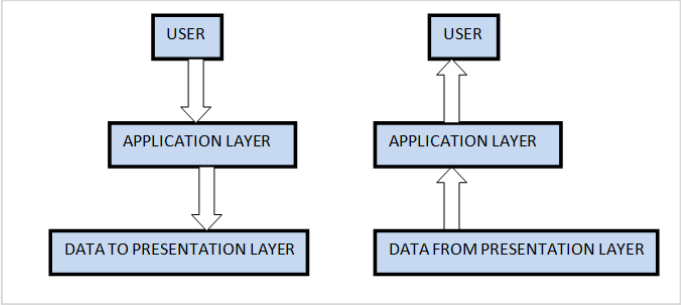
6.1 Web: HTTP & HTTPS
6.2 File Transfer: FTP, PuTTY, WinSCP
6.3 Electronic Mail: SMTP, POP3, IMAP
6.4 DNS
6.5 P2P Applications
6.6 Socket Programming
6.7 Application server concept: proxy caching, Web/Mail/DNS server optimization
6.8 Concept of traffic analyzer: MRTG, PRTG, SNMP, Packet tracer, Wireshark.

- Application layer is the top most layer in OSI and TCP/IP layered model. This layer exists in both layered Models because of its significance, of interacting with user and user applications. This layer is for applications which are involved in communication system.
- A user may or may not directly interacts with the applications. Application layer is where the actual communication is initiated and reflects. Application layer takes the help of transport and all layers below it to communicate or transfer its data to the remote host.
- When an application layer protocol wants to communicate with its peer application layer protocol on remote host, it hands over the data or information to the Transport layer. The transport layer does the rest with the help of all the layers below it.
- The application layer is not the application itself that is doing the communication. It is a service layer that provides these services:
  - Simple Mail Transfer Protocol
  - File transfer
  - Web surfing
  - Web chat
  - Email clients
  - Network data sharing
  - Virtual terminals
  - Various file and data operations



6.1 Web : *HTTP & HTTPS*

HTTP **-** On the other hand, when we use a Web Browser, which is actually using Hyper Text Transfer Protocol (HTTP) to interact with the network. HTTP is Application Layer protocol.

The Hyper Text Transfer Protocol (HTTP) is the foundation of World Wide Web. Hypertext is well organized documentation system which uses hyperlinks to link the pages in the text documents. HTTP works on client server model. When a user wants to access any HTTP page on the internet, the client machine at user end initiates a TCP connection to server on port 80. When the server accepts the client request, the client is authorized to access web pages.

To access the web pages, a client normally uses web browsers, who are responsible for initiating, maintaining, and closing TCP connections. HTTP is a stateless protocol, which means that the connection between the browser and the server is lost once the transaction ends, which means does not require the server to retain session information or status about each communications partner for the duration of multiple requests.

- HTTP is an application layer protocol
- The default port if not mentioned in the request, is assumed as 80
- The hostname in the request is case insensitive
- World Wide Web Consortium (W3C) and the Internet Engineering Task Force (IETF), both coordinates in the standardization of the HTTP protocol
- HTTP allows the improvement of its request and response with the help of intermediates in between (*e.g. a gateway, a proxy, or a tunnel*)
- The resources that can be requested by using HTTP protocol is made available with the help of a type of **URI** (Uniform Resource Identifier) called **URL** (Uniform Resource Locator).
- TCP (Transmission Control Protocol), is used to establish a connection to the application layer port 80 used by HTTP.
- A series of request and response in http is called as a session in HTTP
- HTTP version 0.9 was the first documented version of HTTP
- HTTP is a stateless protocol (which means each and every connection is independent of each other.)

How HTTP works?

HTTP is a request response protocol to communicate asynchronously between client and server. Mostly in HTTP a browser acts as a client and a web-server like Apache or IIS acts as server. The server which hosts the files (like html, audio, video files etc.) responses to the client. Depending on the request a response contains the status of the request.

Types of HTTP requests

- **GET:** This request is Used to get the Response header and the response body!!
- **HEAD:** This request is used to get back the response header only (not the response body as returned by the GET request.)
- **POST:** This request is used to submit data (e.g.: for to be used in HTML forms etc.)
- **PUT:** Used for uploading resource
- **PATCH:** Is used to modify the resource
- **DELETE:** Used for deleting resource

HTTPS (also called HTTP over TLS, HTTP over SSL, and HTTP Secure) is a protocol for secure communication over a computer network which is widely used on the Internet. HTTPS consists of communication over Hypertext Transfer Protocol (HTTP) within a connection encrypted by Transport Layer Security (TLS) or its predecessor, Secure Sockets Layer. The main motivation for HTTPS is authentication of the visited website and protection of the privacy and integrity of the exchanged data.

In its popular deployment on the internet, HTTPS provides authentication of the website and associated web server with which one is communicating, which protects against man-in-the-middle attacks. Additionally, it provides bidirectional encryption of communications between a client and server, which protects against eavesdropping and tampering with or forging the contents of the communication. In practice, this provides a reasonable guarantee that one is communicating with precisely the website that one intended to communicate with (as opposed to an impostor), as well as ensuring that the contents of communications between the user and site cannot be read or forged by any third party.

Historically, HTTPS connections were primarily used for payment transactions on the World Wide Web, e-mail and for sensitive transactions in corporate information systems. In the late 2000s and early 2010s, HTTPS began to see widespread use for protecting page authenticity on all types of websites, securing accounts and keeping user communications, identity and web browsing private.

Difference from HTTP

- HTTPS URLs begin with "https://" and use port 443 by default, whereas HTTP URLs begin with "http://" and use port 80 by default.
- HTTP is not encrypted and is vulnerable to man-in-the-middle and eavesdropping attacks, which can let attackers gain access to website accounts and sensitive information, and modify webpages to inject malware or advertisements. HTTPS is designed to withstand such attacks and is considered secure against them (with the exception of older, deprecated versions of SSL).

6.2 File Transfer : FTP, PuTTY, WinSCP

* File Transfer Protocol(FTP)

FTP helps a user to transfer text based or binary files across the network. A user can use this protocol in either GUI based software like FileZilla or CuteFTP and the same user can use FTP in Command Line mode.

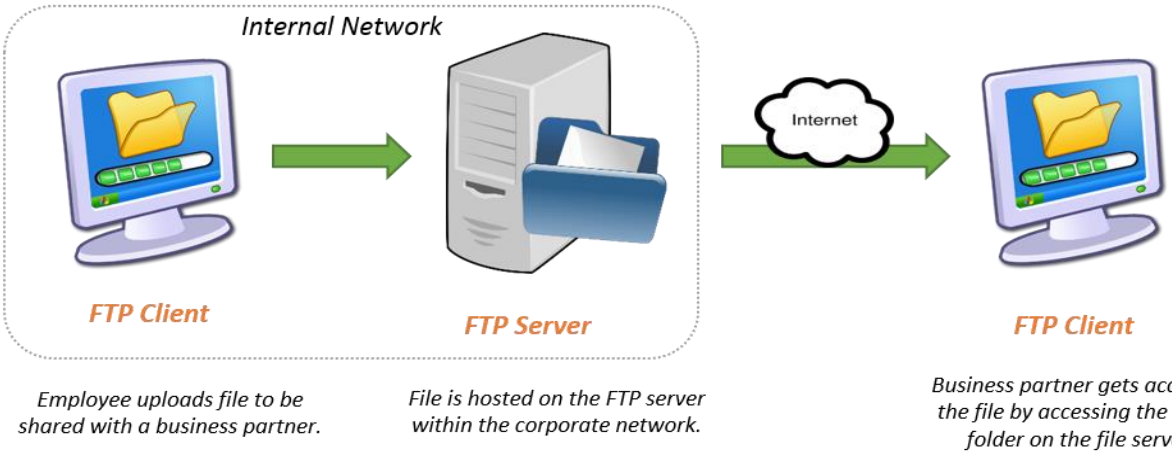
Hence, irrespective of which software you use, it is the protocol which is considered at Application Layer used by that software. DNS is a protocol which helps user application protocols such as HTTP to accomplish its work.

Features of FTP

- Access control : hosts normally use for access control on files, login by username and password.
- Filename : native filenames or universal filenames. FTP uses native filename.
- File translation : two types of file format : local types or universal file format. Universal type: *files to be translated from local to the universal type on transmission and from universal to local type translation for storage.*

The FTP operation supports two types of actions:

- *Get* — Used to download data from the FTP server.
- *Send* — Used to upload data to the FTP server.



For example, if you create a web page on your computer, you would use FTP to transfer your web page design to your actual website.

