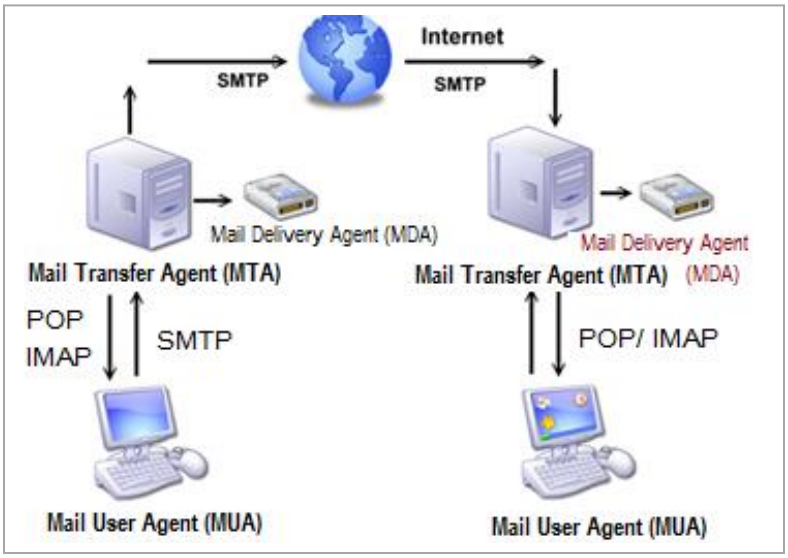
Internet Protocol (IP)

- Internet Protocol (IP) is *the principal set (or communications protocol) of digital message formats and rules for exchanging messages between computers* across a single network or a series of interconnected networks, using the Internet Protocol Suite (often referred to as TCP/IP). *Messages are exchanged as datagrams, also known as data packets or just packets.*
- IP is the primary protocol in the Internet Layer of the Internet Protocol Suite, which is a set of communications protocols consisting of four abstraction layers: link layer (lowest), Internet layer, transport layer and application layer (highest).
- *The main purpose and task of IP is the delivery of datagrams from the source host (source computer) to the destination host (receiving computer) based on their addresses.* To achieve this, IP includes methods and structures for putting tags (address information, which is part of metadata) within datagrams. The process of putting these tags on datagrams is called encapsulation.

3.1. Standard protocols: SMTP, E-mail Message (RFC22),PGP, POP, IMAP, HTTP, FTP



***SMTP** : SMTP is part of the application layer of the TCP/IP protocol. Using a process called *"store and forward," SMTP moves your email on and across networks*. It works closely with the Mail Transfer Agent (MTA) to send your communication to the right computer and email inbox.

-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.

*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 you can cut the internet connection and read your emails at your leisure without suffering further communication costs.

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

***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.
You can access your mail from multiple mail clients and each client detects the change in real-time. Suppose mail server is connected with two different mail clients (let's say Client 1 and Client 2) on different computers. If the user deletes a message in mail client 1, the change will appear on mail server immediately and also on mail client 2. In IMAP all messages from mail clients(Outlook) and servers are synced with each other.

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

Comparison between IMAP and POP

Features	IMAP	POP
Email Store	Emails are stored on the server.	Emails are stored on a single device
Sent Message	Sent messages are stored on the server.	Sent messages are stored on a single device
Accessible	Messages can be synced and accessed across multiple devices.	Emails can only be accessed from a single device. If you want to keep messages on the server, make sure the setting "Keep email on server" is enabled or all messages are deleted from the server once downloaded to the app or software.

*** E-mail Message (RFC22)**

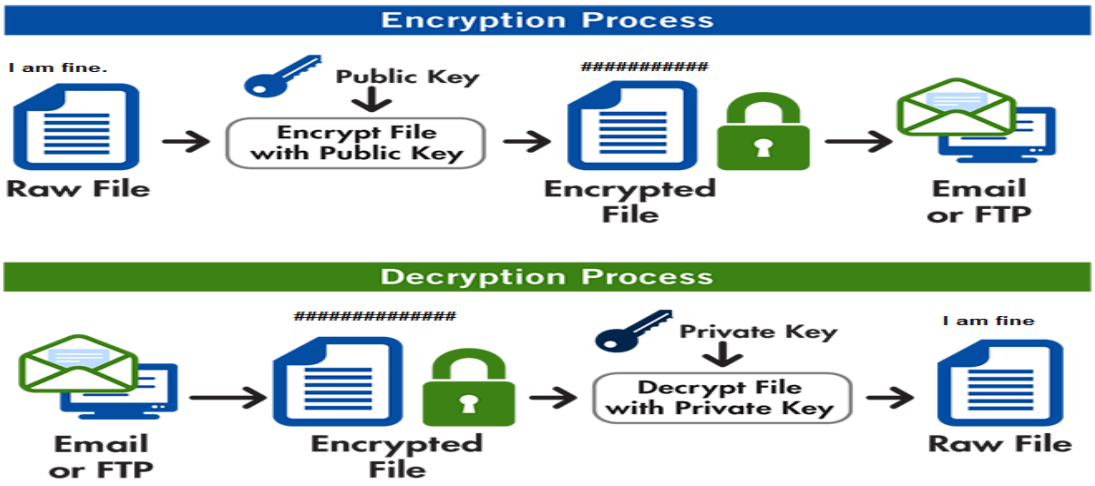
*** Pretty Good Privacy (PGP):** is a popular *program used to encrypt and decrypt email over the Internet, as well as authenticate messages with digital signatures and encrypted stored files.*

- **Pretty Good Privacy** (PGP) allows you to send files and messages securely over the Internet
- It follows cryptography technology during sending and receiving message
- PGP generates a **public key** (to encrypt messages) and a **private key** (to decrypt messages)
- OpenPGP RFC4880 is an e-mail encryption standard

How PGP works?

