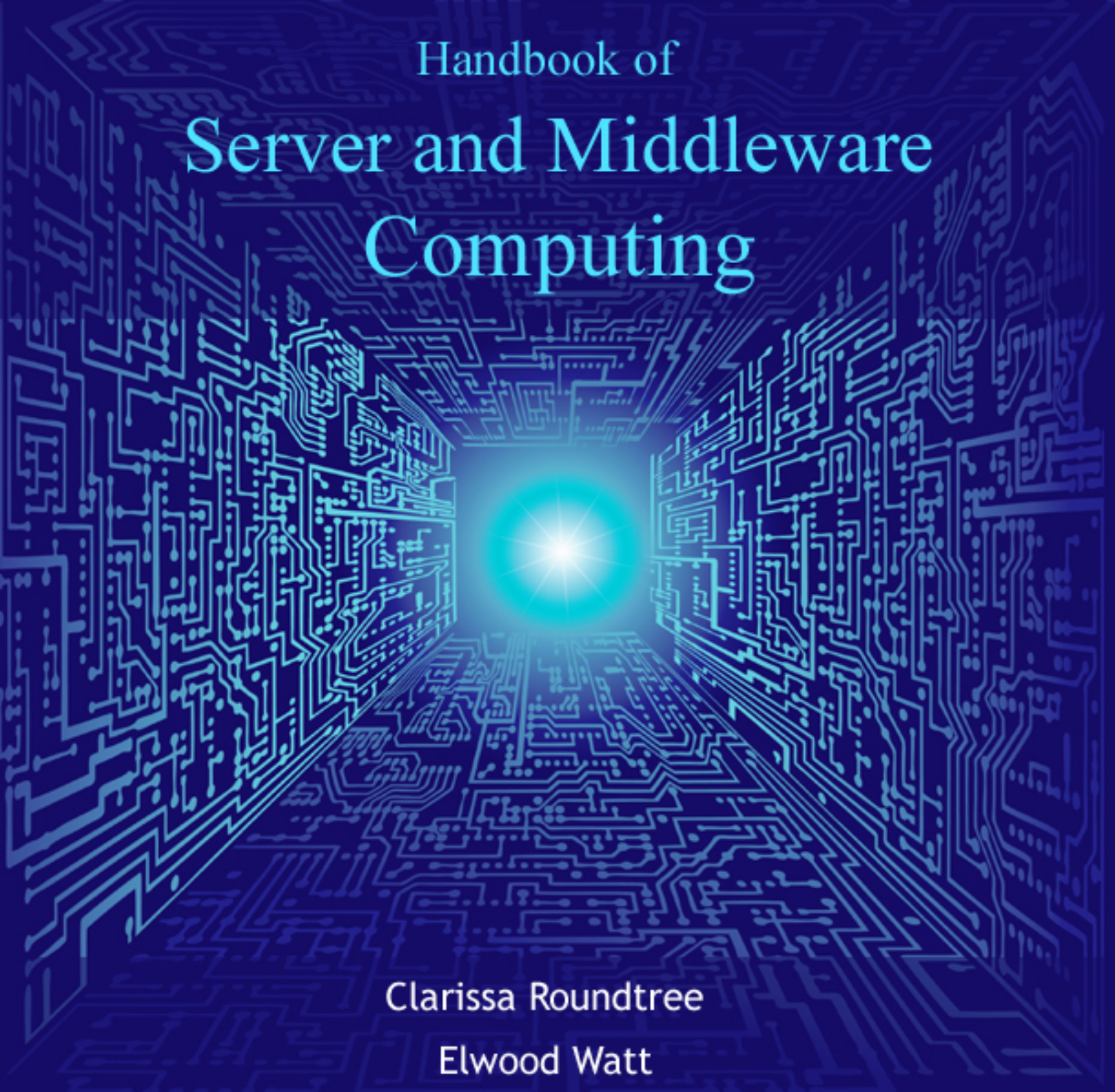

Handbook of
Server and Middleware
Computing
Clarissa Roundtree
Elwood Watt

First Edition, 2012

ISBN 978-81-323-0928-4



*Published by:*
**Academic Studio**
4735/22 Prakashdeep Bldg,
Ansari Road, Darya Ganj,
Delhi - 110002
Email: info@wtbooks.com

# Table of Contents

## Chapter-1

# Server

Several servers mounted on a rack, connected to a display

A rack-mountable server

In computing, the term **server** is used to refer to one of the following:

- a computer program running as a service, to serve the needs or requests of other programs (referred to in this context as "clients") which may or may not be running on the same computer.
- a physical computer dedicated to running one or more such services, to serve the needs of programs running on other computers on the same network.
- a software/hardware system (i.e. a software service running on a dedicated computer) such as a database server, file server, mail server, or print server.

In computer networking, a **server** is a program that operates as a socket listener. The term **server** is also often generalized to describe a host that is deployed to execute one or more such programs.

A **server computer** is a computer, or series of computers, that link other computers or electronic devices together. They often provide essential services across a network, either to private users inside a large organization or to public users via the internet. For example, when you enter a query in a search engine, the query is sent from your computer over the internet to the servers that store all the relevant web pages. The results are sent back by the server to your computer.

Many servers have dedicated functionality such as web servers, print servers, and database servers. **Enterprise servers** are servers that are used in a business context.

## *Usage*

Servers provide essential services across a network, either to private users inside a large organization or to public users via the Internet. For example, when you enter a query in a search engine, the query is sent from your computer over the internet to the servers that store all the relevant web pages. The results are sent back by the server to your computer.

The term *server* is used quite broadly in information technology. Despite the many server-branded products available (such as server versions of hardware, software or operating systems), in theory any computerised process that shares a resource to one or more client processes is a server. To illustrate this, take the common example of file sharing. While the existence of files on a machine does not classify it as a server, the mechanism which shares these files to clients by the operating system is the server.

Similarly, consider a web server application (such as the multiplatform "Apache HTTP Server"). This web server software can be *run* on any capable computer. For example, while a laptop or personal computer is not typically known as a server, they can in these situations fulfill the role of one, and hence be labelled as one. It is in this case that the machine's purpose as a web server classifies it in general as a server.

In the hardware sense, the word *server* typically designates computer models intended for hosting software applications under the heavy demand of a network environment. In this client–server configuration one or more machines, either a computer or a computer appliance, share information with each other with one acting as a host for the other.

While nearly any personal computer is capable of acting as a network server, a dedicated server will contain features making it more suitable for production environments. These features may include a faster CPU, increased high-performance RAM, and typically more than one large hard drive. More obvious distinctions include marked redundancy in power supplies, network connections, and even the servers themselves.

Between the 1990s and 2000s an increase in the use of *dedicated hardware* saw the advent of self-contained **server appliances**. One well-known product is the Google Search Appliance, a unit that combines hardware and software in an out-of-the-box packaging. Simpler examples of such appliances include switches, routers, gateways, and print server, all of which are available in a near plug-and-play configuration.

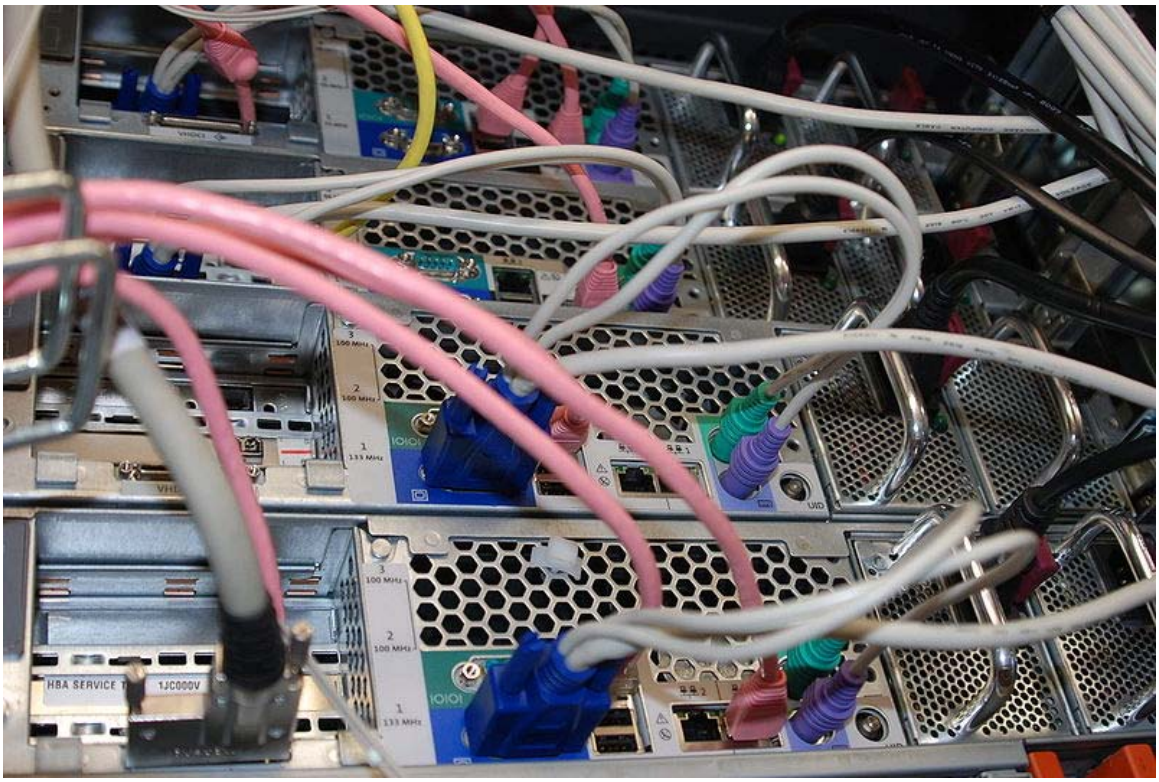
Modern operating systems such as Microsoft Windows or Linux distributions rightfully seem to be designed with a client–server architecture in mind. These operating systems attempt to abstract hardware, allowing a wide variety of software to work with components of the computer. In a sense, the operating system can be seen as *serving* hardware to the software, which in all but low-level programming languages must interact using an API.

These operating systems may be able to run programs in the background called either services or daemons. Such programs may wait in a sleep state for their necessity to

become apparent, such as the aforementioned *Apache HTTP Server* software. Since any software that provides services can be *called* a server, modern personal computers can be seen as a forest of servers and clients operating in parallel.

The Internet itself is also a forest of servers and clients. Merely requesting a web page from a few kilometers away involves satisfying a stack of protocols that involve many examples of hardware and software servers. The least of these are the routers, modems, domain name servers, and various other servers necessary to provide us the world wide web.

## *Server hardware*



A server rack seen from the rear

Hardware requirements for servers vary, depending on the server application. Absolute CPU speed is not usually as critical to a server as it is to a desktop machine. Servers' duties to provide service to many users over a network lead to different requirements like fast network connections and high I/O throughput. Since servers are usually accessed over a network, they may run in headless mode without a monitor or input device. Processes that are not needed for the server's function are not used. Many servers do not have a graphical user interface (GUI) as it is unnecessary and consumes resources that could be allocated elsewhere. Similarly, audio and USB interfaces may be omitted.

Servers often run for long periods without interruption and availability must often be very high, making hardware reliability and durability extremely important. Although servers

can be built from commodity computer parts, mission-critical enterprise servers are ideally very fault tolerant and use specialized hardware with low failure rates in order to maximize uptime, for even a short-term failure can cost more than purchasing and installing the system. For example, it may take only a few minutes of down time at a national stock exchange to justify the expense of entirely replacing the system with something more reliable. Servers may incorporate faster, higher-capacity hard drives, larger computer fans or water cooling to help remove heat, and uninterruptible power supplies that ensure the servers continue to function in the event of a power failure. These components offer higher performance and reliability at a correspondingly higher price. Hardware redundancy—installing more than one instance of modules such as power supplies and hard disks arranged so that if one fails another is automatically available—is widely used. ECC memory devices that detect and correct errors are used; non-ECC memory is more likely to cause data corruption.

To increase reliability, most of the servers use memory with error detection and correction, redundant disks, redundant power supplies and so on. Such components are also frequently hot swappable, allowing to replace them on the running server without shutting it down. To prevent overheating, servers often have more powerful fans. As servers are usually administered by qualified engineers, their operating systems are also more tuned for stability and performance than for user friendliness and ease of use, Linux taking noticeably larger percentage than for desktop computers.

As servers need stable power supply, good Internet access, increased security and are also noisy, it is usual to store them in dedicated server centers or special rooms. This requires to reduce power consumption as extra energy used generates more heat and the temperature in the room could exceed the acceptable limits. Normally server rooms are equipped with air conditioning devices. Server casings are usually flat and wide, adapted to store many devices next to each other in server rack. Unlike ordinary computers, servers usually can be configured, powered up and down or rebooted remotely, using out-of-band management.

Many servers take a long time for the hardware to start up and load the operating system. Servers often do extensive pre-boot memory testing and verification and startup of remote management services. The hard drive controllers then start up banks of drives sequentially, rather than all at once, so as not to overload the power supply with startup surges, and afterwards they initiate RAID system pre-checks for correct operation of redundancy. It is common for a machine to take several minutes to start up, but it may not need restarting for months or years.

## *Server operating systems*

Server-oriented operating systems tend to have certain features in common that make them more suitable for the server environment, such as

- GUI not available or optional

- ability to reconfigure and update both hardware and software to some extent without restart,
- advanced backup facilities to permit regular and frequent online backups of critical data,
- transparent data transfer between different volumes or devices,
- flexible and advanced networking capabilities,
- automation capabilities such as daemons in UNIX and services in Windows, and
- tight system security, with advanced user, resource, data, and memory protection.

Server-oriented operating systems can, in many cases, interact with hardware sensors to detect conditions such as overheating, processor and disk failure, and consequently alert an operator or take remedial measures itself.

Because servers must supply a restricted range of services to perhaps many users while a desktop computer must carry out a wide range of functions required by its user, the requirements of an operating system for a server are different from those of a desktop machine. While it is possible for an operating system to make a machine both provide services and respond quickly to the requirements of a user, it is usual to use different operating systems on servers and desktop machines. Some operating systems are supplied in both server and desktop versions with similar user interface.

The desktop versions of the Windows and Mac OS X operating systems are deployed on a minority of servers, as are some proprietary mainframe operating systems, such as z/OS. The dominant operating systems among servers are UNIX-based or open source kernel distributions, such as Linux (the kernel).

The rise of the microprocessor-based server was facilitated by the development of Unix to run on the x86 microprocessor architecture. The Microsoft Windows family of operating systems also runs on x86 hardware, and since Windows NT have been available in versions suitable for server use.

While the role of server and desktop operating systems remains distinct, improvements in the reliability of both hardware and operating systems have blurred the distinction between the two classes. Today, many desktop and server operating systems share similar code bases, differing mostly in configuration. The shift towards web applications and middleware platforms has also lessened the demand for specialist application servers.

## *Servers on the Internet*

Almost the entire structure of the Internet is based upon a client–server model. High-level root nameservers, DNS servers, and routers direct the traffic on the internet. There are millions of servers connected to the Internet, running continuously throughout the world.

- World Wide Web
- Domain Name System
- E-mail

- FTP file transfer
- Chat and instant messaging
- Voice communication
- Streaming audio and video
- Online gaming
- Database servers

Virtually every action taken by an ordinary Internet user requires one or more interactions with one or more servers.

There are also technologies that operate on an inter-server level. Other services do not use dedicated servers; for example peer-to-peer file sharing, some implementations of telephony (e.g. Skype), and supplying television programs to several users (e.g. Kontiki, SlingBox).

## *Energy consumption of servers*

In 2010, servers were responsible for 2.5% of energy consumption in the United States. A further 2.5% of United States energy consumption was used by cooling systems required to cool the servers. It was estimated in 2010, that if trends continued, by 2020, servers would use more of the world's energy than air travel.

## Chapter-2

# Web Server

The inside and front of a Dell PowerEdge Web server

A **web server** can be referred to as either the hardware (the computer) or the software (the computer application) that helps to deliver content that can be accessed through the Internet.

The most common use of Web servers is to host Web sites but there are other uses like data storage or for running enterprise applications.

## *Overview*

The primary function of a web server is to deliver web pages on the request to clients. This means delivery of HTML documents and any additional content that may be included by a document, such as images, style sheets and JavaScripts.

A client, commonly a web browser or web crawler, initiates communication by making a request for a specific resource using HTTP and the server responds with the content of that resource or an error message if unable to do so. The resource is typically a real file on the server's secondary memory, but this is not necessarily the case and depends on how the web server is implemented.

While the primary function is to serve content, a full implementation of HTTP also includes ways of receiving content from clients. This feature is used for submitting web forms, including uploading of files.

Many generic web servers also support server-side scripting, e.g., Apache HTTP Server and PHP. This means that the behaviour of the web server can be scripted in separate files, while the actual server software remains unchanged. Usually, this function is used to create HTML documents "on-the-fly" as opposed to returning fixed documents. This is referred to as dynamic and static content respectively. The former is primarily used for retrieving and/or modifying information from databases. The latter is, however, typically much faster and more easily cached.

Web servers are not always used for serving the world wide web. They can also be found embedded in devices such as printers, routers, webcams and serving only a local network. The web server may then be used as a part of a system for monitoring and/or administrating the device in question. This usually means that no additional software has to be installed on the client computer, since only a web browser is required (which now is included with most operating systems).

## *History of web servers*


The world's first web server.

In 1989 Tim Berners-Lee proposed to his employer CERN (European Organization for Nuclear Research) a new project, which had the goal of easing the exchange of information between scientists by using a hypertext system. As a result of the implementation of this project, in 1990 Berners-Lee wrote two programs:

- a browser called WorldWideWeb;
- the world's first web server, later known as CERN httpd, which ran on NeXTSTEP.

Between 1991 and 1994 the simplicity and effectiveness of early technologies used to surf and exchange data through the World Wide Web helped to port them to many different operating systems and spread their use among lots of different social groups of people, first in scientific organizations, then in universities and finally in industry.

In 1994 Tim Berners-Lee decided to constitute the World Wide Web Consortium (W3C) to regulate the further development of the many technologies involved (HTTP, HTML, etc.) through a standardization process.

## Common features

1. **Virtual hosting** to serve many Web sites using one IP address.
2. **Large file support** to be able to serve files whose size is greater than 2 GB on 32 bit OS.
3. **Bandwidth throttling** to limit the speed of responses in order to not saturate the network and to be able to serve more clients.
4. **Server-side scripting** to generate dynamic Web pages, still keeping Web server and Web site implementations separate from each other.

## Path translation

Web servers are able to map the path component of a Uniform Resource Locator (**URL**) into:

- a local file system resource (for static requests);
- an internal or external program name (for dynamic requests).

For a *static request* the URL path specified by the client is relative to the Web server's root directory.

Consider the following URL as it would be requested by a client:

```
http://www.example.com/path/file.html
```

The client's user agent will translate it into a connection to `www.example.com` with the following HTTP 1.1 request:

```
GET /path/file.html HTTP/1.1
Host: www.example.com
```

The Web server on `www.example.com` will append the given path to the path of its root directory. On an Apache server, this is commonly `/home/www` (On Unix machines, usually `/var/www`). The result is the local file system resource:

```
/home/www/path/file.html
```

The Web server then reads the file, if it exists and sends a response to the client's Web browser. The response will describe the content of the file and contain the file itself or an error message will return saying that the file does not exist or is unavailable.

## Load limits

A Web server (program) has defined load limits, because it can handle only a limited number of concurrent client connections (usually between 2 and 80,000, by default

between 500 and 1,000) per IP address (and TCP port) and it can serve only a certain maximum number of requests per second depending on:

- its own settings;
- the HTTP request type;
- content origin (static or dynamic);
- the fact that the served content is or is not cached;
- the hardware and software limitations of the OS where it is working;

When a Web server is near to or over its limits, it becomes unresponsive.

### *Kernel-mode and user-mode Web servers*

A Web server can be either implemented into the OS kernel, or in user space (like other regular applications).

An in-kernel Web server (like TUX on GNU/Linux or Microsoft IIS on Windows) will usually work faster, because, as part of the system, it can directly use all the hardware resources it needs, such as non-paged memory, CPU time-slices, network adapters, or buffers.

Web servers that run in user-mode have to ask the system the permission to use more memory or more CPU resources. Not only do these requests to the kernel take time, but they are not always satisfied because the system reserves resources for its own usage and has the responsibility to share hardware resources with all the other running applications.

Also, applications cannot access the system's internal buffers, which causes useless buffer copies that create another handicap for user-mode web servers. As a consequence, the only way for a user-mode web server to match kernel-mode performance is to raise the quality of its code to much higher standards, similar to that of the code used in web servers that run in the kernel. This is a significant issue under Windows, where the user-mode overhead is about six times greater than that under Linux.

### Overload causes

At any time web servers can be overloaded because of:

- **Too much legitimate web traffic.** Thousands or even millions of clients connecting to the web site in a short interval, e.g., Slashdot effect;
- **Distributed Denial of Service** attacks;
- **Computer worms** that sometimes cause abnormal traffic because of millions of infected computers (not coordinated among them);
- **XSS viruses** can cause high traffic because of millions of infected browsers and/or Web servers;
- **Internet bots.** Traffic not filtered/limited on large web sites with very few resources (bandwidth, etc.);

- **Internet (network) slowdowns**, so that client requests are served more slowly and the number of connections increases so much that server limits are reached;
- **Web servers (computers) partial unavailability.** This can happen because of required or urgent maintenance or upgrade, hardware or software failures, back-end (e.g., database) failures, etc.; in these cases the remaining web servers get too much traffic and become overloaded.

## Overload symptoms

The symptoms of an overloaded Web server are:

- requests are served with (possibly long) delays (from 1 second to a few hundred seconds);
- 500, 502, 503, 504 HTTP errors are returned to clients (sometimes also unrelated 404 error or even 408 error may be returned);
- TCP connections are refused or reset (interrupted) before any content is sent to clients;
- in very rare cases, only partial contents are sent (but this behavior may well be considered a bug, even if it usually depends on unavailable system resources).

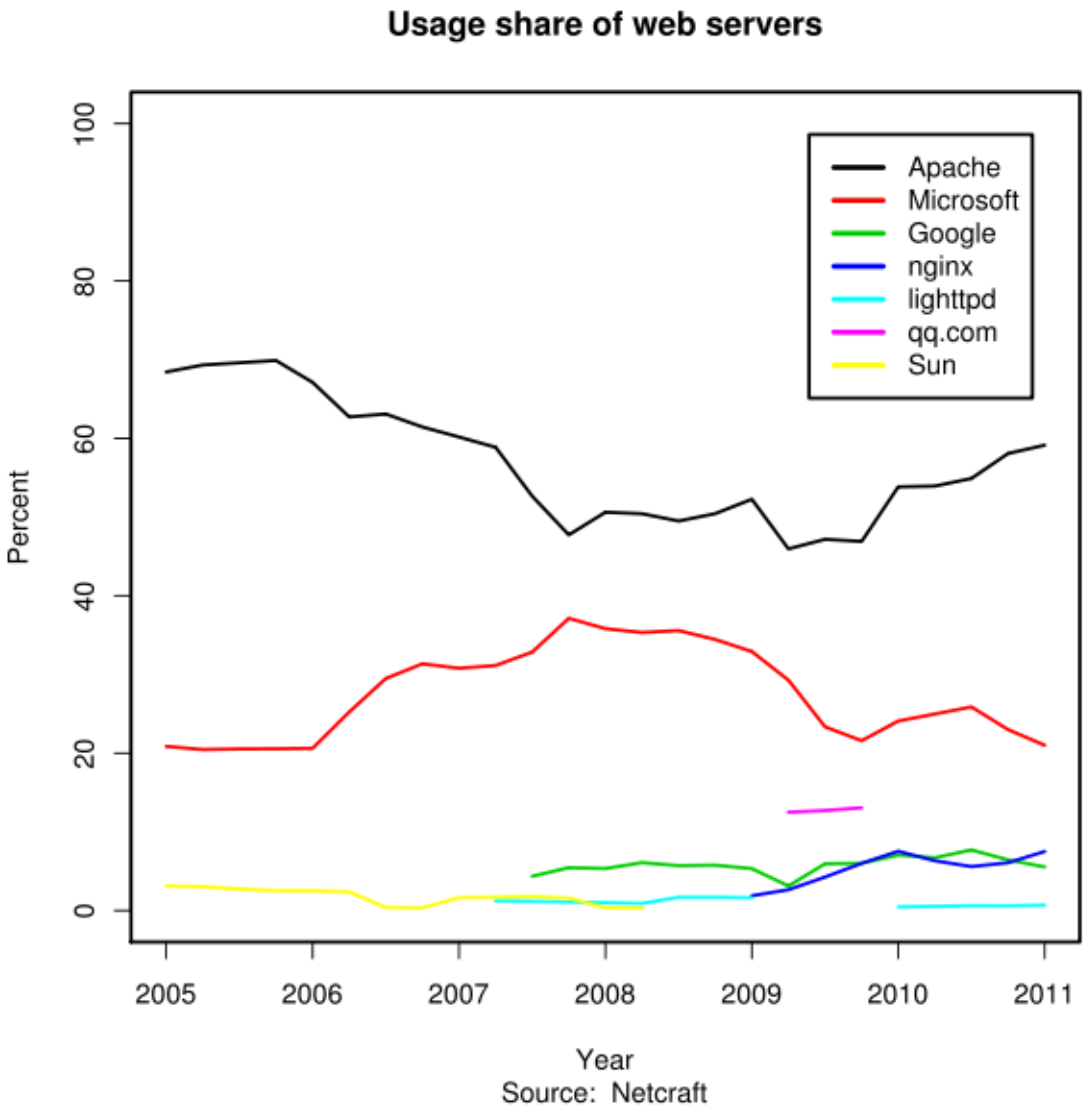
## Anti-overload techniques

To partially overcome above load limits and to prevent overload, most popular Web sites use common techniques like:

- **managing network traffic**, by using:
  - **Firewalls** to block unwanted traffic coming from bad IP sources or having bad patterns;
  - **HTTP traffic managers** to drop, redirect or rewrite requests having bad HTTP patterns;
  - **Bandwidth management** and **traffic shaping**, in order to smooth down peaks in network usage;
- deploying **Web cache** techniques;
- using different domain names to serve different (static and dynamic) content by separate Web servers, i.e.:
- using different domain names and/or computers to separate big files from small and medium sized files; the idea is to be able to fully cache small and medium sized files and to efficiently serve big or huge (over 10 - 1000 MB) files by using different settings;
- using many Web servers (programs) per computer, each one bound to its own network card and IP address;
- using many Web servers (computers) that are grouped together so that they act or are seen as one big Web server;
- adding more hardware resources (i.e. RAM, disks) to each computer;
- tuning OS parameters for hardware capabilities and usage;
- using more efficient computer programs for Web servers, etc.;

- using other workarounds, especially if dynamic content is involved.

## *Market structure*




Market share of major Web servers

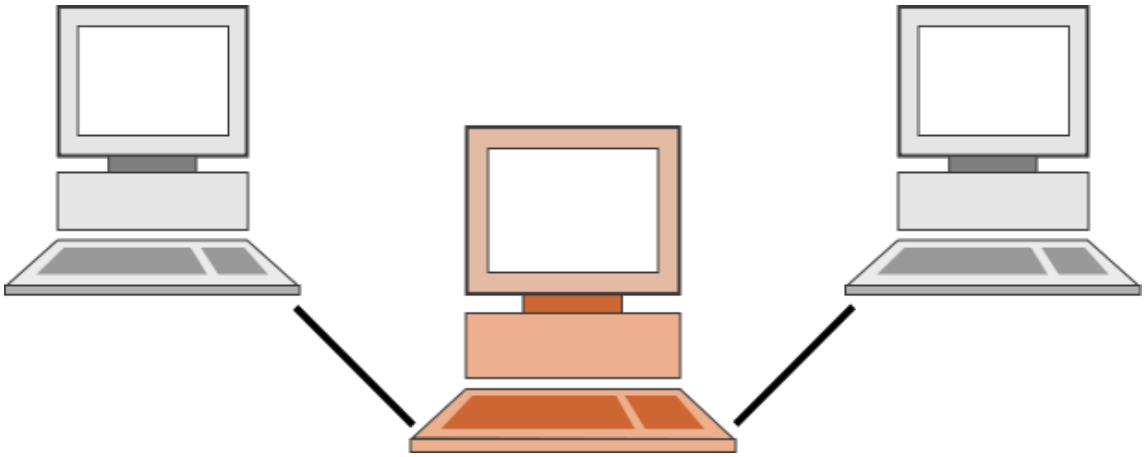
Below is the most recent statistics of the market share of the top web servers on the internet by Netcraft survey in March 2011.

| Vendor | Product | Web Sites Hosted | Percent |
|---|---|---|---|
| Apache | Apache | 179,720,332 | 60.31% |
| Microsoft | IIS | 57,644,692 | 19.34% |
| Igor Sysoev | nginx | 22,806,060 | 7.65% |
| Google | GWS | 15,161,530 | 5.09% |

| | | | |
|---|---|---|---|
| lighttpd | lighttpd | 1,796,471 | 0.60% |

## Chapter-3

# Proxy Server



Schematic representation of a proxy server, where the computer in the middle acts as the proxy server between the other two.

In computer networks, a **proxy server** is a server (a computer system or an application) that acts as an intermediary for requests from clients seeking resources from other servers. A client connects to the proxy server, requesting some service, such as a file, connection, web page, or other resource, available from a different server. The proxy server evaluates the request according to its filtering rules. For example, it may filter traffic by IP address or protocol. If the request is validated by the filter, the proxy provides the resource by connecting to the relevant server and requesting the service on behalf of the client. A proxy server may optionally alter the client's request or the server's response, and sometimes it may serve the request without contacting the specified server. In this case, it 'caches' responses from the remote server, and returns subsequent requests for the same content directly.

Most proxies are a **web proxy**, allowing access to content on the World Wide Web.

A proxy server has a large variety of potential purposes, including:

- To keep machines behind it anonymous (mainly for security).
- To speed up access to resources (using caching). Web proxies are commonly used to cache web pages from a web server.

- To apply access policy to network services or content, e.g. to block undesired sites.
- To log / audit usage, i.e. to provide company employee Internet usage reporting.
- To bypass security/ parental controls.
- To scan transmitted content for malware before delivery.
- To scan outbound content, e.g., for data leak protection.
- To circumvent regional restrictions.

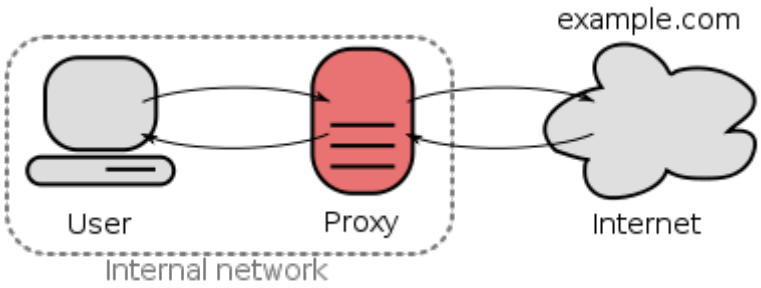
A proxy server that passes requests and replies unmodified is usually called a gateway or sometimes *tunneling proxy*.

A proxy server can be placed in the user's local computer or at various points between the user and the destination servers on the Internet.

A reverse proxy is (usually) an Internet-facing proxy used as a front-end to control and protect access to a server on a private network, commonly also performing tasks such as load-balancing, authentication, decryption or caching.

## *Types of proxy*

### Forward proxies

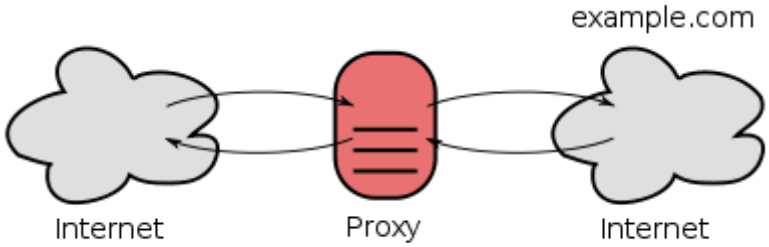



A forward proxy taking requests from an internal network and forwarding them to the Internet.

Forward proxies are proxies where the client server names the target server to connect to. Forward proxies are able to retrieve from a wide range of sources (in most cases anywhere on the Internet).

The terms "forward proxy" and "forwarding proxy" are a general description of behaviour (forwarding traffic) and thus ambiguous. Except for Reverse proxy, the types of proxies described here are more specialized sub-types of the general forward proxy concept.

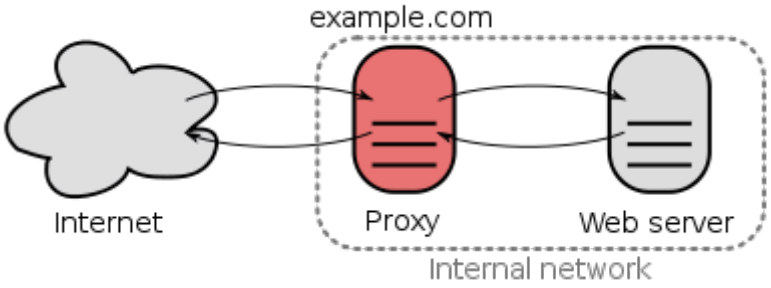
## Open proxies




An open proxy forwarding requests from and to anywhere on the Internet.