* PuTTY

- PuTTY is a free and open-source terminal emulator, serial console and network file transfer application. It supports several network protocols, including SCP, SSH, Telnet, rlogin, and raw socket connection. It can also connect to a serial port.
- PuTTY was originally written for Microsoft Windows, but it has been ported to various other operating systems. Official ports are available for some Unix-like platforms, with work-in-progress ports to Classic Mac OS and macOS, and unofficial ports have been contributed to platforms such as Symbian, Windows Mobile and Windows Phone.
- PuTTY was written and is maintained primarily by Simon Tatham.

Features:

- PuTTY supports many variations on the secure remote terminal, and provides user control over the SSH encryption key and protocol version, alternate ciphers such as 3DES, Arcfour, Blowfish, DES, and Public-key authentication.

- It also can emulate control sequences from xterm, VT102 or ECMA-48 terminal emulation, and allows local, remote, or dynamic port forwarding with SSH (including X11 forwarding).
- The network communication layer supports IPv6, and the SSH protocol supports the zlib@openssh.com delayed compression scheme. It can also be used with local serial port connections.
- PuTTY comes bundled with command-line SCP and SFTP clients, called "pscp" and "psftp" respectively, and plink, a command-line connection tool, used for non-interactive sessions.

PuTTY consists of several components:

- PuTTY: the Telnet, rlogin(login on another host via network), and SSH(Secure Shell) client itself, which can also connect to a serial port
- PSCP: an SCP(Secure Copy) client, i.e. command-line secure file copy
- PSFTP: an SFTP(SSH FTP) client, i.e. general file transfer sessions much like FTP
- PuTTYtel: a Telnet-only client
- Plink: a command-line interface to the PuTTY back ends
- Pageant: an SSH authentication agent for PuTTY, PSCP and Plink
- PuTTYgen: key generation utility e.g. an RSA(Rivest–Shamir–Adleman cryptosystem), DSA(Digital Signature Algorithm), ECDSA(Elliptic Curve Digital Signature Algorithm) and EdDSA(Edwards-curve Digital Signature Algorithm)
- pterm: a standalone terminal emulator

*** WinSCP**

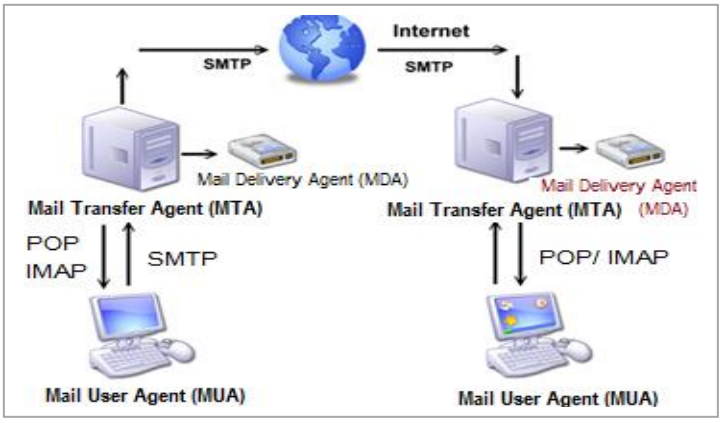
WinSCP (***Win****dows* ***S****ecure* ***Cop****y*) is a free and open-source SFTP, FTP, WebDAV and SCP client for Microsoft Windows. Its main function is secure file transfer between a local and a remote computer. Beyond this, WinSCP offers basic file manager(or file browser is a computer program that provides a user interface to manage files and folders.) and file synchronization(process of ensuring that computer files in two or more locations are updated used for backups or mirroring) functionality. For secure transfers, it uses Secure Shell (SSH) and supports the SCP protocol in addition to SFTP.

WinSCP is for file transfer to and from your server while PuTTY is used to interact with the server directly. Putty is just a command line interface to your server. WinSCP is a file transfer application using Secure FTP.

Features:

- Microsoft Windows based Graphical User Interface (GUI)
- Translated into several languages
- Integration with Windows (Drag-and-drop, URL, shortcut icons)
- All common operations with files
- Support for SFTP and SCP protocols over SSH-1 and SSH-2, FTP protocol, WebDAV protocol and S3 protocol.[9]
- Batch file scripting, command-line interface and .NET wrapper
- Directory synchronization in several semi or fully automatic ways
- Integrated text editor
- Support for SSH password, keyboard-interactive, public key and Kerberos (GSS) authentication
- Integrates with Pageant (PuTTY authentication agent) for full support of public key authentication with SSH
- Choice of Windows Explorer–like or Norton Commander–like interfaces
- Optionally stores session information
- Optionally import session information from PuTTY sessions in the registry
- Able to upload files and retain associated original date/timestamps, unlike FTP clients

6.3 Electronic Mail : SMTP, POP3, IMAP



***** SMTP : The ***Simple Mail Transfer Protocol (SMTP)*** is used to transfer electronic mail from one user to another..

- ***Mail User Agent (MUA)*** : is an *application (e.g., Outlook Express, Thunderbird) that runs on a user's computer.* Mail user agents are used to compose and send messages, as well as to display and manage messages in a user's mailbox.
- ***Mail transfer agents (MTA)*** : are used to *pass emails between different mail servers.* When a mail user agent passes a message to a mail transfer agent, the latter passes the message to another transfer agent (or possibly many other transfer agents). Transfer agents are responsible for properly routing messages to the destination.
- ***Mail delivery agents (MDA) :*** are used to *place messages into a local user's mailbox.* When the message arrives at its destination, the final transfer agent gives the message to the appropriate delivery agent, and the latter delivers the message to the user's mailbox.

***** IMAP : **IMAP (Internet Message Access Protocol)** – Is a standard protocol for *accessing e-mail from your local server*. IMAP is a client/server protocol in which e-mail is received and held for you by your Internet server. As this requires only a small data transfer this works well even over a slow connection such as a modem. Only if you request to read a specific email message will it be downloaded from the server. You can also create and manipulate folders or mailboxes on the server, delete messages etc.

IMAP it keeps copy of the emails on server and it constantly synchronizes with email client likes emails all folder and sub folders.

IMAP service:

- IMAP is designed for the situation where you need *to work with your email from multiple computers,* such as your workstation at work, your desktop computer at home, or a laptop computer while traveling.
- *Messages are displayed on your local computer but are kept and stored on the mail* server -you can work with all your mail, old and new, from any computer connected to the internet.
- You can create subfolders on the mail server to organize the mail you want to keep. However, these subfolders, as well as its contents work against your total email quota of 1GB.

IMAP Workflow:

- Connect to server
- Fetch user requested content and cache it locally, e.g. list of new mail, message summaries, or content of explicitly selected emails
- Process user edits, e.g. marking email as read, deleting email etc.
- Disconnect

* POP : The **POP (Post Office Protocol)** protocol *provides a simple, standardized way for users to access mailboxes and download messages* to their computers. When using the POP protocol all your e-*mail messages will be downloaded from the mail server to your local computer*. The advantage is that once your messages are downloaded and read your emails at your leisure without suffering further communication costs even in no internet connection.

POP keeps emails on the client pc and it sync only inbox with server. POP works faster than IMAP.

POP service:

- POP was designed for, and works best in, the situation where you *use only a single desktop computer.*
- Normally, messages are *downloaded to your desktop computer and then deleted from the mail server.*
- If you choose to work with your POP mail on more than one machine, you may have trouble with email messages getting downloaded on one machine that you need to work with on another machine; for example, you may need a message at work that was downloaded to your machine at home.
- If you choose the POP option "keep mail on server", your POP "inbox" can grow large and unwieldy, and email operations can become inefficient and time-consuming.
- Your archive of mail, if you have one, is kept on your desktop computer - you generally need little storage space on the mail server.

POP Workflow:

- Connect to server
- Retrieve all mail
- Store locally as new mail
- Delete mail from server*
- Disconnect

6.4 DNS - Domain Name System

The first thing to understand about the Internet is that it doesn't understand English (despite everything AI enthusiasts tell you). That is why we have DNS servers. DNS servers translate plain English website names into machine readable IP-addresses. Once the website names are translated into IP addresses they can then be returned to the user browser. The user's browser then uses the IP address to query the website server and receive the content.

- The Domain Name System (DNS) works on Client Server model. It uses UDP protocol for transport layer communication. DNS uses hierarchical domain based naming scheme. The DNS server is configured with Fully Qualified Domain Names (FQDN) and email addresses mapped with their respective Internet Protocol (IP) addresses.
- A DNS server is requested with FQDN and it responds back with the IP address mapped with it. DNS uses UDP port 53.
- The domain name system (DNS) maps internet domain names to the internet protocol (IP) network addresses they represent and enables websites to use names, rather than difficult-to-remember IP addresses.
- Domain names give people a more easier way to access content or services than IP addresses.

Example:

When you type in a web address, e.g., www.hcoe.edu.np, your Internet Service Provider views the DNS associated with the domain name, translates it into a machine friendly IP address (i.e. 192.185.35.17) and directs your Internet connection to the correct website.