PGP uses a variation of the public key system, each user has an *Encryption Key (Public Key)* that is publicly known and *a Decryption Key (Private Key)* that is known only to that user. You encrypt a message you send to someone else using their public key. When they receive it, they decrypt it using their private key. Since encrypting an entire message can be time-consuming, PGP uses a faster encryption algorithm to encrypt the message and then uses the public key to encrypt the shorter key that was used to encrypt the entire message.

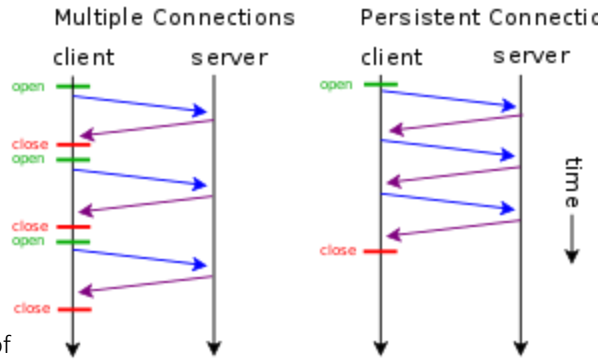
PGP is very easy to understand, on the surface. Imagine you want to send your credit card information to a friend and you write it on a piece of paper. You then put the paper in a box and send it by mail.
A thief can easily steal the box and look at the paper that contains your credit card information. What could you do instead?
You decide to put a key lock on the box, but you realize that you have to send the key along with the box. That's no good.
What if you meet your friend in person to share the key beforehand? That could work, right? It could, but then both of you have a key that allows to unlock the box. You, as the sender, will never need to open the box again after closing it. By keeping a copy of a key that can unlock the box, you are creating a vulnerability.
Finally, you found just the right solution: you'll have two keys. The first key will only be able to lock the box. The second key will only be able to open the box. That way, only the person who needs to get the content of the box has the key that allows them to unlock it.
*This is how PGP works. You have a **public key** (to lock/encrypt the message) and a **private key** (to unlock/decrypt the message). You would send the public key to all your friends so that they can encrypt sensitive messages that they want to send to you. Once you receive an encrypted message, you use your private key to decrypt it.*

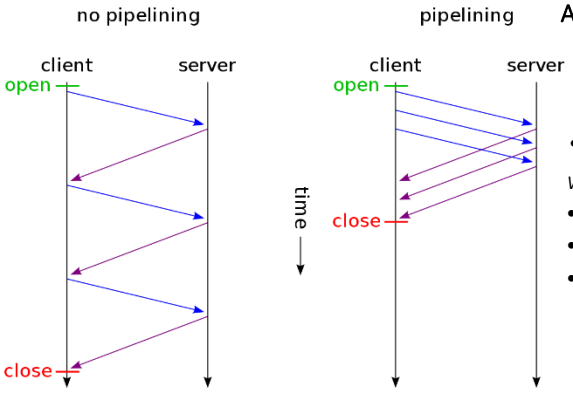


*HTTP (*HyperText Transfer Protocol) : HTTP* is the *stateless protocol* i.e. server does not maintain information about past client requests *used by the World Wide Web* and this protocol *defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response* to various commands. For example, when you enter a URL in your browser, this actually sends an HTTP command to the Web server directing it to fetch and transmit the requested Web page. The other main standard that controls how the World Wide Web works is HTML, which covers how Web pages are formatted and displayed.

HTTP Connection

1. **Persistent Connection:** Persistent HTTP uses a single TCP connection to send and receive multiple HTTP requests/responses, as opposed to opening a new connection for every single request/response pair.
2. **Non-Persistent Connection:** Nonpersistent has to create a new connection for every request whereas persistent can just use the connection it already has and pick up every new request through that one connection.

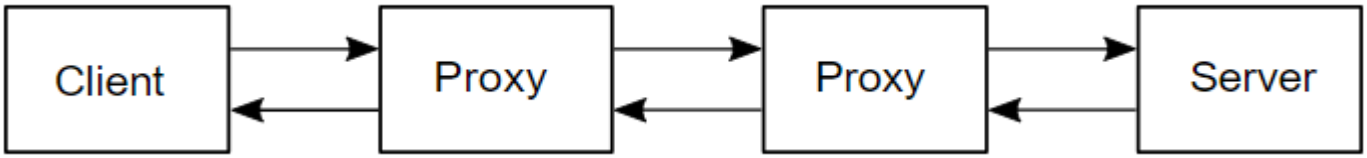




Advantages of persistent connections:

- Lower CPU and memory usage because there are less number of connections.
- Allows HTTP pipelining (*multiple HTTP requests are sent on a single TCP connection without waiting for the corresponding responses*.) of requests and responses.
- Reduced network congestion (fewer TCP connections).
- Reduced latency in subsequent requests (no handshaking).
- Errors can be reported without the penalty of closing the TCP connection.