An open proxy is a forward proxy server that is accessible by any Internet user. Gordon Lyon estimates there are "hundreds of thousands" of open proxies on the Internet. An *anonymous open proxy* allows users to conceal their IP address while browsing the Web or using other Internet services.

## Reverse proxies




A reverse proxy taking requests from the Internet and forwarding them to servers in an internal network. Those making requests connect to the proxy and may not be aware of the internal network.

A **reverse proxy** is a proxy server that appears to clients to be an ordinary server. Requests are forwarded to one or more origin servers which handle the request. The response is returned as if it came directly from the proxy server.

Reverse proxies are installed in the neighborhood of one or more web servers. All traffic coming from the Internet and with a destination of one of the web servers goes through the proxy server. The use of "reverse" originates in its counterpart "forward proxy" since the reverse proxy sits closer to the web server and serves only a restricted set of websites.

There are several reasons for installing reverse proxy servers:

- Encryption / SSL acceleration: when secure web sites are created, the SSL encryption is often not done by the web server itself, but by a reverse proxy that is equipped with SSL acceleration hardware. Furthermore, a host can provide a

single "SSL proxy" to provide SSL encryption for an arbitrary number of hosts; removing the need for a separate SSL Server Certificate for each host, with the downside that all hosts behind the SSL proxy have to share a common DNS name or IP address for SSL connections. This problem can partly be overcome by using the *SubjectAltName* feature of X.509 certificates.
- Load balancing: the reverse proxy can distribute the load to several web servers, each web server serving its own application area. In such a case, the reverse proxy may need to rewrite the URLs in each web page (translation from externally known URLs to the internal locations).
- Serve/cache static content: A reverse proxy can offload the web servers by caching static content like pictures and other static graphical content.
- Compression: the proxy server can optimize and compress the content to speed up the load time.
- Spoon feeding: reduces resource usage caused by slow clients on the web servers by caching the content the web server sent and slowly "spoon feeding" it to the client. This especially benefits dynamically generated pages.
- Security: the proxy server is an additional layer of defense and can protect against some OS and WebServer specific attacks. However, it does not provide any protection to attacks against the web application or service itself, which is generally considered the larger threat.
- Extranet Publishing: a reverse proxy server facing the Internet can be used to communicate to a firewalled server internal to an organization, providing extranet access to some functions while keeping the servers behind the firewalls. If used in this way, security measures should be considered to protect the rest of your infrastructure in case this server is compromised, as its web application is exposed to attack from the Internet.

## *Uses of proxy servers*

### Filtering

A content-filtering web proxy server provides administrative control over the content that may be relayed through the proxy. It is commonly used in both commercial and non-commercial organizations (especially schools) to ensure that Internet usage conforms to acceptable use policy. In some cases users can circumvent the proxy, since there are services designed to proxy information from a filtered website through a non filtered site to allow it through the user's proxy.

A content filtering proxy will often support user authentication, to control web access. It also usually produces logs, either to give detailed information about the URLs accessed by specific users, or to monitor bandwidth usage statistics. It may also communicate to daemon-based and/or ICAP-based antivirus software to provide security against virus and other malware by scanning incoming content in real time before it enters the network.

Many work places, schools, and colleges restrict the web sites and online services that are made available in their buildings. This is done either with a specialized proxy, called a

content filter (both commercial and free products are available), or by using a cache-extension protocol such as ICAP, that allows plug-in extensions to an open caching architecture.

Some common methods used for content filtering include: URL or DNS blacklists, URL regex filtering, MIME filtering, or content keyword filtering. Some products have been known to employ content analysis techniques to look for traits commonly used by certain types of content providers.

Requests made to the open internet must first pass through an outbound proxy filter. The web-filtering company provides a database of URL patterns (regular expressions) with associated content attributes. This database is updated weekly by site-wide subscription, much like a virus filter subscription. The administrator instructs the web filter to ban broad classes of content (such as sports, pornography, online shopping, gambling, or social networking). Requests that match a banned URL pattern are rejected immediately.

Assuming the requested URL is acceptable, the content is then fetched by the proxy. At this point a dynamic filter may be applied on the return path. For example, JPEG files could be blocked based on fleshtone matches, or language filters could dynamically detect unwanted language. If the content is rejected then an HTTP fetch error is returned and nothing is cached.

Most web filtering companies use an internet-wide crawling robot that assesses the likelihood that a content is a certain type (e.g. "This content is 70% chance of porn, 40% chance of sports, and 30% chance of news" could be the outcome for one web page). The resultant database is then corrected by manual labor based on complaints or known flaws in the content-matching algorithms.

Web filtering proxies are not able to peer inside secure sockets HTTP transactions, assuming the chain-of-trust of SSL/TLS has not been tampered with. As a result, users wanting to bypass web filtering will typically search the internet for an open and anonymous HTTPS transparent proxy. They will then program their browser to proxy all requests through the web filter to this anonymous proxy. Those requests will be encrypted with https. The web filter cannot distinguish these transactions from, say, a legitimate access to a financial website. Thus, content filters are only effective against unsophisticated users.

As mentioned above, the SSL/TLS chain-of-trust does rely on trusted root certificate authorities; in a workplace setting where the client is managed by the organization, trust might be granted to a root certificate whose private key is known to the proxy. Concretely, a root certificate generated by the proxy is installed into the browser CA list by IT staff. In such scenarios, proxy analysis of the contents of a SSL/TLS transaction becomes possible. The proxy is effectively operating a man-in-the-middle attack, allowed by the client's trust of a root certificate the proxy owns.

A special case of web proxies is "CGI proxies". These are web sites that allow a user to access a site through them. They generally use PHP or CGI to implement the proxy functionality. These types of proxies are frequently used to gain access to web sites blocked by corporate or school proxies. Since they also hide the user's own IP address from the web sites they access through the proxy, they are sometimes also used to gain a degree of anonymity, called "Proxy Avoidance".

## Caching

A **caching proxy** server accelerates service requests by retrieving content saved from a previous request made by the same client or even other clients. Caching proxies keep local copies of frequently requested resources, allowing large organizations to significantly reduce their upstream bandwidth usage and costs, while significantly increasing performance. Most ISPs and large businesses have a caching proxy. Caching proxies were the first kind of proxy server.

Some poorly-implemented caching proxies have had downsides (e.g., an inability to use user authentication). Some problems are described in RFC 3143 (Known HTTP Proxy/Caching Problems).

Another important use of the proxy server is to reduce the hardware cost. An organization may have many systems on the same network or under control of a single server, prohibiting the possibility of an individual connection to the Internet for each system. In such a case, the individual systems can be connected to one proxy server, and the proxy server connected to the main server.

## Bypassing filters and censorship

If the destination server filters content based on the origin of the request, the use of a proxy can remove this filter. For example, a server using IP-based geolocation to restrict its service to a certain country can be accessed using a proxy located in that country to access the service.

Likewise, a badly configured proxy can provide access to a network otherwise isolated from the Internet.

## Logging and eavesdropping

Proxies can be installed in order to eavesdrop upon the data-flow between client machines and the web. All content sent or accessed – including passwords submitted and cookies used – can be captured and analyzed by the proxy operator. For this reason, passwords to online services (such as webmail and banking) should always be exchanged over a cryptographically secured connection, such as SSL.

By chaining proxies which do not reveal data about the original requester, it is possible to obfuscate activities from the eyes of the user's destination. However, more traces will be

left on the intermediate hops, which could be used or offered up to trace the user's activities. If the policies and administrators of these other proxies are unknown, the user may fall victim to a false sense of security just because those details are out of sight and mind.

In what is more of an inconvenience than a risk, proxy users may find themselves being blocked from certain Web sites, as numerous forums and Web sites block IP addresses from proxies known to have spammed or trolled the site. Proxy bouncing can be used to maintain your privacy.

### Gateways to private networks

Proxy servers can perform a role similar to a network switch in linking two networks.

### Accessing services anonymously

An anonymous proxy server (sometimes called a web proxy) generally attempts to anonymize web surfing. There are different varieties of anonymizers. The destination server (the server that ultimately satisfies the web request) receives requests from the anonymizing proxy server, and thus does not receive information about the end user's address. However, the requests are not anonymous to the anonymizing proxy server, and so a degree of trust is present between the proxy server and the user. Many of them are funded through a continued advertising link to the user.

**Access control**: Some proxy servers implement a logon requirement. In large organizations, authorized users must log on to gain access to the web. The organization can thereby track usage to individuals.

Some anonymizing proxy servers may forward data packets with header lines such as HTTP_VIA, HTTP_X_FORWARDED_FOR, or HTTP_FORWARDED, which may reveal the IP address of the client. Other anonymizing proxy servers, known as elite or high anonymity proxies, only include the REMOTE_ADDR header with the IP address of the proxy server, making it appear that the proxy server is the client. A website could still suspect a proxy is being used if the client sends packets which include a cookie from a previous visit that did not use the high anonymity proxy server. Clearing cookies, and possibly the cache, would solve this problem.

## *Implementations of proxies*

### Web proxy

A proxy that focuses on World Wide Web traffic is called a "web proxy". The most common use of a web proxy is to serve as a web cache. Most proxy programs provide a means to deny access to URLs specified in a blacklist, thus providing content filtering. This is often used in a corporate, educational, or library environment, and anywhere else

where content filtering is desired. Some web proxies reformat web pages for a specific purpose or audience, such as for cell phones and PDAs.

### Suffix proxies

A **suffix proxy server** allows a user to access web content by appending the name of the proxy server to the URL of the requested content. Suffix proxy servers are easier to use than regular proxy servers.

### Transparent proxies

An **intercepting proxy** (also **forced proxy** or **transparent proxy**) combines a proxy server with a gateway or router (commonly with NAT capabilities). Connections made by client browsers through the gateway are diverted to the proxy without client-side configuration (or often knowledge). Connections may also be diverted from a SOCKS server or other circuit-level proxies.

RFC 2616 (Hypertext Transfer Protocol—HTTP/1.1) offers standard definitions:

"A 'transparent proxy' is a proxy that does not modify the request or response beyond what is required for proxy authentication and identification".
"A 'non-transparent proxy' is a proxy that modifies the request or response in order to provide some added service to the user agent, such as group annotation services, media type transformation, protocol reduction, or anonymity filtering".

A security flaw in the way that transparent proxies operate was published by Robert Auger in 2009 and advisory by the Computer Emergency Response Team was issued listing dozens of affected transparent, and intercepting proxy servers.

## Purpose

Intercepting proxies are commonly used in businesses to prevent avoidance of acceptable use policy, and to ease administrative burden, since no client browser configuration is required. This second reason however is mitigated by features such as Active Directory group policy, or DHCP and automatic proxy detection.

Intercepting proxies are also commonly used by ISPs in some countries to save upstream bandwidth and improve customer response times by caching. This is more common in countries where bandwidth is more limited (e.g. island nations) or must be paid for.

## Issues

The diversion / interception of a TCP connection creates several issues. Firstly the original destination IP and port must somehow be communicated to the proxy. This is not always possible (e.g. where the gateway and proxy reside on different hosts). There is a class of cross site attacks which depend on certain behaviour of intercepting proxies that

do not check or have access to information about the original (intercepted) destination. This problem can be resolved by using an integrated packet-level and application level appliance or software which is then able to communicate this information between the packet handler and the proxy.

Intercepting also creates problems for HTTP authentication, especially connection-oriented authentication such as NTLM, since the client browser believes it is talking to a server rather than a proxy. This can cause problems where an intercepting proxy requires authentication, then the user connects to a site which also requires authentication.

Finally intercepting connections can cause problems for HTTP caches, since some requests and responses become uncacheble by a shared cache.

Therefore intercepting connections is generally discouraged. However due to the simplicity of deploying such systems, they are in widespread use.

## Implementation Methods

Interception can be performed using Cisco's WCCP (Web Cache Control Protocol). This proprietary protocol resides on the router and is configured from the cache, allowing the cache to determine what ports and traffic is sent to it via transparent redirection from the router. This redirection can occur in one of two ways: GRE Tunneling (OSI Layer 3) or MAC rewrites (OSI Layer 2).

Once traffic reaches the proxy machine itself interception is commonly performed with NAT (Network Address Translation). Such setups are invisible to the client browser, but leave the proxy visible to the web server and other devices on the Internet side of the proxy. Recent releases of Linux and some BSD provide TPROXY (Transparent Proxy) which performs IP-level (OSI Layer 3) transparent interception and Spoofing of outbound traffic. Hiding the proxy IP address from other network devices.

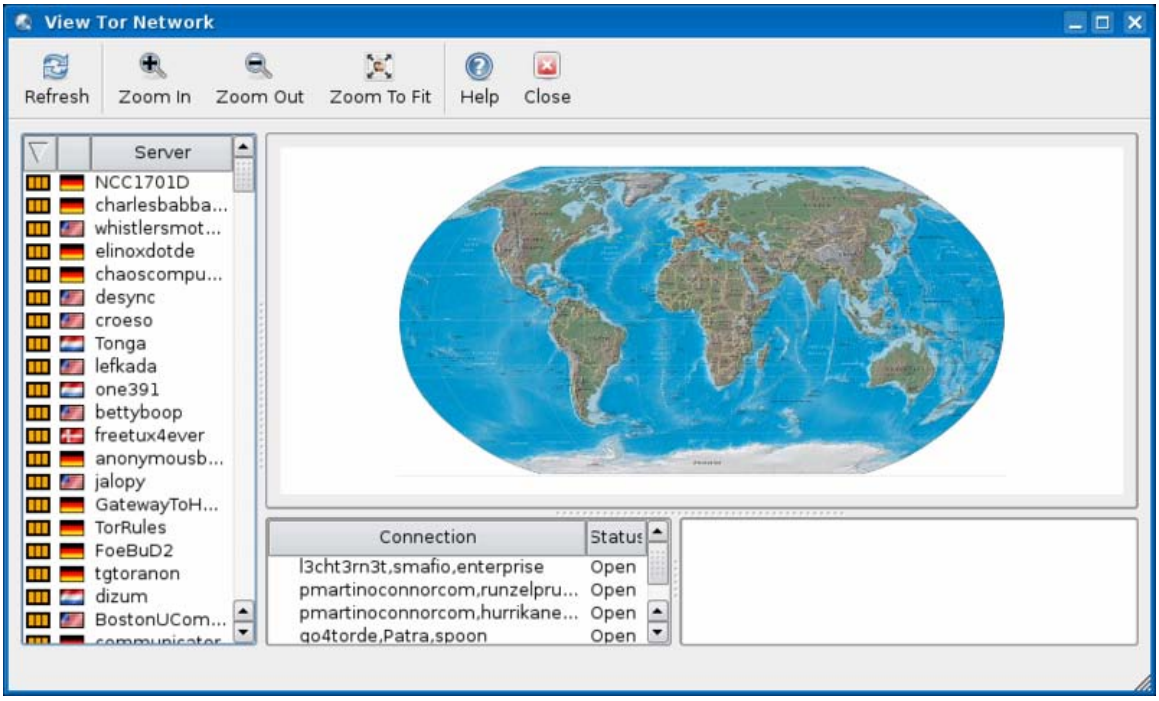
## Detection

There are several methods that can often be used to detect the presence of an intercepting proxy server:

- By comparing the client's external IP address to the address seen by an external web server, or sometimes by examining the HTTP headers received by a server. A number of sites have been created to address this issue, by reporting the user's IP address as seen by the site back to the user in a web page.
- By comparing the sequence of network hops reported by a tool such as traceroute for a proxied protocol such as http (port 80) with that for a non proxied protocol such as SMTP (port 25). ,
- By attempting to make a connection to an IP address at which there is known to be no server. The proxy will accept the connection and then attempt to proxy it on. When the proxy finds no server to accept the connection it may return an error

message or simply close the connection to the client. This difference in behaviour is simple to detect. For example most web browsers will generate a browser created error page in the case where they cannot connect to an HTTP server but will return a different error in the case where the connection is accepted and then closed.

## Tor onion proxy software




The Vidalia Tor-network map.

The Tor anonymity network ('Tor' for short) is a system aiming at online anonymity. Tor is an implementation of onion routing. It works by relaying communications through a network of systems run by volunteers in various locations. By keeping some of the network entry points hidden, Tor is also able to evade internet censorship. Tor is intended to protect users' personal freedom, privacy, and ability to conduct confidential business.

Users of a Tor network run an onion proxy software on their computer. The Tor software periodically negotiates a virtual circuit through the Tor network. At the same time, the onion proxy software presents a SOCKS interface to its clients or users. SOCKS-ifying applications like Polipo may be linked with the Tor onion proxy software, which then multiplexes the traffic through a Tor virtual circuit.

The software is open-source and the network is free of charge to use. Vidalia is a cross-platform controller GUI for Tor.

## I2P anonymous proxy

The I2P anonymous network ('I2P') is a proxy network aiming at online anonymity. It implements garlic routing, which is an enhancement of Tor's onion routing. I2P is fully distributed and works by encrypting all communications in various layers and relaying them through a network of routers run by volunteers in various locations. By keeping the source of the information hidden, I2P offers censorship resistance. The goals of I2P are to protect users' personal freedom, privacy, and ability to conduct confidential business.

Each user of I2P runs an I2P router on their computer (node). The I2P router takes care of finding other peers and building anonymizing tunnels through them. I2P provides proxies for all protocols (HTTP, irc, SOCKS, ...).

## Chapter-4

# MUD

A **MUD** (originally **Multi-User Dungeon**, with later variants **Multi-User Dimension** and **Multi-User Domain**), is a multiplayer real-time virtual world described primarily in text. MUDs combine elements of role-playing games, hack and slash, player versus player, interactive fiction, and online chat. Players can read or view descriptions of rooms, objects, other players, non-player characters, and actions performed in the virtual world. Players typically interact with each other and the world by typing commands that resemble a natural language.

Traditional MUDs implement a role-playing video game set in a fantasy world populated by fictional races and monsters, with players choosing classes in order to gain specific skills or powers. The object of this sort of game is to slay monsters, explore a fantasy world, complete quests, go on adventures, create a story by roleplaying, and advance the created character. Many MUDs were fashioned around the dice-rolling rules of the *Dungeons & Dragons* series of games.

Such fantasy settings for MUDs are common, while many others have science fiction settings or are based on popular books, movies, animations, history, and so on. Not all MUDs are games; some are designed for educational purposes, while others are purely chat environments, and the flexible nature of many MUD servers leads to their occasional use in areas ranging from computer science research to geoinformatics to medical informatics. MUDs have attracted the interest of academic scholars from many fields, including communications, sociology, law, and economics. At one time, there was interest from the United States military in using them for teleconferencing.

Most MUDs are run as hobbies and are free to players; some may accept donations or allow players to purchase virtual items, while others charge a monthly subscription fee. MUDs can be accessed via standard telnet clients, or specialized MUD clients which are designed to improve the user experience. Numerous games are listed at various web portals, such as The Mud Connector.

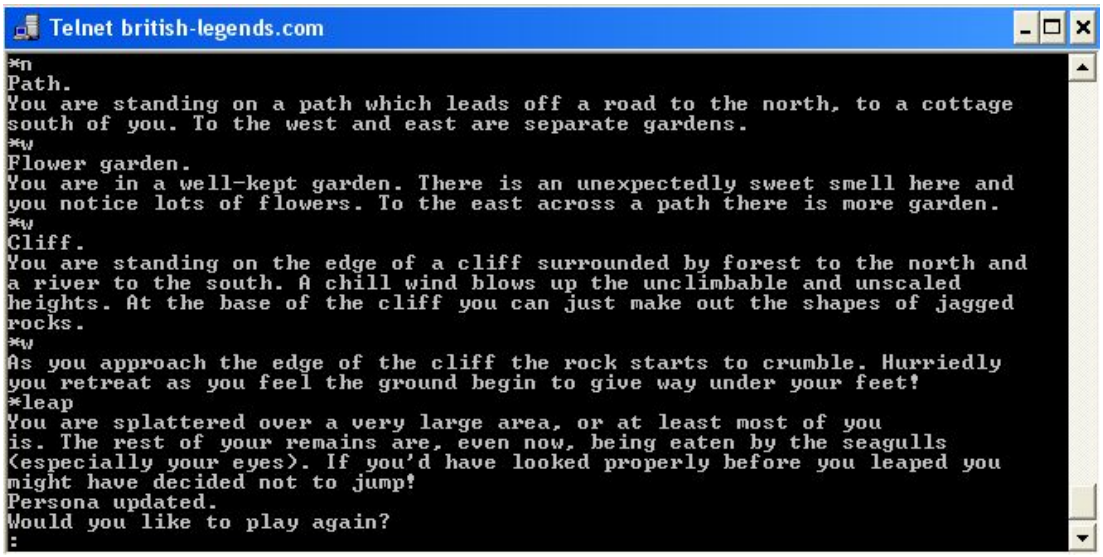
The history of modern Massively Multiplayer Online Role-Playing Games (MMORPGs) like *World of Warcraft*, and related virtual world genres such as the social virtual worlds exemplified by *Second Life*, traces directly back to the MUD genre. Indeed, before the invention of the term MMORPG, games of this style were simply called graphical MUDs. A number of influential MMORPG designers began as MUD developers and/or

players (such as Raph Koster, Brad McQuaid, Matt Firor, and Brian Green) or were involved with early MUDs (like Mark Jacobs and J. Todd Coleman).

## Origins

```
.RUN ADV11

WELCOME TO ADVENTURE!!  WOULD YOU LIKE INSTRUCTIONS?


YES
SOMEWHERE NEARBY IS COLOSSAL CAVE, WHERE OTHERS HAVE FOUND
FORTUNES IN TREASURE AND GOLD, THOUGH IT IS RUMORED
THAT SOME WHO ENTER ARE NEVER SEEN AGAIN. MAGIC IS SAID
TO WORK IN THE CAVE.  I WILL BE YOUR EYES AND HANDS. DIRECT
ME WITH COMMANDS OF 1 OR 2 WORDS.
(ERRORS, SUGGESTIONS, COMPLAINTS TO CROWTHER)
(IF STUCK TYPE HELP FOR SOME HINTS)


YOU ARE STANDING AT THE END OF A ROAD BEFORE A SMALL BRICK
BUILDING . AROUND YOU IS A FOREST. A SMALL
STREAM FLOWS OUT OF THE BUILDING AND DOWN A GULLY.


GO IN
YOU ARE INSIDE A BUILDING, A WELL HOUSE FOR A LARGE SPRING.


THERE ARE SOME KEYS ON THE GROUND HERE.


THERE IS A SHINY BRASS LAMP NEARBY.


THERE IS FOOD HERE.


THERE IS A BOTTLE OF WATER HERE.
```

Will Crowther's *Adventure*




You haven't lived until you've died in MUD. — The *MUD1* Slogan

*Colossal Cave Adventure*, created in 1975 by Will Crowther on a DEC PDP-10 computer, was the first widely used adventure game. The game was significantly expanded in 1976 by Don Woods. Also called *Adventure*, it contained many D&D features and references, including a computer controlled dungeon master.

Inspired by *Adventure*, a group of students at MIT wrote a game called *Zork* in the summer of 1977 for the PDP-10 minicomputer which became quite popular on the ARPANET. *Zork* was ported under the name *Dungeon* to FORTRAN by a programmer working at DEC in 1978.

In 1978 Roy Trubshaw, a student at Essex University in the UK, started working on a multi-user adventure game in the MACRO-10 assembly language for a DEC PDP-10. He named the game *MUD* (*Multi-User Dungeon*), in tribute to the *Dungeon* variant of *Zork*, which Trubshaw had greatly enjoyed playing. Trubshaw converted MUD to BCPL (the predecessor of C), before handing over development to Richard Bartle, a fellow student at Essex University, in 1980.

*MUD*, better known as *Essex MUD* and *MUD1* in later years, ran on the Essex University network until late 1987, becoming the first Internet multiplayer online role-playing game in 1980, when Essex University connected its internal network to ARPANet. The game revolved around gaining points till one achieved the wizard rank, giving the player immortality and certain powers over mortals. The game became more widely accessible when a guest account was set up that allowed users on JANET (a British academic X.25 computer network) to connect on weekends and between the hours of 2 AM and 8 AM on weekdays. *MUD1* was reportedly closed down when Richard Bartle licensed *MUD1* to CompuServe, and was getting pressure from them to close *Essex MUD*. This left *MIST*, a derivative of *MUD1* with similar gameplay, as the only remaining MUD running on the Essex University network, becoming one of the first of its kind to attain broad popularity. *MIST* ran until the machine that hosted it, a PDP-10, was superseded in early 1991.

During the Christmas of 1985, Neil Newell, an avid *MUD1* player, started programming his own MUD called *SHADES* because *MUD1* was closed down during the holidays. Starting out as a hobby, *SHADES* became accessible in the UK as a commercial MUD via British Telecom's Prestel and Micronet networks. A scandal on *SHADES* led to the closure of Micronet, as described in Indra Sinha's net-memoir, *The Cybergypsies*.

In 1985 Pip Cordrey gathered some people on a BBS he ran to create a *MUD1* clone that would run on a home computer. The tolkienesque MUD went live in 1986 and was named *MirrorWorld*.

1985 also saw the creation of *Gods* by Ben Laurie, a *MUD1* clone that included online creation in its endgame. *Gods* became a commercial MUD in 1988.

In 1985 CompuNet started a project named *Multi-User Galaxy Game* as a Science Fiction alternative to *MUD1* which ran on their system at the time. When one of the two programmers left CompuNet, the remaining programmer, Alan Lenton, decided to

rewrite the game from scratch and named it Federation II (at the time no Federation I existed). The MUD was officially launched in 1989. Federation II was later picked up by AOL, where it became known simply as "Federation: Adult Space Fantasy". Federation later left AOL to run on its own after AOL began offering unlimited service.

In 1978, around the same time Roy Trubshaw wrote MUD, Alan E. Klietz wrote a game called *Milieu* using Multi-Pascal on a CDC Cyber 6600 series mainframe which was operated by the Minnesota Educational Computing Consortium. Klietz ported *Milieu* to an IBM XT in 1983, naming the new port *Scepter of Goth*. *Scepter* supported 10 to 16 simultaneous users, typically connecting in by modem. It was one of the first commercial MUDs; franchises were sold to a number of locations. *Scepter* was first owned and run by GamBit (of Minneapolis, Minnesota), founded by Bob Alberti. GamBit's assets were later sold to Interplay Productions. Interplay eventually went bankrupt.

In 1984, Mark Peterson wrote *The Realm of Angmar*, beginning as a clone of *Scepter of Goth*. In 1994, Peterson rewrote *The Realm of Angmar*, adapting it to MS-DOS (the basis for many dial-in BBS systems), and renamed it *Swords of Chaos*. For a few years this was a very popular form of MUD, hosted on a number of BBS systems, until widespread Internet access eliminated most BBSes.

In 1984, Mark Jacobs created and deployed a commercial gaming site, *Gamers World*. The site featured two games coded and designed by Jacobs, a MUD called *Aradath* (which was later renamed, upgraded and ported to *GEnie* as *Dragon's Gate*) and a 4X science-fiction game called *Galaxy*, which was also ported to *GEnie*. At its peak, the site had about 100 monthly subscribers to both *Aradath* and *Galaxy*. GEnie was shut down in the late 1980s, although *Dragon's Gate* was later brought to *America Online* before it was finally released on its own. Dragon's Gate was closed on February 10, 2007.

In the summer of 1980 University of Virginia classmates John Taylor and Kelton Flinn wrote *Dungeons of Kesmai*, a six player game inspired by *Dungeons & Dragons* which used Roguelike ASCII graphics. They founded the Kesmai company in 1982 and in 1985 an enhanced version of *Dungeons of Kesmai*, *Island of Kesmai*, was launched on CompuServe. Later, its 2-D graphical descendant *Legends of Kesmai* was launched on AOL in 1996. The games were retired commercially in 2000.

The popularity of MUDs of the Essex University tradition escalated in the USA during the late 1980s when affordable personal computers with 300 to 2400 bit/s modems enabled role-players to log into multi-line Bulletin Board Systems and online service providers such as CompuServe. During this time it was sometimes said that MUD stands for "Multi Undergraduate Destroyer" due to their popularity among college students and the amount of time devoted to them.

## *Spread*

### AberMUD

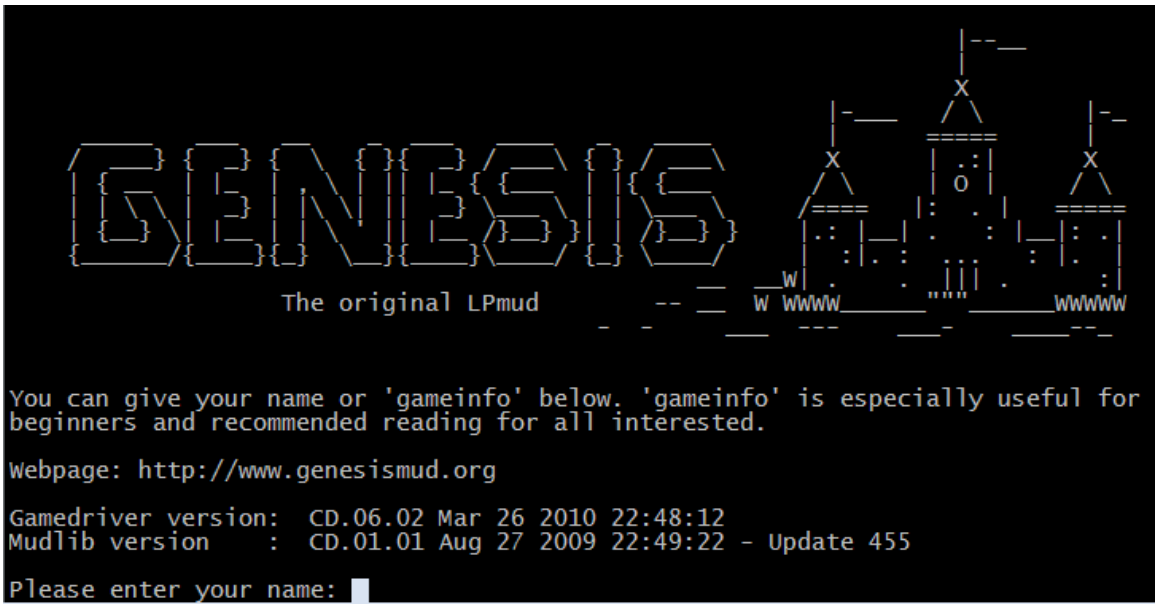
The first popular MUD codebase was AberMUD, written in 1987 by Alan Cox, named after the University of Wales, Aberystwyth. Alan Cox had played the original University of Essex MUD, and the gameplay was heavily influenced by it. AberMUD was initially written in B for a Honeywell L66 mainframe under GCOS3/TSS. In late 1988 it was ported to C, which enabled it to spread rapidly to many Unix platforms upon its release in 1989. AberMUD's popularity resulted in several inspired works, the most notable of which were TinyMUD, LPMud, and DikuMUD.

### TinyMUD

*Monster* was a multi-user adventure game created by Richard Skrenta for the VAX and written in VMS Pascal. It was publicly released in November 1988. *Monster* was disk-based and modifications to the game were immediate. *Monster* pioneered the approach of allowing players to build the game world, setting new puzzles or creating dungeons for other players to explore. Monster, which comprised about 60,000 lines of code, had a lot of features which appeared to be designed to allow *Colossal Cave Adventure* to work in it. Though there never were many network-accessible Monster servers, it inspired James Aspnes to create a stripped down version of *Monster* which he called TinyMUD.

TinyMUD, written in C and released in late 1989, spawned a number of descendants, including TinyMUCK and TinyMUSH. TinyMUCK version 2 contained a full programming language named MUF (Multi-User Forth), while MUSH greatly expanded the command interface. To distance itself from the combat-oriented traditional MUDs it was said that the "D" in TinyMUD stood for Multi-User "Domain" or "Dimension"; this, along with the eventual popularity of acronyms other than MUD (such as MUCK, MUSH, MUSE, and so on) for this kind of server, led to the eventual adoption of the term MU* to refer to the TinyMUD family. UberMUD, UnterMUD, and MOO were inspired by TinyMUD but are not direct descendants.

## LPMud




The login screen from *Genesis*, the first LPMud

In 1989 LPMud was developed by Lars Pensjö (hence the **LP** in LPMud). Pensjö had been an avid player of TinyMUD and AberMUD and wanted to create a world with the flexibility of TinyMUD and the gameplay of AberMUD. In order to accomplish this he wrote what is nowadays known as a virtual machine, which he called the LPMud driver, that ran the C-like LPC programming language used to create the game world. Pensjö's interest in LPMud eventually waned and development was carried on by others such as Jörn "Amylaar" Rennecke, Felix "Dworkin" Croes, Tim "Beek" Hollebeek and Lars Düning. During the early 1990s, LPMud was one of the most popular MUD codebases. Descendants of the original LPMud include MudOS, DGD, SWLPC, FluffOS, and the Pike programming language, the latter the work of long-time LPMud developer Fredrik "Profezzorn" Hübinette.