Types of DNS Domain Names

i. Root domain: A single period (.) or a period used at the end of a name, such as "**example.microsoft.com**."
ii. Top level domain: "**.com**", which indicates a name registered to a business for commercial use on the Internet.
iii. Second level domain: "**microsoft.com**. ", which is the second-level domain name registered to Microsoft by the Internet DNS domain name registrar.
iv. Subdomain: "**example**.microsoft.com. ", which is a fictitious subdomain assigned by Microsoft for use in documentation example names.
v. Host or resource name: "**media**.example.microsoft.com.", where the first label ("media") is the DNS host name for a specific computer on the network.

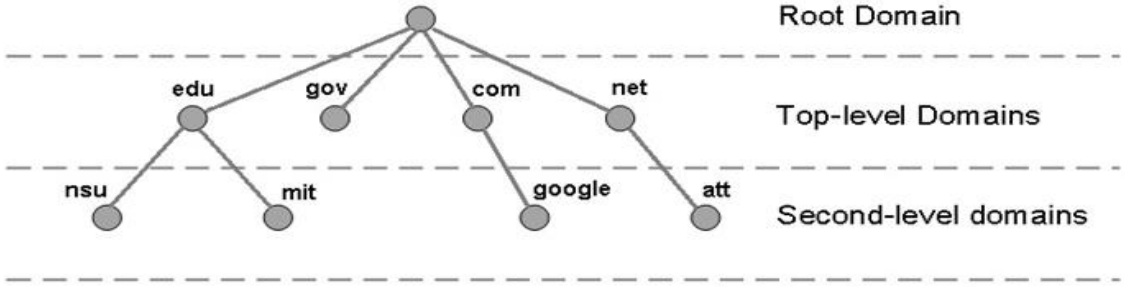
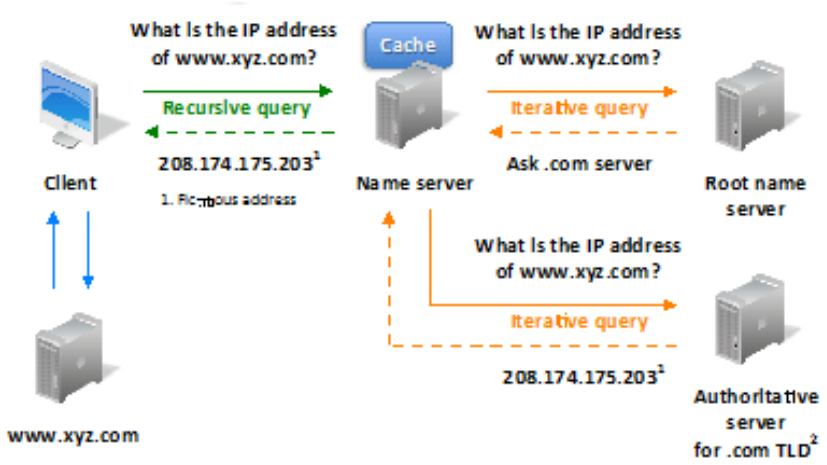


Fig. DNS Domain Name Hierarchy

How does DNS work?

When you visit a domain such as *dyn.com,* your computer follows a series of steps to turn the human-readable web address into a machine-readable IP address. This happens every time you use a domain name, whether you are viewing websites, sending email or listening to Internet radio stations like Pandora.



Step 1: Request information

The process begins when you ask your computer to resolve a hostname, such as visiting *http://xyz.com*. The first place your computer looks is its local DNS cache, which stores information that your computer has recently retrieved.

If your computer doesn't already know the answer, it needs to perform a **DNS query** to find out.

Step 2: Ask the recursive DNS servers or Name server

If the information is not stored locally, your computer queries (contacts) your ISP's **recursive DNS servers**. These specialized computers perform the legwork of a DNS query on your behalf. Recursive servers have their own caches, so the process usually ends here and the information is returned to the user.

Step 3: Ask the root nameservers

If the recursive servers don't have the answer, they query the **root nameservers**. A **nameserver** a computer that answers questions about domain names, such as IP addresses, they can direct our query to someone that knows where to find it.

The root nameservers will look at the first part of our request, reading from right to left — *www.dyn.**com*** — and direct our query to the Top-Level Domain (TLD) nameservers for *.com*. These servers don't have the information we need, but they can refer us directly to the servers that *do* have the information.

Step 4: Ask the authoritative DNS servers

The TLD nameservers review the next part of our request — *www.**xyz.com*** — and direct our query to the nameservers responsible for this *specific* domain. These authoritative nameservers are responsible for knowing all the information about a specific domain, which are stored in DNS records. There are many types of records, which each contain a different kind of information. In this example, we want to know the IP address for *www.xyz.com*, so we ask the authoritative nameserver for the Address Record (A).

Step 5: Retrieve the record

The recursive server retrieves the A record for *xyz.com* from the authoritative nameservers and stores the record in its local cache. If anyone else requests the host record for *dyn.com*, the recursive servers will already have the answer and will not need to go through the lookup process again. All records have a **time-to-live** value, which is like an expiration date. After a while, the recursive server will need to ask for a new copy of the record to make sure the information doesn't become out-of-date.

Step 7: Receive the answer

Armed with the answer, recursive server returns the A record back to your computer. Your computer stores the record in its cache, reads the IP address from the record, then passes this information to your browser. The browser then opens a connection to the webserver and receives the website.

6.5 P2P Applications

P2P Architecture

- All nodes are both clients and server: provide and consume data and any node can initiate connection
- No centralized data source

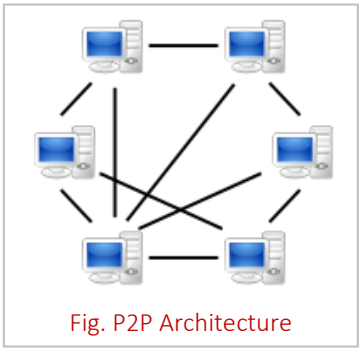


Fig. P2P Architecture

P2P is nothing but just Peer to Peer networking. As we have Server - Client Model and Peer to Peer network in the same way these P2P applications work. You need a P2P program that will be installed on your computer it creates a community of P2P application users and it creates a virtual network between these users. For the user it will look as it is in a Peer to Peer network and he can share files from his local computer and download files shared by other users. It is very similar to our Instant Messaging like Yahoo, AOL or GTalk where even though to whom we are taking to are on a different network but a virtual network is created where it looks we are on a same network and we can share files and chat. The P2P application has been very much in demand from last couple of years. A P2P application is mainly used for sharing Music, Movies, Games and other files.

Examples: **BitTorrent**: "swarm" (a group of computers downloading and uploading the same torrent) transfer data between each other without the need for a central server. Once connected, a BitTorrent client downloads bits of the files in the torrent in small pieces, downloading all the data it can get. Once the BitTorrent client has some data, it can then begin to upload that data to other BitTorrent clients in the swarm. In this way, everyone downloading a torrent is also uploading the same torrent. This speeds up everyone's download speed. If 10,000 people are downloading the same file, it doesn't put a lot of stress on a central server. Instead, each downloader contributes upload bandwidth to other downloaders, ensuring the torrent stays fast.

6.6 Socket Programming

A network socket is a presented as the endpoint of an inter-process communication flow across a computer network. Today, most communication systems carried out between computers is based on the Internet Protocols. So, nowadays most network sockets that we use are Internet sockets. A socket address possesses the combination between the IP address and a port number, much like one of the ends of a telephone connection has the combination between a phone number and a particular extension in a telephone system. Based on this address, the internet sockets used for communication delivers incoming data packets of information to the appropriate application process ends or the threads which help to establish the better communication between the components of the system.

Steps that occurs when establishing connection using sockets between two computers:

- The server instantiates a **ServerSocket** object, which represents the port no denoting which specific communication to occur on.
- The server invokes the **accept**() method of the **ServerSocket** class. This method waits for a client until it connects to the server on the given port.
- After the server is waiting, a client instantiates a **Socket** object, specifying which server name and the port number to connect to.
- The specified server and the specific port number is used to connect the client by the constructor of the socket class. If communication is successfully established then the connected client possesses the Socket object capable of communicating with the server.
- In the server side, the **accept** () method returns a reference to a new socket on the server that is connected to the client's socket.