Components of HTTP-based systems



Proxies

Between the Web browser and the server, numerous computers and machines relay the HTTP messages. Due to the layered structure of the Web stack, most of these operate at either the transport, network or physical levels, becoming transparent at the HTTP layer and potentially making a significant impact on performance. Those operating at the application layers are generally called **proxies**. These can be transparent, or not (changing requests not going through them), and may perform numerous functions:

- Caching (the cache can be public or private, like the browser cache)
- Filtering (like an antivirus scan, parental controls, ...)
- Load balancing (to allow multiple servers to serve the different requests)

- Authentication (to control access to different resources)
- Logging (allowing the storage of historical information)

HTTP Operations or Methods:

The set of common methods for HTTP/1.1 is defined below and this set can be expanded based on requirements. These method names are case sensitive and they must be used in uppercase.

S.N.	Method and Description
1	**GET :** The GET method is used to retrieve information from the given server using a given URI. Requests using GET should only retrieve data and should have no other effect on the data. **E.g.** `test/demo_form.php?name1=value1&name2=value2`
2	**HEAD :** Same as GET, but transfers the status line and header section only.
3	**POST :** A POST request is used to send data to the server, for example, customer information, file upload, etc. using HTML forms. You POST to example.com/users since you don't know the URL of the user yet, you want the server to create it.
4	**PUT :** Replaces all current representations of the target resource with the uploaded content. You PUT to example.com/users/id since you want to replace/ create a specific user.
5	**DELETE :** The DELETE method is used to request the server to delete a file at a location specified by the given URL
6	**CONNECT :** The CONNECT method is used by the client to establish a network connection to a web server over HTTP. Establishes a tunnel to the server identified by a given URI.
7	**OPTIONS :** The OPTIONS method is used by the client to find out the HTTP methods and other options supported by a web server.
8	**TRACE :** The TRACE method is used to echo the contents of an HTTP Request back to the requester which can be used for debugging purpose at the time of development.

Compare GET vs. POST

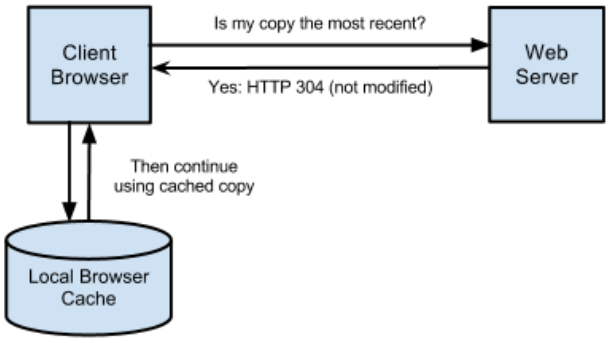
	GET	POST
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding for binary data
History	Parameters remain in browser history	Parameters are not saved in browser history
Restrictions on data length	Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters)	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Security	- GET is less secure compared to POST because data sent is part of the URL - Never use GET when sending passwords or other sensitive information!	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

Conditional GET

The modern day developer has a wide variety of techniques and technologies available to improve application performance and end-user experience. One of the most frequently overlooked technologies is that of the HTTP cache.

Role : Increasing Application Performance "HTTP has a mechanism that allows a cache to verify that its objects are up to date"

Conditional requests are those where the browser can ask the server if it has an updated copy of the resource. The browser will send some information about the cached resource it holds and the server will determine whether updated content should be returned or the browser's copy is the most recent. In the case of the latter an HTTP status of 304 (not modified) is returned.



HTTP conditional requests are requests that are executed differently, depending on the value of specific headers. These headers define a precondition, and the result of the request will be different if the precondition is matched or not.

The different behaviors are defined by the method of the request used, and by the set of headers used for a precondition:

- **for safe methods**, like **GET**, which usually tries to fetch a document, the conditional request can be used to send back the document, if relevant only. Therefore, this spares bandwidth.

- **for unsafe methods,** like **PUT**, which usually uploads a document, the conditional request can be used to upload the document, only if the original it is based on is the same as that stored on the server.

The browser will send some information about the cached resource it holds and the server will determine whether updated content should be returned or the browser's copy is the most recent. In the case of the latter an HTTP status of 304 (not modified) is returned.
Though conditional requests do invoke a call across the network, unmodified resources result in an empty response body – saving the cost of transferring the resource back to the end client. The backend service is also often able to very quickly determine a resource's last modified date without accessing the resource which itself saves non-trivial processing time.

So consider a document /sample.html on example.com. Consider the very first request of the Client, Since this is the first request, the client does not know about the modified time, etc…

Here goes the **request** header as follows…

```
GET /sample.html HTTP/1.1
Host: example.com
```

Now the response that comes from the server is the document with the response headers. The response headers would be…

```
HTTP/1.x 200 OK
Via: The-proxy-name
Content-Length: 32859
Expires: Tue, 27 Dec 2005 11:25:11 GMT
Date: Tue, 27 Dec 2005 05:25:11 GMT
Content-Type: text/html; charset=iso-8859-1
Server: Apache/1.3.33 (Unix) PHP/4.3.10
Cache-Control: max-age=21600
Last-Modified: Wed, 01 Sep 2004 13:24:52 GMT
Etag: "4135cda4"
```

Next time when the user calls for the same document /sample.html within the specified cache time frame. The browser(client) will make a conditional get request, try to ask the server that if the document is modified after the specified time zone whose hashed value was the Etag value, ONLY THEN return a new document or else confirm that it is an old document.