## DikuMUD

In 1990, the release of DikuMUD, which was inspired by AberMUD, led to a virtual explosion of hack and slash MUDs based upon its code. DikuMUD inspired numerous derivative codebases, including CircleMUD, Merc, ROM, SMAUG, and GodWars. The original Diku team comprised Sebastian Hammer, Tom Madsen, Katja Nyboe, Michael Seifert, and Hans Henrik Staerfeldt. DikuMUD had a key influence on the early evolution of the MMORPG genre, with *EverQuest* (created by avid DikuMUD player Brad McQuaid) displaying such Diku-like gameplay that Verant developers were made to issue a sworn statement that no actual DikuMUD code was incorporated.

### Simutronics

In 1987 David Whatley, having previously played *Scepter of Goth* and *Island of Kesmai*, founded Simutronics with Tom and Susan Zelinski. In the same year they demonstrated a prototype of *GemStone* to GEnie. After a short-lived instance of *GemStone II*, *GemStone III* was officially launched in February 1990. *GemStone III* became available on AOL in September 1995, followed by the release of *DragonRealms* in February 1996. By the end of 1997 *GemStone III* and *DragonRealms* had become the first and second most played games on AOL.

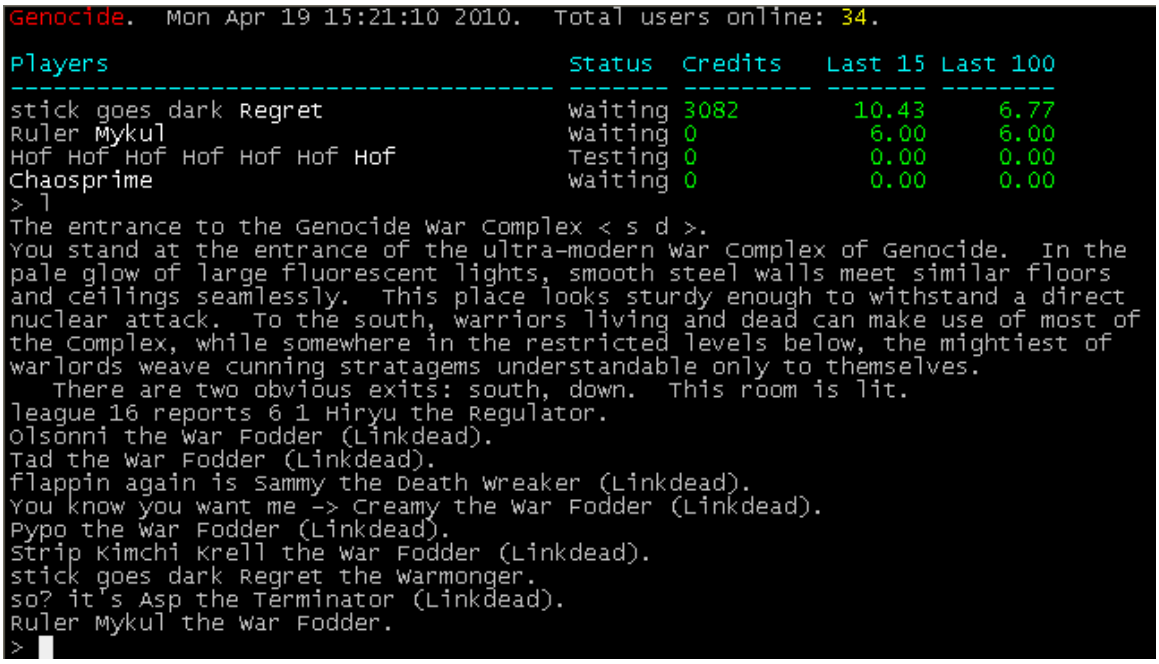
## *Style*

While there have been many variations in overall focus, gameplay and features in MUDs, some distinct sub-groups have formed that can be used to help categorize different game mechanics, game genres and non-game uses.

### Hack and Slash MUDs

Perhaps the most common approach to game design in MUDs is to loosely emulate the structure of a *Dungeons & Dragons* campaign focused more on fighting and advancement than role-playing. When these MUDs restrict player-killing in favor of player versus environment conflict and questing, they are labeled **Hack and Slash MUDs**. This may be considered particularly appropriate since, due to the room-based nature of traditional MUDs, ranged combat is typically difficult to implement, resulting in most MUDs equipping characters mainly with close-combat weapons. This style of game was also historically referred to within the MUD genre as "adventure games", but video gaming as a whole has developed a meaning of "adventure game" that is greatly at odds with this usage.

## Player versus player MUDs




A screenshot from *Genocide* showing its War Complex

Most MUDs restrict player versus player combat, often abbreviated as PK (Player Killing). This is accomplished through hard coded restrictions and various forms of social intervention. MUDs without these restrictions are commonly known as **PK MUDs**. Taking this a step further are MUDs devoted *solely* to this sort of conflict, called **pure PK** MUDs, the first of which was *Genocide* in 1992. *Genocide*'s ideas were influential in the evolution of player versus player online gaming.

## Roleplaying MUDs

**Roleplaying MUDs**, generally abbreviated as **RP MUDs**, encourage or enforce that players act out the role of their playing characters at all times. Some RP MUDs provide an immersive gaming environment, while others only provide a virtual world with no game elements. MUDs where roleplay is enforced and the game world is heavily computer-modeled are sometimes known as **Roleplay Intensive MUDs**, or **RPIMUDs**.

## Social MUDs

**Social MUDs** de-emphasize game elements in favor of an environment designed primarily for socializing. They are differentiated from talkers by retaining elements beyond online chat, typically online creation as a community activity and some element of role-playing — often such MUDs have broadly defined contingents of socializers and roleplayers. Server software in the TinyMUD family, or MU*, is traditionally used to implement social MUDs.

## Talkers

A less-known MUD variant is the **talker**, a variety of online chat environment typically based on server software like ew-too or NUTS. Most of the early Internet talkers were LPMuds with the majority of the complex game machinery stripped away, leaving just the communication commands. The first Internet talker was *Cat Chat* in 1990. Avid users of talkers are called spods.

## Educational MUDs

Taking advantage of the flexibility of MUD server software, some MUDs are designed for educational purposes rather than gaming or chat. *MicroMUSE* is considered by some to have been the first educational MUD, but it can be argued that its evolution into this role was not complete until 1994, which would make the first of many educational MOOs, *Diversity University* in 1993, also the first educational MUD. The MUD medium lends itself naturally to constructionist learning pedagogical approaches.

## Graphical MUDs


A combat in *The Shadow of Yserbius*, an early graphical MUD

A **graphical MUD** is a MUD that uses computer graphics to represent parts of the virtual world and its visitors. A prominent early graphical MUD was *Habitat*, written by Randy Farmer and Chip Morningstar for Lucasfilm in 1985. Graphical MUDs require players to download a special client and the game's artwork. They range from simply enhancing the

user interface to simulating 3D worlds with visual spatial relationships and customized avatar appearances.

Games such as *Meridian 59*, *EverQuest*, *Ultima Online* and *Dark Age of Camelot* were routinely called graphical MUDs in their earlier years. *RuneScape* was actually originally intended to be a *text-based* MUD, but graphics were added very early in development. However, with the increase in computing power and Internet connectivity during the late nineties, and the shift of online gaming to the mass market, the term "graphical MUD" fell out of favor, being replaced by MMORPG, Massively Multiplayer Online Role-Playing Game, a term coined by Richard Garriott in 1997.

## Psychology and engagement

Sherry Turkle developed a theory that the constant use (and in many cases, overuse) of MUDs allows users to develop different personalities in their environments. She uses examples, dating back to the text-based MUDs of the mid-1990s, showing college students who simultaneously live different lives through characters in separate MUDs, up to three at a time, all while doing schoolwork. The students claimed that it was a way to "shut off" their own lives for a while and become part of another reality. Turkle claims that this could present a psychological problem of identity for today's youths.

"A Story About A Tree" is a short essay written by Raph Koster regarding the death of a *LegendMUD* player named Karyn, raising the subject of inter-human relationships in virtual worlds.

Observations of MUD-play show styles of play that can be roughly categorized. Achievers focus on concrete measurements of success such as experience points, levels, and wealth; Explorers investigate every nook and cranny of the game, and evaluate different game mechanical options; Socializers devote most of their energy to interacting with other players; and then there are Killers who focus on interacting negatively with other players, if permitted, killing their characters or otherwise thwarting their play. Few players play only one way, or play one way all the time; most exhibit a diverse style. According to Richard Bartle, "People go there as part of a hero's journey — a means of self-discovery".

Research has suggested that various factors combine in MUDs to provide users with a sense of *presence* rather than simply communication.

## Grammatical usage and derived terms

As a noun, the word **MUD** is variously written MUD, Mud, and mud, depending on speaker and context. It is also used as a verb, with **to mud** meaning to play or interact with a MUD and **mudding** referring to the act of doing so. A **mudder** is, naturally, one who MUDs. Compound words and portmanteaux such as **mudlist**, **mudsex**, and **mudflation** are also regularly coined. Puns on the "wet dirt" meaning of "mud" are

endemic, as with, for example, the names of the ROM (**R**ivers **o**f **M**UD), MUCK and MUSH codebases and the MUD *Muddy Waters*.

## Chapter-5

# Message Transfer Agent

Within Internet message handling services (MHS), a **message transfer agent** or **mail transfer agent** (**MTA**) or **mail relay** is software that transfers electronic mail messages from one computer to another using a client–server application architecture. An MTA implements both the client (sending) and server (receiving) portions of the Simple Mail Transfer Protocol.

The terms *mail server*, *mail exchanger*, and *MX host* may also refer to a computer performing the MTA function. The Domain Name System (DNS) associates a mail server to a domain with mail exchanger (MX) resource records containing the domain name of a host providing MTA services.

## *Operation*

A message transfer agent receives mail from either another MTA, a mail submission agent (MSA), or a mail user agent (MUA). The transmission details are specified by the Simple Mail Transfer Protocol (SMTP). When a recipient mailbox of a message is not hosted locally, the message is relayed, that is, forwarded to another MTA. Every time an MTA receives an email message, it adds a `Received` trace header field to the top of the header of the message, thereby building a sequential record of MTAs handling the message. The process of choosing a target MTA for the next hop is also described in SMTP, but can usually be overridden by configuring the MTA software with specific routes.

A MTA works in the background, while the user usually interacts directly with a mail user agent. One may distinguish initial submission as first passing through an MSA – port 587 is used for communication between an MUA and an MSA while port 25 is used for communication between MTAs, or from an MSA to an MTA; this distinction is first made in RFC 2476.

For recipients hosted locally, the final delivery of email to a recipient mailbox is the task of a message delivery agent (MDA). For this purpose the MTA transfers the message to the message handling service component of the message delivery agent. Upon final delivery, the `Return-Path` field is added to the envelope to record the return path.

## *Transfer versus access*

The function of an MTA is usually complemented with some means for email clients to access stored messages. This function typically employs a different protocol. The most widely implemented open protocols for the MUA are the Post Office Protocol (POP3) and the Internet Message Access Protocol (IMAP), but many proprietary systems exist (Exchange, Lotus Domino/Notes) for retrieving messages. Many systems also offer a web interface for reading and sending email that is independent of any particular MUA.

At its most basic, an MUA using POP3 downloads messages from the server mailbox onto the local computer for display in the MUA. Messages are generally removed from the server at the same time but most systems also allow a copy to be left behind as a backup. In contrast, an MUA using IMAP displays messages directly from the server, although a download option for archive purposes is usually also available. One advantage this gives IMAP is that the same messages are visible from any computer accessing the email account, since messages aren't routinely downloaded and deleted from the server. If set up properly, sent mail can be saved to the server also, in contrast with POP mail, where sent messages exist only in the local MUA and are not visible by other MUAs accessing the same account.

The IMAP protocol has features that allow uploading of mail messages and there are implementations that can be configured to also *send* messages like an MTA, which combine sending a copy and storing a copy in the *Sent* folder in one upload operation.

The reason for using SMTP as a standalone transfer protocol is twofold:

- *To cope with discontinuous connections.* Historically, inter-network connections were not continuously available as they are today and many readers didn't need an access protocol, as they could access their mailbox directly (as a file) through a terminal connection. SMTP, if configured to use backup MXes, can transparently cope with temporary local network outages. A message can be transmitted along a variable path by choosing the next *hop* from a preconfigured list of MXes with no intervention from the originating user.
- *Submission policies.* Modern systems are designed for users to submit messages to their local servers for policy, not technical, reasons. It was not always that way. For example, the original Eudora email client featured direct delivery of mail to the recipients' servers, out of necessity. Today, funneling email through MSA systems run by providers that in principle have some means of holding their users accountable for the generation of the email is a defense against spam and other forms of email abuse.

## *List of MTA software for Unix-like operating systems*

- Apache James
- Courier Mail Server
- Dragonfly Mail Agent - A lightweight mail transport agent

- esmtp
- exim
- masqmail
- meldware
- meta1
- MMDF
- msmtp-mta (a wrapper around msmtp providing a sendmail executable)
- nbsmtp
- norelaysmtp
- postfix
- qmail
- qpsmtpd
- qwik-smtpd
- sendmail
- smail
- ssmtp
- Synovel (email server)
- zimbra
- zmailer

## Chapter-6

# IRCd

An **IRCd**, short for **Internet Relay Chat daemon**, is server software that implements the IRC protocol, enabling people to talk to each other via the Internet (exchanging textual messages in real time). It is distinct from an IRC bot that connects outbound to an IRC channel.

The server listens to connections from IRC clients on a set of TCP ports. When the server is part of an IRC network, it also keeps one or more established connections to other servers/daemons.

The term *ircd* originally referred to only one single piece of software, but it eventually became a generic reference to any implementation of an IRC daemon. However, the original version is still distributed under the same name, and here we, discusses both uses.

### *History*

The original IRCd was known as 'ircd', and was authored by Jarkko Oikarinen (WiZ on IRC) in 1988. He received help from a number of others, such as Markku Savela (msa on IRC), who helped with the 2.2+msa release, etc.

In its first incarnations, IRC did not have many features that are taken for granted today, such as named channels and channel operators. Channels were numbered – channel 4 and channel 57, for example – and the channel **topic** described the kind of conversation that took place in the channel. One holdover of this is that joining channel 0 causes a client to leave all the channels it is presently on: "CHANNEL 0" being the original command to leave the current channel.

The first major change to IRC, in version 2.5, was to add **named channels** – "+channels". "+channels" were later replaced with "#channels" in version 2.7, numeric channels were removed entirely and channel bans (mode +b) were implemented.

Around version 2.7, there was a small but notable dispute, which led to ircu – the Undernet fork of ircd.

irc2.8 added "&channels" (those that exist only on the current server, rather than the entire network) and "!channels" (those that are theoretically safe from suffering from the many ways that a user could exploit a channel by "riding a netsplit"), and is the baseline release from which nearly all current implementations are derived.

Around 2.8 came the concept of nick and channel delay, a system designed to help curb abusive practices such as takeovers and split riding. This was not agreed on by the majority of modern IRC (EFnet, DALnet, Undernet, etc.) - and thus, 2.8 was forked into a number of different daemons using an opposing theory known as TS – or time stamping, which stored a unique time stamp with each channel or nickname on the network to decide which was the 'correct' one to keep.

Time stamping itself has been revised several times to fix various issues in its design. The latest versions of such protocols are:

- the TS6 protocol, which is used by EFnet, and Hybrid and Ratbox based servers amongst others
- the P10 protocol, which is used by Undernet and ircu based servers.

While the client-to-server protocols are at least functionally similar, server-to-server protocols differ widely (TS5, P10, and ND/CD server protocols are incompatible), making it very difficult to "link" two separate implementations of the IRC server. Some "bridge" servers do exist, to allow linking of, for example, 2.10 servers to TS5 servers, but these are often accompanied with restrictions of which parts of each protocol may be used, and are not widely deployed.

Significant releases based on 2.8 included:

- 2.8.21+CS, developed by Chris Behrens (**Comstud**)
- 2.8+th, Taner's patchset, which later became
    - Hybrid IRCd, originally developed by Jon Lusky (**Rodder**) and Diane Bruce (**Dianora**) as 2.8/hybrid, later joined by a large development team.
- 2.9, 2.10, 2.11, ... continue the development of the original codebase,

The original code base continued to be developed mainly for use on the IRCnet network. New server-to-server protocols were introduced in version 2.10, released in 1998, and in 2.11, first released in 2004, and current as of 2007. This daemon is used by IRCnet and it can be found at ftp://ftp.irc.org/irc/server/ The original ircd is free software, licensed under the GNU General Public License. This development line produced the 4 IRC RFCs released after RFC 1459, which document this server protocol exclusively.

2.8.21+CS and Hybrid IRCd continue to be used on EFnet, with ircd-ratbox (an offshoot of ircd-hybrid) as of 2004 being the most popular.

## Sidestream versions

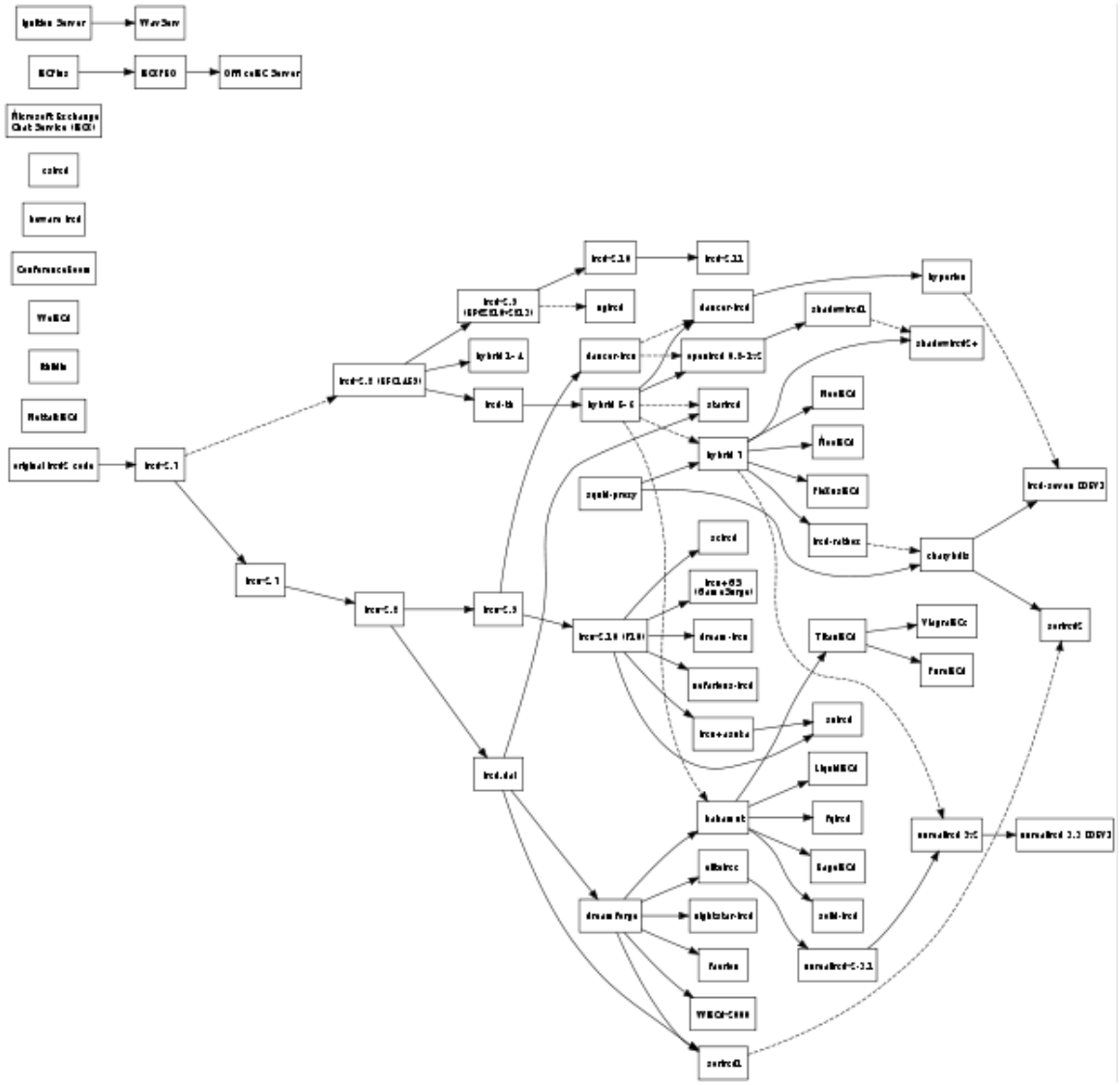


Diagram of derivations and relations for common IRCd implementations.

More recently, several irc daemons were written from scratch, such as ithildin, InspIRCd, csircd (also written by Chris Behrens), ConferenceRoom, Microsoft Exchange Chat Service, WeIRCd, or IRCPlus/IRCXPro.

These attempts have met with mixed success, and large doses of skepticism from the existing IRC development community. With each new IRCd, a slightly different version of the IRC protocol is used, and many IRC clients and bots are forced to compromise on features or vary their implementation based on the server to which they are connected. These are often implemented for the purpose of improving usability, security, separation of powers, or ease of integration with services. Possibly one of the most common and visible differences is the inclusion or exclusion of the half-op channel operator status (which is not a requirement of the RFCs).

## *Features*

### Ports

The officially assigned port numbers are 194 ("irc"), 529 ("irc-serv"), and 994 ("ircs"). However, these ports are in the *privileged* range (0-1024), which on a Unix-like system means that the daemon would have to have superuser privileges in order to open them. For various security reasons this is undesirable.

The common ports for an IRCd process are 6665 to 6669, with 6667 being the historical default. These ports can be opened by a non-superuser process, and they became widely used.

### Connections

Running a large IRC server, one that has more than a few thousand simultaneous users, requires keeping a very large number of TCP connections open for long periods. Very few ircds are multithreaded as nearly every action needs to access (at least read and possibly modify) the global state.

The result is that the best platforms for ircds are those that offer efficient mechanisms for handling huge numbers of connections in a single thread. Linux offers this ability in the form of epoll, in kernel series newer than 2.4.x. FreeBSD (since 4.1) and OpenBSD (since 2.9) offers kqueue. Solaris has had /dev/poll since version 7, and from version 10 onwards has IOCP (I/O Completion Ports). Windows has supported IOCP since Windows NT 3.5. The difference made by these new interfaces can be dramatic. IRCU coders have mentioned increases in the practical capacity per server from 10,000 users to 20,000 users.

### SSL

Some IRCd support SSL, for those who don't, it is still possible to use SSL via Stunnel. The unofficial, but most often used port for SSL IRCd connections is 6697. More recently, as a security enhancement and usability enhancement, various client and server authors have begun drafting a standard known as the STARTTLS standard which allows for SSL and plain text connections to co-exist on the same TCP port.

### IPv4 and IPv6

IRC daemons support IPv4, and some also support IPv6. In general, the difference between IPv6 and IPv4 connections to IRC is purely academic and the service operates in much the same manner through either protocol.

### Clustering

Large IRC networks consist of multiple servers for horizontal scaling purposes. There are several IRC protocol extensions for these purposes.

### IRCX

IRCX (Internet Relay Chat eXtensions) is an extension to the IRC protocol developed by Microsoft

### P10

The **P10** protocol is an extension to the Internet Relay Chat protocol for server to server communications developed by the Undernet Coder Committee to use in their ircu server software. It is similar in purpose to IRCX and EFnet TS5/TS6 protocols and implements nick and channel timestamping for handling nick collisions and netsplit channel riding, respectively. Other IRCd's that utilize this protocol extension include beware ircd.

## *Configuration*

### Jupe

In Internet Relay Chat (IRC), **juping** a server, a channel, or a nickname refers to the practice of blocking said channel or nickname on the server or network or said server on the network. One possible explanation of how this term came about is that it is named after the oper named Jupiter, who gained control of the nickname NickServ on EFnet. EFnet does not offer services such as NickServ; Jupiter gained control of the nickname as he (among other operators) did not believe nicknames should be owned. Today, EFnet opers jupe nicknames that are used as services on other networks.

A nickname or server jupe takes advantage of the fact that certain identifiers are unique; by using an identifier, one acquires an exclusive lock that prevents other users from making use of it.

Officially sanctioned jupes may also utilize services or server configuration options to enforce the jupe, such as when a compromised server is juped to prevent it from harming the network.

In practice IRC operators now use jupe configurations to administratively make channel or nick names unavailable. A channel jupe refers to a server specific ban of a channel, which means that a specific channel cannot be joined when connected to a certain server, but other servers may allow a user to join the channel. This is a way of banning access to problematic channels.

## O-line

An **O-line**, shortened from *Operator Line*, is a line of code in an IRC daemon configuration file that determines which users can become an IRC operator and which permissions they get upon doing so. The name comes from the prefix used for the line in the original ircd, a capital O. The O-line specifies the username, password, operator flags, and hostmask restrictions for a particular operator. A server may have many O-lines depending on the administrative needs of the server and network.

Operator flags are used to describe the permissions an operator is granted. While some IRC operators may be in charge of network routing, others may be in charge of network abuse, making their need for certain permissions different. Operator flags available vary widely depending on which IRC daemon is in use. Generally, more feature rich IRC daemons tend to have more operator flags, and more traditional IRC daemons have fewer.

An O-line may also be set so that only users of a certain hostmask or IP address can gain IRC operator status using that O-line. Using hostmasks and IP addresses in the O-line require the IP address to remain the same but provide additional security.

## K-line

A **k-line** or **kill line** (also written **K:line**) is an Internet Relay Chat term, applied to a specific user. When a user is *k-lined*, it bans the user from a certain server, either for a certain amount of time or permanently. Once the user is banned, they are not allowed back onto that server; they have to join a different server to get onto IRC. This is recorded as a line in the server's IRC daemon configuration file prefixed with the letter "K", hence "K-line".

While the precise reason for the disconnection varies from case to case, usual reasons involve some aspect of the client or the user it is issued against.

User behavior
: K-lines can be given due to inappropriate behavior on the part of the user, such as nickname colliding, mode "hacking", multiple channel flooding, harassing other users via private messaging features, spamming etc., or in the case of older networks without timestamping, split riding, which cannot be corrected through use of channel operator privileges alone.

Client software
: Some IRC daemons can be configured to scan for viruses or other vulnerabilities in clients connecting to them, and will react in various ways according to the result. Outdated and insecure client software might be blocked to protect other network users from vulnerabilities, for instance. Some networks, e.g. freenode, will disconnect clients operating on/via open proxies, or running an insecure web server.

Geographic location

> An IRC network operating multiple servers in different locales will attempt to reduce the distance between a client and a server. This is often achieved by disconnecting (and/or banning) clients from distant locales in favour of local ones.

There are a number of other network "lines" relating to the K-line. Modern IRC daemons will also allow IRC operators to set these lines during normal operation, where access to the server configuration file is not routinely needed.

## G-line

A **G-line** or **global kill line** (also written **G:line**) is a global network ban applied to a user; the term comes from Undernet but on DALnet a similar concept known as an AKill was used.

G-lines are sometimes stored in the configuration file of the IRCd, although some networks, who handle K-lines through the IRC services, prefer to have them stored in their service's configuration files. Whenever a G-lined person attempts to connect to the IRC network, either the services or the IRC daemon will automatically disconnect the client, often displaying a message explaining the "reasoning" behind the ban.

G-lines are a variant of K-lines, which work in much the same way, except K-lines only disconnect clients on one server of the network. G-lines are normally applied to a user who has received a K-line on one server but continues to abuse the network by connecting via a different server. G-lines are often regarded as an extreme measure, only to be used in cases of repeated abuse when extensive attempts have been made to reason with the offending user. Therefore, especially on larger networks, often only very high ranking global IRC operators are permitted to set them, while K-lines, which are mostly regarded as a local affair, are left to the operators of the individual server in the network.

G-lines also work slightly differently from K-lines. G-lines are typically set as *@IPaddress or *@host, with the first being the better option. G-lines do still wait for an ident response from the connecting user (if specified in the banmask), but immediately close the socket once the user's IP address is compared to the G-line list and a match is found. If the *@host option is used, the server must conduct a reverse DNS lookup on the user and then compare the returned host to the hosts in the G-line list. This results in delay, and, if the DNS doesn't return correct results, the banned user may still get on the network.

## Z-line

A **Z-line** or **zap line** (also written **Z:line**) is similar to a K-line, but applied to a client's IP address range, and is considered to be used in extreme cases. Because a Z-line does not have to check usernames (identd) or *resolved* hostnames, it can be applied to a user before they send any data at all upon connection. Therefore a Z-line is more efficient and uses fewer resources than a K-line or G-line when banning large numbers of users.

Because not all IRCds are the same, others such as Charybdis use a D-line or X-line instead.

Z-lines are sometimes stored in the configuration file of the IRCd, although some networks, who handle lines through the IRC services, prefer to have them stored in their service's configuration files. Whenever a Z-lined person attempts to connect to the IRC network, either the services or the IRC daemon will automatically disconnect the client, often displaying a message explaining the reasoning behind the ban.

Z-lines are a variant of K-lines, which work in much the same way. Most Z-lines are "awarded" to people who abuse the network as a whole (on smaller networks, these are more frequently issued for isolated incidents).

Z-lines also work slightly differently than K-lines. Z-lines are typically set as *@IP or *@host, with the first being the better option. Z-lines do not wait for an ident response from the connecting user, but immediately close the socket once the user's IP is compared to the Z-line list and a match is found. If the *@host option is used, the server must conduct a reverse DNS lookup on the user and then compare the returned host to the hosts in the Z-line list. This can result in delays, or if the DNS doesn't return correctly, banned users could still get on the network. In actuality, the *@host option is completely against the intentions of using a Z-line, and therefore some IRCd programs will not allow anything other than *@IP, with wildcards (?,*) allowed in the IP section to block entire subnets. Another difference from K-lines (which affect only IRC clients) is if an IP is banned, nothing, not even other servers, can connect from this IP (or IP range, depending on the banmask).

One advantage to using Z-lines over K-lines and G-lines, from a server or network administrator's perspective, a Z-line uses less bandwidth than a K-line, mainly because it doesn't wait for an ident response or DNS lookup.

A disadvantage to using Z-line over K-line or G-line is that it becomes more difficult to ban entire ISPs and very dynamic IP addresses, common with some dialup and DSL connections. For example, if a network administrator wants to ban all of ISP example.com (with hypothetical IP address ranges of 68.0.0.0 - 68.255.255.255 and 37.0.0.0 - 38.255.255.255), a G-line regex could be *@*example.com, whereas Z-line would require *@37.*.*.*, *@38.*.*.*, and *@68.*.*.* to accomplish the same thing.

Z-lines can also be global, in which case they are called **GZ-lines**. GZ-lines work in the same manner as Z-lines, except that they propagate to every server on the network.

## Q-line

On some IRCds, such as UnrealIRCd, a **Q-line** forbids a nickname, or any nickname matching a given pattern. This is most often used to forbid use of services nicknames (such as 'X', or NickServ) or forbid use of IRC operator nicknames by non-operators. Some IRC daemons may disconnect users when initially applying the Q-line, whilst

others will force a nickname change, or do nothing until the user covered by the Q-line reconnects. Other IRCds, like Charybdis, use the 'RESV' command instead, with the stats letter remaining as Q.

## Chapter-7

# Data Center

An operation engineer overseeing a Network Operations Control Room of a data center.

A **data center** (or **data centre** or **datacentre** or **datacenter**) is a facility used to house computer systems and associated components, such as telecommunications and storage systems. It generally includes redundant or backup power supplies, redundant data communications connections, environmental controls (e.g., air conditioning, fire suppression) and security devices.

## *History*

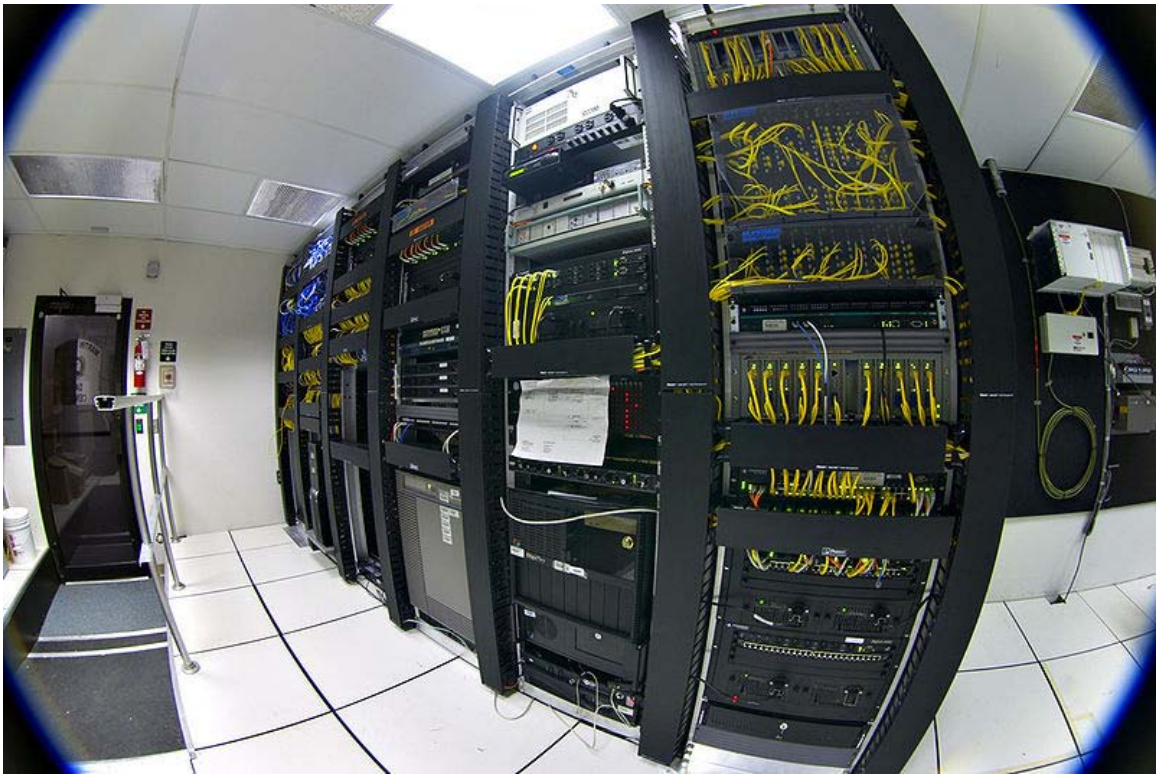
Data centers have their roots in the huge computer rooms of the early ages of the computing industry. Early computer systems were complex to operate and maintain, and required a special environment in which to operate. Many cables were necessary to connect all the components, and methods to accommodate and organize these were devised, such as standard racks to mount equipment, elevated floors, and cable trays (installed overhead or under the elevated floor). Also, old computers required a great deal of power, and had to be cooled to avoid overheating. Security was important – computers were expensive, and were often used for military purposes. Basic design guidelines for controlling access to the computer room were therefore devised.