Primitive	Meaning	Telephone Dial Analogue
SOCKET	Create a new communication end point	end point for communication
BIND	Attach a local address to a socket	Assign a unique telephone number
LISTEN	Announce willingness to accept connections: give queue size	Wait for caller
ACCEPT	Block the caller until a connection attempt arrives	Dial a number
CONNECT	Actively attempt to establish a connection	Receive a call
SEND	Send some data over the connection	Talk or Message exchange
RECEIVE	Receive some data from the connection	
CLOSE	Release the connection	Hang up

There are two different types of sockets:

1. **TCP(Stream) Socket (connection-oriented):** uses TCP; telnet, ssh, HTTP...
2. **UDP(Datagram) Socket (connection-less):** uses UDP; audio/video streaming

TCP (Stream) Sockets

It is a connection Oriented Sockets used in TCP. The stream socket may serve several clients concurrently, by creating a child process for each client which helps to establish a TCP connection between the child process and the client. But those unique dedicated sockets are needed to be created for each connection when it is essential. Each socket is identified by both the local and remote IP address and port as it needs to establish the connection.

server => socket(), bind(), **listen(), accept(),** recv(), send()
client => socket(), **connect(),** send(), recv()

UDP (Datagram) Sockets

Datagram Sockets are connectionless sockets used in UDP. Here, no child process is created for multiple clients. Single Socket serves all the clients. The socket is identified by local port no. and IP address.

server => socket(), bind(), recvfrom(), sendto()
client => socket(), sendto(), recvfrom()

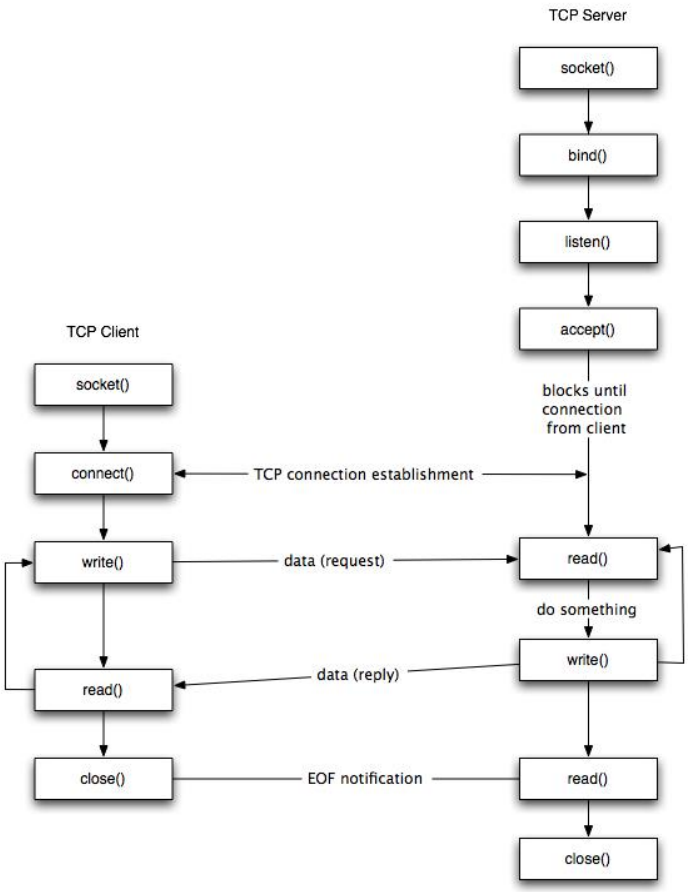


Fig: TCP(Stream) Sockets

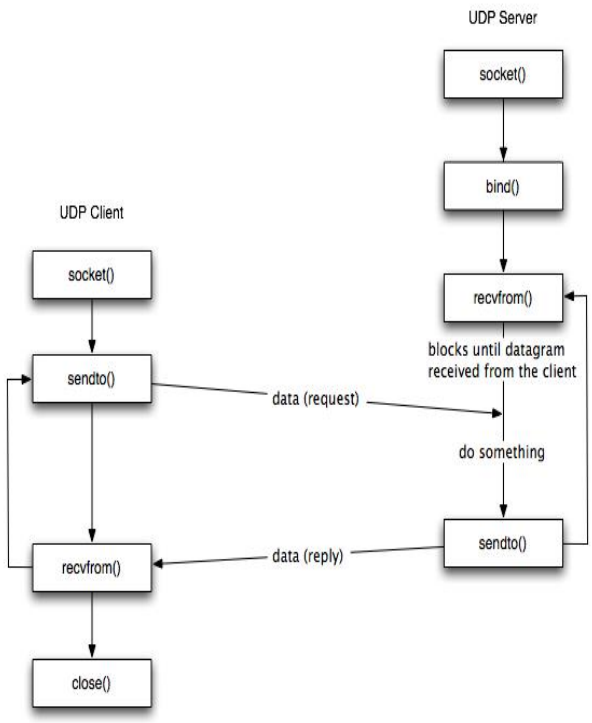


Fig: Datagram Sockets

6.7 Application server concepts : *Proxy caching, Web/Mail/DNS server optimization*

Terminology:

- **Origin Server:-** It represents the original source of an object
- **Proxy Server:-** It supplies the object instead of origin server

Some servers (caches) are demand-driven which acts as both:

- **Server:-** responding to client's requests
- **Client:-** forwarding requests that it cannot respond to towards the origin server.

Proxy caching allows a server to act as an intermediary between a user and a provider of web content. When a user accesses a website, proxies interpret and respond to requests on behalf of the original server.

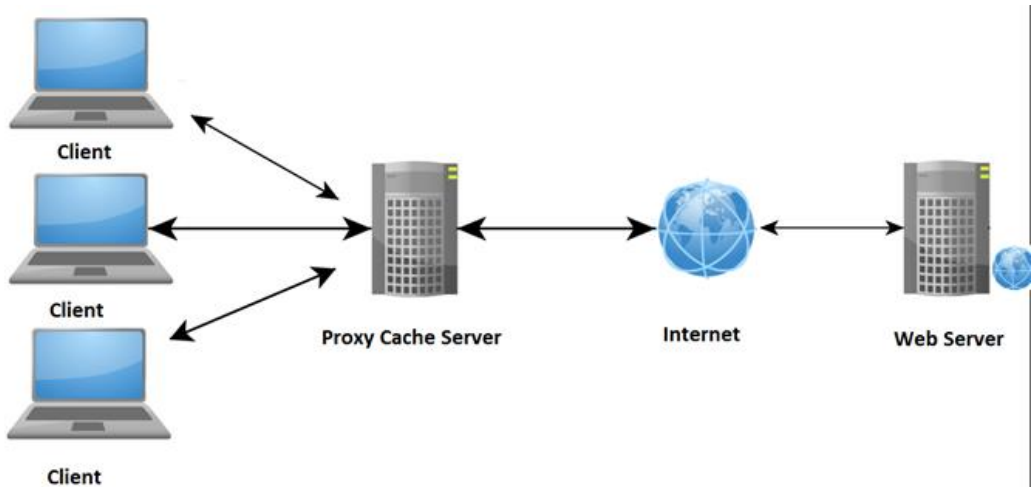


Fig Cache Server

For much of the modern web, requests sent to a web service are handled by an intermediate server. This intermediate – or proxy – server processes requests to determine how they should be handled. Proxy servers allow enterprises to structure and safeguard their networks while transparently providing additional features to users.

Proxy caching is a feature of proxy servers that stores content on the proxy server itself, allowing web services to share those resources to more users. The proxy server coordinates with the source server to cache documents such as files, images and web pages.

Caching Principle

In caching principle, fast resources are scarce e.g. storage close to clients (low propagation delay & fast links) It is aimed to locate commonly used objects in fast resource, other objects can remain in slower resources. After determining which objects are commonly accessed then common Accesses are often correlated, and are said to exhibit "locality".

- **Temporal Locality:-** Information accessed at one time is likely to be accessed again in near upcoming future.
- **Spatial Locality:-** Information accessed at one point is likely to be accessed also by nearby points.

HOW PROXY CACHING WORKS ?

Proxies act as a gateway between the user and the source server, storing (or caching) the server's resources. When the user attempts to access a resource, the proxy checks to see if it has a recent copy of the resource on hand. If so, the resource is immediately delivered to the user. If not, the resource is retrieved from the source and simultaneously cached to the proxy and delivered to the user. Step-by-step, here's how proxy caching works

- A user opens a webpage containing static content such as media, JavaScript or CSS.
- The user's browser automatically connects to a proxy server. This could be a content delivery network (CDN) which caches content in various locations around the world.
- The browser requests resources from the proxy server. The proxy server checks to see if it not only has the resource, but if the resource is recent. If the resource is old or missing, the proxy fetches a new copy from the source.
- The proxy delivers the resource to the browser. If the proxy had to fetch a new copy, it caches the copy for future use. Once a proxy refreshes a resource, it resets the resource's "expiration date" to prevent it from being reflagged as out-of-date. The resource will be delivered as-is until a certain time passes, at which point the proxy server will re-check the source.

EXAMPLE OF PROXY CACHING

CDNs like MaxCDN perform proxy caching on a massive scale. When a user accesses a static website hosted by MaxCDN, content is provided directly from MaxCDN's servers instead of the original web server. The user only connects to the original server when dynamic content is required or when MaxCDN's copy is outdated.