So the request header would be as…

```
GET /sample.html HTTP/1.1
Host: example.com
If-Modified-Since: Wed, 01 Sep 2004 13:24:52 GMT
If-None-Match: "4135cda4"
```

The response to the above request would be as.

```
HTTP/1.x 304 Not Modified
Via: The-proxy-server
Expires: Tue, 27 Dec 2005 11:25:19 GMT
Date: Tue, 27 Dec 2005 05:25:19 GMT
Server: Apache/1.3.33 (Unix) PHP/4.3.10
Keep-Alive: timeout=2, max=99
Etag: "4135cda4"
Cache-Control: max-age=21600
```

There are two ways to utilize conditional GETs:

1. **Cache-Control**: It tells the client the maximum time in seconds to cache the document.
2. **Last-Modified**: The document's last modified date
3. **Etag**: A unique hash for the document.

Conditional headers

Several HTTP headers, called conditional headers, lead to conditional requests. These are:

- **If-Match:** Succeeds if the ETag of the distant resource is equal to one listed in this header. By default, unless the etag is prefixed with 'W/', it performs a strong validation.
- **If-None-Match:** Succeeds if the ETag of the distant resource is different to each listed in this header. By default, unless the etag is prefixed with 'W/', it performs a strong validation.
- **If-Modified-Since:** Succeeds if the Last-Modified date of the distant resource is more recent than the one given in this header.
- **If-Unmodified-Since:** Succeeds if the Last-Modified date of the distant resource is older or the same than the one given in this header.
- **If-Range:** Similar to If-Match, or If-Unmodified-Since, but can have only one single etag, or one date. If it fails, the range request fails, and instead of a 206 Partial Contentresponse, a 200 OK is sent with the complete resource.

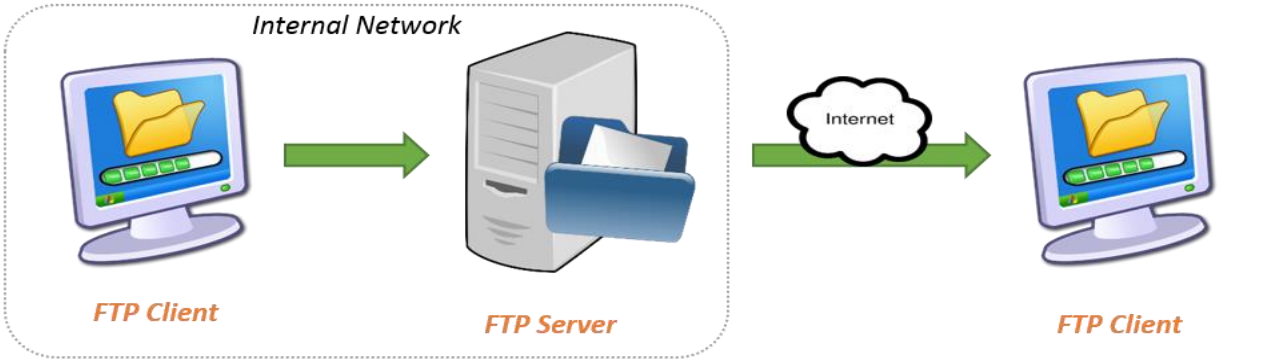
***FTP : *File Transfer Protocol* (FTP)** is Client Server based protocol that commonly used *protocol for exchanging files(sending and receiving or downloading) over the Internet.* FTP uses the Internet's TCP/IP protocols to enable data transfer. FTP uses a client-server architecture, often secured with SSL/TLS. FTP promotes sharing of files via remote computers with reliable and efficient data transfer.
FTP provides mechanism for authenticating users for the access control, setting file transmission parameters, identifying the files to be transferred and some file and directory maintenance functions. There are three relatively independent features to the FTP : access control, filename and file translation.
* 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.

Types of FTP

I. FTP, or File Transfer Protocol, is a protocol used frequently in website creation that allows you to transfer data. FTP enables one to transfer information from their computer to their web hosting account. *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.*



Employee uploads file to be shared with a business partner.

File is hosted on the FTP server within the corporate network.

Business partner gets access to the file by accessing the share folder on the file server.

II. FTPS, or File Transfer Protocol Secure, is a more secure form of FTP and is also known as FTP-SSL. In short, FTPS is basic FTP with some security added to the data transfer. These added security protocols, such as TLS (Transport Layer Security) and SSL (Secure Sockets Layer), are cryptographic and provide encryption of data to protect your information as it moves from point A to point B. *So, with FTPS, this added layer of security would encrypt your login information so that those attempting to steal your password would end up stealing an encrypted version, which ends up being completely worthless.*

III. FTPES is just another form of FTPS, only the difference is that it connects to your web hosting account explicitly, rather than FTPS's implicit connection. More simply, the difference is primarily how and when the login information is encrypted. FTPES is known to be the safest FTP connection, and that's exactly what SmartFile has. *For example, FTPES is what is used when making online purchases at 'secure' websites*

Overall, knowing the difference between **FTP, FTPS, and FTPES** is important mostly because it involves the security and privacy of your data.

