CSNB113: System Administration
-
13th Topic:
Services Using the Network

Services - Servers

With a clear definition of services and servers, plus an understanding of networking, we can now start to think of using the network, and our server; into which we install **more services**.

So, what server-based-services can we install?

web-server
file server
print server
mail server
database server

There are more that we have already mentioned:
ssh server
ftp server
name server

When you look at servers in a ***task-list***, you find – at least in *nix – a lot of '**d**' at the end: that stands for ***d****aemon*, and identifies a process running a server.

Protocols

The term ***Protocol*** is used in a number of meanings. There is 'protocol' as the way how we as humans **interact**; there is something like 'Diplomatic Protocol', that decides on how Heads of State need to be welcomed at the airport when visiting another country; for example. The 'Red Carpet' rolled out is part of this **set of rules**.

We can say that a **protocol is a set of rules**.

Why do we need to explain this? What does it have to do with servers?

This is answered easily: **Different** servers fulfill **different tasks** and therefore there will be **different ways to interact** with these servers.

When you send a mail, you have to specify the sender and the addressee; maybe a subject, and a body.

When you contact a webserver, you need to pass a URL (Universal Resource Locator), and download the HTML data.

When you contact a Domain Name Server, you will pass a domain name and have the IP-address returned.

Overall: The way to use / interact with a server, is governed by a set of rules, the so-called protocol

Web-Server

This type of servers has become the most often used ones in the last decade. Actually, it was 'invented' rather late; when the Internet had already been in existence for a long time.

In 1990, Tim Berners-Lee suggested to develop an information system that would create a ***web of information***. In 1990, he wrote the Hypertext Transfer Protocol (HTTP)—the language computers would use to communicate hypertext documents over the Internet and designed a scheme to give documents addresses on the Internet.

By the end of the year he had also written a client program (***browser***) to retrieve and view ***hypertext*** documents. He called this client "WorldWideWeb." Hypertext pages were formatted using the Hypertext Markup Language (HTML) that Berners-Lee had written. He also wrote the first web server. A web server is the software that stores web pages on a computer and makes them available to be accessed by others.

Berners-Lee set up the first web server known as "info.cern.ch." at CERN.

(adapted from http://www.ibiblio.org/pioneers/lee.html)

This is a good example for the idea of a *protocol*. Berners-Lee used this term, like many others before him. That's why we have so many 'P' in the servers.

(Actually, a web-server is usually found in a system as 'httpd': a ***d***aemon for ***h***yper***t***ext ***t***ransfer ***p***rotocol.)

Mail Server

A mail server is used to send e-mail from one terminal to another. This method has evolved very much since the first e-mail ever. (It remains unclear who actually **did** it, but Ray Tomlinson is credited to have formulated the concept of "dumping a message on someone else's terminal".)

Ray Tomlinson was the first to use the '@' symbol to indicate the user who was "at" what computer.

The original concept of mail was to transfer a message. That's why the protocol is also known as **S**imple **M**ail **T**ransfer **P**rotocol – **SMTP**.

(It should be noted, that the 'simple' does not stand for 'simple protocol' – SMTP actually is anything but simple - , but rather for 'simple mail', because it was written for simple, unformatted, short mails.)

Also, SMTP does not care about delivery. It only cares about transmitting / ***transfer***ring the message. The ***delivery*** is done in another manner, though that will be covered in a future course ... .

File Transfer

File Transfer has become much less fashionable over the last decade. More modern protocols have superseded the rather antique **F**ile **T**ransfer **P**rotocol (**FTP**).

One of the reasons is, that this protocol sends all data as ***plaintext***; that means ***unencrypted***. So anyone in between could read the so-called message; that is everyone can ***intercept*** the files during transfer and read, copy, and use the data.

On the other hand, it is still a very convenient protocol to transfer files, and in **both directions**. Especially web-servers are not written with upload in mind.

FTP is still used worldwide to update the websites of users on web servers. The users can log on to an FTP-server (usually on the same machine as the web server), and then they can handle their files: delete, rename, as well as upload new files. Whatever they do, will automatically be reflected in the website, because the users have access to their data in their respective directories on that web server through FTP).

Also, there is so-called ***anonymous ftp***, which allows anyone to **download** (only!) files from an anonymous ftp-server. This can be done easily with any web browser; because all modern web browsers have at least an ftp-download functionality build-in.

Try this out:

`http://metalab.uniten.edu.my`

`ftp://metalab.uniten.edu.my`

are two totally different things; and both lead you to very different protocols.

(Remember lab exercise 1; when we downloaded the server-install-CD -image from ftp://metalab.uniten.edu.my?)

Print Server

Print servers are a great idea. While people tend to prefer to have their own printer, any more advanced printer (fast, laser, colour) is expensive, consumes a lot of energy. And when used infrequently, becomes inefficient.

There are (unfortunately) a number of different protocols for printing; so the system administrator needs to learn more here.

- The most ancient one is **lpd**, the **L**ine **P**rinter **D**aemon. It is – like all servers – waiting for a client to connect: and dump the data to be printed into its ***queue***. That's a directory, the so-called ***spooler***, where all files (data) are printed one after the other.