Proxy caching allows enterprises to quickly and easily manage their websites. CDNs perform real-time analytics on data, maintain copies of cached content across the globe, and can keep a website up and running even when the source server goes offline.

Main Uses of proxy server:

- Caching: When a user accesses a web page, that page is temporarily stored in the proxy cache. Then, when a subsequent user requests for the same web page which is stored in the proxy cache they access the copy in the proxy cache, rather than having the web page sent again from the originating server. It improves performance and frees up Internet bandwidth for other tasks.
- Filtering: Allows to block specific sites
- Maintain Privacy: Can hide actual IP address of Client from the outside world

BENEFITS OF PROXY CACHING

Proxy caching makes it easier for enterprises to manage their web services and deliver content to their customers.

- ➢ Users enjoy a faster, more reliable experience by retrieving files that are already cached.
- ➢ Enterprises see higher customer satisfaction since proxies allow content to be delivered to customers faster.
- ➢ Enterprises also see lower administration costs since infrastructure can be outsourced to third parties such as CDNs. Send request to origin server (reducing delay, and link use).
  - Send full response from origin server (reducing link use)
  - Directly benefits end-user.
  - Caching benefits the service providers by making their service more popular.
  - Reduced traffic
  - Reduces load on network links
  - Reduces load on server: e.g. "flash crowds" reduction
  - Mask unavailability of origin server: e.g. when working offline, or during faults
  - Filter Requests
  - Security schemes
  - Logging

* Web Server optimization is a vital part of web development and maintenance but also something often overlooked by webmasters. Just think of the money you can save, and how it can potentially help increase your readership and traffic when they are properly done.

If you have not done any optimization to your website (or blog) so far or merely curious how it can help speed up your site, please take a look at this list of optimization tips we've put together.

We've split things up into 3 separate sections for better readability – respectively ***server-side optimization***, ***assets optimization*** (which includes web components like CSS, Javascript, images, etc) and ***platform***, where we'll focus on *WordPress optimization*.

Server side optimization

- *Choose a decent Web Host*

Your web hosting account has no direct relationship with the optimizations you are about to perform but we figured choosing the right web hosting account so important we decided to bring it to your attention first. Hosting account is the foundation of your website/blog where it's security, accessibility (cPanel, FTP, SSH), server stability, prices and customer supports all play important roles. You need to make sure you are in good hands.

Recommended reading: How to Choose a Web Host by wikiHow is a great article that gives you steps and tips you should know before purchasing any web hosting account.

- *Host Assets Separately*

When we mention assets, we meant web components like images and static scripts that don't require server-side processing. These includes any web graphics, images, Javascripts, Cascading Style Sheets (CSS), etc. Hosting assets separately is not a must, but we've seen tremendous result in terms of server stability with this implementation when the blog was having a traffic spike.

Recommended reading: Maximizing Parallel Downloads in the Carpool Lane.

- *Compression with GZip*

In short, contents travel from server side to client side (vicet versa) whenever a HTTP request is make. Compressing the content for sending greatly reduce the time needed to process each request.

GZip is one of the best ways to do this and it various according to the type of servers you are using. For instance, Apache 1.3 uses mod_zip, Apache 2.x uses mod_deflate and here's how you do it in Nginx. Here are some really good articles to get you familiar with server side compressions:

Speed up a web site by enabling Apache file compression
Compress Web Output Using mod_gzip and Apache
How To Optimize Your Site With GZIP Compression
Server-side compression for ASP

- *Minimize Redirects*

Webmasters do URL redirect (whether it's Javascript or META redirects) all the time. Sometimes it's purpose is to point users from an old page to new, or merely guide users to the correct page. Each redirect create an additional HTTP request and RTT (round-trip-time). The more redirection you have, the slower user will get to the destination page.

Recommended reading: Avoid Redirects by Google Code gives you a good overview on the matter. The article also recommends some practical ways to minimize redirection to increase serving speed.

- *Reduce DNS Lookups*

According to Yahoo! Developer Network Blog, it takes about 20-120 milliseconds for DNS (Domain Name System) to resolve IP address for a given hostname or domain name and the browser cannot do anything until the process is properly completed.

Author *Steve Souders* suggested that splitting these components across at least two but no more than four hostnames reduces DNS lookups and allow high degree parallel downloads. Read more on the article.

The assets Optimization : CSS, JavaScript, Images

Merge Multiple Javascripts into One

Compress Javascript & CSS

There are also some web services that allow you to manually compress your Javascripts and CSS files online. Here are few we come to know:

- compressor.ebiene (*Javascript*, *CSS*)
- javascriptcompressor.com (*Javascript*)
- jscompress.com (*Javascript*)
- CleanCSS (*CSS*)
- CSS Optimizer (*CSS*)

- *Customize Header Expiry/Caching*

By using a customized Expiry header, your web components like images, static files, CSS, Javascript skipped unnecessary HTTP request when the same user reload the page for the second time. It reduces the bandwidth needed and definetely help in serving the page faster.

- *Off-load The Assets*

By off-loading, we mean separating the Javascripts, images, CSS and static files from main server where the website is hosted and place them on another server or rely on other web services. We've seen significant improvement here in Hongkiat by off-loading assets to other web services available, leaving the server to mainly do the PHP processing. Here are some suggestions of online services for off-loading:

- Images: Flickr, Smugmug, Paid hostings*
- Javascripts: Google Ajax Library, Google App Engine, Paid hostings*
- Web Forms: WuFoo, FormStack
- RSS: Google Feedburner
- Survey and Polls: SurveyMonkey, PollDaddy

- *Handling Web Images*

Images are important part of your website. However if they are not properly optimize, they can become a burden and end up utilizing unpredictably large amount of bandwidths on daily basis. Here are some best practices to optimize your images:

- Optimize PNG ImagesFolks at Smashing Magazine describes some clever techniques that may help you optimize your PNG-images.
- Optimizing for Web – Things you need to know (the formats)Learn more about Jpeg, GIF, PNG and how you should save your images for web.
- Don't Scale Images Always practice inserting the width and height for each images. Also don't scale down an image just because you need a smaller version on the web. For example: Do not force scale a 200×200 px image to 50×50 px for your website by altering the width and height. Get a 50×50 px instead.

Optimizing with Web Services and Tools. Good news is, you don't need to be a Photoshop expert to optimize your images. There are plenty of web services and tools to help you do the job.

Handling CSS

Modern websites uses CSS as the foundation of the style, as well as the look and feel. Not only CSS gives great flexibility to changes, it is also lesser in terms of codes needed. However, if they are badly coded, it could be a backfire. Here are some checklists, or rather guides to you make sure your CSS are properly optimized:

- Keeping Your Elements' Kids in Line with Offspring How to keep your markup clean with CSS Selectors.
- Keep CSS short When they give the same style, the codes are better the shorter they are. Here's a CSS Shorthand guide you'll probably need.
- CSS Sprite technique reduce HTTP request each time a page is load by combining several (or all) images together on a single image file and control it's output with CSS background-position attribute. Here are some useful guides and techniques to create CSS Sprites:
  - Creating easy and useful CSS Sprites
  - Online CSS Sprite Generator
  - Best Online And Offline CSS Sprites Generator
- Using every declaration just once When looking to optimize your CSS files, one of the most powerful measures you can employ is to use every declaration just once.
- Reduce amount of CSS files The reason is simple, the more CSS files you have the more HTTP request it'll have to make when a webpage is being requested. For example, instead of having multiple CSS files like the following:

```
1<link rel="stylesheet" type="text/css" href="main.css" />
2<link rel="stylesheet" type="text/css" href="body.css" />
3<link rel="stylesheet" type="text/css" href="footer.css" />
```

- You can combine them into the one single CSS:

```
1<link rel="stylesheet" type="text/css" href="layout.css" />
```

*** Mail Server Optimization**

**How we went from 2 000 000 to 200 000 daily SMTP connections overnight*

If you are running a huge pile of email servers, like we do, you deal with a lot of SPAM messages on a daily basis. In which case – god speed, but this article is not about the way you find, flag, and prevent SPAM. It is about its impact on the computational resources practically wasted by the servers for processing messages that afterwards ended up in the junk folder. Turns out, you can effectively find a way to reclaim all that computational power and use it for better purposes. Even if you don't deal with huge volumes of SPAM messages in particular, our quest could surely shed some light on some simple server optimizations that can help you use your resources much more efficiently.

**Measuring SPAM in terms of server resources*

aims to be a general and flexible mailer with extensive facilities for checking incoming email.

**Asking the right questions*

We started working on the problem by breaking it down to several simple questions:

- Can we stop SMTP connections known to be associated with previous SPAM attempts from happening entirely?
- How we can do that at the first possible sign and save most of the steps associated with the connection handling overhead?
- What system resources we will be able to save if we manage to achieve that?
- How we can do all of the above lightening-fast without wasting more resources than we were actually going to save?