FTP Modes

FTP supports two transfer mode or FTP uses two TCP connections for communication.

i. **Control Connection using port 21:** Only to pass control information and is not used to send files on port 21.
ii. **Data Connection using port 20:** a data connection on **port 20** to send the data files between the client and the server.

The connection has to be established before the files can actually be sent across. I think the user authentication takes place over the control connection on port 21. After which, the data starts transferring over the data connection on port 20.

1. **Active Mode**: With the first active mode, the client initiates the *connection* to the server on *port 21(Command)* and the server then *binds* his on *port 20(Data)* and opens a connection to a port above 1023 to the client.
2. **Passive Mode**: While using passive FTP both connections are established from the client to port tcp 21 and 20 to the server.

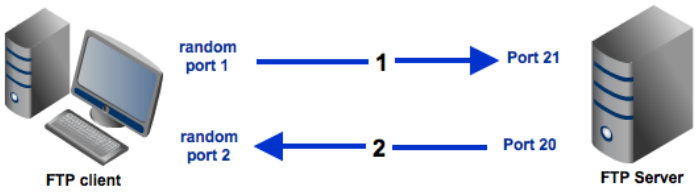


Fig. Active

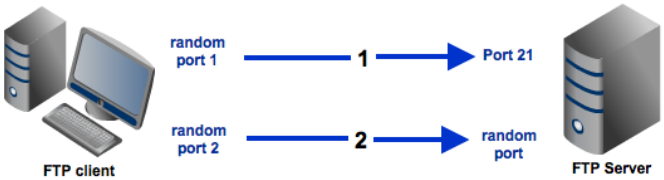
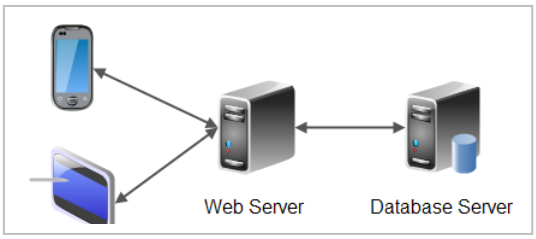


Fig. Passive

FTP Server

1. **Anonymous Server** - No need of password
2. **Non Anonymous Server –** Need password

3.2. N-Tiered Client/Server Architecture



Client/server architecture is a *producer-consumer computing architecture where the server acts as the producer and the client as a consumer.* The server produces services like *applications access, storage, file sharing, printer access and/or direct access to the server's raw computing power.*

Client/server architecture works when *the client computer sends a resource or process request to the server over the network connection, which is then processed and delivered to the client.*

- A server is a computer, a device or a program that is *dedicated to managing network resources. E.g.* web browsers, E-mail clients, and online chat clients.
- A client can be a simple application or a whole system that *accesses services being provided by a server and sending request to server*.

Clients are classified into three types:

- *Thin Client:* Thin *clients use the resources of the host computer*. A thin client generally *only presents processed data provided by an application server, which performs the bulk of any required data processing. A device using web application (e.g. Office Web Apps) is a thin client. Programming environments for thin clients include JavaScript, ASP.NET, JSP, Ruby on Rails, Django, PHP and others.*
- *Thick/Fat Client*: A *fat client* or *rich client* or *thick client*, is a client that performs the *bulk of any data processing operations itself, and does not necessarily rely on the server*. E.g. personal computer, because of its relatively large set of features and capabilities and its light reliance upon a server. *For example, a computer running a CAD program (such as AutoCAD or CATIA) that ultimately shares the result of its work on a network is a fat client. Common development tools for rich clients use Delphi, NetBeans and Visual Studio.*
- *Hybrid*: A *hybrid client* is a combination of thin client and fat client. E.g. video game Diablo III
  Specific types of clients include: web browsers, E-mail clients, and online chat clients.
  Specific types of servers include: web servers, FTP servers, database servers, E-mail servers, file servers, print servers.

Clients characteristics

- Always *initiates* requests to servers.
- *Waits* for replies.
- *Receives* replies.
- Usually connects to a small number of servers *at one time.*
- Usually *interacts* directly with end-users using any user interface such as graphical user interface.

Server characteristics

- Always *wait for a request from* one of the clients.
- *Serve* clients requests then replies with requested data to the clients.
- A server *may communicate with other servers* in order to serve a client request.
- If additional information is required to process a request (or security is implemented), a server may request additional data (passwords) from a client before processing a request.
- End users typically do not interact directly with a server, but use a client.

N-Tiered Client/Server Architecture : is a client-server architecture concept in software engineering where the *presentation, processing and data management functions are both logically and physically separated.*

Applications in N-tier architecture are basically separated into : *a presentation tier, a middle tier, and a data tier.* The easiest way to separate the various tiers in an n-tier application is to create discrete projects for each tier that you want to include in your application.

Components of Client server architecture

Client layer: this layer is involved with users directly. There may be several different types of clients coexisting, such as WPF, Window form, HTML web page and etc.