  (This is why the pages of multiple print jobs are not getting mixed up.)
- Another one is the so-called HP **JetDirect** method (guess: invented by which company??) that is usually implemented on stand-alone printers, that are directly connected to the network.
- A modern one is the **I**nternet **P**rinting **P**rotocol (**IPP**). It is modern, because it allows the client (e.g. your workstation) to query the remote printer about its capabilities; the paper reservoir,the driver, the margins, etc. The IPP is a 'real' service to which the clients connect that want to get something printed. On the client machines, the printer often looks like a local printer.

With respect to administration: Modern print servers can be administrated remotely. So the administrator can log in through the network, and change settings, paper sizes, permissions, etc.

File Server

File Servers are a somewhat contentious item. Though it is a convincing idea to (transparently) store all user data on a central file server, often this is not done. Why? Because it takes a great effort on the side of the system administrator. It also requires a good and reliable network infrastructure (when the network is down, the users cannot do any work). Also, the system administrator becomes responsible and accountable for all data belonging to a user.

File servers put a lot of work load on the system administrators.

On the other hand, file servers – if managed properly – help a lot in the organisation: Users are not responsible any longer, and when a machine goes down, crashes, the hard disk dies, this same user can continue without any problems doing the work immediately on another machine; because the logon associates ('mounts') the remote data to the work station to which the user is logged on.

There are principally two methods to **handle shared folders / directories**. One is based on a concepts from Windows; on the Network Neighborhood. The directories can be administrated by the Windows system administrator (via Active Directory).

Access from and to *nix-machines is possible using a *Samba*-server (smbd).

The other one (found on *nix) is the **Network File System** (NFS), developed by SUN.

Whichever method used, there is a lot of (administrative) overhead still needed.

Backup Server

The term ***Backup Server*** can have two meanings:

- a server used to backup data (this is what we discuss here)
- a server that stands ready to take over when the main server goes down (not our topic here)

A backup server is a not-so-interesting item; and can become very relevant. It is often not powerful, not fast. The main idea is to have it ***off-site***. That means, when a main site, or the main machine, crashes, goes down, gets stolen, burns in a fire or is 'hacked', the data on the backup server are still available.

Since it is connected over the network, it makes not much sense to send all data all the time. Rather, it is like a strategy that we saw earlier with respect to backup: Starting with a **complete copy**; followed by only storing the **changes** of the file; storing only the data that have actually changed.

A handy utility for this is **rsync** (***remote synchronisation***). This means that the content of a remote site (server) reflects the content of the actual site (server).

For a complete, enterprise-ready backup solution ***Bacula*** can be used.

Database Server

The database server is slightly different from the other types of servers, since databases are systems of their own. Databases are usually administrated by ***database administrators***. (Check the courses of your degree program: there is at least one course totally dedicated to databases.) Databases can be very complex systems, with their administrative tools; their own generation and administration of users, their own backup utilities to solve backup problems specific to databases.

The system administrator still has some work to do with databases: Usually the system administrator ***hosts*** the database. That means he / she provides the system (hardware and operating system), and often also the installation, including the account of the database administrator. From here on, usually the database administrator takes over.

The system administrator could come in for patches and upgrades, that have to be communicated and agreed upon with the database administrator.

Server Administration Tasks

Again, we come back to this topic. We discussed these earlier, in the beginning of this course. Then, we picked up on this topic when we introduced services on the network: DHCP and DNS need to be administrated.

(Routing is usually done by the network administrator.)

Here we have introduced more services. They need to be administrated:

The user accounts and URLs need to be created and set up. Users need to be given a website (or just not be given a website). They must be allowed (or just not be allowed) to update their websites.

Some user, **not** the system administrator (!!) must receive the permissions to update for example the main website of an organisation. This latter person is usually called the ***web-master***. (Usually the web-master is not the system administrator.)

Permissions have to be given out for users to use printers; we might want to keep a record of number of pages printed overall or by a specific user.

File servers need a lot of work: the shares have to be created, the user permissions given, and – since the users tend to put their relevant data on those shares – reliability must be guaranteed. Backups of the shared files need to be done frequently.

Printers need to be shared, databases installed, etc.

Differences??

Hopefully, everyone is aware now about the differences between a server and a desktop (laptop, workstation)!?

We have pointed out a number of differences, with respect to hardware, operating systems, processing power.

Though mainly, in the previous and this lecture we have identified typical server tasks, which are like servants' tasks, often network-related, and almost never directly accessed by a user; be it a typist, a scientist, a doctor or a lecturer.

Many – if not all – computer users administrate their machine (or have them administrated); but then it is always the administration of their own machines, for their own purpose.

A system administrator in the strict sense is the person that administrates a machine for a plurality of users, often user over the network, and often focuses on providing **services** for the user that we can by now consider **servers**. Running a local program, be it an office program, a web-browser, a mail-client, a drawing program, an image editing program, is not running services.

Services, their users, features, accessibilities, updates and upgrades are the responsibility of a system administrator.

All in all, though the tasks of a system administrator can all be done manually, any clever system administrator will think hard to simplify his / her task as much as possible by writing shell scripts.

References

- http://www.ibiblio.org/pioneers/lee.html
- http://inventors.about.com/od/estartinventions/a/email.htm
- http://en.wikipedia.org/wiki/Internet_Printing_Protocol
- http://en.wikipedia.org/wiki/Samba_(software)
- http://nfs.sourceforge.net/
- http://en.wikipedia.org/wiki/Rsync
- http://www.bacula.org/en/