During the boom of the microcomputer industry, and especially during the 1980s, computers started to be deployed everywhere, in many cases with little or no care about operating requirements. However, as information technology (IT) operations started to grow in complexity, companies grew aware of the need to control IT resources. With the advent of client-server computing, during the 1990s, microcomputers (now called "servers") started to find their places in the old computer rooms. The availability of inexpensive networking equipment, coupled with new standards for network cabling, made it possible to use a hierarchical design that put the servers in a specific room inside the company. The use of the term "data center," as applied to specially designed computer rooms, started to gain popular recognition about this time,

The boom of data centers came during the dot-com bubble. Companies needed fast Internet connectivity and nonstop operation to deploy systems and establish a presence on the Internet. Installing such equipment was not viable for many smaller companies. Many companies started building very large facilities, called Internet data centers (IDCs), which provide businesses with a range of solutions for systems deployment and operation. New technologies and practices were designed to handle the scale and the operational requirements of such large-scale operations. These practices eventually migrated toward the private data centers, and were adopted largely because of their practical results.

As of 2007, data center design, construction, and operation is a well-known discipline. Standard Documents from accredited professional groups, such as the Telecommunications Industry Association, specify the requirements for data center design. Well-known operational metrics for data center availability can be used to evaluate the business impact of a disruption. There is still a lot of development being done in operation practice, and also in environmentally-friendly data center design. Data centers are typically very expensive to build and maintain. For instance, Amazon.com's new 116,000 sq ft (10,800 $m^2$) data center in Oregon is expected to cost up to $100 million.

## *Requirements for modern data centers*



Racks of telecommunications equipment in part of a data center.

IT operations are a crucial aspect of most organizational operations. One of the main concerns is **business continuity**; companies rely on their information systems to run their operations. If a system becomes unavailable, company operations may be impaired or stopped completely. It is necessary to provide a reliable infrastructure for IT operations, in order to minimize any chance of disruption. Information security is also a concern, and for this reason a data center has to offer a secure environment which minimizes the chances of a security breach. A data center must therefore keep high standards for assuring the integrity and functionality of its hosted computer environment. This is accomplished through redundancy of both fiber optic cables and power, which includes emergency backup power generation.

Telcordia GR-3160, *NEBS Requirements for Telecommunications Data Center Equipment and Spaces*, provides guidelines for data center spaces within telecommunications networks, and environmental requirements for the equipment intended for installation in those spaces. These criteria were developed jointly by Telcordia and industry representatives. They may be applied to data center spaces housing data processing or Information Technology (IT) equipment. The equipment may be used to:

- Operate and manage a carrier's telecommunication network
- Provide data center based applications directly to the carrier's customers

- Provide hosted applications for a third party to provide services to their customers
- Provide a combination of these and similar data center applications.

Effective data center operation requires a balanced investment in both the facility and the housed equipment. The first step is to establish a baseline facility environment suitable for equipment installation. Standardization and modularity can yield savings and efficiencies in the design and construction of telecommunications data centers.

Standardization means integrated building and equipment engineering. Modularity has the benefits of scalability and easier growth, even when planning forecasts are less than optimal. For these reasons, telecommunications data centers should be planned in repetitive building blocks of equipment, and associated power and support (conditioning) equipment when practical. The use of dedicated centralized systems requires more accurate forecasts of future needs to prevent expensive over construction, or perhaps worse — under construction that fails to meet future needs.

## *Data center classification*

The *TIA-942:Data Center Standards Overview* describes the requirements for the data center infrastructure. The simplest is a Tier 1 data center, which is basically a server room, following basic guidelines for the installation of computer systems. The most stringent level is a Tier 4 data center, which is designed to host mission critical computer systems, with fully redundant subsystems and compartmentalized security zones controlled by biometric access controls methods. Another consideration is the placement of the data center in a subterranean context, for data security as well as environmental considerations such as cooling requirements.

The four levels are defined, and copyrighted, by the Uptime Institute, a Santa Fe, New Mexico-based think tank and professional services organization. The levels describe the availability of data from the hardware at a location. The higher the tier, the greater the accessibility. The levels are:

| Tier Level | Requirements |
|---|---|
| 1 | • Single non-redundant distribution path serving the IT equipment<br>• Non-redundant capacity components<br>• Basic site infrastructure guaranteeing 99.671% availability |
| 2 | • Fulfills all Tier 1 requirements<br>• Redundant site infrastructure capacity components guaranteeing 99.741% availability |

| | |
|---|---|
| 3 | • Fulfills all Tier 1 & Tier 2 requirements<br>• Multiple independent distribution paths serving the IT equipment<br>• All IT equipment must be dual-powered and fully compatible with the topology of a site's architecture<br>• Concurrently maintainable site infrastructure guaranteeing 99.982% availability |
| 4 | • Fulfills all Tier 1, Tier 2 and Tier 3 requirements<br>• All cooling equipment is independently dual-powered, including chillers and Heating, Ventilating and Air Conditioning (HVAC) systems<br>• Fault tolerant site infrastructure with electrical power storage and distribution facilities guaranteeing 99.995% availability |

## *Design considerations*

A typical server rack, commonly seen in colocation.

A data center can occupy one room of a building, one or more floors, or an entire building. Most of the equipment is often in the form of servers mounted in 19 inch rack cabinets, which are usually placed in single rows forming corridors (so-called aisles) between them. This allows people access to the front and rear of each cabinet. Servers differ greatly in size from 1U servers to large freestanding storage silos which occupy many tiles on the floor. Some equipment such as mainframe computers and storage devices are often as big as the racks themselves, and are placed alongside them. Very large data centers may use shipping containers packed with 1,000 or more servers each;

when repairs or upgrades are needed, whole containers are replaced (rather than repairing individual servers).

Local building codes may govern the minimum ceiling heights.

A bank of batteries in a large data center, used to provide power until diesel generators can start.

## Environmental control

The physical environment of a data center is rigorously controlled. Air conditioning is used to control the temperature and humidity in the data center. ASHRAE's "Thermal Guidelines for Data Processing Environments" recommends a temperature range of 16–24 °C (61–75 °F) and humidity range of 40–55% with a maximum dew point of 15°C as optimal for data center conditions. The temperature in a data center will naturally rise because the electrical power used heats the air. Unless the heat is removed, the ambient

temperature will rise, resulting in electronic equipment malfunction. By controlling the air temperature, the server components at the board level are kept within the manufacturer's specified temperature/humidity range. Air conditioning systems help control humidity by cooling the return space air below the dew point. Too much humidity, and water may begin to condense on internal components. In case of a dry atmosphere, ancillary humidification systems may add water vapor if the humidity is too low, which can result in static electricity discharge problems which may damage components. Subterranean data centers may keep computer equipment cool while expending less energy than conventional designs.

Modern data centers try to use economizer cooling, where they use outside air to keep the data center cool. Washington State now has a few data centers that cool all of the servers using outside air 11 months out of the year. They do not use chillers/air conditioners, which creates potential energy savings in the millions.

Telcordia GR-2930, *NEBS: Raised Floor Generic Requirements for Network and Data Centers*, presents generic engineering requirements for raised floors that fall within the strict NEBS guidelines.

There are many types of commercially available floors that offer a wide range of structural strength and loading capabilities, depending on component construction and the materials used. The general types of raised floors include stringerless, stringered, and structural platforms, all of which are discussed in detail in GR-2930 and summarized below.

- ***Stringerless Raised Floors*** - One non-earthquake type of raised floor generally consists of an array of pedestals that provide the necessary height for routing cables and also serve to support each corner of the floor panels. With this type of floor, there may or may not be provisioning to mechanically fasten the floor panels to the pedestals. This stringerless type of system (having no mechanical attachments between the pedestal heads) provides maximum accessibility to the space under the floor. However, stringerless floors are significantly weaker than stringered raised floors in supporting lateral loads and are not recommended.

- ***Stringered Raised Floors*** - This type of raised floor generally consists of a vertical array of steel pedestal assemblies (each assembly is made up of a steel base plate, tubular upright, and a head) uniformly spaced on two-foot centers and mechanically fastened to the concrete floor. The steel pedestal head has a stud that is inserted into the pedestal upright and the overall height is adjustable with a leveling nut on the welded stud of the pedestal head.

- ***Structural Platforms*** - One type of structural platform consists of members constructed of steel angles or channels that are welded or bolted together to form an integrated platform for supporting equipment. This design permits equipment to be fastened directly to the platform without the need for toggle bars or

supplemental bracing. Structural platforms may or may not contain panels or stringers.

## Electrical power

Backup power consists of one or more uninterruptible power supplies and/or diesel generators.

To prevent single points of failure, all elements of the electrical systems, including backup systems, are typically fully duplicated, and critical servers are connected to both the "A-side" and "B-side" power feeds. This arrangement is often made to achieve N+1 redundancy in the systems. Static switches are sometimes used to ensure instantaneous switchover from one supply to the other in the event of a power failure.

Data centers typically have raised flooring made up of 60 cm (2 ft) removable square tiles. The trend is towards 80–100 cm (31–39 in) void to cater for better and uniform air distribution. These provide a plenum for air to circulate below the floor, as part of the air conditioning system, as well as providing space for power cabling.

## Low-voltage cable routing

Data cabling is typically routed through overhead cable trays in modern data centers. But some are still recommending under raised floor cabling for security reasons and to consider the addition of cooling systems above the racks in case this enhancement is necessary. Smaller/less expensive data centers without raised flooring may use anti-static tiles for a flooring surface. Computer cabinets are often organized into a hot aisle arrangement to maximize airflow efficiency.

## Fire protection

Data centers feature fire protection systems, including passive and active design elements, as well as implementation of fire prevention programs in operations. Smoke detectors are usually installed to provide early warning of a developing fire by detecting particles generated by smoldering components prior to the development of flame. This allows investigation, interruption of power, and manual fire suppression using hand held fire extinguishers before the fire grows to a large size. A fire sprinkler system is often provided to control a full scale fire if it develops. Fire sprinklers require 18 in (46 cm) of clearance (free of cable trays, etc.) below the sprinklers. Clean agent fire suppression gaseous systems are sometimes installed to suppress a fire earlier than the fire sprinkler system. Passive fire protection elements include the installation of fire walls around the data center, so a fire can be restricted to a portion of the facility for a limited time in the event of the failure of the active fire protection systems, or if they are not installed. For critical facilities these firewalls are often insufficient to protect heat-sensitive electronic equipment, however, because conventional firewall construction is only rated for flame penetration time, not heat penetration. There are also deficiencies in the protection of vulnerable entry points into the server room, such as cable penetrations, coolant line

penetrations and air ducts. For mission critical data centers fireproof vaults with a Class 125 rating are necessary to meet NFPA 75 standards.

### Security

Physical security also plays a large role with data centers. Physical access to the site is usually restricted to selected personnel, with controls including bollards and mantraps. Video camera surveillance and permanent security guards are almost always present if the data center is large or contains sensitive information on any of the systems within. The use of finger print recognition man traps is starting to be commonplace.

## *Energy use*

Energy use is a central issue for data centers. Power draw for data centers ranges from a few kW for a rack of servers in a closet to several tens of MW for large facilities. Some facilities have power densities more than 100 times that of a typical office building. For higher power density facilities, electricity costs are a dominant operating expense and account for over 10% of the total cost of ownership (TCO) of a data center. By 2012 the cost of power for the data center is expected to exceed the cost of the original capital investment.

### Greenhouse gas emissions

In 2007 the entire information and communication technologies or ICT sector was estimated to be responsible for roughly 2% of global carbon emissions with data centers accounting for 14% of the ICT footprint. The US EPA estimates that servers and data centers are responsible for up to 1.5% of the total US electricity consumption, or roughly .5% of US GHG emissions, for 2007. Given a business as usual scenario greenhouse gas emissions from data centers is projected to more than double from 2007 levels by 2020.

Siting is one of the factors that affect the energy consumption and environmental effects of a datacenter. In areas where climate favors cooling and lots of renewable electricity is available the environmental effects will be more moderate. Thus countries with favorable conditions, such as Finland, Sweden and Switzerland, are trying to attract cloud computing data centers.

In an 18-month investigation by scholars at Rice University's Baker Institute for Public Policy in Houston and the Institute for Sustainable and Applied Infodynamics in Singapore, data center-related emissions will more than triple by 2020.

### Energy efficiency

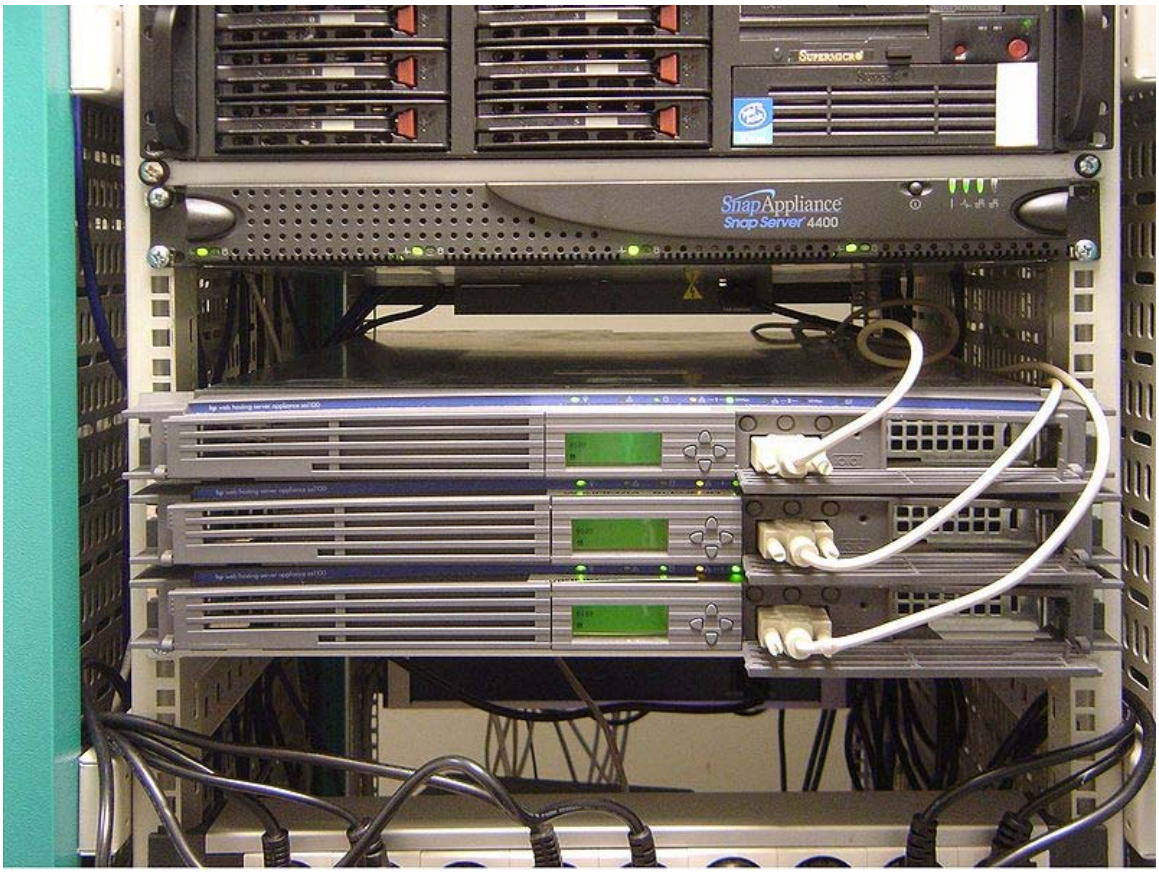
The most commonly used metric to determine the energy efficiency of a data center is power usage effectiveness, or PUE. This simple ratio is the total power entering the data center divided by the power used by the IT equipment.

$$\mathrm{PUE} = \frac{\text{Total Facility Power}}{\text{IT Equipment Power}}$$

Power used by support equipment, often referred to as overhead load, mainly consists of cooling systems, power delivery, and other facility infrastructure like lighting. The average data center in the US has a PUE of 2.0, meaning that the facility uses one Watt of overhead power for every Watt delivered to IT equipment. State-of-the-art data center energy efficiency is estimated to be roughly 1.2. Some large data center operators like Microsoft and Yahoo! have published projections of PUE for facilities in development; Google publishes quarterly actual efficiency performance from data centers in operation.

The U.S. Environmental Protection Agency has an Energy Star rating for standalone or large data centers. To qualify for the ecolabel, a data center must be within the top quartile of energy efficiency of all reported facilities.

## *Network infrastructure*




An example of "rack mounted" servers.

Communications in data centers today are most often based on networks running the IP protocol suite. Data centers contain a set of routers and switches that transport traffic between the servers and to the outside world. Redundancy of the Internet connection is often provided by using two or more upstream service providers.

Some of the servers at the data center are used for running the basic Internet and intranet services needed by internal users in the organization, e.g., e-mail servers, proxy servers, and DNS servers.

Network security elements are also usually deployed: firewalls, VPN gateways, intrusion detection systems, etc. Also common are monitoring systems for the network and some of the applications. Additional off site monitoring systems are also typical, in case of a failure of communications inside the data center.

## *Applications*

Multiple racks of servers, and how a data center commonly looks.

The main purpose of a data center is running the applications that handle the core business and operational data of the organization. Such systems may be proprietary and developed internally by the organization, or bought from enterprise software vendors. Such common applications are ERP and CRM systems.

A data center may be concerned with just operations architecture or it may provide other services as well.

Often these applications will be composed of multiple hosts, each running a single component. Common components of such applications are databases, file servers, application servers, middleware, and various others.

Data centers are also used for off site backups. Companies may subscribe to backup services provided by a data center. This is often used in conjunction with backup tapes. Backups can be taken of servers locally on to tapes. However, tapes stored on site pose a security threat and are also susceptible to fire and flooding. Larger companies may also send their backups off site for added security. This can be done by backing up to a data center. Encrypted backups can be sent over the Internet to another data center where they can be stored securely.

For disaster recovery, several large hardware vendors have developed mobile solutions that can be installed and made operational in very short time. Vendors such as Cisco Systems, Sun Microsystems, IBM, and HP have developed systems that could be used for this purpose.

## Chapter-8

# File Server

In computing, a **file server** is a computer attached to a network that has the primary purpose of providing a location for shared disk access, i.e. shared storage of computer files (such as documents, sound files, photographs, movies, images, databases, etc.) that can be accessed by the workstations that are attached to the computer network. The term *server* highlights the role of the machine in the client–server scheme, where the *clients* are the workstations using the storage. A file server is usually not performing any calculations, and does not run any programs on behalf of the clients. It is designed primarily to enable the rapid storage and retrieval of data where the heavy computation is provided by the workstations.

File servers are commonly found in schools and offices and rarely seen in local internet service providers using LAN to connect their client computers.

## *History of file servers*

Novell proposed an approach using software to connect each workstation to a network file server that would manage both the network and access to network resources. Novell grew upon the strength of its Netware operating system, used for file serving, and by the late 1980s had a 50% market share of local area networks.

## *Types of file servers*

A file server may be dedicated or non-dedicated. A dedicated server is generally designed specifically for use as a file server, with workstations attached for reading and writing files and databases. File servers may also be categorized by the method of access: Internet file servers are frequently accessed by File Transfer Protocol (FTP) or by HTTP (but are different from web servers, that often provide dynamic web content in addition to static files). Servers on a LAN are usually accessed by SMB/CIFS protocol (Windows and Unix-like) or NFS protocol (Unix-like systems). Database servers, that provide access to a shared database via a database device driver, are *not* regarded as file servers. Most file servers are simultaneously print servers too, as they provide access to printers via network. A single file serving computer may be accessible by multiple means: it may run an FTP server, an SMB server, etc., serving the same files.

## *Design of file servers*

In actually In modern businesses the design of file servers is complicated by competing demands for storage space, access speed, recoverability, ease of administration, security, and budget. This is further complicated by a constantly changing environment, where new hardware and technology rapidly obsolesces old equipment, and yet must seamlessly come online in a fashion compatible with the older machinery. To manage throughput, peak loads, and response time, vendors may utilize queuing theory to model how the combination of hardware and software will respond over various levels of demand. Servers may also employ dynamic load balancing scheme to distribute requests across various pieces of hardware.

The primary piece of hardware equipment for servers over the last couple of decades has proven to be the hard disk drive. Although other forms of storage are viable (such as magnetic tape and solid-state drives) disk drives have continued to offer the best fit for cost, performance, and capacity.

### Storage

Since the crucial function of a file server is storage, technology has been developed to operate multiple disk drives together as a team, forming a disk array. A disk array typically has cache (temporary memory storage that is faster than the magnetic disks), as well as advanced functions like RAID and storage virtualization. Typically disk arrays increase level of availability by using redundant components other than RAID, such as power supplies. Disk arrays may be consolidated or virtualized in a storage area network (SAN).

### Network-attached storage

Network-attached storage (NAS) is file-level computer data storage connected to a computer network providing data access to heterogeneous clients. As of 2010 NAS devices are gaining popularity, as a convenient method of sharing files between multiple computers. Potential benefits of network-attached storage, compared to non-dedicated file servers, include faster data access, easier administration, and simple configuration.

NAS systems are networked appliances which contain one or more hard drives, often arranged into logical, redundant storage containers or RAID arrays. Network Attached Storage removes the responsibility of file serving from other servers on the network. They typically provide access to files using network file sharing protocols such as NFS, SMB/CIFS (Server Message Block/Common Internet File System), or AFP.

### Security

File servers generally offer some form of system security to limit access to files to specific users or groups. In large organizations, this is a task usually delegated to what is

known as directory services such as openLDAP, Novell's eDirectory or Microsoft's Active Directory.

These servers work within the hierarchical computing environment which treat users, computers, applications and files as distinct but related entities on the network and grant access based on user or group credentials. In many cases, the directory service spans many file servers, potentially hundreds for large organizations. In the past, and in smaller organizations, authentication can take place directly to the server itself.

## Performance

While file servers can provide an organization with a highly centralized method of sharing data, they frequently run up against performance bottlenecks that limit the speed at which data can be passed to and from the network clients.

For example, while gigabit backbones are common in organizations, as of 2009 it is still difficult for a single server using SAS disk storage to fully utilize a gigabit link at maximum speed, often only achieving up to 750 megabit of the rated link speed. Ten-gigabit backbones generally cannot be fully utilized by a single file server, but instead are used to aggregate client data across multiple servers operating in parallel.

Network performance also hindered by the desktop client computer, which has generally lower performance memory and storage than most servers, such that 100 megabit to the desktop is still considered acceptable for business networks. Many desktop computers are unable to achieve much greater than 250 megabit, using gigabit server connections.

Part of the difficulty is that current mechanical disk technology slows down severely when very small files are involved, due to relatively slow head seeking speed between sectors on the storage device. A desktop computer downloading a Microsoft windows roaming profile on a gigabit link can drop to no more than 20 megabit speed when downloading the user's cookies, favorites, and recent files list, which can consist of thousands of individual files in a roaming profile, with each file less than 4 kilobytes in size.

As desktop and server storage moves towards solid state technology, these smaller files can be read and written much more quickly because sector seek time does not apply to solid-state storage.

## Chapter-9

# Home Server

A **home server** is a server located in a private residence providing services to other devices inside and/or outside the household through a home network and/or the internet. Such services may include file and/or printer serving, media center serving, web serving, web caching, account authentication and backup services. Because of the relatively low number of computers on a typical home network, a home server commonly does not require significant computing power. Often, users reuse older systems, and home servers with specifications as low as 1 GHz CPU and 256 MB of RAM can be used. Large, preferably fast hard drives (ATA-100 or Serial ATA) and a network interface card are usually all the hardware required for home file serving. An uninterruptible power supply is recommended in case of power outages that can possibly corrupt data.

## *Commercial home server products*

A common type of home server is the Plug computer form factor. Most of these are small ARM-based devices running Linux, these have an integrated AC to DC power converter and come pre-loaded with various server applications.

## *Operating systems*

Home servers run many different operating systems. Enthusiasts who build their own home servers can use whatever OS is conveniently available or familiar to them, such as Microsoft Windows, Mac OS X, GNU/Linux, Solaris or BSD.

## *Services provided by home servers*

### Administration and configuration

Home servers often run headless, and can be administered remotely through a command shell, or graphically through a remote desktop system such as RDP, VNC, Webmin, or many others.

Some home server operating systems, such as Windows Home Server include a consumer-focused graphical user interface for setup and configuration that is available on home computers on the home network (and remotely over the Internet via remote access). Others simply enable users to use native operating system tools for configuration.

## Centralized storage

Home servers often act as network-attached storage providing the major benefit that all users' files can be centrally and securely stored, with flexible permissions applied to them. Such files can be easily accessed from any other system on the network, provided the correct credentials are supplied. This also applies to shared printers.

Such files can also be shared over the internet to be accessible from anywhere in the world using remote access.

Servers running UNIX or Linux with the free Samba suite (or certain Windows Server products - Windows Home Server excluded) can provide domain control, custom logon scripts, and roaming profiles to users of certain versions of Windows. This allows a user to log on from any machine in the domain and have access to his/her "My Documents" and personalized Windows and application preferences - multiple accounts on each computer in the home are not needed.

## Media serving

Home servers are often used to serve multi-media content, including photos, music, and video to other devices in the household (and even to the Internet). Using standard protocols such as DLNA or proprietary systems such as iTunes users can access their media stored on the home server from any room in the house. Windows XP Media Center Edition, Windows Vista, and Windows 7 can act as a home server, supporting a particular type of media serving that streams the interactive user experience to Media Center Extenders including the Xbox 360.


A typical MythTV menu.

Windows Home Server supports media streaming to Xbox 360 and other DLNA based media receivers via the built-in Windows Media Connect technology. Some Windows Home Server device manufacturers such as Hewlett-Packard extend this functionality with a full DLNA implementation such as PacketVideo TwonkyMedia server.

On a Linux server, there are many free, open-source, fully-functional, all-in-one software solutions for media serving available. One such program is LinuxMCE, which allows other devices to boot off a hard drive image on the server, allowing them to become appliances such as set-top boxes. Amahi is a free Linux Home Server that provides shared storage, automated backups, secure VPN, and shared applications like calendar and Asterisk, Xine, MythTV (another media serving solution), VideoLAN, SlimServer, DLNA, and many other open-source projects are fully integrated for a seamless home theater/automation/telephony experience.

Because a server is typically always on, it is often a more logical choice to put a TV tuner or radio tuner for recording broadcasts into a server, than it is to use e.g. a desktop for recording, as it allows recording to be scheduled at any time.

On an Apple Macintosh server, options include iTunes, PS3 Media Server, Twonky Media Server, Nullriver MediaLink, and Elgato EyeConnect. Additionally, for Macs directly connected to TVs, the built-in FrontRow program or Boxee can act as a full-featured media center interface.

Some home servers provide remote access to media and entertainment content.

## Remote access


The Webmin Interface as it would appear in a standard browser.

A home server can be used to provide remote access into the home from devices on the Internet, using remote desktop software and other remote administration software. For example, Windows Home Server provides remote access to files stored on the home server via a web interface as well as remote access to Remote Desktop sessions on PCs in the house. Similarly, Tonido provides direct access via a web browser from the internet without requiring any port forwarding or other setup. Some enthusiasts often use VPN technologies as well.

On a Linux server, two popular tools are (among many) VNC and Webmin. VNC allows clients to remotely view a server GUI desktop as if the user was physically sitting in front of the server. A GUI need not be running on the server console for this to occur; there can be multiple 'virtual' desktop environments open at the same time. Webmin allows users to control many aspects of server configuration and maintenance all from a simple web interface. Both can be configured to be accessed from anywhere on the internet.

Servers can also be accessed remotely using the command line-based Telnet and SSH protocols.

### Web serving

Some users choose to run a web server in order to share files easily and publicly (or privately, on the home network). Others set up web pages and serve them straight from their home, although this may be in violation of some ISPs terms of service. Sometimes these webservers are run on a nonstandard port in order to avoid the ISP's port blocking. Example web servers used on home servers include Apache and IIS.

### Web proxy

Some networks have an HTTP proxy which can be used to speed up web access when multiple users visit the same websites, and to get past blocking software while the owner is using the network of some institution that might block certain sites. Public proxies are often slow and unreliable and so it is worth the trouble of setting up one's own private proxy.

Some proxies can be configured to block websites on the local network if it is set up as a transparent proxy.

### E-mail

Many home servers also run e-mail servers that handle e-mail for the owner's domain name. The advantages are having much bigger mailboxes and maximum message size than most commercial e-mail services. Access to the server, since it is on the local network is much faster than using an external service. This also increases security as e-mails do not reside on an off-site server.

## BitTorrent

Home servers are ideal for utilizing the BitTorrent protocol for downloading and seeding files as some torrents can take days, or even weeks to complete and perform better on an uninterrupted connection. There are many command-line based clients such as rTorrent and web-based ones such as TorrentFlux and Tonido available for this purpose. BitTorrent also makes it easier for those with limited bandwidth to distribute large files over the internet.

## Gopher

An unusual service is the Gopher protocol, a hypertext document retrieval protocol which pre-dated the World Wide Web and was popular in the early 1990s. Many of the remaining gopher servers are run off home servers utilizing PyGopherd and the Bucktooth gopher server.

## Home automation

Home automation requires a device in the home that is available 24/7. Often such home automation controllers are run on a home server.

## Security monitoring

Relatively low cost CCTV DVR solutions are available that allow recording of video cameras to a home server for security purposes. The video can then be viewed on PCs or other devices in the house.

A series of cheap Universal serial bus-based webcams can be connected to a home server as a makeshift CCTV system. Optionally these images and video streams can be made available over the internet using standard protocols.

## Family applications

Home servers can act as a host to family oriented applications such as a family calendar, to-do lists, and message boards.

## IRC and instant messaging

Because a server is always on, an IRC client or IM client running on it will be highly available to the Internet. This way, the chat client will be able to record activity that occurs even while the user is not at the computer, e.g. asleep or at work or school. Textual clients such as Irssi and tmsnc can be detached using GNU Screen for example, and graphical clients such as Pidgin can be detached using xmove. Quassel provides a specific version for this kind of use. Home servers can also be used to run personal XMPP servers and IRC servers as these protocols can support a large number of users on very little bandwidth

## Online gaming

Some multiplayer games such as Continuum, and Tremulous have server software available which users may download and use to run their own private game server. Some of these servers are password protected, so only a selected group of people such as clan members can gain access to the server. Others are open for public use and may move to colocation or other forms of paid hosting if they gain a large number of players.

## 3rd Party Platform

Home servers often are platforms that enable 3rd party products to be built and added over time. For example Windows Home Server provides a Software Development Kit and over 60 3rd party products are available for it. Similarly Tonido provides an application platform that can be extended by writing new applications using their SDK.

Chapter-10

# Server Farm and Sound Server

## Server farm

A row of racks in a server farm.

A **server farm** or **server cluster** is a collection of computer servers usually maintained by an enterprise to accomplish server needs far beyond the capability of one machine. Server farms often have backup servers, which can take over the function of primary servers in the event of a primary server failure. Server farms are typically co-located with the network switches and/or routers which enable communication between the different parts of the cluster and the users of the cluster. The computers, routers, power supplies,

and related electronics are typically mounted on 19-inch racks in a server room or data center.

## Applications

This server farm supports the various computer networks of the Joint Task Force Guantanamo

Server farms are commonly used for cluster computing. Many modern supercomputers comprise giant server farms of high-speed processors connected by either Gigabit Ethernet or custom interconnects such as Infiniband or Myrinet. Web hosting is a common use of a server farm; such a system is sometimes collectively referred to as a *web farm*. Other uses of server farms include scientific simulations (such as computational fluid dynamics) and the rendering of 3D computer generated imagery.