Client presenter layer: contains the presentation logic needed by clients, such as ASP.NET MVC in IIS web server. Also it adapts different clients to the business layer.

Business layer: Business layer that responsible for all business logic e.g. displaying the results from business logic, next control flow. "business logic" is what rules the company has, while "presentation logic" is how the details are shown to users.

Persistence layer: handles the read/write of the business data to the data layer, also called data access layer (DAL) and perform CRUD (Create, Read, Update, Delete) operations. It helps easier migration to other storage engines, better encapsulation of database logic in a single layer (easier to replace or modify later depending on how well you have designed your cross-layer interfaces etc...)

Data layer: the external data source, such as a database.

Major Quality Attributes on Tier Architecture

- **Secure:** You can secure each of the three tiers separately using different methods.
- **Easy to manage:** You can manage each tier separately, adding or modifying each tier without affecting the other tiers.
- **Scalable:** If you need to add more resources, you can do it per tier, without affecting the other tiers.
- **Flexible:** Apart from isolated scalability, you can also expand each tier in any manner that your requirements dictate.
- **More efficient development.** N-tier architecture is very friendly for development, as different teams may work on each tier.
- **Easy to add new features.** If you want to add a new feature, you can add it to the appropriate tier without affecting the other tiers.
- **Easy to reuse.** Because the application is divided into independent tiers, you can easily reuse each tier for other software projects. For instance, if you want to use the same program, but for a different data set, you can just replicate the logic and presentation tiers and then create a new data tier.

Comparison of Architectures

Architecture	Pros	Cons
One tier	Simple Very high performance Self-contained	No networking – can't access remote services Potential for spaghetti code
Two tiers	Clean, modular design Less network traffic Secure algorithms Can separate UI from business logic	Must design/implement protocol and reliable data storage
Three tiers	Can separate UI, logic, and storage Reliable, replicable data Concurrent data access via transactions Efficient data access	Need to buy DB product and Need to hire DBA Need to learn SQL Object-relational mapping is difficult
N tiers	Support multiple applications more easily Common protocol/API	Less inefficient Must learn API (CORBA, RMI, etc.) Expensive products More complex, more faults Load balancing is hard

3.3. Universal Internet Browsing

NTERNET has become a playground of the mind, where anyone with the time and inclination can travel the globe. The main purpose of an internet browser is to translate, or render, the code that websites are designed in into the text, graphics, and other features of the web pages that we are all used to seeing today.

The first web browser was called WorldWideWeb, and later changed its name toNexus. Created by Sir Tim Berners-Lee, it was released in 1990, and at least gave people a basic way to view web pages. But it was a long way from the immersive online experience we have today.

Basic functionality of Mosaic web browser

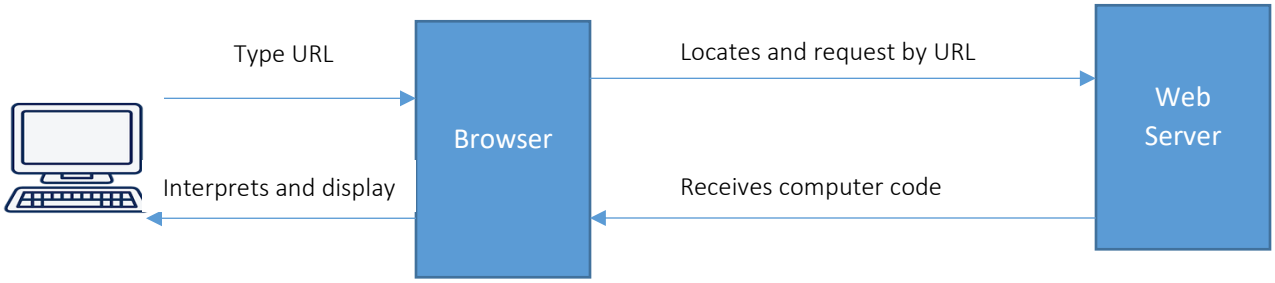
- Mouse driver graphical interface
- Ability to display hypertext, hypermedia documents, electronic text in different style(fonts, bold, italic, layouts- paragraph, bullets)
- Supports sounds, movies, animimated picture
- Support http, ftp based applications
- Opens URL and does Print/ Save/ Save AS/ Reload current document, Back/ Forward from history links, copy/ paste content in clipboard

`What makes a good browser ?`

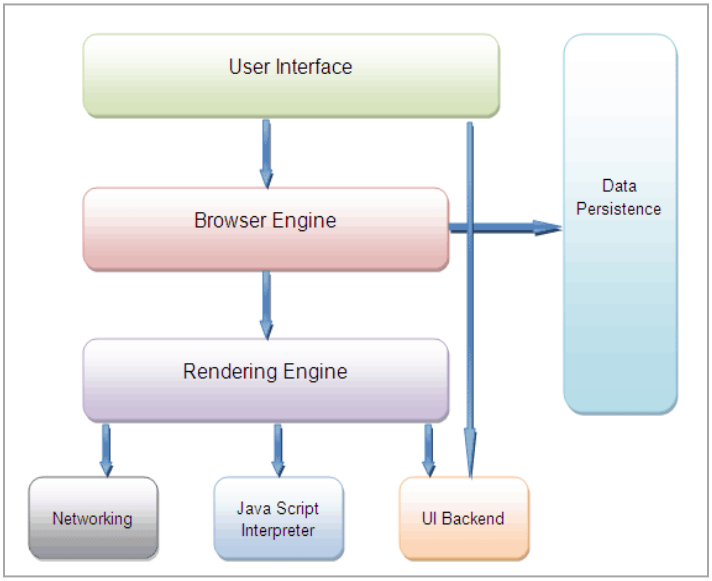
- Installation : automate setup process
- Security : offer basic authentication : login process, SSL, S-HTTP
- Navigability : viewing progress of download
- Data capture : easy to download text and picture, audio/ videos in background
- Interoperability : should support web based email, ftp, news
- Performance : analytic way to compare browser speed.