**A reliable way to identify IPs associated with SPAM messages*

If you pass certain known-to-be-SPAM-mail-message via SPAM assassin, dSPAM or similar solutions and this message is known to be SPAM you have two types of classifications. Low-to-medium probability of SPAM and screamingly high probability of SPAM.

For the purpose of our system we started to flag message/IP pairs where messages were identified only with high probability of SPAM. We used SPAMAssassin with a combination of custom EXIM ACLs we built over time that were 99.9% accurate to identify such high-probability-of-SPAM messages. With EXIM we also started to flag message/IP pairs where IPs were known to deliver good messages or such with low probability of SPAM.

**EXIM readsocket - a fast and easy method for communication between the MTA and the rest of the world*

While looking at the EXIM documentation, we discovered EXIM readsocket string expansion. For us it was one of the super-handy features provided by the MTA that allowed us to report to the external processor what is actually happening inside the MTA during ACL processing. Any EXIM variable that can be used inside the ACLs can be reported to an external program. Awesome, right?

**Building the stats collector*

We decided to create a simple daemon that listens on unix domain socket. It accepts data on that socket and uses it to sample statistics about IP addresses and whether they are delivering SPAM or SPAM-free mail messages.

**Blocking bad with ipset - low overhead and close-to-zero maintenance costs*

If you have been working in the network world enough and managing a lot of systems, sooner or later you notice that having 10k iptables rules in your firewall is not something you really want. At least most of the time. I suppose that other people had the same problem and that's how Ipset was born.

**Expanding the logic of the blocker inspiration*

Having different ipset lists with different expiration times allowed us to do some more stuff.

If we add an IP object to a list with expiration 1 hour that particular IP address will be unable to connect to the SMTP service for 1 hour. When it is removed from the list it will be able to re-connect to the SMTP server again.

However if the very same IP continues to attempt to deliver mail messages mainly flagged as SPAM it is then added to a set block list with expiration 2 hours. The third time, the expiration period is increased to 4 hours, the fourth to 8 hours and so on. The more SPAM you send, the longer the block gets. Finally, we started adding "bad" IPs to 24-hour block lists directly.

* DNS Optimization

DNS or domain name system is the way the internet makes sense of website names and connects users to the correct web server. The time it takes for DNS queries to be sent, resolved and received has a huge impact on the speed of a website or application.

DNS also plays its part in failover and load balancing. DNS failover allows DNS queries to be re-routed among a set of servers, in the event of a server failure. DNS load balancing architectures distribute traffic over multiple servers.

Optimizing DNS queries for the quickest response times, high availability and failover is essential to ensuring a great end user experience.

Why DNS Optimization matters?

The average internet connection speed in the United States has more than doubled in the past five years. Akamai's "State of the Internet" site reports that speeds increased from 4,163 kpbs to 8,675 kbps between 2008 and 2013 (Q2 data).

Web app users appreciate the difference: a faster connection means faster apps.

- **Faster ISP:** One way to upgrade your connection speed is to call to your internet service provider — or a competitor. In most cities, if you pay a bit more, you can get a faster connection. This may be less feasible in rural areas.
- **Faster network hardware:** New network hardware also may provide faster performance. A high-speed router and firewall are essential, as are client systems (and devices) that support high-bandwidth connections.
- **Faster DNS:** You might get an internet performance boost by changing one setting: the Domain Name Server, or DNS, address. For less-technical folks, think of the DNS as a web-site search service: every time you request a website, your DNS provider translates your request into numbers and routes your computer to the requested site's servers.

When you change the DNS setting on your router (or device), you change where your router "looks up" servers on the internet (i.e., DNS translates "www.techrepublic.com" into numerical server addresses). It doesn't change the bandwidth of your connection. Instead, your router sends requests to a faster DNS server, which means you spend less time waiting for a DNS server to respond. The decreased wait feels like increased speed.

Small businesses or home users typically use the default DNS server of their internet service provider, but other DNS providers are often faster.

DNS matters because it takes time for DNS queries to travel across the network to the DNS resolver and back again. The time required translates into load times for websites and applications.

Therefore, optimizing DNS queries for faster response time is essential to ensuring a fast browsing experience. Following are some tips to ensure that your DNS infrastructure is optimized for blazing fast response times.

- Time to live (TTL)

TTL is the time that you allow the IP address to be cached. TTL presents a catch 22 situation. Setting low TTLs can lead to slower performance because the client has to go up to the resolver or the authoritative domain to get the IP. Higher TTLs side step this issue but create different problems. Longer TTLs can lead to situations where a server goes down and is no longer available but since the IP is cached, the client goes to the server that is down.

Setting TTLs is a compromise between having faster DNS resolving and aiming for maximum availability.

- Anycast DNS

Another way of optimizing DNS server performance is to use anycast. Anycast allows multiple geographically distributed DNS nameservers to advertise the same IP address. DNS queries can then be served by any one of a number of anycasted DNS servers. This lowers DNS response times and in turn improves performance.

- DNS Load Balancing

Anycast DNS can also be used for DNS load balancing. Popular websites that get hundreds of thousands of hits every hour usually distribute the load over many DNS servers. This ensures that no one server has too much load at one time.

- DNS failover

Anycast can also be used for DNS failover. It allows automatic and predictable switching between servers, in the event of failures. If a server goes down, DNS queries can be re-routed to redundant servers without any disruption. Datapath.io's anycast DNS architecture reduces failover times to less than 10 seconds and ensures that sessions are not lost.

6.8 Concept of traffic analyser: *MRTG, PRTG, SNMP, Packet Tracer, Wireshark*

Monitoring...

- avoids performance bottlenecks and breakdowns
- delivers better quality of service proactively
- reduces costs buy only what is needed
- increases company profits by minimizing downtime
- provides peace of mind: No notifications mean that everything is running fine

Basic tasks that fall under this category are:

- Configuration Management: Keeping track of device settings and how they function
- Fault Management: Dealing with problems and emergencies in the network (router stops routing, server loses power, etc.)
- Performance Management: How smoothly is the network running? Can it handle the workload it currently has?
- Security Management – Hardware and Network security in networking

* Simple Network Management Protocol (SNMP)

SNMP is a tool (protocol) that allows for remote and local management of items on the network including servers, workstations, routers, switches and other managed devices.

SNMP is an Internet-standard protocol for collecting and organizing information about managed devices on IP networks and for modifying that information to change device behavior. Devices that typically support SNMP include routers, switches, servers, workstations, printers, modem racks and more.

SNMP is widely used in network management for network monitoring. SNMP exposes management data in the form of variables on the managed systems organized in a management information base which describe the system status and configuration. These variables can then be remotely queried (and, in some circumstances, manipulated) by managing applications.

Basic commands or Operations of SNMP

The simplicity in information exchange has made the SNMP as widely accepted protocol. The main reason being concise set of commands, here are they listed below:

- GET: The GET operation is a request sent by the manager to the managed device. It is performed to retrieve one or more values from the managed device.
- GET NEXT: This operation is similar to the GET. The significant difference is that the GET NEXT operation retrieves the value of the next OID in the MIB(Management Information Base- *hierarchical order of all managed objects and how they are accessed*) tree.
- GET BULK: The GETBULK operation is used to retrieve voluminous data from large MIB table.
- SET: This operation is used by the managers to modify or assign the value of the Managed device.
- TRAPS: Unlike the above commands which are initiated from the SNMP Manager, TRAPS are initiated by the Agents. It is a signal to the SNMP Manager-(*process running on a management workstation that requests information about devices on the network.)* by the Agent on the occurrence of an event.
- INFORM: This command is similar to the TRAP initiated by the Agent, additionally INFORM includes confirmation from the SNMP manager on receiving the message.
- RESPONSE: It is the command used to carry back the value(s) or signal of actions directed by the SNMP Manager.

For more SNMP detail visit : https://www.manageengine.com/network-monitoring/what-is-snmp.html

*** Multi Router Traffic Grapher (MRTG)** is a tool to monitor the traffic load on network links. MRTG generates HTML pages containing PNG images which provide a LIVE visual representation of this traffic.

- Free monitoring tool that shows graphs in web pages
  - Optimized for traffic monitoring, but can be used to monitor other parameters (e.g., CPU use, disk use,…)
  - Graphs are updated periodically and web page reloaded
- Obtains data from SNMP agents using SNMP – It's a SNMP manager

MRTG consists of a Perl script which uses SNMP to read the traffic counters of your routers and a fast C program which logs the traffic data and creates beautiful graphs representing the traffic on the monitored network connection. These graphs are embedded into webpages which can be viewed from any modern Web-browser.

In addition to a detailed daily view, MRTG also creates visual representations of the traffic seen during the last seven days, the last five weeks and the last twelve months. This is possible because MRTG keeps a log of all the data it has pulled from the router. This log is automatically consolidated so that it does not grow over time, but still contains all the relevant data for all the traffic seen over the last two years. This is all performed in an efficient manner. Therefore, you can monitor 200 or more network links from any halfway decent UNIX box.