Server farms are increasingly being used instead of or in addition to mainframe computers by large enterprises, although server farms do not yet reach the same reliability levels as mainframes. Because of the sheer number of computers in large server farms, the failure of individual machines is a commonplace event, and the management of large server farms needs to take this into account, by providing support for redundancy, automatic failover, and rapid reconfiguration of the server cluster.

## Performance

The performance of the very largest server farms (thousands of processors and up) is typically limited by the performance of the data center's cooling systems and the total electricity cost rather than by the performance of the processors. A computer that runs 24/7 consumes (over its lifetime) electricity worth many times its initial purchase cost. For this reason, the critical design parameter for both large and continuous systems tends to be performance per watt, rather than cost of peak performance or (peak performance / (unit * initial cost)). Also, for high availability systems that must run 24/7 (unlike supercomputers that can be power-cycled to demand, and also tend to run at much higher utilizations), there is more attention placed on power saving features such as variable

clock-speed and the ability to turn off both computer parts, processor parts, and entire computers (WoL and virtualization) according to demand without bringing down services.

### Performance per watt

The EEMBC EnergyBench, SPECpower, and the Transaction Processing Performance Council TPC-Energy are benchmarks designed to predict performance per watt in a server farm. The power used by each rack of equipment can be measured at the power distribution unit. Some servers include power tracking hardware so the people running the server farm can measure the power used by each server. The power used by the entire server farm may be reported in terms of power usage effectiveness or data center infrastructure efficiency.

According to some estimates, for every 100 watts spent on running the servers, roughly another 50 watts is needed to cool them. Iceland, which has a cold climate all year as well as cheap and carbon-neutral geothermal electricity supply, is building its first site. Fibre optic cables are being laid from Iceland to North America and Europe to enable companies there to locate their servers in Iceland.

Siting is one of the factors that affect the energy consumption and environmental effects of a server farm. In areas where climate favors cooling and lots of renewable electricity is available the environmental effects will be more moderate. Thus countries with favorable conditions, such as Finland, Sweden and Switzerland, are trying to attract cloud computing data centers.

# Sound server

A **sound server** is software that manages the use of and access to audio devices, most notably, the soundcard. It usually runs as a background process. The term could also apply to a complete computer which is in a server role, dedicated to audio streaming or a networked or stand-alone appliance for playing sounds and sound files.

## *Sound server in an operating system*

In a Unix-like operating system, the main task of a sound server is performing the mix of different data streams and send out a single unified audio output device of system. This mixture is usually done by software, or hardware if there is a supported sound card.

### Layers

The "sound stack" can be visualized as follows, with programs in the upper layers calling elements in the lower layers:

- Applications (e.g. mp3 player, web video)

- Sound server (e.g. aRts, ESD, JACK, PulseAudio)
- Sound subsystem (described as kernel modules or drivers; e.g. OSS, ALSA)
- Operating system kernel (e.g. Linux, Unix)

### Motivation

Sound servers appeared in Unix-like operating systems after limitations in Open Sound System were recognized. OSS is a basic sound interface that was incapable of playing multiple streams simultaneously, dealing with multiple sound cards, or streaming sound over the network.

A sound server can provide these features by running as a daemon. It receives calls from different programs and sound flows, mixes the streams, and sends raw audio out to the audio device.

With a sound server, users can also configure global and per-application sound preferences.

### Diversification and problems

Currently, there are multiple sound servers; some are focused on providing very low latency while others concentrate on features suitable for general desktop systems. While diversification allows a user to choose just the features that are important to a particular application, it also forces developers to accommodate these options by necessitating code that is compatible with the various sound servers available. Consequently, this variety has resulted in a desire for a standard API to unify efforts.

## *List of sound servers*

- aRts
- aucat - OpenBSD audio server
- Bergen Sound Server
- JACK
- Enlightened Sound Daemon
- Network Audio System
- PulseAudio

### Streaming

- SHOUTcast
- Icecast

## Chapter-11

# Squeezebox Server and yDecode

# Squeezebox Server

Squeezebox Server

Squeezebox Server Web Interface

| | |
|---|---|
| **Developer(s)** | Logitech |
| **Stable release** | 7.5.3 |
| **Written in** | Perl |
| **Operating system** | Linux, Windows, Mac OS X, BSD, Solaris |
| **Type** | Streaming audio server |
| **License** | GNU General Public License |

**Squeezebox Server** (formerly SlimServer and SqueezeCenter) is a streaming audio server supported by Logitech (formerly Slim Devices), developed in particular to support their Squeezebox range of digital audio receivers.

The software is designed for streaming music over a network, allowing users to play their music collections from virtually anywhere there is an Internet connection. It supports a large number of audio formats including MP3, FLAC, WAV, Ogg, and AAC, as well as transcoding. It can stream to both software and hardware receivers, including the various

Squeezebox models, as well as any media player capable of playing MP3 streams. Plugins from Logitech and third-party sources are also supported, allowing additional functionality to be added, and there is integration with Logitech's mysqueezebox.com online service.

Squeezebox Server is open source software, released under the terms of the GNU General Public License.

## *Compatible Players*

### Hardware


An O2 Joggler running SqueezePlay

Logitech's own Squeezebox hardware players exist in a variety of configurations, offering wired and wireless Ethernet, analog and digital audio outputs, touchscreen interfaces and a variety of remote controlled options.

Squeezebox Server also works with networked music players such as the Roku SoundBridge M1001, although Logitech does not officially support these competing products. Chumby devices also support streaming music from a Squeezebox Server, as does the Rio Receiver when running replacement software (slimrio) to emulate the SliMP3 device, although it is limited to modest bitrates (<128kps).

Recently the O2 Joggler has proven a popular device for running Logitech's open source SqueezePlay software, providing a similar interface to the Squeezebox Touch on a 7" display.

### Software

SqueezePlay is based on SqueezeOS, the operating system that drives the hardware devices *Squeezebox Duet, Radio* and *Touch.* Written in Lua, it is also open source software and sees regular updates through Logitech's SVN releases. There is also a free software emulator version of the Squeezebox, called Softsqueeze, which is written in Java and can be run easily as an applet inside a web page. A third player, SqueezeSlave, is also available, which operates similarly but without any display. SqueezeSlave is designed to be run on a server connected to an amplifier/speakers, and can be controlled through the standard Squeezebox Server web interface. At this time, SqueezeSlave is incompatible with Logitech's Spotify plugin due to a lack of support for 'direct streaming'.

Other popular software players capable of playing music from Squeezebox Server include, among others, Apple's iTunes.

## *Server Hardware and Plugins*

The Squeezebox Server software is written in Perl, and will run on Linux, Microsoft Windows, Apple Macintosh, BSD platforms.

Squeezebox Server itself can run on a number of NAS devices, such as QNAP Turbo NAS, Synology Disk Station, NETGEAR ReadyNAS, Buffalo Linkstation, Linksys NSLU2, Xtreamer eTRAYz and any device running FreeNAS software. Squeezebox Server also comes pre-installed on the VortexBox Linux distribution and VortexBox appliance. This generally results in lower energy consumption than running Squeezebox Server on a personal computer whilst offering the same feature set (albeit with a slightly less responsive web interface under certain circumstances). Some NAS devices may require more effort than others to get Squeezebox Server running, though. Logitech only supports the Netgear ReadyNAS NAS devices.

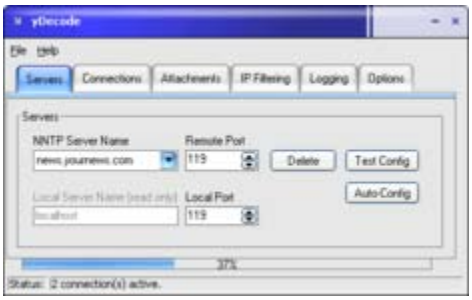
More information on NAS-based Logitech solutions can be found in the Logitech Squeezebox forum for third-party hardware.

### Plugins

There are numerous plug-ins and device drivers available for Squeezebox Server, which include features such as support for automation systems from AMX LLC and Crestron Electronics. Plugins also provide access to additional services, such as the live radio and 'listen-again' features of BBC iPlayer in the UK.

# yDecode

yDecode




yDecode screenshot

| | |
|---|---|
| **Developer(s)** | yDecode |
| **Stable release** | 1.72 / June 22, 2010 |
| **Operating system** | Microsoft Windows |
| **Type** | Network Utility |
| **License** | Proprietary |

yDecode is a NNTP proxy server for the Windows operating system. yDecode works in the background and converts yEnc encoded attachments to UU encoded attachments on the fly. It also automatically joins multipart posts into a single part post. This way popular news readers like Outlook Express, Windows Mail, Windows Live Mail or Mozilla Thunderbird get additional functionality.

## *Overview*

yDecode is an application that works as an intermediate between a news reader client (e.g. Outlook Express) and news server, therefore similar in technical functionality to proxy server software. However, this is not its main purpose - it recognizes and decodes yEnc messages and converts them into UU format which is then sent to a newsreader. The proxy mechanism is chosen purely for compatibility reasons so that it could be applied to any news reader. Even though yDecode can be installed on any computer and accessed from another computer usually it is installed on a computer where a newsreader resides and accessed via localhost address.

While some newsreaders have native yEnc support, some of the most popular ones don't and yDecode has been designed to enable a user to continue using his preferred news client rather than switching to a new one. It supports automatic configuration of Outlook

Express, Windows Mail, Windows Live Mail and Mozilla Thunderbird but works with any other newsreader.

## Features

yDecode has the following features:

1. Decodes yEnc encoded attachments and sends them to client as UU encoded
2. Allows news client to download all parts of a multipart post as if it were a single article
3. Saves attachments automatically to a user chosen folder
4. Monitoring of connections
5. Supports SSL (Secure Sockets Layer) connections
6. Automatic progress indicator for single or multipart posts
7. Built-in wizard for automatic configuration and testing of Outlook Express, Windows Mail, Windows Live Mail and Mozilla Thunderbird accounts
8. Support for Windows Vista and Windows 7

It has been successfully tested with the following popular newsreaders:

1. Outlook Express
2. Windows Mail
3. Windows Live Mail
4. Mozilla Thunderbird
5. Agent (older versions that doesn't support yEnc)
6. Netscape News

Additionally, it works with other standards-complying newsreaders.

## History

First version of yDecode was released in 2004 and had a basic functionality as a yEnc decoder. Since then it has been gradually improved and simplified.

In version 1.3 an automatic multipart joiner has been added. In the following releases this feature has been made more compatible with the majority of newsgroup posts.

Today, yDecode is one of the most popular yEnc decoders and is being recommended by some of the largest Usenet providers like Giganews, Uncensored Newsfeeds and others.

## How yDecode works

yDecode stands in-between newsgroups server and a newsreader. It is installed usually on a computer that also runs a newsreader and must be running when newsgroups are accessed.

Once a newsreader connects, it redirects all the NNTP commands to the server and awaits for a response. When the response comes, yDecode analyzes and modifies the traffic. The main issues it works with are yEnc attachments and multipart split posts.

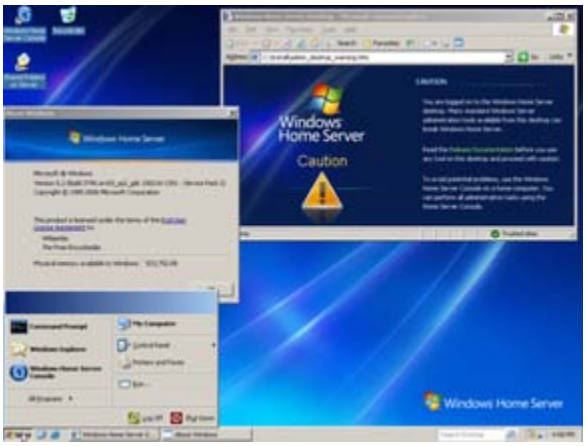
yEnc attachments are converted to standardized UU format which most newsreaders can easily decode and display. Multipart download support simplifies downloading big posts that are split into many parts on news servers. yDecode allows any news client to download all parts as if it was a single article.

## Chapter-12

# Windows Home Server

### Windows Home Server

Part of the Microsoft Windows family




Screenshot of the Windows Home Server Desktop

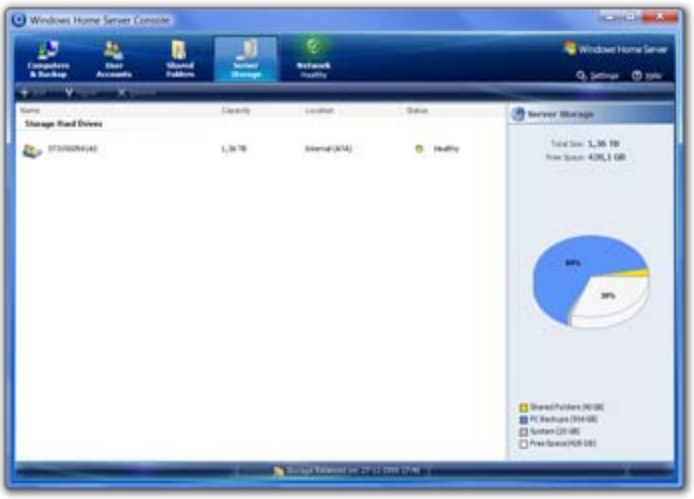
| Developer | |
|---|---|
| Microsoft | |
| **Releases** | |
| **Release date** | 4 November 2007; 3 years ago (info) |
| **Current version** | 6.0.2423.0 (Power Pack 3) (24 November 2009; 16 months ago) (info) |
| **Source model** | Closed source / Shared source |
| **License** | Proprietary commercial software |
| **Kernel type** | Hybrid |
| **Support status** | |

Mainstream support until 8 January 2013

**Windows Home Server**, code-named *Quattro*, is a home server operating system from Microsoft. Announced on 7 January 2007, at the Consumer Electronics Show by Bill Gates, Windows Home Server is intended to be a solution for homes with multiple connected PCs to offer file sharing, automated backups, and remote access. It is based on Windows Server 2003 Service Pack 2.

Windows Home Server was released to manufacturing on 16 July 2007 and officially released on 4 November 2007. Power Pack 1 for Windows Home Server was released 20 July 2008, Power Pack 2 was released 24 March 2009, and Power Pack 3 was released 24 November 2009. The next version of Windows Home Server, Windows Home Server 2011 will be released in April 2011.

## *Features*



Windows Home Server Console

- **Centralized backup** - Allows backup of up to 10 PCs, using Single-instance storage technology to avoid multiple copies of the same file, even if that file exists on multiple PCs.
- **Health monitoring** - Can centrally track the health of all PCs on the network, including antivirus and firewall status.
- **File sharing** - Offers network shares for computers to store the files remotely, acting as a network-attached storage device. Separate categories are provided for common file types like Documents, Music, Pictures and Videos. The files are indexed for fast searching.
- **Printer sharing** - Allows a centralized print server to handle print jobs for all users.
- **Shadow Copy** - Takes advantage of Volume Shadow Copy Service to take point in time snapshots that allow older versions of files to be recovered.

- **Headless operation** - No monitor or keyboard is required to manage the device. Remote administration is performed by using the *Windows Home Server Console* client software provided in the bundle. Also supports Remote Desktop Services connections to the server while connected to the same LAN.
- **Remote access gateway** - Allows remote access to any connected PC on the network, including the server itself, over the Internet.
- **Media streaming** - Can stream media to an Xbox 360 or other devices supporting Windows Media Connect.
- **Selective data redundancy** - Guards against a single drive failure by duplicating selected data across multiple drives.
- **Expandable storage** - Provides a unified single and easily expandable storage space, removing the need for drive letters.
- **Extensibility through add-Ins** - Add-Ins allow third-party developers to extend the features and functionality of the server. Add-Ins can be developed using the Windows Home Server SDK, to provide additional services to the client computers or work with the data already on the server. Add-Ins can also be ASP.NET applications, hosted in IIS 6 running on WHS.
- **Server backup** - Backs up files which are stored within shared folders on the server to an external hard drive.

## *Technology*

Windows Home Server is built on the same codebase as Windows Server 2003 SP2. It includes almost all technologies found in Windows Server 2003 SP2 but has been modified in some areas to remove or limit features. It also includes some new capabilities not found in Windows Server 2003 SP2:

### Home Server Console

While the underlying operating system is built on Windows Server 2003 with Service Pack 2, the configuration interface is designed to be user friendly enough that it can be set up without prior knowledge of server administration. The configuration interface, called the *Home Server Console*, is delivered as a Remote Desktop Protocol application to remote PCs - while the application runs on the server itself, the UI is rendered on the remote system. The Home Server Console client application can be accessed from any Windows PC. The server itself requires no video card or peripherals; it is designed to require only an Ethernet card and at least one Windows XP, Windows Vista or Windows 7 computer.

### Drive Extender

**Windows Home Server Drive Extender** is a file-based replication system that provides three key capabilities:

- Multi-disk redundancy so that if any given disk fails, data is not lost

- Arbitrary storage expansion by supporting any type of hard disk drive (Serial ATA, USB, FireWire etc.) in any mixture and capacity — similar in concept to JBOD
- A single folder namespace (no drive letters)

Users (specifically those who configure a family's home server) deal with storage at two levels: Shared Folders and Disks. The only concepts relevant regarding disks is whether they have been "added" to the home server's storage pool or not and whether the disk appears healthy to the system or not. This is in contrast with Windows' Logical Disk Manager which requires a greater degree of technical understanding in order to correctly configure a RAID array.

Shared Folders have a name, a description, permissions, and a flag indicating whether duplication (redundancy) is on or off for that folder.

If duplication is on for a Shared Folder (which is the default on multi-disk Home Server systems and not applicable to single disk systems) then the files in that Shared Folder are duplicated and the effective storage capacity is halved. However, in situations where a user may not want data duplicated (e.g. TV shows that have been archived to a Windows Home Server from a system running Windows Media Center), Drive Extender provides the capability to not duplicate such files if the server is short on capacity or manually mark a complete content store as not for duplication.

Additional information can be found on the *Windows Home Server Technical Brief for Drive Extender*.

A known limitation of Drive Extender is that it in some cases changes date/timestamp of directories and files when data is moved around between disks. According to Microsoft this is currently expected behaviour. This causes unexpected behaviour when using clients that sort media based on date. Examples are Xbmc, MediaPortal, Squeezebox Server. The aforementioned programs will work fine with WHS; however, files may appear out of order due to this caveat.

**Drive Extender Cancellation**

On 23 November 2010, Microsoft announced that Drive Extender would be removed from the next version of Windows Home Server code name "Vail". This announcement has led to public outcry in the announcement's comments section. Criticism of Drive Extender's removal is mainly related to it being seen as a core feature of Windows Home Server and a key reason for adoption. As a replacement for Drive Extender, Microsoft states that OEMs will use RAID on their Windows Home Server products.

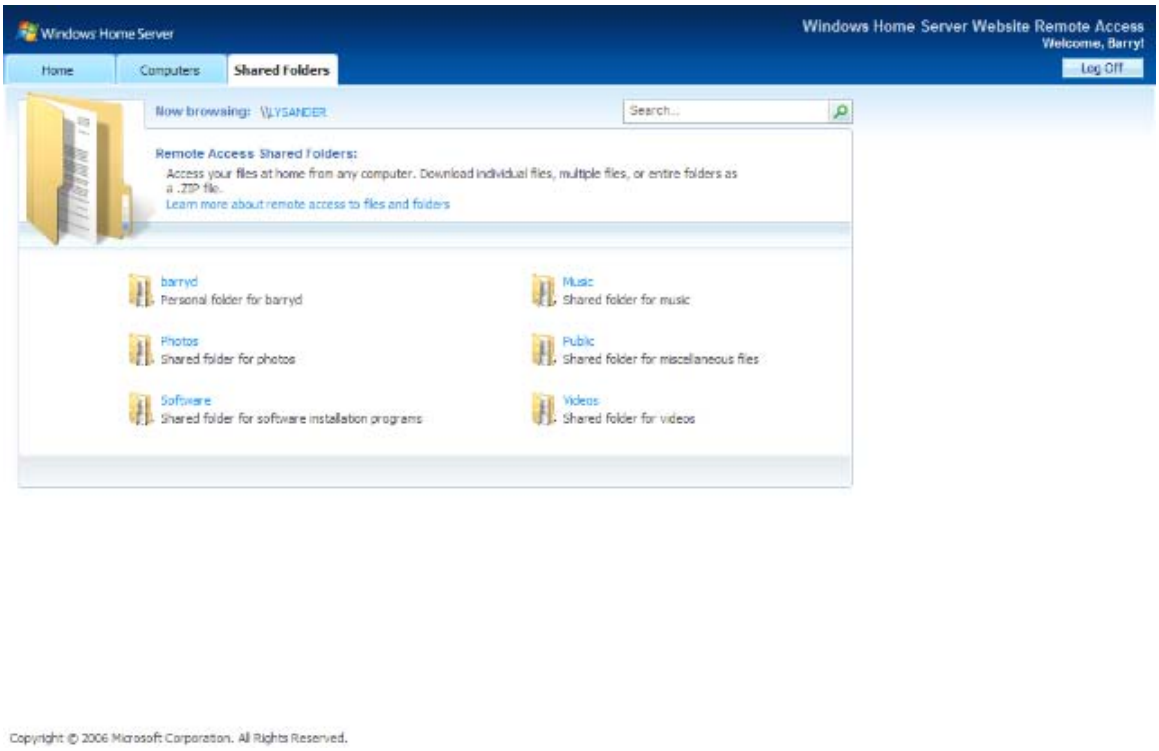
## Computer Backup and Restore

**Windows Home Server Computer Backup** automatically backs up all of the computers in a home to the server using an image-based system that ensures point-in-time-based

restoration of either entire PCs or specific files and folders. Complete computer restores are initiated through a restore bootable CD, file based restores are initiated through the WHS client software which allows the users to open a backup and "drag and drop" files from it. This technology uses Volume Shadow Services (VSS) technology on the client computer to take an image based backup of a running computer. Because the backup operates on data at the cluster level, single instancing can be performed to minimize the amount of data that travels over the network and that will ultimately be stored on the home server. This single instancing gives the server the ability to store only one instance of data, no matter if the data originated from another computer, another file, or even data within the same file.

Computer backup images are not duplicated on the server, so if a server hard drive fails, backups could be lost. The "Server Backup" feature added in Power Pack 1 does not include duplication of backup images.

## Remote File Access




Web Interface showing the shared files UI

The system also offers an SSL secured web browser based interface over the Internet to the shared file stores. The release version offers access to the web interface via a free Windows Live-provided URL, which uses Dynamic DNS. The web interface also allows the uploading to and downloading of files from the content stores. However, there is a limit of 2 GB for a single batch of upload.

### Remote Desktop Services

The system also supports Terminal Services Gateway, allowing remote control of the desktop of any Windows computer on the home network. Currently supported systems are those which would normally support Remote Desktop: Windows XP Professional, Tablet and Media Center editions, Windows Vista Business, Enterprise and Ultimate editions and Windows 7 Professional, Enterprise and Ultimate editions. The web interface also supports embedding the Remote Desktop ActiveX control, to provide remote access to home computers from within the web interface directly. Remote sessions can also connect to the Home Server console to configure the server over the internet.

### Add-Ins

Windows Home Server allows for developers to publish community and commercial add-ins designed to enhance the Windows Home Server with added functionality. As of January 2010, nearly 100 of these add-ins have been developed for WHS, including applications for antivirus & security, backups, disk management, automation, media, network/power management, remote access, BitTorrent and more. The Windows Home Server SDK (Software Development Kit) provides developers with a set of APIs and tools to use when developing for and extending Windows Home Server.

## *Compatibility*

Windows Home Server features integration with Windows XP (SP2 or newer), Windows Vista, and Windows 7 (after the release of Power Pack 3) through a software installation, either from a client CD or via a network share. Files stored on Windows Home Server are also available through a Windows share, opening compatibility to a wide variety of operating systems. Also, the Administration console is available via Remote Desktop, allowing administration from unsupported platforms.

Windows Home Server does not support Microsoft's own MSE anti-virus program.

64-bit Windows client support was introduced in Power Pack 1, though the Restore Wizard on the Windows Home Server Restore CD is unable to restore clients running 64-bit operating systems. Windows XP Professional x64 isn't officially supported. However, unofficial workarounds allow Connector software to work on XP x64.

Integration of the file sharing service as a location for Mac OS X's Time Machine was apparently being considered, but upon Mac OS X Leopard's release, Apple had removed the ability to use the SMB file sharing protocol for Time Machine backups. One WHS provider, HP, provides their own plug-in with their home server line capable of Time Machine backup to a home server.

Windows Home Server has not officially supported Domain Controller capability and cannot readily join a Windows Server domain. Wireless networking is not currently supported.

## *Minimum system requirements*

The following minimum specs are needed:

- 1.0 GHz Intel Pentium 3 (or equivalent) processor
- 512 MB RAM
- 80 GB internal hard drive as primary drive
- 100 Mbit/s wired Ethernet

Additionally, the following are required for installation of the operating system only:

- Bootable DVD drive or USB stick
- Display
- Keyboard and mouse

Dedicated devices will have the operating system pre-installed and may be supplied with a server recovery disk which reloads the OS over a network connection. This is utilized on the HP MediaSmart Server, and the Fujitsu Siemens Scaleo Home Server.

## *Resolved issues*

### File corruption

The first release of Windows Home Server, RTM (Release to manufacturing), suffered from a file corruption flaw whereby files saved directly to or edited on shares on a WHS device could become corrupted. Only the files that had NTFS Alternate Data Streams were susceptible to the flaw. The flaw led to data corruption only when the server was under heavy load at the time when the file (with ADS) was being saved onto a share.

Backups of client PCs made by Windows Home Server were not susceptible to the flaw.

Even though the issue was first acknowledged in October 2007, Microsoft formally warned users of the seriousness of the flaw on 20 December 2007. Microsoft then issued a list of applications, including Windows Live Photo Gallery, Microsoft OneNote, Microsoft Outlook and SyncToy 2.0, which might have triggered the flaw if they were used to edit the files on a WHS share directly.

This issue was fixed by Power Pack 1, released on July 21, 2008.

### No native backup

Windows Home Server RTM did not include a mechanism for backing up the server. Power Pack 1 added the ability to back up files stored on the Shared Folders, to an external drive. Users can also subscribe to 3rd-party online services, for a fee. However, there remains no way to back up the installed server operating system. Backing-up of the client backup database is available either manually using the instructions provided by Microsoft on page 24 of this document or can be done using the WHS BDBB add-in written by Alex Kuretz and available from this website.

## *Pricing*

While some hardware manufacturers have developed dedicated boxes, Microsoft has also released Windows Home Server under the OEM/System Builder license. In November 2008 Microsoft lowered the price of the WHS System Builder SKU to US$100.

Users can also choose to use an existing PC or build their own systems, which would include the use of WHS System Builder.

As of March 23, 2009, Microsoft has also made Windows Home Server available to MSDN and Microsoft Technet subscribers.

### Refund of Windows Home Server license fees

Some computer systems are available only with a bundled Windows Home Server license. As is the case with other versions of Windows it is possible to request a refund of the license fees paid for Windows Home Server.

## Chapter-13

# Middleware

**Middleware** is computer software that connects software components or some people and their applications. The software consists of a set of services that allows multiple processes running on one or more machines to interact. This technology evolved to provide for interoperability in support of the move to coherent distributed architectures, which are most often used to support and simplify complex distributed applications. It includes web servers, application servers, and similar tools that support application development and delivery. Middleware is especially integral to modern information technology based on XML, SOAP, Web services, and service-oriented architecture.

Middleware sits "in the middle" between application software that may be working on different operating systems. It is similar to the middle layer of a three-tier single system architecture, except that it is stretched across multiple systems or applications. Examples include EAI software, telecommunications software, transaction monitors, and messaging-and-queueing software.

The distinction between operating system and middleware functionality is, to some extent, arbitrary. While core kernel functionality can only be provided by the operating system itself, some functionality previously provided by separately sold middleware is now integrated in operating systems. A typical example is the TCP/IP stack for telecommunications, nowadays included in virtually every operating system.

In simulation technology, *middleware* is generally used in the context of the high level architecture (HLA) that applies to many distributed simulations. It is a layer of software that lies between the application code and the run-time infrastructure. Middleware generally consists of a library of functions, and enables a number of applications– simulations or federates in HLA terminology—to page these functions from the common library rather than re-create them for each application.

## *Definitions*

Software that provides a link between separate software applications. Middleware is sometimes called plumbing because it connects two applications and passes data between them. Middleware allows data contained in one database to be accessed through another. This definition would fit enterprise application integration and data integration software.

ObjectWeb defines middleware as: "The software layer that lies between the operating system and applications on each side of a distributed computing system in a network."

## Origins

Middleware is a relatively new addition to the computing landscape. It gained popularity in the 1980s as a **solution to the problem of how to link newer applications to older legacy systems,** although the term had been in use since 1968. It also facilitated distributed processing, the connection of multiple applications to create a larger application, usually over a network.

## Organizations

IBM, Red Hat, and Oracle Corporation are major vendors providing middleware software. Vendors such as Axway, SAP, TIBCO, Informatica, Pervasive and webMethods were specifically founded to provide Web-oriented middleware tools. Groups such as the Apache Software Foundation, OpenSAF and the ObjectWeb Consortium (now OW2) encourage the development of open source middleware. Microsoft .NET “Framework” architecture is essentially “Middleware” with typical middleware functions distributed between the various products, with most inter-computer interaction by industry standards, open APIs or RAND software licence. Solace Systems provides middleware in purpose-built hardware for implementations that may experience scale or speed limitations when using software.

## Use of middleware

Middleware services provide a more functional set of application programming interfaces to allow an application to:

- Locate transparently across the network, thus providing interaction with another service or application
- Filter data to make them friendly usable or public via anonymization process for privacy protection (for example)
- Be independent from network services
- Be reliable and always available
- Add complementary attributes like semantics

when compared to the operating system and network services.

Middleware offers some unique technological advantages for business and industry. For example, traditional database systems are usually deployed in closed environments where users access the system only via a restricted network or intranet (e.g., an enterprise’s internal network). With the phenomenal growth of the World Wide Web, users can access virtually any database for which they have proper access rights from anywhere in the world. Middleware addresses the problem of varying levels of interoperability among different database structures. Middleware facilitates transparent access to legacy database

management systems (DBMSs) or applications via a web server without regard to database-specific characteristics.

Businesses frequently use middleware applications to link information from departmental databases, such as payroll, sales, and accounting, or databases housed in multiple geographic locations. In the highly competitive healthcare community, laboratories make extensive use of middleware applications for data mining, laboratory information system (LIS) backup, and to combine systems during hospital mergers. Middleware helps bridge the gap between separate LISs in a newly formed healthcare network following a hospital buyout.

Wireless networking developers can use middleware to meet the challenges associated with wireless sensor network (WSN), or WSN technologies. Implementing a middleware application allows WSN developers to integrate operating systems and hardware with the wide variety of various applications that are currently available.

Middleware can help software developers avoid having to write application programming interfaces (API) for every control program, by serving as an independent programming interface for their applications. For Future Internet network operation through traffic monitoring in multi-domain scenarios, using mediator tools (middleware) is a powerful help since they allow operators, searchers and service providers to supervise Quality of service and analyse eventual failures in telecommunication services.

Finally, e-commerce uses middleware to assist in handling rapid and secure transactions over many different types of computer environments. In short, middleware has become a critical element across a broad range of industries, thanks to its ability to bring together resources across dissimilar networks or computing platforms.

In 2004 members of the European Broadcasting Union (EBU) carried out a study of Middleware with respect to system integration in broadcast environments. This involved system design engineering experts from 10 major European broadcasters working over a 12 month period to understand the effect of predominantly software based products to media production and broadcasting system design techniques. The resulting reports Tech 3300 and Tech 3300s were published and are freely available from the EBU web site.

## *Types of middleware*

### Message-oriented Middleware

Message-oriented middleware is middleware where transactions or event notifications are delivered between disparate systems or components by way of messages, often via an enterprise messaging system.

## Enterprise messaging system

An enterprise messaging system is a type of middleware that facilitates message passing between disparate systems or components in standard formats, often using XML, SOAP or web services.

### *Message broker*

Part of an enterprise messaging system, message broker software may queue, duplicate, translate and deliver messages to disparate systems or components in a messaging system.

## Enterprise Service Bus

Enterprise Service Bus (ESB) is defined by the Burton Group as "some type of integration middleware product that supports both MOM and Web services".

## Content-Centric Middleware

Content-centric middleware provides a simple provide/consume abstraction through which applications can issue requests for uniquely identified content, without worrying about where or how it is obtained. Juno is one example, which allows applications to generate content requests associated with high-level delivery requirements. The middleware then adapts the underlying delivery to access the content from the source(s) that are best suited to matching the requirements. This is therefore similar to Publish/subscribe middleware, as well as the Content-centric networking paradigm.

## Hurwitz classification system

Judith Hurwitz created a classification system for middleware in her article *Sorting Out Middleware*.

## Remote Procedure Call

With Remote Procedure Call middleware, a client makes calls to procedures running on remote systems. Can be asynchronous or synchronous.

## Message Oriented Middleware

With Message Oriented Middleware, messages sent to the client are collected and stored until they are acted upon, while the client continues with other processing.