How Internet Browsers Work:

1. You type a website's URL into your browser's address bar; "http://www.hcoe.edu.np" is an example of a URL.
2. The browser locates and requests that page's information from a web server.
3. The browser receives a file in a computer code like HTML or JavaScript, which includes instructions about how to display the information on that page.
4. The browser interprets that file and displays the page for you to read and interact with. And it does all of this in just a few seconds, usually.



The browser's main functionality is to fetch the files from the server and to display them on the screen. It basically displays html files containing images, PDF, videos, flashes, etc. in an ordered layout. A browser is a group of structured codes that performs plenty of tasks to display a webpage on the screen. These codes are separated in to different components according to their tasks performed.



- **User Interface** – It is the space where interaction between users and the browser occurs. Most of the browsers have common inputs for user interface. *Some of them are - an address bar, next and back buttons, buttons for home, refresh and stop, options to bookmark web pages, etc.*
- **Browser Engine** – It is the piece of code that communicates the inputs of user interface with the rendering engine. It is responsible for *querying and manipulating the rendering engine according to the inputs from various user interfaces.*
- **Rendering Engine** – It is the part thoroughly responsible for displaying the requested content on the screen. It first parses the html tags and then using the styles, it builds a render tree and finally a render layout, which displays the content on the screen.
- **Networking** – The fraction of the code written in the browser, responsible to send various network calls. For example, sending the http requests to the server.
- **Java Script Interpreter** – It is the component of the browser written to interpret the java script code presented in a web page.
- **UI Backend** – This draws basic widgets on the browser like combo boxes, windows, etc.
- **Data Storage** – It is small database created on the local drive of the computer where the browser is installed. This database stores various files like cache, cookies, etc.

Sample of available browsers

MS Internet Explorer V8	www.microsoft.com/downloads – the default Windows browser
Mozilla Firefox V 8.0	http://www.mozilla.org/ Mozilla's mission is to promote openness, innovation and opportunity on the web and offer a number of products, namely Firefox as a browser. It is a global non-profit organisation. They now offer a browser for phones as well. Has hundreds of plug-ins, add-ons and free additional resources
Google Chrome	http://www.google.com/chrome – for Windows, Mac and Linux – mainly boasts speed and performance with a clean, uncluttered interface
Safari	http://www.apple.com/safari/ For MAC and Windows – very intuitive, has multi windowed site reference with elegant look and feel and boasts having some powerful tools. Also now connects seamlessly to the iCloud.
Opera	http://www.opera.com/ For Windows Mac and Linux – as well as Smartphones and tablets – mainly boasts having increased speed, add-ons and extensions Opera was designed to run on low-end computers with a strong commitment to computer accessibility for users who may have visual or mobility impairments. It has keyboard control over all main functions of the browser and the default keyboard shortcuts can be modified. Opera also supports the use of access keys to allow a computer user to immediately jump to a specific part of a web page via the keyboard. Opera was also one of the first browsers to support mouse gestures allowing patterns of mouse movements.
Midori	http://www.twotoasts.de/index.php?/pages/midori_summary.html – is a web browser that aims to be lightweight and fast. It also provides mouse gestures i.e. a pointing device gesture or mouse gesture is a way of combining pointing device movements and clicks which the software recognizes as a specific command

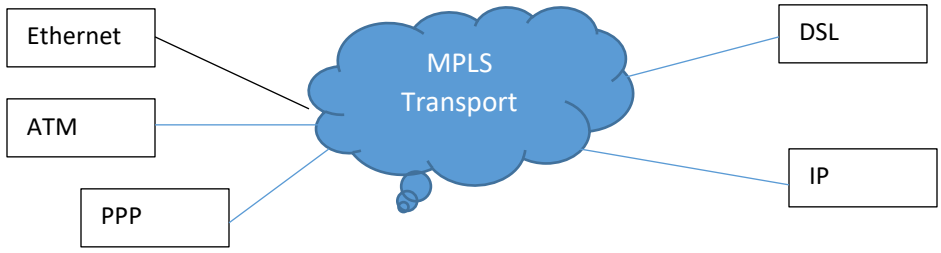
3.4. Multiprotocol Support

Multiprotocol Label Switching (MPLS) is a protocol for *speeding up and shaping network traffic flows by rapid forwarding transaction data.* MPLS allows most packets to be forwarded at Layer 2 (the switching level) rather than having to be passed up to Layer 3 (the routing level). *Each packet gets labeled on entry into the service provider's network by the ingress router. All the subsequent routing switches perform packet forwarding based only on those labels—they never look as far as the IP header. Finally, the egress router r//emoves the label(s) and forwards the original IP packet toward its final destination.*

The label determines which pre-determined path the packet will follow. The paths, which are called label-switched paths (LSPs), allow service providers to decide ahead of time what will be the best way for certain types of traffic to flow within a private or public network.