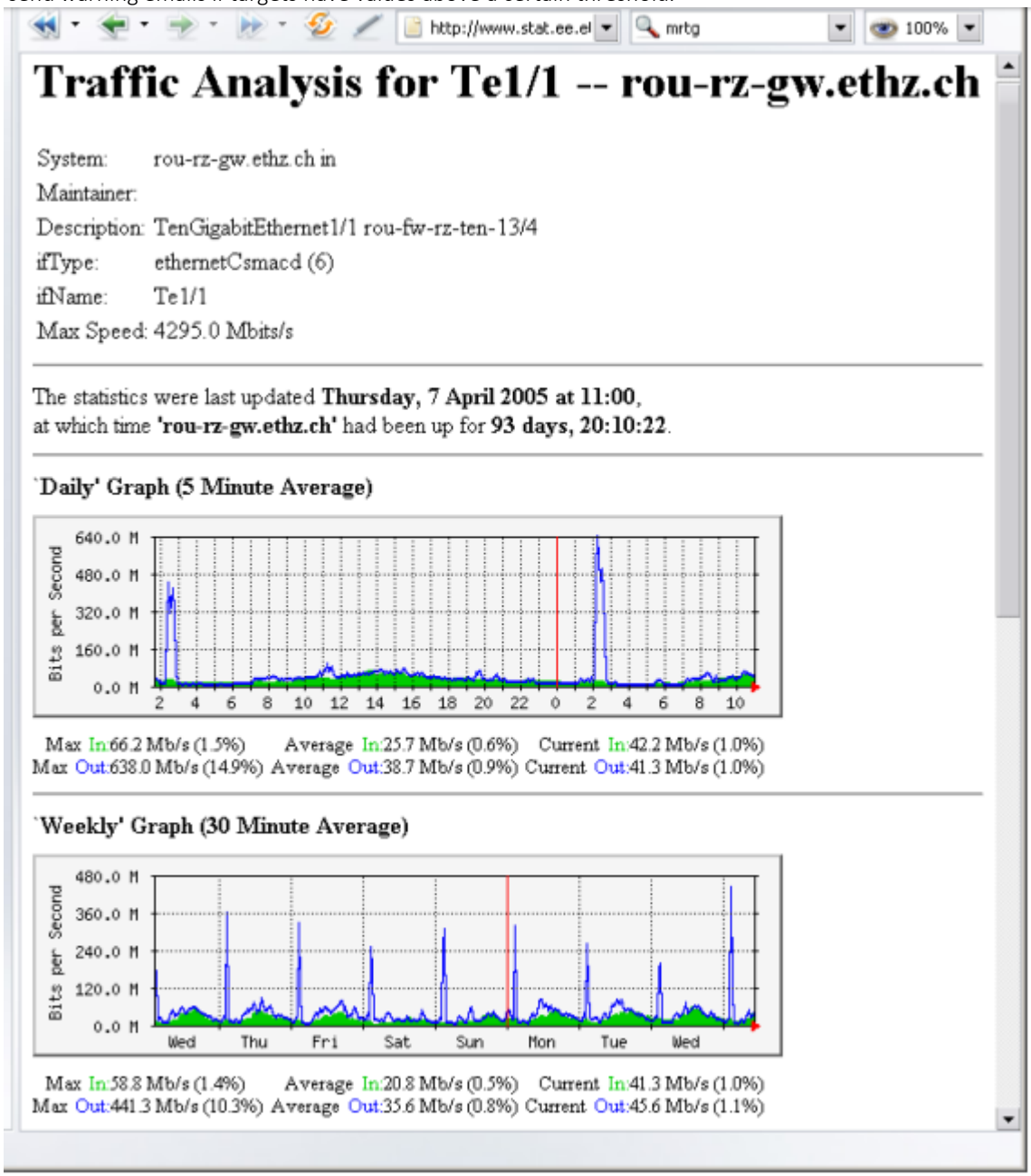
MRTG is not limited to monitoring traffic, though. It is possible to monitor any SNMP variable you choose. You can even use an external program to gather the data which should be monitored via MRTG. People are using MRTG, to monitor things such as System Load, Login Sessions, Modem availability and more. MRTG even allows you to accumulate two or more data sources into a single graph.

- MRTG works on most UNIX platforms and Windows NT
- MRTG is written in Perl and comes with full source.
- MRTG Uses a highly portable SNMP implementation written entirely in Perl (thanks to Simon Leinen). There is no need to install any external SNMP package.
- MRTG can read the new SNMPv2c 64bit counters. No more counter wrapping.

- Router interfaces can be identified by IP address, description and ethernet address in addition to the normal interface number.
- MRTG's logfiles do NOT grow thanks to the use of a unique data consolidation algorithm.
- MRTG comes with a set of configuration tools which make configuration and setup very simple.

Features:

- It measures two values (I for Input, O for Output) per target.
- It gets its data via an SNMP agent, or through the output of a command line.
- It typically collects data every five minutes (it can be configured to collect data less frequently).
- It creates an HTML page that features four graphs (GIF or PNG images).
- It results are plotted vs time into day, week, month and year graphs.
- It automatically scales the Y-axis of the graphs to show the most detail.
- It adds calculated Max, Average and Current values for both I and O to the target's HTML page.
- It can also send warning emails if targets have values above a certain threshold.

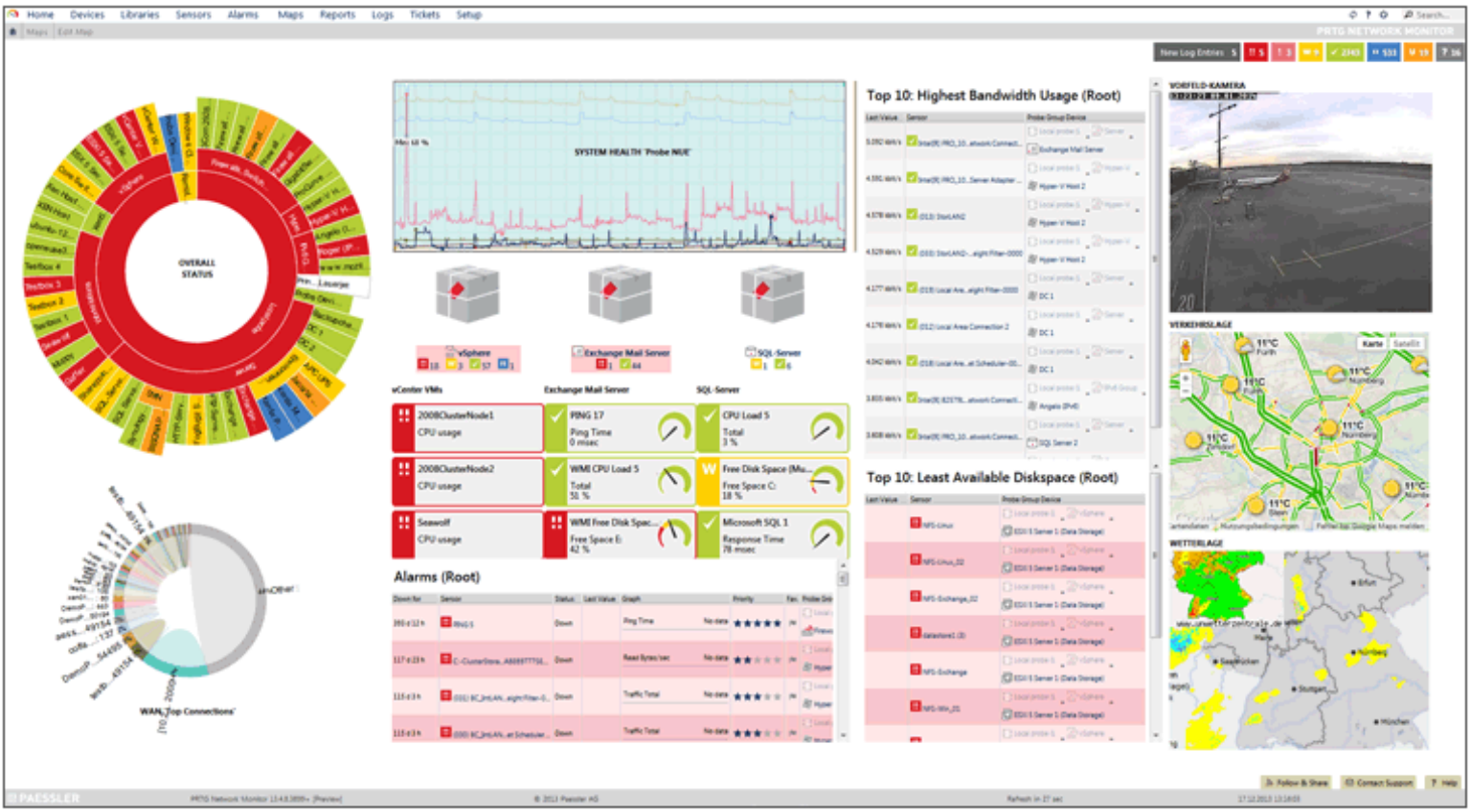


* PRTG (Paessler Router Traffic Grapher), renamed PRTG Network Monitor from version 7 in 2008, is a server up-time and utilisation, network monitoring and bandwidth usage software package for server infrastructure from Paessler AG. It can monitor and classify bandwidth usage in a network using SNMP, packet sniffing and Netflow. It services Microsoft Windows and Linux. It was derived from the open-source Multi Router Traffic Grapher (MRTG) project. A version with a limited number of sensors is available free of charge.