## Object Request Broker

With Object Request Broker middleware, it is possible for applications to send objects and request services in an object-oriented system..

## SQL-oriented Data Access

SQL-oriented Data Access is middleware between applications and database servers.

## Embedded middleware

Embedded middleware provides communication services and integration interface software/firmware that operates between embedded applications and the real time op.

## Other

Other sources include these additional classifications:

- Transaction processing monitors — Provides tools and an environment to develop and deploy distributed applications.
- Application servers — software installed on a computer to facilitate the serving (running) of other applications.

Chapter-14

# gLite and Flow (Software)

# gLite

**gLite** is the middleware stack for grid computing used by the CERN LHC experiments and a very large variety of scientific domains. Born from the collaborative efforts of more than 80 people in 12 different academic and industrial research centers as part of the EGEE Project, gLite provides a complete set of services for building a production grid infrastructure. gLite provides a framework for building grid applications tapping into the power of distributed computing and storage resources across the Internet. The gLite services are currently adopted by more than 250 Computing Centres and used by more than 15000 researchers in Europe and around the world (Taiwan, Latin America etc.).

## *History*

After prototyping phases in 2004 and 2005, convergence with the LCG-2 distribution was reached in May 2006 when gLite 3.0 was released and became the official middleware of the EGEE project.

## *Middleware description*

### Security

The gLite user community is grouped into Virtual Organisations (VOs). A user must join a VO supported by the infrastructure running gLite to be authenticated and authorized to using grid resources.

The Grid Security Infrastructure (GSI) in WLCG/EGEE enables secure authentication and communication over an open network. GSI is based on public key encryption, X.509 certificates, and the Secure Sockets Layer (SSL) communication protocol, with extensions for single sign-on and delegation.

In order to authenticate himself, a user needs to have a digital X.509 certificate issued by a Certification Authority (CA) trusted by the infrastructure running the middleware.

The authorisation of a user on a specific Grid resource can be done in two different ways. The first is simpler, and relies on the grid-mapfile mechanism. The second way relies on the Virtual Organisation Membership Service (VOMS) and the LCAS/LCMAPS mechanism, which allow for a more detailed definition of user privileges.

## User interface

The access point to the gLite Grid is the User Interface (UI). This can be any machine where users have a personal account and where their user certificate is installed. From a UI, a user can be authenticated and authorized to use the WLCG/EGEE resources, and can access the functionalities offered by the Information, Workload and Data management systems. It provides CLI tools to perform some basic Grid operations:

- list all the resources suitable to execute a given job;
- submit jobs for execution;
- cancel jobs;
- retrieve the output of finished jobs;
- show the status of submitted jobs;
- retrieve the logging and bookkeeping information of jobs;
- copy, replicate and delete files from the Grid;
- retrieve the status of different resources from the Information System.

## Computing element

A Computing Element (CE), in Grid terminology, is some set of computing resources localized at a site (i.e. a cluster, a computing farm). A CE includes a Grid Gate (GG)1, which acts as a generic interface to the cluster; a Local Resource Management System (LRMS) (sometimes called batch system), and the cluster itself, a collection of Worker Nodes (WNs), the nodes where the jobs are run.

There are two GG implementations in gLite 3.1: the LCG CE, developed by EDG and used in LCG-22, and the gLite CE, developed by EGEE. Sites can choose what to install, and some of them provide both types. The GG is responsible for accepting jobs and dispatching them for execution on the WNs via the LRMS.

In gLite 3.1 the supported LRMS types are OpenPBS/PBSPro, Platform LSF, Maui/Torque, BQS and Condor, and Sun Grid Engine.

## Storage element

A Storage Element (SE) provides uniform access to data storage resources. The Storage Element may control simple disk servers, large disk arrays or tape-based Mass Storage Systems (MSS). Most WLCG/EGEE sites provide at least one SE.

Storage Elements can support different data access protocols and interfaces. Simply speaking, GSIFTP (a GSI-secure FTP) is the protocol for whole-file transfers, while local and remote file access is performed using RFIO or gsidcap.

Most storage resources are managed by a Storage Resource Manager (SRM), a middleware service providing capabilities like transparent file migration from disk to tape, file pinning, space reservation, etc. However, different SEs may support different versions of the SRM protocol and the capabilities can vary.

There is a number of SRM implementations in use, with varying capabilities. The Disk Pool Manager (DPM) is used for fairly small SEs with disk-based storage only, while CASTOR is designed to manage large-scale MSS, with front-end disks and back-end tape storage. dCache is targeted at both MSS and large-scale disk array storage systems. Other SRM implementations are in development, and the SRM protocol specification itself is also evolving.

Classic SEs, which do not have an SRM interface, provide a simple disk-based storage model. They are in the process of being phased out.

## Information service

The Information Service (IS) provides information about the WLCG/EGEE Grid resources and their status. This information is essential for the operation of the whole Grid, as it is via the IS that resources are discovered. The published information is also used for monitoring and accounting purposes.

Much of the data published to the IS conforms to the GLUE Schema, which defines a common conceptual data model to be used for Grid resource monitoring and discovery.

The Information System that is used in gLite 3.1 inherits its main concepts from the Globus Monitoring and Discovery Service (MDS). However, the GRIS and GIIS in MDS has been replaced by the Berkeley Database Information Index which is essentially an OpenLDAP server that is updated by an external process.

## Workload management

The purpose of the Workload Management System (WMS) is to accept user jobs, to assign them to the most appropriate Computing Element, to record their status and retrieve their output. The Resource Broker (RB) is the machine where the WMS services run.

Jobs to be submitted are described using the Job Description Language (JDL), which specifies, for example, which executable to run and its parameters, files to be moved to and from the Worker Node on which the job is run, input Grid files needed, and any requirements on the CE and the Worker Node.

The choice of CE to which the job is sent is made in a process called match-making, which first selects, among all available CEs, those which fulfill the requirements expressed by the user and which are close to specified input Grid files. It then chooses the CE with the highest rank, a quantity derived from the CE status information which expresses the *goodness* of a CE (typically a function of the numbers of running and queued jobs).

The RB locates the Grid input files specified in the job description using a service called the Data Location Interface (DLI), which provides a generic interface to a file catalogue. In this way, the Resource Broker can talk to file catalogues other than LFC (provided that they have a DLI interface).

The most recent implementation of the WMS from EGEE allows not only the submission of single jobs, but also collections of jobs (possibly with dependencies between them) in a much more efficient way then the old LCG-2 WMS, and has many other new options.

Finally, the Logging and Bookkeeping service (LB) tracks jobs managed by the WMS. It collects events from many WMS components and records the status and history of the job.

## Software components

Below you can find a list of the current gLite components and services that are deployed or about to be deployed on the production infrastructure, together with the contributing partners (from the JRA1 home page):

- VOMS and VOMSAdmin (INFN)
- Proxy and attribute certificate renewal (CESNET)
- Shibboleth interoperability: SLCS, VASH, STS (SWITCH)
- LCAS/LCMAPS (NIKHEF)
- gLExec (NIKHEF)
- Delegation Framework (CERN, HIP, STFC)
- CGSI_gSOAP (CERN)
- gsoap-plugin (CESNET)
- Trustmanager (HIP)
- Util-java (HIP)
- Gridsite (STFC)
- Authorization Framework (HIP, INFN, NIKHEF, SWITCH)
- BDII (CERN)
- Grid Laboratory Uniform Environment (CERN)
- R-GMA (STFC)
- CREAM (INFN)
- CEMon (INFN)
- BLAH (INFN)
- WMS (INFN, ElsagDatamat)
- LB (CESNET)

- DPM (CERN)
- GFAL (CERN)
- LFC (CERN)
- FTS (CERN)
- lcg_utils (CERN)
- EDS and Hydra (HIP)
- AMGA (CERN, KISTI, INFN)

# Flow (software)

**Flow** is middleware software, which allows data integration specialists to connect disparate systems, whether they are on-premise, hosted or in the cloud; transforming and restructuring data as required between environments. Flow functionality can be utilised for data integration projects, EDI and data conversion activities. Flow has been created by Flow Software Ltd in NZ and is available through a variety of partner companies or directly from Flow Software in NZ and Australia.

Integration software allows organisations to continue using existing applications, overcoming the need to customize or upgrade as their requirements change. By using integration software, many businesses benefit from reduced dependence on manual keying of data and the avoidance of costs and delays caused by keying errors.

## *Flow Software Features*

Flow enables data management:

- Transformation of data, within and between sets
- Generation and consumption of data, accessioning from specified sets within structures
- Transportation of data files, using various transport formats, including secure
- Specification of task work-flows
- Notification of transactions and formats via reports

### Data Generation

Flow accesses and generates data in structured formats, from files or databases.

Flow can access and read from, or write to databases using either the SQL89 or SQL92 specification. Informix provides support for extended SQL use.

- Microsoft SQL Server 2000 & above
- Microsoft Access 97 97 above
- MySQL 4.x

- Oracle 8i
- InterBase 5.6
- Informix
- IBM DB2
- MYOB
- Any ODBC compliant database as per the Microsoft ODBC specification
- Any ADO compliant data source as per the Microsoft ADO specification

Flow can access and read from, or write to various file types.

- Any ASCII format file
- Any EDI type file based on either UN/EDIFACT or ANSI X12 standards
- Any XML file based on XML standards such as SOAP, XHTML or ebXML

## Data Transformation

A visual mapping engine is used to configure data transformation between data sets. Data can be restructured as it is transformed, thus allowing for dissimilar data structures between source and destination. Flow data access operates independently of the mapping layer. The applied mapping logic uses events containing Object Pascal code.

## Data Transportation

Flow transports generated data and files using the following formats:

- Local file access
- LAN
- FTP
- HTTP/SPOST
- HTTP/S GET
- SMTP
- POP
- ebMS
- SOAP

## User Interface

The Flow user interface allows users to create and processes, activate processes and view activity logs.

Email notifications of Flow process activity can also be configured.

## Actions

Flow uses predefined processing of events that can be executed either on schedule, or event driven. Actions and their results are logged and available via the user interface.

Actions include:

- Transformation of data or files
- Generation of specific reports
- Windows-based shell commands
- Outward-bound transports of data or files
- Selected SQL statements
- Custom plugin actions

## Reports

Flow includes a report writer based on the software Report Builder. The report writer can create custom notification reports providing users with details related to their transactions. Reports can be created in XML, PDF, JPEG and XLS. Reports can be embedded into email messages if required.

## *History*

Flow was created and developed by company founder Cameron Hart in North Shore City, New Zealand.

## Chapter-15

# Advanced Message Queuing Protocol

The **Advanced Message Queuing Protocol** (**AMQP**) is an open standard application layer protocol for message-oriented middleware. The defining features of AMQP are message orientation, queuing, routing (including point-to-point and publish-and-subscribe), reliability and security.

AMQP mandates the behaviour of the messaging provider and client to the extent that implementations from different vendors are truly interoperable, in the same way as SMTP, HTTP, FTP, etc. have created interoperable systems. Previous attempts to standardise middleware have happened at the API level (e.g. JMS) and this did not create interoperability. Unlike JMS, which merely defines an API, AMQP is a wire-level protocol. A wire-level protocol is a description of the format of the data that is sent across the network as a stream of octets. Consequently any tool that can create and interpret messages that conform to this data format can interoperate with any other compliant tool irrespective of implementation language.

## *Overview*

AMQP was originally designed to provide a vendor-neutral (i.e. interoperable across multiple vendors) protocol for managing the flow of messages across an enterprise's business systems.

AMQP is middleware to provide a point of rendezvous between backend systems, such as data stores and services and front end systems such as end user applications. The first applications happen to have been in the financial industry, i.e. trading desks, where real time order and market data are transmitted. Though originally used inside of enterprises, AMQP can easily be used to move messages between organizations.

AMQP lets system architects build common messaging patterns out of a simpler underlying model. Typical messaging patterns are: *request-response*, in which messages are sent to or from specific recipients, *publish-and-subscribe*, in which information is distributed to a set of recipients according to various subscription criteria, and *round-robin*, in which tasks are distributed fairly among a set of recipients. Realistic applications combine these, e.g. round-robin for distributing work plus request-response for sending back responses.

The protocol specification defines a binary wire protocol used between a *client* and *server* (also known as a *broker*). In addition the specification outlines a messaging queuing model and services that an implementation provides.

The queuing model of AMQP provides for a wide range of messaging use-cases and further refines the functions of the clients and brokers. The function of brokers can be usefully broken into two kinds: exchanges and message queues. Message queues store messages, and various implementations can achieve various quality of service. For example a slow but tornado-proof message queue would keep redundant copies in multiple geographic regions while a fast but fragile message queue might keep everything in a single process's RAM. To help improve interoperability some of these aspects of the message queues are specified in the protocol, e.g. you can state what you need asking a message queue implementing broker to create a new queue.

The standard AMQP exchanges have no semantics for storing messages. They route them to queues, which store them on behalf of recipients. Exchanges implement a range of message routing techniques: one-to-one message passing (like email to one recipient), one-to-N (like an email list), one-to-one-of-N (like a queue for the next open checkout), and so on. Since all exchanges accept messages from N senders, AMQP allows all one-to-any routing to be N-to-any. The rules that configure an exchange, known as *bindings*, can range from very simple (pass everything into this message queue) to procedural inspections of message content. AMQP allows arbitrary exchange semantics through *custom exchanges* (which can queue, generate, consume, and route messages in any way desired by the implementation).

Messages consist of an envelope of properties used in routing and by applications and a content, of any size. AMQP message contents are opaque binary blobs. Messages are passed between brokers and clients using the protocol commands Basic.Publish and Basic.Deliver. These commands are asynchronous so that conditions that arise from a command's evaluation are signalled by sending additional commands back on the channel that carried the command originally. AMQP also provides a synchronous message delivery command, Basic.Get/Get-Ok.

Examples of error conditions include signalling by an exchange that it could not route a message because no route was found, or signalling that a message queue declined to accept a message (say because it was full). Message brokers may be configured to handle exceptions in different ways. For example, routing the associated message to a dead letter queue or even bringing the broker to a hard stop.

## *Development*

AMQP was developed from mid-2004 to mid-2006 by JPMorgan Chase & Co. and iMatix Corporation who also developed implementations in C/C++ and Java. JPMorgan Chase & Co. and iMatix documented the protocol as an interoperable specification and assigned to a working group that included Red Hat, Cisco Systems, TWIST, IONA, and iMatix. As of November 2009, the working group consists of Bank of America, Barclays,

Cisco Systems, Credit Suisse, Deutsche Börse Systems, Envoy Technologies, Inc., Goldman Sachs, Progress Software, iMatix Corporation, JPMorgan Chase Bank Inc. N.A, Microsoft Corporation, Novell, Rabbit Technologies Ltd., Red Hat, Inc., Solace Systems, Tervela Inc., TWIST Process Innovations ltd, WS02 and 29West Inc.

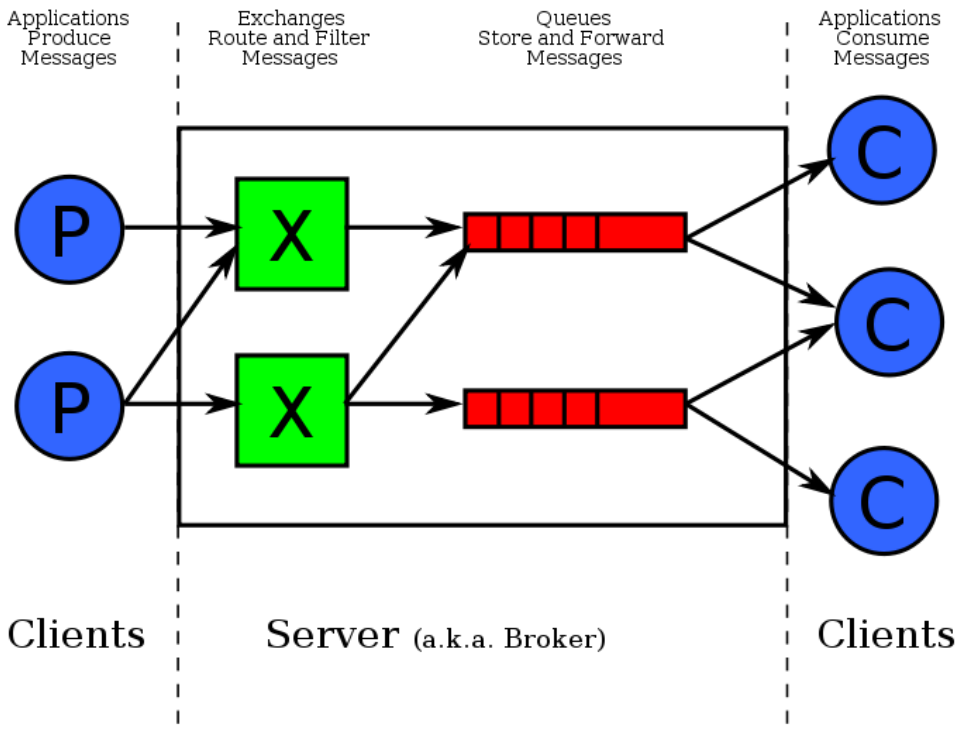
A notable design goal of AMQP was to enable the creation of open standard protocol stacks for business messaging both within and between firms by combining AMQP with one of the many open standards describing business transactions, such as FpML or more generically as a reliable transport for SOAP.

Whilst AMQP originated in the financial services industry, it has general applicability to a broad range of middleware problems.

## *The AMQP model*

AMQP defines a number of entities. From a connection perspective the relevant ones are:

- Message broker: a server to which AMQ clients connect using the AMQ protocol. Message brokers can run in a cluster but these details are implementation specific and are not covered by the specification.
- User: a user is an entity that, by providing credentials in form of a password, may or may not be authorized to connect to a broker.
- Connection: a physical connection e.g. using TCP/IP or SCTP. A connection is bound to a user.
- Channel: a logical connection that is tied to a connection. Hence communication over a channel is stateful. Clients that perform concurrent operations on a connection should maintain a distinct channel for each of those. Clients that use a threaded model of concurrency can for example encapsulate the channel declaration in a thread-local variable.




Entities in the AMQP model used for message transfer

The entities used for the actual sending and receiving of messages are all declared on a channel. A declaration assures the issuing client that the entity exists (or was previously declared by another client). Any attempt to declare a named entity with different properties than it was declared before will result in an error. In order to change the properties of such an entity it must be deleted prior to a re-declaration (with changed properties).

Some of these entities are named. The naming must be unique within the scope of the entity and its broker. Since clients usually (at least no such operations are defined in the AMQP specification) do not have the means to get a list of all available named entities, the knowledge of an entity name is what allows the client to perform operations on it.

Names are encoded in UTF-8, must be between 1 and 255 characters in length and must start with a digit, a letter or an underscore character.

## Exchanges

Exchanges are the entities to which messages are sent. They are named and have a type as well as properties such as:

- passive: the exchange will not get declared but an error will be thrown if it does not exist.
- durable: the exchange will survive a broker restart.
- auto-delete: the exchange will get deleted as soon as there are no more queues bound to it. Exchanges to which queues have never been bound will never get auto deleted.

## Queues

Queues are the entities which receive messages. They are named and have properties but not a type. Clients can subscribe to queues to the effect that the message broker delivers (pushes) the contents of the queue to the client. Alternatively clients can pop (pull) messages from the queue as they see fit.

Messages are guaranteed to be delivered in the order that they were first delivered to a queue, unless certain kinds of rerouting operations (e.g. due to failures) occur.

The properties of queues are:

- alternate-exchange: when messages are rejected by a subscriber or orphaned by queue deletion, its messages get routed to this exchange and get removed from the queue.
- passive: the queue will not get declared but an error will be thrown if it does not exist.
- durable: the queue will survive a broker restart.
- exclusive: there can only be one client for this specific queue.
- auto-delete: the queue will get deleted as soon as no more subscriptions are active on it. This shares the same constraint as the auto-delete property for exchanges: if no subscription has been ever active on the queue it will not get auto-deleted. An exclusive queue however will always get auto-deleted when the client terminates its session.

Note that queues are scheduled to replace exchanges in AMQP/1.0.

## Messages

Messages are unnamed and are published to an exchange. They consist of a header and a content body. While the body is opaque data the header contains a number of optional properties:

- routing-key: this field is used in ways dependent on the type of the exchange.
- immediate: the message will get handled as unroutable if at least one of the queues which would receive the message has no subscription on it.
- delivery-mode: indicates that a message might need persistence. Only for such messages the broker makes a best-effort to prevent a loss of the message before consumption. If there is uncertainty on the broker's end about the successful

delivery of a message (e.g. in case of errors) it might deliver a message more than once. Non persistent delivery modes do not show this kind of behavior.
- priority: an indicator (a range between 0 and 9) that a message has higher precedence than others.
- expiration: the duration in milliseconds before the broker may handle the message as unroutable.

## Bindings

A binding is a relationship between one queue and one exchange that specifies how messages flow from the exchange to the queue. The binding properties match the routing algorithm used in exchanges. Bindings (and exchange algorithms) can be placed on a curve of increasing complexity:

- Unconditional - the binding has no properties and requests "all" messages from the exchange.
- Conditional on a fixed string - the binding has one property, the **routing key** and requests all messages that have an identical routing key.
- Conditional on a pattern match - the binding has one property, the **routing key** and requests all messages that match the routing key using a pattern-matching algorithm. Arbitrary pattern syntaxes could be used. AMQP implements topic matching.
- Conditional on multiple fixed strings - the binding has a table of properties, the **arguments** and requests all messages whose headers match these arguments, using logical ANDs or ORs to combine matches.
- Conditional on multiple patterns - the binding has a table of properties, the **arguments** and requests all messages whose headers match these arguments, using a pattern matching algorithm and logical combinations.
- Conditional on algorithmic comparison - the binding has an algorithmic expression (like an SQL SELECT WHERE clause) and requests all messages whose headers match that expression.
- Conditional on content inspection - the binding specifies arbitrary criteria that are resolved by inspection of the actual message content.

Not all these are implemented as standard, or by all implementations.

## Exchange types and the effect of bindings

These four entities form the basic model of the AMQP. The key to understand how a message is passed to a queue lies in the relationship between the type of an exchange and the resulting interpretation of the routing key.

An exchange will deliver up to one copy of a message to a queue if the routing key in the message matches a binding (subsequent semantically identical bindings will not lead to duplicate copies). What constitutes a match however is solely dependent on the type of an exchange:

- a direct exchange matches when the routing key property of a message and the key of the binding are identical.
- a fanout exchange always matches, even on bindings without a key.
- a topic exchange matches the routing key property of a message on binding key words. Words are strings which are separated by dots. Two additional characters are also valid: the *, which matches 1 word and the #, which matches 0..N words. Example: *.*stock.*# matches the routing keys *usd.stock* and *eur.stock.db* but not *stock.nasdaq*.
- a headers exchange matches on the presence of keys as well as key–value pairs which can be concatenated with logical and–or connections in a messages header. In this case the routing key is not a criterion for matching that is considered by the exchange. Neither does the binding carry a single routing key but special format which contains header keys and / or key-value-pairs which match on the header key being present or the header key being present and the value being the same respectively.

Other e.g. vendor-specific exchanges are explicitly permitted in the specification.

The concept of binding named queues to named exchanges has powerful properties (with binding making those two entities independent of each other). It is, for instance, possible to bind a single queue with multiple bindings to the same or to different exchanges. Or multiple consumers can share the name of a queue and bind to it with the same parameters and will therefore get only message that the other consumers did not consume. Or multiple consumers can declare independent queues but share the bindings and get all the message every other consumer would get on the bound exchange with these bindings.

## *Specification revisions and the future of AMQP*

The following specifications of the AMQ protocol have been published, in chronological order:

- 0-8 in June 2006
- 0-9 in December 2006
- 0-10 (documents are undated)
- 0-9-1 in November 2008
- 1.0 draft in May 2010

The draft 1.0 specification changes the AMQP model illustrated above by removing the concepts of exchanges and bindings, and replacing these with queues and *links*. This change aims to remedy two problems with the previous approach:

1. The publisher needs to know too much about the receivers topology (what exchanges and exchange types are available).

2. Producer flow control is challenging - if an Exchange is routing a message to 2 different queues, one empty and the other nearly full, what flow control information should be relayed to the producer and how would that be determined?

According to John O'Hara however, JPMorganChase and RedHat introduced links into AMQP/1.0 simply to solve an operational problem of slow consumers causing memory build up in brokers.

Other changes include the introduction of a queue addressing schema similar to E-mail and XMPP. This raises addresses to first-class entities, and allows for the publication of service location records using the DNS.

The process of bringing the 1.0 Specification to a Standard involves a requirement elicitation phase, then the release of a "public review" spec (PR) which should be reviewed and asked for comments, optionally resulting in further modifications. When there are no substantive changes to the PR, it is voted to be the 1.0 Recommendation. When there are at least two implementations that pass a special test coverage, the Recommendation is voted to be 1.0 Standard. As of 29-Dec-2010, a Recommendation spec has been produced and is waiting for two or more implementations proven to interoperate.

## *Implementations*

These are the known publicly available AMQP implementations:

- OpenAMQ, an open-source implementation of AMQP, written in C by iMatix. Runs on Linux, AIX, Solaris, Windows, OpenVMS. APIs in C/C++ and Java JMS. Discontinued by iMatix after their switching to ØMQ.
- StormMQ, currently the only message queuing service using AMQP. Offered as a managed service, there is no software to install and no licence fees to pay. Customers pay for a subscription out of OPEX, and use a solution in the Cloud, Co-Hosted or On-Site.
- RabbitMQ, an independent open-source implementation bought by VMware in 2010. The server is written in Erlang.
- Apache Qpid, a project in the Apache Foundation. Bindings to many languages without the use of DLLs.
- Red Hat Enterprise MRG implements the latest version of AMQP 0-10 providing rich set of features like full management, federation, Active-Active clustering using Apache Qpid as upstream, adds a web console and many enterprise features. Also available in the latest 3 versions of Fedora as AMQP Infrastructure.

## *Other Products Integrating with AMQP*

There are many integrations of other products with AMQP, including:

- Zyre, a broker that implements RestMS and AMQP to provide RESTful HTTP access to AMQP networks.

## Clients

There are many clients, including:

- DE.SETF.AMQP, a Common Lisp client library for AMQP.

## Comparative specifications

These are the known open specifications that cover the same or similar space as AMQP:

- Stomp, a text-based pub-sub protocol developed at Codehaus; uses the JMS-like semantics of 'destination'.
- RestMS, an HTTP-based message routing and queuing protocol that provides AMQP interoperability through an optional profile.
- XMPP, the Extensible Messaging and Presence Protocol.

There are also vendor specific, proprietary specifications includes those by the Amazon Simple Queue Service, IBM WebSphere MQ, Microsoft Message Queuing, JMS and the OpenWire as used by ActiveMQ

There has not as yet been a formal comparison of these and other protocols in the same space, although an informal comparison of XMPP and AMQP may be found here. JMS, the Java Messaging service, is often compared to AMQP. However, JMS is an API specification (part of the Java EE specification) that defines how message producers and consumers are implemented. JMS does not guarantee interoperability between implementations, and the JMS-compliant messaging system in use may need to be deployed on both client and server. On the other hand, AMQP is a wire-level protocol specification. In theory AMQP provides interoperability as different AMQP-compliant software can be deployed on the client and server sides. Note that, like HTTP and XMPP, AMQP does not have a standard API.

Chapter-16

# IBM WebSphere Message Broker and HP RTR

## IBM WebSphere Message Broker

**WebSphere Message Broker** (WMB) is IBM's information broker from the WebSphere product family that allows business information to flow between disparate applications across multiple hardware and software platforms. Business rules can be applied to the data flowing through the message broker to route and transform the information. The product can be considered to be an Enterprise Service Bus providing connectivity between disparate applications.

### *History*

Originally the product was developed by a company named 'NEON', short for 'New Era of Networks' which was later acquired by Sybase. The product was developed by New Era of Networks and re-branded as IBM product called 'MQSeries Integrator' or 'MQSI' for short. Versions of MQSI ran up to 2.1. The product was added to the WebSphere family and rebranded 'WebSphere MQ Integrator', still version 2.1. From 2.1 the version numbers are synchronized with the rest of the WebSphere family and jumped to version 5.0. The name changed to 'WebSphere Business Integration Message Broker' (WBIMB). In this version the development environment was redesigned using Eclipse and support for Web services was integrated into the product. Since V6.0 the product has been known as 'WebSphere Message Broker'. WebSphere Message Broker version 7.0 was announced in October 2009.

### *Components*

WebSphere Message Broker consists of the following components:

- WebSphere Message Broker runtime
- WebSphere Message Broker Toolkit
- WebSphere Message Broker Explorer

## *How Message Broker works*

### Overview

The WebSphere Message Broker Toolkit enables developers to graphically design message flows and related artifacts. Once developed, these resources can be packaged into a broker archive (BAR) file and deployed into the runtime environment. At this point, the broker is able to continually process messages according to the logic described by the message flow.

WebSphere Message Broker flows can be used in a Service Oriented Architecture, and if properly designed by Middleware Analysts, integrated into event-driven SOA schemas, sometimes referred to as SOA 2.0.

The workflow a developer would perform to create WMB functionality is cyclical, and probably more agile than most other software development. Developers will create a message flow, generate a BAR file, deploy the message flow contained in the BAR file, test the message flow and repeat as necessary to achieve reliable functionality.

### Expected Performance

Performance varies by platform and application. Reference implementations have been bench tested to achieve transaction rates as high as 10,000 TPS. The best performing WMB system resides on AIX Power7 architecture due to its throughput characteristics, very large memory blocks of contiguous memory, 32 MB of L3 cache, 8 terabytes of local address space, 2 petabytes of global address space and 256 logical CPUs per system. Among the slower performers for WMB runtime bench tests are z/OS implementations because of the lack of very large contiguous memory block allocations within the OS. Whereas on AIX Power7, 10,000 TPS is attainable, on z/OS 400 TPS is more likely.

### Cost Per Transaction

WMB price per transaction on AIX Power7 considering three-year total cost of acquisition of hardware, software, maintenance is scalable much more economically than other platforms. Some people would call this "bang-for-the-buck", value delivery, or Return-On-Investment (ROI). This means that counting all costs involved, not necessarily the original purchase price, the Power7 architecture can be much more economically viable than other offerings, if implemented with the right expertise. Because an AIX Power7 WMB application can achieve a sustained 10,000 TPS throughput, in the course of one day, 864 million transactions can be serviced (10,000 * 60 seconds * 60 minutes * 24 hours). In one year, with ten percent downtime for maintenance, this equates to 283 billion transactions. In three years, this is 849 billion transactions. Cost per transaction then could be as little as 0.25 cents (four transactions per penny) given a three year total cost of ownership around US$200 million. Using a real-life example, when compared to

the price being charged merchants by some credit card networks, of sixty-four cents per transaction or greater, four transactions for a penny seems pretty sensible.

## Broker nodes available

A developer can choose from many pre-designed broker nodes. Nodes have different purposes. Some nodes map data from one format to another (for instance, Cobol or PL/I Copybook to canonical XML). Other nodes evaluate content of data and route the flow differently based on certain criteria.

## Node types

There are many types of node that can be used in developing message flows; the following node transformation technology options are available:

- Extended Structured Query Language (ESQL)
- Graphical Message Mapping
- eXtensible Stylesheet Language Transformations (XSLT)
- JavaCompute (as of version 6)
- WebSphere Transformation Extender (formerly known as Ascential DataStage TX, DataStage TX and Mercator Integration Broker) is available as a separate licensing option
- PhpCompute (as of version 6.1.0.3)

## Patterns

A pattern captures a commonly recurring solution to a problem (example: Request-Reply pattern). The specification of a pattern describes the problem being addressed, why the problem is important, and any constraints on the solution. Patterns typically emerge from common usage and the application of a particular product or technology. A pattern can be used to generate customized solutions to a recurring problem in an efficient way. We can do this pattern recognition or development through a process called service oriented modeling.

WebSphere Message Broker version 7 introduced patterns that:

- Give you guidance in implementing solutions
- Increase development efficiency because resources are generated from a set of predefined templates
- Improve quality through asset reuse and common implementation of functions such as error handling and logging

The patterns cover a range of categories including file processing, application integration, and message based integration.

### Pattern Examples

- Fire-and-Forget (FaF)
- Request-Reply (RR)
- Aggregation (Ag)
- Sequential (Seq)

## *Supported platforms*

### Operating systems

Currently available platforms for WebSphere Message Broker are:

- AIX
- HP-UX (IA64)
- Solaris (SPARC and x86-64)
- Linux (x86, x86-64, PPC and 390x)
- Microsoft Windows
- z/OS

## *Trivia*

- The Configuration Manager repository was named "BERNARD" after the Configuration Manager development team's pet gargoyle.

# HP RTR

**HP Reliable Transaction Router (RTR)** is a transactional middleware and is simply referred to as RTR by its users. RTR is used to integrate with applications that require reliable transaction services. RTR is currently available on HP-UX, Linux, Windows and OpenVMS.

## *Reliable Transaction Support*

RTR manages the messages sent between client-server to provide node and network fail-over for increased reliability, transactional integrity, and interoperability between dissimilar systems. RTR does this with a scalable, easy-to-use design.

## *Three-Tier Architecture*

The RTR software has three logical entities and referred to as Front-End (FE), Back-End (BE) and Transaction-Router(TR). The router is a software component that provides the fail-over intelligence and manages connections to the Back-End. The client applications

running on the Front-End combined with Router and Server applications running on Back-End interact to provide transaction integrity and reliability. The three logical entities can exist on the same node but are usually deployed on different nodes to achieve modularity, scalability and high availability.

The client application interacts with the Front-End which forwards the messages to the Router, the Router in turn routes the message to the intended Back-End where the appropriate Server application is available for processing the message.

## *Message Content Routing*

The RTR routing capability is that it allows to partition data across multiple servers and nodes for increased performance. Within an application, the partition determines how messages are routed between the client and the servers.

### Transactional Messaging

The message exchange happens between the client and server. Transactions start at the client and involve many messages that can go to a number of different servers.

### Broadcast Messaging

Such method of messaging is used in situations where there are multiple recipients for a message, or where unsolicited messages need to be sent.

## *Replication*

RTR can help survive the failures generally seen in distributed application environment which include complete site failure, node failure, network link failure and software process failure. RTR also provides continuous availability by using redundant resources in the distributed environment.

## *RTR Environment*

RTR provides a **Web Interface** and a **Command Line Interface**(CLI) for managing the RTR environment. When RTR and its components are running along with the applications, then Client Application, Server Application, RTR services will be active.

## *RTR Features*

RTR is widely used as both a product which is integrated with client applications and as well as integrated in an infrastructure developed and customized to suit various user needs. It is widely popular amongst its users for its seamless integration into the user's solutions.

## APIs

User and Management Applications can be written using RTR APIs. The C, C++, Java and .Net variants of APIs are available for creating applications to use RTR.

## Compatibility Matrix

RTR is currently supported on the Linux, Windows and OpenVMS platforms.

Chapter-17

# Internet Communications Engine and Volunteer Computing

## Internet Communications Engine

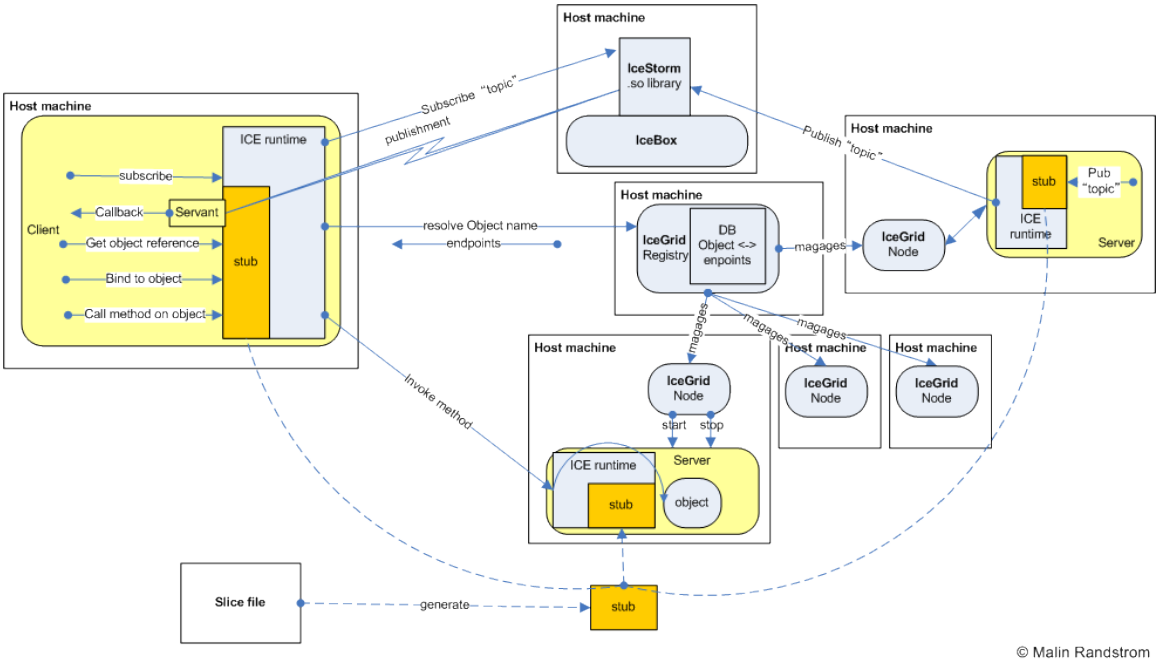
The **Internet Communications Engine**, or **Ice**, is an object-oriented middleware that provides object-oriented Remote Procedure Call, grid computing, and Publish/subscribe functionality developed by ZeroC and dual-licensed under the GNU GPL and a proprietary license. It supports C++, Java, .NET-languages (such as C# or Visual Basic), Objective-C, Python, PHP, and Ruby on most major operating systems such as Linux, Solaris, Windows and Mac OS X. A light variant of ICE runtime, called Ice-e, may run inside mobile phones. As its name indicates, the middleware may be used for internet applications without the need to use the HTTP protocol and is capable of traversing firewalls unlike most other middleware.

### *ICE and CORBA*

ICE was influenced by CORBA in its design, and indeed was created by several influential CORBA developers, including Michi Henning. However, it is much smaller and less complex than CORBA. According to ZeroC's webpages, this is partly a result of being designed by a small group of experienced developers, instead of suffering from design by committee.

### *ICE Components*

ICE is a set of CORBA like components that include object-oriented remote-object-invocation, replication, grid-computing, failover, load-balancing, firewall-traversals, and publish-subscribe services. To gain access to those services, applications are linked to a stub library or assembly, which is generated from a language-independent IDL-like syntax called *slice*.




## IceStorm

is an object-oriented publish-and-subscribe framework that also supports federation and quality-of-service. Unlike other publish-subscribe frameworks such as TIBCOs' Rendezvous or SmartSockets, message content consist of objects of well defined classes rather than of structured text.

## IceGrid

is a suite of frameworks that provide object-oriented load balancing, failover, object-discovery and registry services.

## IcePatch

facilitates the deployment of ICE based software. For example, a user who wishes to deploy new functionality and/or patches to several servers may use IcePatch.

## Glacier

is a proxy-based service to enable communication through firewalls, thus making ICE an internet communication engine.

## IceBox

is a SOA-like container of executable services implemented in .dll or .so libraries. This is a lighter alternative to building entire executable for every service.

### Slice

Slice is a Zeroc-proprietary file format that programmers follow to edit computer-language independent declarations and definitions of classes, interfaces, structures and enumerations. Slice definition files are used as input to the stub generating process. The stub in turn is linked to applications and servers that should communicate with one another based on interfaces and classes as declared/defined by the slice definitions.

Apart from CORBA, classes and interfaces support inheritance and abstract classes. In addition, slice provides configuration options in form of macros and attributes to direct the code generation process. An example is the directive to generate a certain STL `list<double>` template instead of the default, which is to generate a STL `vector<double>` template.

## *Comparisons to other major middleware*

### SOAP

Ice also compares favorably to SOAP, the main advantages being that it's more object oriented, and offers vastly superior performance in terms of both bandwidth and processor load, because SOAP is based on HTTP and XML, which needs to be parsed, while Ice uses a binary protocol designed for high performance and low verbosity. However, Ice might not offer similar performance or compactness advantages when SOAP messages are exchanged using a more efficient transport and message encoding such as SOAP/TCP and Fast Infoset.

### CORBA

The Common Object Request Broker Architecture (CORBA) is a standard defined by the Object Management Group (OMG) that enables software components written in multiple computer languages and running on multiple computers to work together.

### TIBCO Rendezvous/EMS

Rendezvous is an asynchronous publish/subscribe only middleware from TIBCO that provides text based messaging as well as its own proprietary name value pair format. A daemon runs at the client side and communicates with subscribing client processes through IPC pipes or TCP/IP. The daemon mediates between client processes and daemons that handle publishing servers. Those daemons support multicast as well as broadcast communication.

EMS stands for Enterprise Messaging Services and is a JMS server implementation, that also has support for TIBCO Rendezvous.

### Talarian Smartsockets

The differences to Rendezvous/EMS above are the lack of a payload format and a daemon running on the client side. Instead, a number of publish/subscribe daemons run somewhere on the network collectively called a cloud. This provides better performance and failover since the communication is split between several daemons and once a daemon run into an unavailable state, clients may automatically switch to another daemon. The cloud also catches published data and provides an interface for clients to retrieve the data. Any clients may therefore request for last published data anytime without having to wait for sources to republish the data. The latter mechanism is currently missing in ICE.

# Volunteer computing

**Volunteer computing** is a type of distributed computing in which computer owners donate their computing resources (such as processing power and storage) to one or more "projects".

## *History*

The first volunteer computing project was the Great Internet Mersenne Prime Search, which was started in January 1996. It was followed in 1997 by distributed.net. In 1997 and 1998 several academic research projects developed Java-based systems for volunteer computing; examples include Bayanihan, Popcorn, Superweb, and Charlotte.. Another similar concept is Sideband computing which let a user to share his computing power while he is online.

The term "volunteer computing" was coined by Luis F. G. Sarmenta, the developer of Bayanihan. It is also appealing for global efforts on social responsibility, or Corporate Social Responsibility as reported in a Harvard Business Review or used in the Responsible IT forum.

In 1999 the SETI@home and Folding@home projects were launched. These projects received considerable media coverage, and each one attracted several hundred thousand volunteers.

Between 1998 and 2002, several companies were formed with business models involving volunteer computing. Examples include Popular Power, Porivo, Entropia, and United Devices.

In 2002, the Berkeley Open Infrastructure for Network Computing (BOINC) opensource project was founded, and became the software running the largest public computing grid (World Community Grid) in 2007.

## *Middleware for volunteer computing*

The client software of the early volunteer computing projects consisted of a single program that combined the scientific computation and the distributed computing infrastructure. This monolithic architecture was inflexible; for example, it was difficult to deploy new application versions.

More recently, volunteer computing has moved to middleware systems that provide a distributed computing infrastructure independently of the scientific computation. Examples include:

- The Berkeley Open Infrastructure for Network Computing (BOINC). BOINC is the most widely-used middleware system, and is currently used by the World Community Grid. It is open source (LGPL) and is developed by an NSF-funded research project located at the UC Berkeley Space Sciences Laboratory. It offers client software for Windows, Mac OS X, Linux, and other Unix variants.
- XtremWeb is used primarily as a research tool. It is developed by a group based at the University of Paris - South.
- Xgrid is developed by Apple. Its client and server components run only on Mac OS X.
- Grid MP is a commercial middleware platform developed by United Devices and has been used in volunteer computing projects including grid.org, World Community Grid, Cell Computing, and Hikari Grid.

Most of these systems have the same basic structure: a client program runs on the volunteer's computer. It periodically contacts project-operated servers over the Internet, requesting jobs and reporting the results of completed jobs. This "pull" model is necessary because many volunteer computers are behind firewalls that don't allow incoming connections. The system keeps track of each user's "credit", a numerical measure of how much work that user's computers have done for the project.

Volunteer computing systems must deal with several problematic aspects of the volunteered computers: their heterogeneity, their churn (that is, the arrival and departure of hosts), their sporadic availability, and the need to not interfere with their performance during regular use.

In addition, volunteer computing systems must deal with several related problems related to correctness:

- Volunteers are unaccountable and essentially anonymous.
- Some volunteer computers (especially those that are overclocked) occasionally malfunction and return incorrect results.
- Some volunteers intentionally return incorrect results or claim excessive credit for results.

One common approach to these problems is "replicated computing", in which each job is performed on at least two computers. The results (and the corresponding credit) are accepted only if they agree sufficiently.

## *Costs for volunteer computing participants*

- Increased power consumption. A CPU that is idle generally has lower power consumption than when it is active. The desire to participate may also cause the volunteer to leave the PC on overnight, or to disable power-saving features like suspend. Additionally, if adequate cooling is not in place, this constant load on the volunteer's CPU can cause it to overheat.
- Decreased performance of the PC. If the volunteer computing application attempts to run while the computer is in use, it will impact performance of the PC. This is due to increased CPU contention, CPU cache contention, disk I/O contention, and network I/O contention. If RAM is a limitation, increased disk cache misses and/or increased paging can result. Volunteer computing applications typically execute at a lower CPU scheduling priority, which helps to alleviate CPU contention.

These effects may or may not be noticeable, and even if they are noticeable, the volunteer might choose to continue participating. However the increased power consumption can be remedied to some extent by setting the option of desired processor usage percent, that is available e.g. in BOINC client.

Chapter-18

# Message-oriented middleware and SynfiniWay

# Message-oriented middleware

**Message-oriented middleware** (**MOM**) is software or hardware infrastructure focused on sending and receiving messages between distributed systems. MOM allows application modules to be distributed over heterogeneous platforms, and reduces the complexity of developing applications that span multiple operating systems and network protocols by insulating the application developer from the details of the various operating system and network interfaces. APIs that extend across diverse platforms and networks are typically provided by MOM.

MOM is a software that resides in both portions of client/server architecture and typically supports asynchronous calls between the client and server applications. MOM reduces the involvement of application developers with the complexity of the master-slave nature of the client/server mechanism.

## *Origins*

### A requirements story

The case of a large bank provides a good example of how middleware emerged as a business requirement:

The bank had stored all its customer details on its large mainframe since the 1960s. This mainframe remained in heavy use and underwent several upgrades.

Although ground-breaking in its day, the mainframe's usefulness to the bank’s staff diminished as the bank introduced new, separate applications based on personal computers (PCs), allowing the bank’s staff to offer customers new services that the mainframe could not support.

An ideal situation would allow the PC-based application to link to the older mainframe application and allow the mainframe and the PCs to share each other's data. Accessing the mainframe’s data offers two advantages:

1. new front-end PC applications can replace the old user-unfriendly mainframe terminals
2. PC-based systems can use the data from the mainframe in new ways — previously impractical due to the constraints of the mainframe's software

Up until the late 1980s system integrators had no easy way to link these different applications together. Developers faced several challenges:

1. the developers would have to construct a separate software 'adapter' on both systems to translate data from source applications into a format that the destination system could understand (and *vice versa*).
2. the processing speed of each system would constrain the other system. For example, if the mainframe ran slowly, the PC-based application would have to wait until the mainframe caught up, thereby slowing down the PC application. Conversely, processing that had been offloaded to distributed servers for cost reasons would run slowly and the mainframe would have to wait until the server caught up.
3. communications programmers would need to install a network gateway system to form a bridge between the mainframe's network and the PC network if the different systems used different network protocols. The gateway would translate the network packets from the source system and pass them on to the destination system using the destination system's protocol.

Such issues made integration between applications difficult. Much of such integration also required re-engineering every time two applications on disparate platforms needed linking together, as every situation differed to some extent. By devoting effort to linking together applications on different systems, IT departments started to spend an amount significantly greater than that spent on original development per sub-system.

Developers needed a separate piece of software that would sit in the middle of two or more applications and would handle all the 'plumbing' between the two systems. Such software needed the intelligence to handle different platforms, different programming languages, various network protocols and diverse hardware. Developers wanted to remove themselves from the complexities of the underlying computing infrastructure so that they could focus on functionality within actual applications.

Towards the end of the 1980s middleware began to emerge that attempted to address these issues. Initial middleware offerings addressed specific handfuls of platforms or languages and thus had limited usefulness. Over time, however, middleware products have become more and more advanced, supporting multiple platforms, languages and protocols.

The ability of middleware to link together disparate systems across a heterogeneous network environment offers only one example of the benefits of this dominant technology. Middleware as of 2006 provides a whole raft of new functionality that augments and enhances the existing applications that it interconnects.

## *Advantages*

The primary advantage of a message-based communications protocol lies in its ability to store, route or transform messages in the process of delivery.

## *Communication properties*

### Synchronicity

MOM comprises a category of inter-application communication software that generally relies on asynchronous message-passing, as opposed to a request-response metaphor. In asynchronous systems, message queues provide temporary storage when the destination program is busy or not connected. In addition, most asynchronous MOM systems provide persistent storage to back up the message queue. This means that the sender and receiver do not need to connect to the network at the same time (asynchronous delivery), and solves problems with intermittent connectivity. It also means that should the receiver application fail for any reason, the senders can continue unaffected, as the messages they send will simply accumulate in the message queue for later processing when the receiver restarts.

### Routing

Many message-oriented middleware implementations depend on a message queue system. Some implementations permit routing logic to be provided by the messaging layer itself, whilst others depend on client applications to provide routing information or allow for a mix of both paradigms. Some implementations make use of broadcast or multicast distribution paradigms.

### Transformation

In a message-based middleware system, the recipient's message need not replicate the sender's message exactly. A MOM system with built-in intelligence can transform messages en-route to match the requirements of the sender or of the recipient. In conjunction with the routing and broadcast/multicast facilities, one application can send a message in its own native format, and two or more other applications may each receive a copy of the message in their own native format. Many modern MOM systems provide sophisticated message transformation (or mapping) tools which allow programmers to specify transformation rules applicable to a simple GUI drag-and-drop operation.

## *Disadvantages*

The primary disadvantage of many message oriented middleware systems is that they require an extra component in the architecture, the message transfer agent (message broker). As with any system, adding another component can lead to reductions in performance and reliability, and can also make the system as a whole more difficult and expensive to maintain.

In addition, many inter-application communications have an intrinsically synchronous aspect, with the sender specifically wanting to wait for a reply to a message before continuing. Because message-based communication inherently functions asynchronously, it may not fit well in such situations. That said, most MOM systems have facilities to group a request and a response as a single pseudo-synchronous transaction.

### Lack of standards

The lack of standards governing the use of message oriented middleware has caused problems. All the major vendors have their own implementations, each with its own application programming interface (API) and management tools.

The Java EE programming environment provides a standard API called JMS (Java Message Service), which is implemented by most MOM vendors and aims to hide the particular MOM API implementations; however, JMS does not define the format of the messages that are exchanged, so JMS systems are not interoperable. Microsoft's MSMQ doesn't support JMS, although there are third-party products that can offer this. WebSphere Message Broker, from IBM, does provide JMS support, as well as a whole suite of modern functionality.

The Advanced Message Queuing Protocol (AMQP) is an emerging standard that defines the protocol and formats used in the messaging server and client, so implementations are interoperable. AMQP is defined to provide flexible routing, including common messaging paradigms like point-to-point, fanout, publish/subscribe, and request-response. It also supports transaction management, queuing, distribution, security, management, clustering, federation and heterogeneous multi-platform support. Java applications that use AMQP are typically written in Java JMS. Other implementations provide APIs for C#, C++, PHP, Python, Ruby, and other languages.

## *Trends*

AMQP has been gaining adoption in applications that need an interoperable protocol for Message-oriented middleware.

Other protocols used for message-oriented middleware include XMPP and Streaming Text Oriented Messaging Protocol.

Message-oriented messaging protocols under development include RestMS, a protocol similar in nature to AMQP but constructed over a RESTful HTTP transport, and SPB, a minimalist message framing protocol that can be used to carry higher-level MOM protocols.

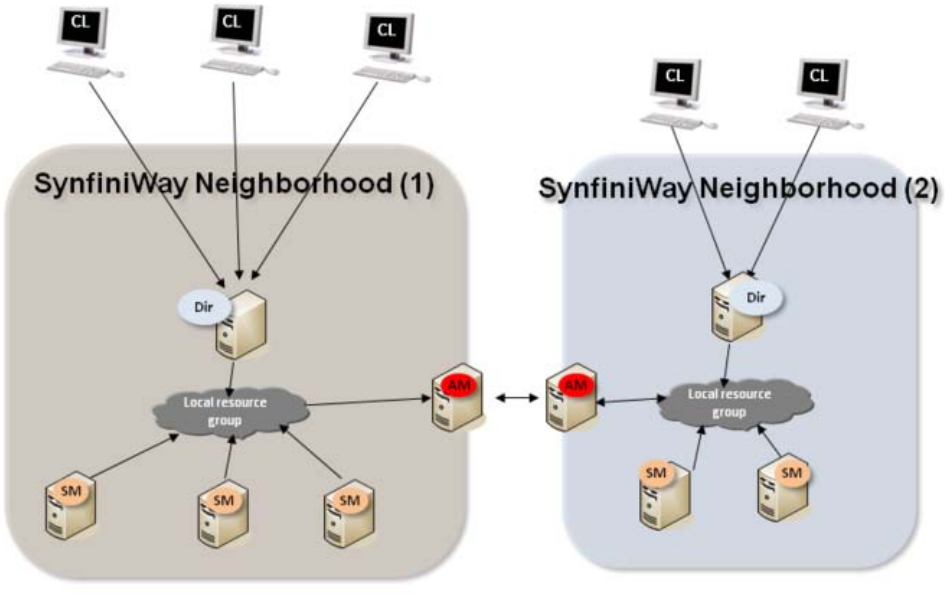
An additional trend sees message-oriented middleware functions being implemented in hardware - usually FPGAs or other specialized silicon chips.

# SynfiniWay

| | SynfiniWay |
|---|---|
| **Developer(s)** | Fujitsu |
| **Stable release** | 4.0 / September 2010 |
| **Operating system** | Linux, Unix, Windows |
| **License** | commercial |

**SynfiniWay** is middleware with which a virtualised IT framework can be created that provides a uniform and global view of resources within a department, a company, or a company with its suppliers . This virtualised IT framework is service-oriented, meaning that applications are run as services, which are a system-independent view of applications. Several applications can be linked in a workflow, and data exchange between the applications participating in the workflow is implicitly managed by the IT framework. SynfiniWay is platform-independent, allowing almost any distributed heterogeneous platform to be linked into its virtualised IT framework.

## *IT framework*




**Cl**: Client **Dir**: Director **AM**: Acquaintance Manager **SM**: Service Manager

A **virtualised IT framework** is implemented with SynfiniWay by installing a component with specific software agents on each of the systems in the framework. There are three major types of components in SynfiniWay:

- Director, which manages end-user connection, authentication & authorisation, and workflow task scheduling and execution.
- Service Manager, which publishes and runs services on behalf of users and which executes data migration.
- Acquaintance Manager, which links one remote network, known as a SynfiniWay neighborhood, to another to allow resource discovery and file transfer between components residing in different neighbourhoods.

All components are based on Java, so that they can be deployed in a multi-platform environment. An example framework with two neighbourhoods is shown in the figure. Adding or removing components is automatically detected by the framework. The SynfiniWay meta-scheduler automatically adjusts to changes in Service Manager or service availability.

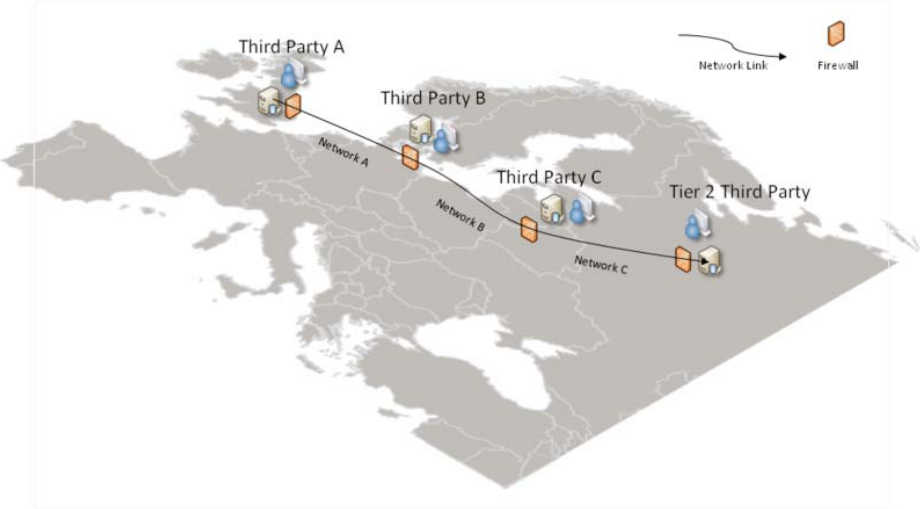
## *Service management*

SynfiniWay is geared towards **service management** . This means that an application or a command that will be utilised is defined as a service and published on the SynfiniWay IT framework. Simple or complex tasks can be abstracted into services for execution. By using these abstracted services, a user can easily run applications or commands regardless of how complex they may be or what underlying IT infrastructure is required. They provide a form of virtualisation of computing resources since the user need not be aware of where the service is available or how it is run.

## *Workflow management*

A technical or business process is created by linking services in a **workflow** . Workflows in SynfiniWay are based on WfMC version 1.0. A workflow defines one or more tasks that will be executed with a given execution logic (branch, loop, conditional). SynfiniWay supports multi-instance tasks which run a service multiple times concurrently. Also it supports a workflow of workflows, whereby a workflow can be executed as a task of a higher level workflow.

## *Data handling*




Files needed by a service are automatically transferred to the computer executing that service so that the user is freed from having to manage file transfers . A file transfer mechanism is used allowing files to be transferred directly from the source to the target computer system, going through any number of firewalls between source and target, without being stored on any of the intermediary systems. This mechanism uses the shortest path for transferring files to a target computer from the source.

## *Meta-scheduling*

SynfiniWay employs a **meta-scheduling** capability , optimizing computational workloads by combining the multiple distributed Resource Managers an organisation is using, into a single aggregated view, allowing batch jobs to be directed to the best location for execution, using local resource managers such as LSF, PBSPro, SGE, LoadLeveler. SynfiniWay is able to schedule and execute services which are deployed on a mixed interlinked set of local resource managers.

Chapter-19

# Yaim and Remote Procedure Call

## Yaim


YAIM's official logo - The Yak

**YAIM** - **YAIM Ain't an Installation Manager**, is a tool to configure the middleware of the EGEE project namely the LCG and gLite software packages.

### *Description*

Yaim is a software to configure Grid services, but YAIM also can be used as a general purpose configuration tool completely independent from the Grid middleware. The aim of YAIM is to provide simple configuration methods that can be used to set up uniform Grid sites but can also be easily adapted to meet the needs of larger sites. To ensure that all system administrators can adapt YAIM to the local requirements, it has been implemented as a set of bash scripts. To support EGEE's component based release model YAIM 4 has been modularized and a YAIM core is supplemented by component specific scripts, distributed as separate rpms.

### *Structure*

YAIM's modular structure allows a distributed and asynchronous development which is essential in implementing the quickly changing configuration requirements of the Grid Middleware.

The hierarchical configuration storage - which is to reflect the architecture of a Grid Site - and the configurable function behavior make it easy to implement the local settings along with YAIM's default configuration.

Multi-level logging messages could provide detailed information on the ongoing configuration process.

## YAIM's directory structure

When a YAIM 4 module is installed the following directory structure is created under /opt/glite/yaim:

- **/functions/**: Contains the functions the configure each node types. They are all bash scripts.
- **/functions/local/**: Site administrators can put here their own function definition files. Those will overwrite the default ones coming with the YAIM rpms. The file name has to be the same as the function name defined inside it.
- **/functions/pre/**: Function definition files having the same name as the original YAIM function but the defined function has the '_pre' suffix. This function - if exists - will be executed before the main function.
- **/functions/post/**: Function definition files having the same name as the original YAIM function but the defined function has the '_post' suffix. This function - if exists - will be executed after the main function.
- **/node-info.d/**: Contains a set of files coming with different YAIM modules and they contain the list of functions to be executed during the configuration the that given nodeype. Their name should be the lower-case variant of the node type.
- **/defaults/**:The filenames in this directory are having the same format as they have in node-info.d with a .pre and/or .post suffix. They are sourced correspondingly before and after the main site-info.def and their purpose is to give meaningful default - if possible - values to the variables used by the given module.
- **/bin/**: Contains the main yaim executable.
- **/log/**: The location of YAIM's logfile, yaimlog.
- **/examples/**: This directory contains an example configuration storage. Its structure is explained in the next section.

## YAIM's configuration storage

The /examples/ directory in /opt/glite/yaim is just a guideline and serious site admins should not store their configuration files in that location. YAIM allows having the site's configuration in a well separated and protected place. The configuration files of a site are to be stored in a directory structure having a fixed layout relative to the site's main site-info.def file. The site-info.def file has to sit in a directory (ex.: /root/siteinfodir/ ) which could contain the followings:

- **/site-info.def**: This is the main configuration file of the site, to be sourced first and other optional configuration files will overwrite the values defined here.

- **/services/**: Site administrators can place files into this directory with their names having the same format as in /opt/glite/yaim/defaults. These files are sourced only

when the given service is configured, thus enabling service-specific configuration settings.

- **/nodes/**: Each file in this directory should be the fully qualified name of a host. Only when that host is to be configured, the corresponding file will be sourced which allows host-specific configuration steps to be performed.
- **/vo.d/**: The entries in this directory are to be named as the lower-case version of a VO to be supported. The file can contain VO specific variables and will overwrite the ones defined in the main site-info.def file.
- **/users.conf/**, **groups.conf**: Usually they reside on the same level in the directory hierarchy where the main site-info.def does, but their location is configurable via the USERS_CONF and GROUPS_CONF variables.

## *The bin/yaim executable*

Usage: `/opt/glite/yaim/bin/yaim <action> <parameters>`

Actions:

```
-i | --install     : Install one or several meta package.
                     Compulsory parameters: -s, -m

-c | --configure   : Configure already installed services.
                     Compulsory parameters: -s, -n

-r | --runfunction : Execute a configuration function.
                     Compulsory parameters: -s, -f
                     Optional parameters  : -n

-v | --verify      : Goes through on all the functions and checks that
                     the necessary variables required for a given
                     configuration target are all defined in site-
info.def.
                     Compulsory parameters: -s -n

-d | --debug       : Turns "set -x" on, rude debug information
                     for development. Probably you don't want to use
it.

-e | --explain     : Doesn't perform configuration but explains what
the
                     functions are doing by printing out the comments
                     found inside them.
                     Compulsory parameters: -s -n

-h | --help        : Prints out this help.
```

Parameters:

```
-s | --siteinfo:   : Location of the site-info.def file
-m | --metapackage : Name of the metapackage(s) to install
-n | --nodetype    : Name of the node type(s) to configure
-f | --function    : Name of the functions(s) to execute
```

Examples:

```
Installation:
  /opt/glite/yaim/bin/yaim -i -s /root/siteinfo/site-info.def -m glite-
SE_dpm_mysql

Configuration:
  /opt/glite/yaim/bin/yaim -c -s /root/siteinfo/site-info.def -n
SE_dpm_mysql

Running a function:
  /opt/glite/yaim/bin/yaim -r -s /root/siteinfo/site-info.def -n
SE_dpm_mysql -f config_mkgridmap

Verify your site-info.def:
  /opt/glite/yaim/bin/yaim -v -s /root/siteinfo/site-info.def -n
SE_dpm_mysql
```

## *The configuration flow*

When launching the configuration the different configuration files will be sourced and will overwrite each other in the following order (supposing that we store the site configuration in /root/siteinfo/ directory and configuring service myservice on host myhost and supporting VO myvo):

1. /opt/glite/yaim/defaults/site-info.pre
2. /opt/glite/yaim/defaults/myservice.pre
3. /root/siteinfo/site-info.def
4. /opt/glite/yaim/defaults/site-info.post
5. /opt/glite/yaim/defaults/myservice.post
6. /root/siteinfo/nodes/mynode
7. /root/siteinfo/vo.d/myvo
8. /opt/glite/yaim/node-info.d/myservice

Then for each function myfunc defined in /opt/glite/yaim/node-info.d/myservice it executes the followings if they are defined:

1. /opt/glite/yaim/functions/pre/myfunc.pre
2. /opt/glite/yaim/functions/myfunc or /opt/glite/yaim/functions/local/myfunc
3. /opt/glite/yaim/functions/post/myfunc

## *YAIM's logo - The Yak*

The Yak (Bos grunniens) is a long-haired humped domestic bovine found in Tibet and throughout the Himalayan region of south Central Asia, as well as in Mongolia. The yak is not a gnu.

The author of the logo is **David O'Callaghan**. When he was asked why he has chosen the yak to be the logo of YAIM, he said:

"*The Jargon File defines 'yak shaving' as "Any seemingly pointless activity which is actually necessary to solve a problem which solves a problem which, several levels of recursion later, solves the real problem you're working on. I think this describes the process of installing and configuring grid systems quite well!*

*YAIM aims to take the pain out of grid system administration. So, I thought a yak would make a good logo for YAIM as we no longer have to spend so much time shaving the yak!*"

# Remote procedure call

In computer science, a **remote procedure call** (**RPC**) is an inter-process communication that allows a computer program to cause a subroutine or procedure to execute in another address space (commonly on another computer on a shared network) without the programmer explicitly coding the details for this remote interaction. That is, the programmer writes essentially the same code whether the subroutine is local to the executing program, or remote. When the software in question uses object-oriented principles, RPC is called **remote invocation** or **remote method invocation**.

Note that there are many different (often incompatible) technologies commonly used to accomplish this.

## *History and origins*

The idea of RPC (Remote Procedure Call) goes back at least as far as 1976, when it was described in RFC 707. One of the first business uses of RPC was by Xerox under the name "Courier" in 1981. The first popular implementation of RPC on Unix was Sun's RPC (now called ONC RPC), used as the basis for NFS (Sun).