PRTG...

- Best price-performance-ratio
- Easy to install and use therefore easy to sell
- includes all 'add-ons' and features from the go, no hidden costs.

- Distributed monitoring, high availability failover cluster included
- 220+ pre-built sensors, everything is covered.
- Fastest database optimized for monitoring data; no SQL-Server needed
- Powerful customizable dashboards, alerts and 1 year reporting
- Web Interface, Enterprise Console
- iPhone app, android, mobile interface and SMS alerts for free.



A network Analyzer is used for

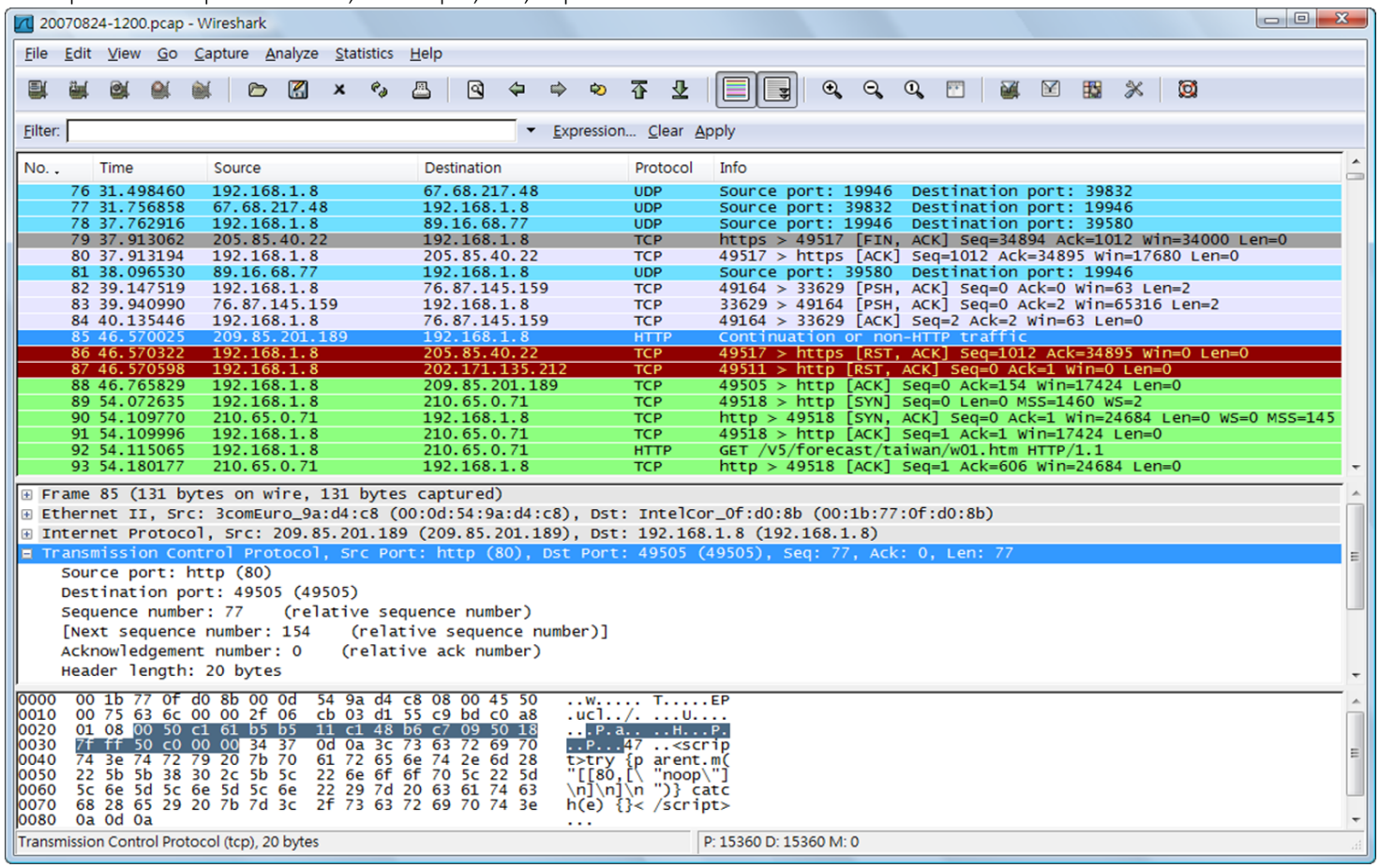
- Troubleshooting problems on the network
- Analysing the performance of a network to discover bottlenecks
- Network intrusion detection
- Analysing the operations of applications

- Wireshark is the world's foremost and widely-used network protocol analyzer. It lets you see what's happening on your network at a microscopic level and is the de facto (and often de jure) standard across many commercial and non-profit enterprises, government agencies, and educational institutions. Wireshark development thrives thanks to the volunteer contributions of networking experts around the globe and is the continuation of a project started by Gerald Combs in 1998.

Wireshark has a rich feature set which includes the following:

- Deep inspection of hundreds of protocols, with more being added all the time
- Live capture and offline analysis
- Standard three-pane packet browser
- Multi-platform: Runs on Windows, Linux, macOS, Solaris, FreeBSD, NetBSD, and many others
- Captured network data can be browsed via a GUI, or via the TTY-mode TShark utility
- The most powerful display filters in the industry
- Rich VoIP analysis
- Read/write many different capture file formats: tcpdump (libpcap), Pcap NG, Catapult DCT2000, Cisco Secure IDS iplog, Microsoft Network Monitor, Network General Sniffer® (compressed and uncompressed), Sniffer® Pro, and NetXray®, Network Instruments Observer, NetScreen snoop, Novell LANalyzer, RADCOM WAN/LAN Analyzer, Shomiti/Finisar Surveyor, Tektronix K12xx, Visual Networks Visual UpTime, WildPackets EtherPeek/TokenPeek/AiroPeek, and many others
- Capture files compressed with gzip can be decompressed on the fly
- Live data can be read from Ethernet, IEEE 802.11, PPP/HDLC, ATM, Bluetooth, USB, Token Ring, Frame Relay, FDDI, and others (depending on your platform)
- Decryption support for many protocols, including IPsec, ISAKMP, Kerberos, SNMPv3, SSL/TLS, WEP, and WPA/WPA2
- Coloring rules can be applied to the packet list for quick, intuitive analysis

- Output can be exported to XML, PostScript®, CSV, or plain text



- Packet Tracer's traffic simulation tools are similar to Wireshark in the sense that you can click on a PDU (in this case, an envelope) and look at the bytes in the message as well as the decoded meaning of the message at the different layers of the stack, but that's really as far as the similarities go.
  - Packet Tracer is not only not a real network, but it's not a virtualized network either, at least not in the same sense as something like GNS3 (which can run real Cisco IOS and create real packets, even those leaving a physical network card). Packet Tracer is limited to its own sandbox and exists solely for training purposes, whereas Wireshark has a greater scope. Wireshark can look at "real" packets from actual networks, both from a network card directly or saved/distributed in a standardized packet capture file format. In short, Wireshark's scope extends to the real world, and the real network administration workforce, whereas Packet Tracer is a classroom training tool.
  - Cisco Packet Tracer is a powerful network simulation program that allows students to experiment with network behavior and ask “what if” questions. As an integral part of the Networking Academy comprehensive learning experience, Packet Tracer provides simulation, visualization, authoring, assessment, and collaboration capabilities and facilitates the teaching and learning of complex technology concepts.
  - Packet Tracer supplements physical equipment in the classroom by allowing students to create a network with an almost unlimited number of devices, encouraging practice, discovery, and troubleshooting. The simulation-based learning environment helps students develop 21st century skills such as decision making, creative and critical thinking, and problem solving. Packet Tracer complements the Networking Academy curricula, allowing instructors to easily teach and demonstrate complex technical concepts and networking systems design.

  - The Packet Tracer software is available free of charge ONLY to Networking Academy instructors, students, alumni, and administrators that are registered Academy Connection users.