Another early Unix implementation was Apollo Computer's Network Computing System (NCS). NCS later was used as the foundation of DCE/RPC in the OSF's Distributed Computing Environment (DCE). A decade later Microsoft adopted DCE/RPC as the basis of the Microsoft RPC (MSRPC) mechanism, and implemented DCOM on top of it. Around the same time (mid-90's), Xerox PARC's ILU, and the Object Management Group's CORBA, offered another RPC paradigm based on distributed objects with an inheritance mechanism.

## *Message passing*

An RPC is initiated by the *client*, which sends a request message to a known remote *server* to execute a specified procedure with supplied parameters. The remote server sends a response to the client, and the application continues its process. There are many variations and subtleties in various implementations, resulting in a variety of different

(incompatible) RPC protocols. While the server is processing the call, the client is blocked (it waits until the server has finished processing before resuming execution).

An important difference between remote procedure calls and local calls is that remote calls can fail because of unpredictable network problems. Also, callers generally must deal with such failures without knowing whether the remote procedure was actually invoked. Idempotent procedures (those that have no additional effects if called more than once) are easily handled, but enough difficulties remain that code to call remote procedures is often confined to carefully written low-level subsystems.

### Sequence of events during a RPC

1. The client calls the Client stub. The call is a local procedure call, with parameters pushed on to the stack in the normal way.
2. The client stub packs the parameters into a message and makes a system call to send the message. Packing the parameters is called marshalling.
3. The kernel sends the message from the client machine to the server machine.
4. The kernel passes the incoming packets to the server stub.
5. Finally, the server stub calls the server procedure. The reply traces the same steps in the reverse direction.

### Standard contact mechanisms

To let different clients access servers, a number of standardized RPC systems have been created. Most of these use an interface description language (IDL) to let various platforms call the RPC. The IDL files can then be used to generate code to interface between the client and server. The most common tool used for this is RPCGEN.

## *Other RPC analogues*

RPC analogues found elsewhere:

- Java's Java Remote Method Invocation (Java RMI) API provides similar functionality to standard UNIX RPC methods.
- Modula-3's Network Objects, which were the basis for Java's RMI
- XML-RPC is an RPC protocol that uses XML to encode its calls and HTTP as a transport mechanism.
- JSON-RPC is an RPC protocol that uses JSON encoded messages
- Microsoft .NET Remoting offers RPC facilities for distributed systems implemented on the Windows platform.
- RPyC implements RPC mechanisms in Python, with support for asynchronous calls.
- Pyro Object Oriented form of RPC for Python.
- Etch (protocol) framework for building network services.
- Facebook's Thrift protocol and framework.

- CORBA provides remote procedure invocation through an intermediate layer called the "Object Request Broker"
- DRb allows Ruby programs to communicate with each other on the same machine or over a network. DRb uses remote method invocation (RMI) to pass commands and data between processes.
- AMF allows Flex applications to communicate with back-ends or other applications that support AMF.
- Libevent provides a framework for creating RPC servers and clients.
- Windows Communication Foundation is an application programming interface in the .NET Framework for building connected, service-oriented applications.

Chapter-20

# OpenLink ODBC Drivers and Oracle Fusion Middleware

## OpenLink ODBC Drivers

High-performance, standards-compliant data access drivers, the **OpenLink Drivers for Open Database Connectivity (ODBC)** were arguably the first ODBC driver implementation available for non-Windows platforms such as Unix-like environments. OpenLink has continued to innovate in their drivers with the incorporation of a multi-tier security model, simplification of single-tier and multi-tier administration, and performance that often beats the DBMS-vendor's own drivers, while providing transparent, ODBC-based access to data sources (such as databases) from Desktop Productivity Tools, Application Development Environments, and Web & Internet Points of Presence.

### *History*

First shipped in 1992, by the then PAL Consulting Ltd, the **OpenLink Drivers for Open Database Connectivity (ODBC)** started as Windows-only components, as ODBC was then a Windows-only technology, and were strictly "Single-Tier." At first, they only supported connections to Progress databases, but this quickly expanded to include Unify, Ingres, Oracle, Informix, Sybase, and Microsoft SQL Server databases.

Innovative from the start, in 1993, OpenLink Software introduced a new product that replaced the database-specific proprietary communication layers with its own network layer for serving ODBC clients. The new architecture used a Session Rulebook and a generic client driver, used for connections to all supported database engines, for Windows, Unix, Macintosh, and OS/2 clients. Server components were shipped to run on Windows NT, OS/2, VMS, and Unix-like systems. All database engines supported by the Single-Tier were also handled by the new Multi-Tier implementation.

### A Philosophy of Open Access

OpenLink believed that the benefits of ODBC should be available to users on any platform. However, there was no ODBC Driver Manager available on any other platform.

Therefore, initially, the Windows-only ODBC drivers were accompanied by drivers for UDBC, or Universal Database Connectivity, for use on Unix-like and other non-Windows platforms. In this packaging, all functions of the ODBC Driver Manager were built directly into the data access driver, getting around limitations of the Unix-like environments which generally could not handle dynamic libraries as they do today. This shifted the generic abstraction layer completely from the Driver Manager to the Driver, and was never intended as a permanent solution.

Inspired by their UDBC products, Ke Jin partnered with OpenLink to develop the non-Windows ODBC Driver Manager which they named iODBC, for Independent Open Database Connectivity. This project brought full-powered ODBC support to many non-Windows operating systems including Solaris, AIX, HP-UX, OpenVMS, the BSD and Linux variants, and Mac OS 9.

## Open Access Doesn't Mean Unprotected

By splitting the elements of the driver in this way, OpenLink was able to incorporate several significant layers of centrally-administered additional security. The Session Rulebook restricts client access to the Database server based on multiple-access criteria including the requested DBMS Engine, Database Catalog and/or Schema, and the requesting client Username, Application, OS, and Hostname.

This gives the company Network/Database Administrator ultimate control of who or what groups of users are allowed access to the database, and what sort of access they get. Possible restrictions include limiting any Sales user's query result sets to 100 records (so the contents of an entire customer database cannot be taken to a competitor, and Cartesian product-based Denial-of-Service on the database server instance are impossible); restricting Microsoft Access users to Read Only connections (minimizing accidental database wipe-outs); entirely preventing connections from outside the enterprise LAN IP address space.

Wire-level encryption between the OpenLink client and server components was also added to the mix circa 1994 , long before SSL was an available option -- and SSL itself was also added in due course (circa 2004).

The security models and features provided by OpenLink's Multi-Tier Rulebook remain a central element of their Multi-Tier implementation, and have yet to be matched in any other data access implementation, ODBC or otherwise.

## The Need for Speed

The X/Open and SQL/CLI API efforts were initially supported by the various DBMS vendors, at a time when they recognized that more client tools would be available for use with each DBMS if the application vendors could code to a single API, rather than refactoring their application for each and every supported DBMS.

When ODBC started getting significant uptake, the DBMS vendors seemed to forget this vision. Directly and indirectly, they began supporting grumblings from the field, that ODBC was slow by nature. This generalized knock on the protocol was disproven by published studies, but there was still a hesitance to trust performance claims by ODBC driver vendors.

OpenLink recognized that there were several ways to implement an ODBC driver, and further that their Multi-Tier "Enterprise Edition" was not the fastest of these, even with its Multi-Threaded architecture and support for Advanced Data Access API calls. Although the security and other administrative features of the Multi-Tier implementation have significant value to the enterprise, there are times that raw performance is more important in a deployment.

To deliver the required performance, OpenLink expanded and improved their Single-Tier "Lite Edition" offerings. These drivers typically require the additional installation of database-specific networking components (e.g., Oracle Instant Client, Progress Client Access) on the client host. In some cases, where the database and/or its network protocol is open source (e.g., PostgreSQL, MySQL, TDS for Sybase & Microsoft SQL Server), it is built directly into the driver, and no additional component installation is required.

Once installed, the Lite Edition provides connectivity to both local and remote databases. The user must generally specify a few connection attributes, such as the database instance listening port and hostname, in addition to the instance name. Of course, these Single-Tier drivers maintained the multi-threaded architecture and inherited full support for Advanced Data Access API calls that had been implemented in the Enterprise Edition.

Testing with the open source OpenLink ODBC Bench in one's own environment shows that the Lite Edition drivers measure up to -- and often surpass -- the performance of all other drivers in their class, including those from the DBMS vendor. The Enterprise Edition can also be seen to be no slouch.

## Simplify, Simplify

More recently, complaints about ODBC have focused on the perceived complexity of setting up connections, especially from the end-user's perspective. OpenLink innovation has continued to match.

First, Zeroconf (also known as Bonjour) functionality was implemented in the existing components.

One of the original features of the Enterprise Edition was the ability to force database connection attributes on the user, but the user had to know the basic connection attributes of the Multi-Tier components. With Zeroconf implementation, even this was no longer required, as the administrator could now register "network DSNs" similarly to the old practice of registering network printers -- and the user could choose them through their

local driver manager's setup dialogs, just as they already did with their Printer setup dialogs.

Some DBMS vendors added similar network broadcast or advertising features features to their database engines, and the Lite Edition drivers were enhanced to recognize such broadcasts and make those instances available to users.

Finally, a new breed of driver, the "Express Edition" was created, enabling connections in most cases when the user knows only the instance name. More importantly, the Express Edition requires no secondary component installation, as it wraps an ODBC driver around a Java-based wire protocol library, typically from the DBMS vendor.

### Universal Benefits of Universal Data Access

Through any of these drivers, OpenLink provides the freedom to mix and match "best of class" IT infrastructure components, preventing vendor lock-in that can impede enterprise agility.

By supporting open standards and specifications, these components help preserve existing investment in IT infrastructure, while empowering Information and Knowledge Workers to create new market opportunities.

- Support Advanced Data Access API functionality
- Support all ODBC Scrollable Cursor Models
- Enhance the security of as-shipped DBMS Engines (Multi-Tier only)
- Deliver performance benefits over DBMS vendor-supplied drivers (primarily Single-Tier)

# Oracle Fusion Middleware

**Oracle Fusion Middleware** (OFM, also known as **Fusion Middleware**) consists of several software products from Oracle Corporation. OFM spans multiple services, including Java EE and developer tools, integration services, business intelligence, collaboration, and content management. OFM depends on open standards such as BPEL, SOAP, XML and JMS.

Oracle Fusion Middleware provides software for the development, deployment, and management of service-oriented architecture (SOA). It includes what Oracle calls "hot-pluggable" architecture, designed to facilitate integration with existing applications and systems from other software vendors such as IBM, Microsoft, and SAP AG.

## Evolution

Many of the products included under the OFM banner do not themselves qualify as middleware products: "Fusion Middleware" essentially represents a re-branding of many of Oracle products outside of Oracle's core database and applications-software offerings—compare Oracle Fusion.

According to Oracle, by 2006 over 30,000 organizations had become Fusion Middleware customers, including over 35 of the world's 50 largest companies and more than 750 of the *BusinessWeek* Global 1000, with OFM also supported by 7,500 partners.

In order to provide standards-based software to assist with business process automation, HP has incorporated OFM into its "service-oriented architecture (SOA) portfolio".

Oracle leveraged its Configurable Network Computing (CNC) technology acquired from its PeopleSoft/JD Edwards 2005 purchase.

Oracle Fusion Applications, based on Oracle Fusion Middleware, were finally released in September, 2010.

## Assessments

In January 2008 Oracle Universal Content Management won *InfoWorld*'s "Technology of the Year" award for "Best Enterprise Content Manager", with Oracle SOA Suite winning the award for "Best Enterprise Service Bus".

In 2007 Gartner, Inc. wrote that "OFM has reached a degree of completeness that puts it on par with, and in some cases ahead of, competing software stacks", and reported revenue from the suite of over US$1 billion during FY06, estimating the revenue from the genuinely middleware aspects at US$740 million.

## Oracle Fusion Middleware components

- Enterprise application server
  - Oracle Weblogic Server
  - Oracle Application Server
  - JRockit (a JVM)
  - Tuxedo (software)
- Integration- and process-management
  - BPEL Process Manager
  - Business activity monitoring
  - business rules
  - Business Process Analysis Suite
  - Business process management
  - Oracle Data Integrator (ODI): an application using the database for set-based data integration

    - Enterprise connectivity (adapters)
    - Oracle Enterprise Messaging Service
    - Oracle Enterprise Service Bus
    - Oracle Application server B2B
    - Oracle Service Registry
    - Oracle Web Services Manager (OWSM), a security and monitoring product for web services
- Application development tools
    - Oracle Application Development Framework
    - JDeveloper
    - Oracle SOA Suite
    - TopLink, a Java object-relational mapping package
    - Oracle Forms services
    - Oracle Developer Suite
- Business intelligence
    - Oracle Business Intelligence 10g
    - Oracle Business Activity Monitoring (Oracle BAM)
    - Oracle Crystal Ball - does "predictive modeling, forecasting, and simulation"
    - Oracle Discoverer
    - Data hubs
    - Oracle BI Publisher
    - Oracle Reports services
- Systems management
    - Oracle Enterprise Manager
    - Web services manager
- User interaction
    - Oracle Beehive collaboration software
    - Oracle Portal
    - Oracle WebCenter
    - Real-time collaboration
    - Unified messaging
    - Workspaces
- Content management
    - Oracle Imaging and Process Management
    - Web content management
    - Records management
    - Enterprise search
    - Digital asset management
    - Email archiving
    - Oracle Universal Content Management: In November 2006 Oracle Corporation acquired Stellent, a software-development company (based in Eden Prairie, Minnesota) which provided content management systems. Stellent's primary product, "Universal Content Management" (UCM), the foundation of most of its other content-management products, became *Oracle Universal Content Management* as a part of the Oracle Fusion

Middleware stack. Oracle retained the name "Stellent" for this suite of applications. (Before 2001 Stellent had used the name "Intranet Solutions" and called its product first "IntraDoc!", then briefly "Xpedio".

- Identity management
  - Enterprise Single sign-on
  - Oracle Entitlements Server
  - Oracle Identity Manager
  - Oracle Access Manager
  - Oracle Adaptive Access Manager
  - Oracle Information Rights Management
- Grid infrastructure
  - Services registry
  - application-server security

## *Integration, pricing and bundling*

Apart from selling licenses to run OFM components, Oracle Corporation also markets a managed option via the SaaS Oracle On Demand service.

## Chapter-21

# ProActive

| | ProActive |
|---|---|
| **Developer(s)** | OW2 Consortium |
| **Written in** | Java |
| **Operating system** | Cross-platform |
| **Type** | Grid Computing |
| **License** | GNU General Public License |

**ProActive** is Java grid middleware for parallel, distributed, and multi-threaded computing. It is developed by the OW2 Consortium, including INRIA, CNRS, University of Nice Sophia Antipolis, and ActiveEon. It is open-source software released under the GPL license.

ProActive provides a comprehensive framework and parallel programming model to simplify the programming and execution of parallel applications running on multi-core processors, distributed on Local Area Network (LAN), on clusters and data centers, on intranets, and on Internet grids.

The ProActive programming model combines the active object design pattern with futures objects.

## *Programming model*

The model was created by Denis Caromel, professor at University of Nice Sophia Antipolis. Several extensions of the model were made later on by members of the OASIS team at INRIA. The book *A Theory of Distributed Objects* presents the ASP calculus that formalizes ProActive features, and provides formal semantics to the calculus, together with properties of ProActive program execution.

## *Active objects*

Active objects are the basic units of activity and distribution used for building concurrent applications using ProActive. An active object runs with its own thread. This thread only

executes the methods invoked on this active object by other active objects, and those of the passive objects of the subsystem that belongs to this active object. With ProActive, the programmer does not have to explicitly manipulate Thread objects, unlike in standard Java.

Active objects can be created on any of the hosts involved in the computation. Once an active object is created, its activity (the fact that it runs with its own thread) and its location (local or remote) are perfectly transparent. Any active object can be manipulated as if it were a passive instance of the same class.

An *active object* is composed of two objects: a *body*, and a standard Java object. The body is not visible from the outside of the active object.

The body is responsible for receiving calls (or *requests*) on the active object and storing them in a queue of pending calls. It executes these calls in an order specified by a synchronization policy. If a synchronization policy is not specified, calls are managed in a "First in, first out" (FIFO) manner.

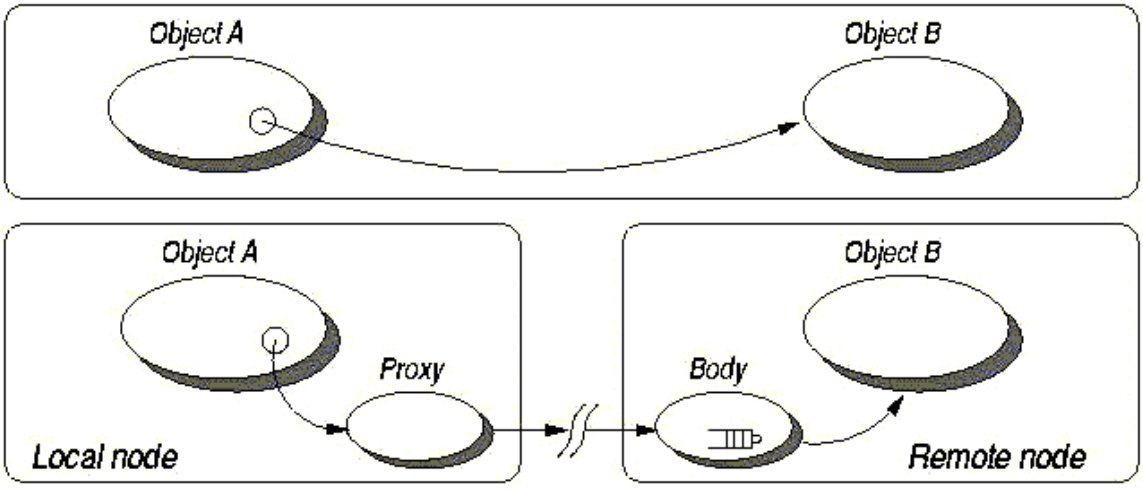
The thread of an active object then chooses a method in the queue of pending requests and executes it. No parallelism is provided inside an active object; this is an important decision in ProActive's design, enabling the use of "pre-post" conditions and class invariants.

On the side of the subsystem that sends a call to an active object, the active object is represented by a *proxy*. The proxy generates future objects for representing future values, transforms calls into Request objects (in terms of metaobject, this is a reification) and performs deep copies of passive objects passed as parameters.

## Active object basis

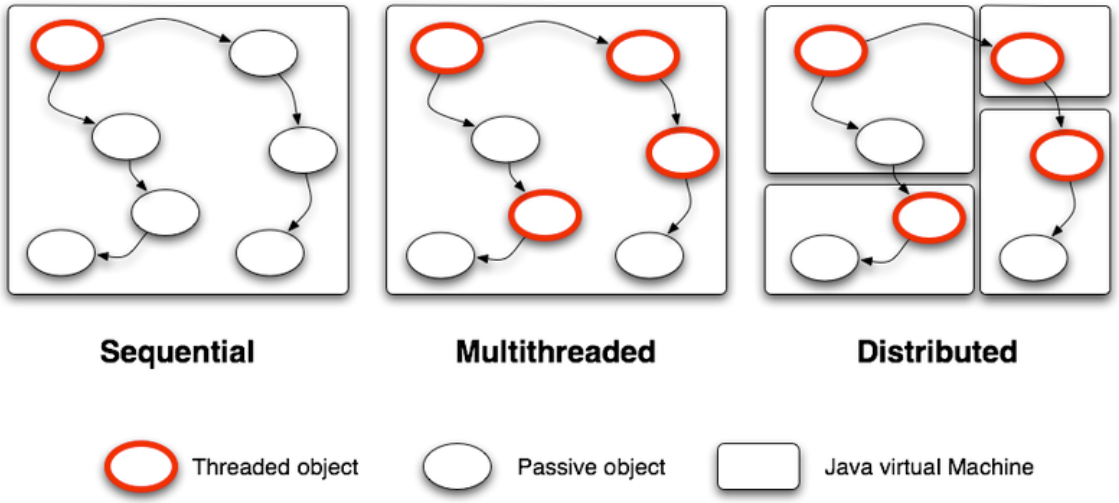
ProActive is a library designed for developing applications in the model introduced by Eiffel//, a parallel extension of the Eiffel programming language.

In this model, the application is structured in *subsystems*. There is one active object (and therefore one thread) for each subsystem, and one subsystem for each active object (or thread). Each subsystem is thus composed of one active object and any number of passive objects—possibly no passive objects. The thread of one subsystem only executes methods in the objects of this subsystem. There are no "shared passive objects" between subsystems.




A call onto an active object, as opposed to a call onto passive one

These features impact the application's topology. Of all the objects that make up a subsystem—the active object and the passive objects—only the active object is known to objects outside of the subsystem. All objects, both active and passive, may have references onto active objects. If an object *o1* has a reference onto a passive object *o2*, then *o1* and *o2* are part of the same subsystem.




The model: Sequential, multithreaded, distributed

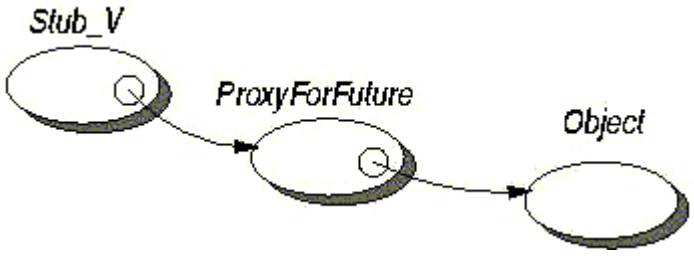
This has also consequences on the semantics of message-passing between subsystems. When an object in a subsystem calls a method on an active object, the parameters of the call may be references on passive objects of the subsystem, which would lead to shared passive objects. This is why passive objects passed as parameters of calls on active

objects are always passed by deep-copy. Active objects, on the other hand, are always passed by reference. Symmetrically, this also applies to objects returned from methods called on active objects.

Thanks to the concepts of asynchronous calls, futures, and no data sharing, an application written with ProActive doesn't need any structural change—actually, hardly any change at all—whether it runs in a sequential, multi-threaded, or distributed environment.

## *Asynchronous calls and futures*

Whenever possible, a method call on an active object is reified as an asynchronous request. If not possible, the call is synchronous, and blocks until the reply is received. If the request is asynchronous, it immediately returns a *future object*.




A future object

The future object acts as a placeholder for the result of the not-yet-performed method invocation. As a consequence, the calling thread can go on with executing its code, as long as it doesn't need to invoke methods on the returned object. If the need arises, the calling thread is automatically blocked if the result of the method invocation is not yet available. Although a future object has structure similar to that of an active object, a future object is not active. It only has a Stub and a Proxy.

## A simple example

The code excerpt below highlights the notion of future objects. Suppose a user calls a method `foo` and a method `bar` from an active object `a`; the `foo` method returns void and the `bar` method returns an object of class `V`:

```
// a one way typed asynchronous communication towards the (remote) AO a
// a request is sent to a
a.foo (param);

// a typed asynchronous communication with result.
// v is first an awaited Future, to be transparently filled up after
// service of the request, and reply
V v = a.bar (param);
...
// use of the result of an asynchronous call.
// if v is still an awaited future, it triggers an automatic
```

```
// wait: Wait-by-necessity
v.gee (param);
```

When `foo` is called on an active object `a`, it returns immediately (as the current thread cannot execute methods in the other subsystem). Similarly, when `bar` is called on `a`, it returns immediately but the result `v` can't be computed yet. A future object, which is a placeholder for the result of the method invocation, is returned. From the point of view of the caller subsystem, there is no difference between the future object and the object that would have been returned if the same call had been issued onto a passive object.

After both methods have returned, the calling thread continues executing its code as if the call had been effectively performed. The role of the future mechanism is to block the caller thread when the `gee` method is called on `v` and the result has not yet been set : this inter-object synchronization policy is known as *wait-by-necessity*.

## Chapter-22

# Jsonwsp

JSON-WSP is short for JavaScript Object Notation Web-Service Protocol. Simply put it is a web-service protocol that uses JSON for service description, requests and responses. It is very much inspired from JSON-RPC, but The lack of a service description specification with documentation in JSON-RPC sparked the design of JSON-WSP.

The description format has the same purpose for JSON-WSP as WSDL has for SOAP or IDL for CORBA which is to describe the types and methods used in a given service. It also describes intertype relations (i.e. nestet types) and defines which types are expected as method arguments and which types the user can expect to receive as method return values. Finally the description opens the possibility to add documentation on service, method, parameter and return levels.

Communication between clients and a JSON-WSP server is carried out using HTTP POST requests and responses, with the JSON objects as data with the content-type application/json.

## *Specifications*

JSON-WSP consists of 4 JSON object specifications:

| Specification | Description |
|---|---|
| description | Service description specification (like WSDL). This specification describes methods, method parameters, types and return types. It also supports user documentation on service, method and parameter levels. |
| request | Specification for JSON requests. It contains information about which method that is to be invoked and all the arguments for the method call. Arguments in the request must obey the parameter definition of the same method described in the corresponding JSON-WSP description. |
| response | Specification for JSON responses. The response object contains the reslult of a service method invokation. The return type must obey the defined return type of the same method in the corrosponding JSON-WSP description. |

| | |
|---|---|
| fault | Specification for JSON fault responses. The fault object contains a fault code and a fault string. The fault information specifies whether the fault occurred on the client or server side. Depending on the server side service framework more detailed information can be extracted, i.e. the filename and line number where the fault occurred. |

## Understanding the specification notation

## Building blocks

- If the name of the building-block being defined starts with **rx-** it means that the definition is a regular expression. In these definitions square brackets have the role of defining character classes and parentheses have the role of defining capturing groups.
- In all other cases square brackets notate lists and parentheses notate either a decition:
- `( d1 | d2 | ... )`

  a repetition of 0-many:

  `( ... )*`

  a repetition 1-many:

  `( --- )+`

  or something optional:

  `( ... )?`

## Common building-blocks

```
<rx-freetext> = ".*"

<rx-identifier> = "[a-zA-Z_][a-zA-Z0-9_]*"

<rx-number> = [0-9]+

<rx-boolean> = (true|false)

<key> = <rx-identifier>

<primitive-value> = ( <rx-freetext> | <rx-number> | <rx-boolean> )

<value> = (
    <primitive-value> |
    [ ( <value>, )* ] |
    { ( <key>: <value>, )* } )

<method-name> = <rx-identifier>
```

```
<service-name> = <rx-identifier>
```

### The JSON-WSP description object

## Additional Building-blocks

```
<primitive> = ( "string" | "number" | "float" )

<service-locator> = <rfc-1738 compliant string>

<type-name> = <rx-identifier>

<member-name> = <rx-identifier>

<multi-type> = ( <primitive> | <type-name> | [<primitive>] | [<type-
name>] )

<doc-string> = <rx-freetext>

<param-name> = <rx-identifier>

<def-order> = <rx-number>

<param-optional> = <rx-boolean>
```

## Specification

```
{
    "type": "jsonwsp/description",
    "version": "1.0",
    "servicename": <service-name>,
    "url": <service-locator>,
    "types": { (
        <type-name>: { (
            <member-name>: <multi-type> )+
        } )*
    },
    "methods": { (
        <method-name>: {
            "doc_lines": [ ( <doc-string>, )* ],
            "params": { (
                <param-name>: {
                    "doc_lines": [ ( <doc-string>, )* ],
                    "def_order": <def-order>,
                    "type": <multi-type>,
                    "optional": <param-optional>
                }, )*
            },
            "ret_info": {
                "doc_lines": [ ( <doc-string>, )* ],
                "type": <multi-type>
            }
        } )+
    }
}
```

## Descriptions

**<service-locator>:** The service endpoint URL that accepts JSON-WSP POST request objects.

**<service-name>:** Service name is case sensitive. It identifies a specific service exposed on a specific server.

**doc_lines:** Each doc-string contained in a doc_lines list reflects a single line of documentation that relates to the parent object of the doc_lines.

### The JSON-WSP request object

The request object contains information about which method to invoke and what arguments to invoke the method with. It also stores information about the type and version of itself.

The optional **mirror** value can be used to send information from the client which will then be reflected by the server and returned unchanged in the response object's **reflection** value. This feature allows clients to send multiple requests to a method and send request identification values that can be intercepted by the client's response handler. This is often nessecary from javascript if more than one request is being processed simultaneously by the server and the response order is unknown by the client.

## Specification

```
{
    "type": "jsonwsp/request",
    "version": "1.0",
    "methodname": <method-name>,
    "args": { ( <key>: <value>, )* }(,
    "mirror": <value> )?
}
```

### The JSON-WSP response object

## Specification

The **reflection** value is an unchanged server reflection of the request object's **mirror** value. It is marked as optional because it is the client that controls via the request whether it is there or not.

```
{
    "type": "jsonwsp/response",
    "version": "1.0",
    "servicename": <service-name>,
    "methodname": <method-name>,
    "result": <value>(,
    "reflection": <value> )?
```

```
}
```

### The JSON-WSP fault response object

## Additional Building-blocks

```
<fault-code> = ( "incompatible" | "client" | "server" )

<fault-string> = <rx-freetext>

<fault-filename> = <rx-freetext>

<fault-lineno> = <rx-number>
```

## Specification

```
{
    "type": "jsonwsp/fault",
    "version": "1.0",
    "fault": {
        "code": <fault-code>,
        "string": <fault-string>,
        ("detail": [ ( <fault-string>, )* ] ,)?
        ("filename": <fault-filename>,)?
        ("lineno": <fault-lineno>,)?
    },
}
```

## Descriptions

**<fault-code>:** The meanings of the possible fault-codes:

- "incompatible": Client version of JSON-WSP is incompatible with the server version of JSON-WSP. Typically you will encounter this type of fault-code if there is a version major in difference between the client and the server.
- "server": An error occurred on the server side after the client request has been successfully consumed.
- "client": The clients request could not be consumed by the server due to incorrect format or missing required arguments etc.

## *Real world example*

### Description

```
{
    "type": "jsonwsp/description",
    "version": "1.0",
    "servicename": "UserService",
    "url":
"http://testladon.org:80/proxy.php?path=UserService/jsonwsp",
    "types": {
        "Group": {
            "group_id": "number",
            "display_name": "string",
```

```
            "name": "string",
            "members": ["User"]
        },
        "User": {
            "username": "string",
            "user_id": "number",
            "mobile": "string",
            "age": "number",
            "given_name": "string",
            "surname": "string"
        },
        "CreateUserResponse": {
            "user_id": "number",
            "success": "boolean"
        }
    },
    "methods": {
        "listUsers": {
            "doc_lines": ["List Users that have a username, given_name
or surname that matches a given filter."],
            "params": {
                "name_filter": {
                    "def_order": 1,
                    "doc_lines": ["String used for filtering the
resulting list of users."],
                    "type": "string",
                    "optional": false
                }
            },
            "ret_info": {
                "doc_lines": ["List of users."],
                "type": ["User"]
            }
        },
        "listGroups": {
            "doc_lines": ["List Groups that have a name or display_name
that matches a given filter."],
            "params": {
                "name_filter": {
                    "def_order": 1,
                    "doc_lines": ["String used for filtering the
resulting list of groups."],
                    "type": "string",
                    "optional": false
                }
            },
            "ret_info": {
                "doc_lines": ["List of groups."],
                "type": ["Group"]
            }
        },
        "createUser": {
            "doc_lines": ["Create a new user account."],
            "params": {
                "username": {
                    "def_order": 1,
```

```
                    "doc_lines": ["Unique username for the new user
account."],
                    "type": "string",
                    "optional": false
                },
                "given_name": {
                    "def_order": 2,
                    "doc_lines": ["First name."],
                    "type": "string",
                    "optional": false
                },
                "surname": {
                    "def_order": 3,
                    "doc_lines": ["Last name."],
                    "type": "string",
                    "optional": false
                },
                "mobile": {
                    "def_order": 4,
                    "doc_lines": ["Optional mobile number."],
                    "type": "string",
                    "optional": true
                },
                "age": {
                    "def_order": 5,
                    "doc_lines": ["Optional age of the person behind
the account."],
                    "type": "number",
                    "optional": true
                }
            },
            "ret_info": {
                "doc_lines": [],
                "type": "CreateUserResponse"
            }
        }
    }
}
```

### Service call 1

## Request

```
{
    "type": "jsonwsp/request",
    "version": "1.0",
    "methodname": "createUser",
    "args": {
        "username": "bettyw",
        "given_name": "Betty",
        "surname": "Wilson",
        "mobile": "555-3423444"
    },
    "mirror": {
        "id": 2
    }
}
```

## Response

```
{
    "type": "jsonwsp/response",
    "version": "1.0",
    "servicename": "UserService",
    "method": "createUser",
    "result": {
        "user_id": 324,
        "success": true
    }
    "reflection": {
        "id": 2
    }
}
```

## Service call 2

## Request

```
{
    "type": "jsonwsp/request",
    "version": "1.0",
    "methodname": "listUsers",
    "args": {
        "name_filter": "jack"
    }
}
```

## Response

```
{
    "type": "jsonwsp/response",
    "version": "1.0",
    "servicename": "UserService",
    "method": "listUsers",
    "result": [{
        "username": "jackp",
        "user_id": 153,
        "mobile": "555-377843",
        "age": 34,
        "given_name": "Jack",
        "surname": "Petersen"
    }, {
        "username": "bradj",
        "user_id": 321,
        "mobile": "555-437546",
        "age": 27,
        "given_name": "Brad",
        "surname": "Jackson"
```