Service providers can use MPLS to improve quality of service (QoS) by defining LSPs that can meet specific service level agreements (SLAs) on traffic latency, jitter, packet loss and downtime. *For example, a network might have three service levels -- one level for voice, one level for time-sensitive traffic and one level for "best effort" traffic*. MPLS also supports traffic separation and the creation of virtual private networks (VPNs) virtual private LAN services (VPLS) and virtual leased lines (VLLs).

MPLS got its name because it works with the Internet Protocol (IP), Asynchronous Transport Mode (ATM) and Frame Relay network protocols; any of these protocols can be used to create an LSP. It was created in the late 1990s to avoid having routers waste time by having to stop and look up routing tables. *A common misconception is that MPLS is only used on private networks, but the protocol is used for all service provider networks -- including Internet backbones. Today, Generalized Multi-Protocol Label Switching (GMPLS) extends MPLS to manage time division multiplexing (TDM), lambda switching and other classes of switching technologies beyond packet switching.*



MPLS Operation

How Does MPLS Work?

MPLS works by tagging the traffic, in these example packets, with an Identifier (a Label) to distinguish the LSPs (Label Switched Path- *a specific path between PE (Provider Edge) routers on the MPLS core that the traffic will traverse*). When a packet is received, the router uses this label (and sometimes also the link over which it was received) to identify the LSP (Label Switched Path). It then looks up the LSP in its own forwarding table to determine the best link over which to forward the packet, and the label to use on this next hop.

A different label is used for each hop, and it is chosen by the router or switch performing the forwarding operation. This allows the use of very fast and simple forwarding engines, which are often implemented in hardware.

Ingress routers at the edge of the MPLS network classify each packet potentially using a range of attributes, not just the packet's destination address, to determine which LSP to use. Inside the network, the MPLS routers use only the LSP labels to forward the packet to the egress router.

- Label switched routers capable of switching and routing packets based on label appended to packet
- Labels define a flow of packets between end points or multicast destinations
- Each distinct flow (forward equivalence class – FEC) has specific path through LSRs (Label Switched Routers) defined - Connection oriented
- IP header not examined - Forward based on label value

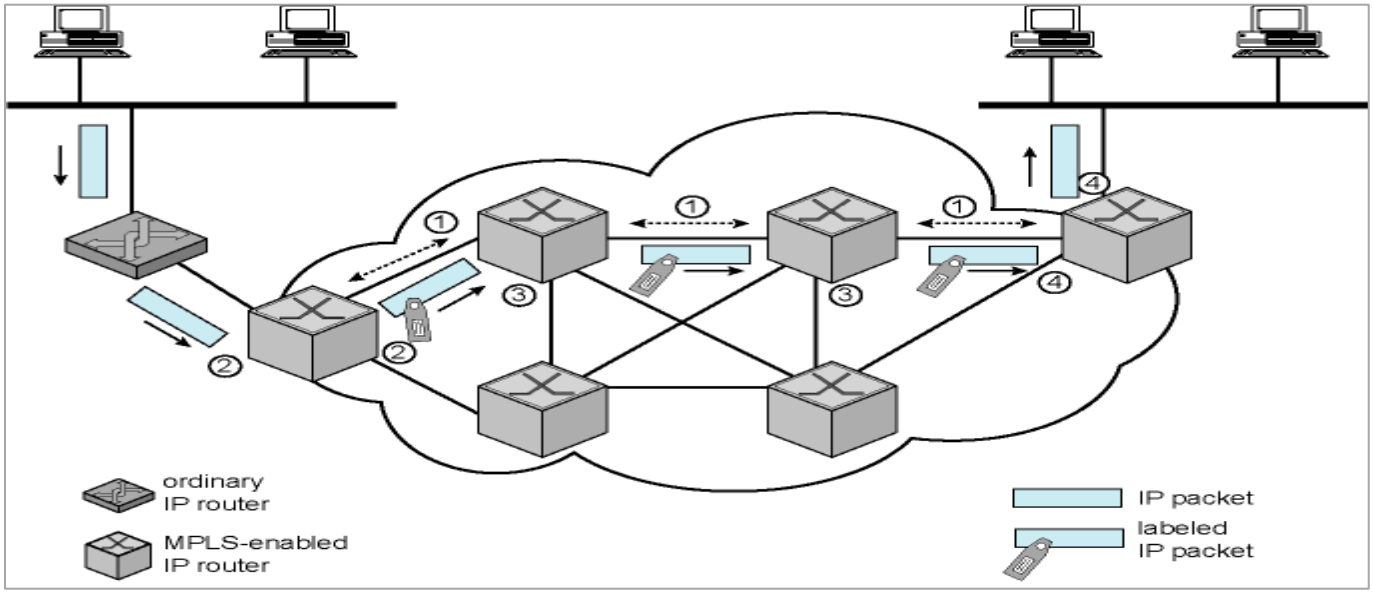


Fig. MPLS Operation Diagram

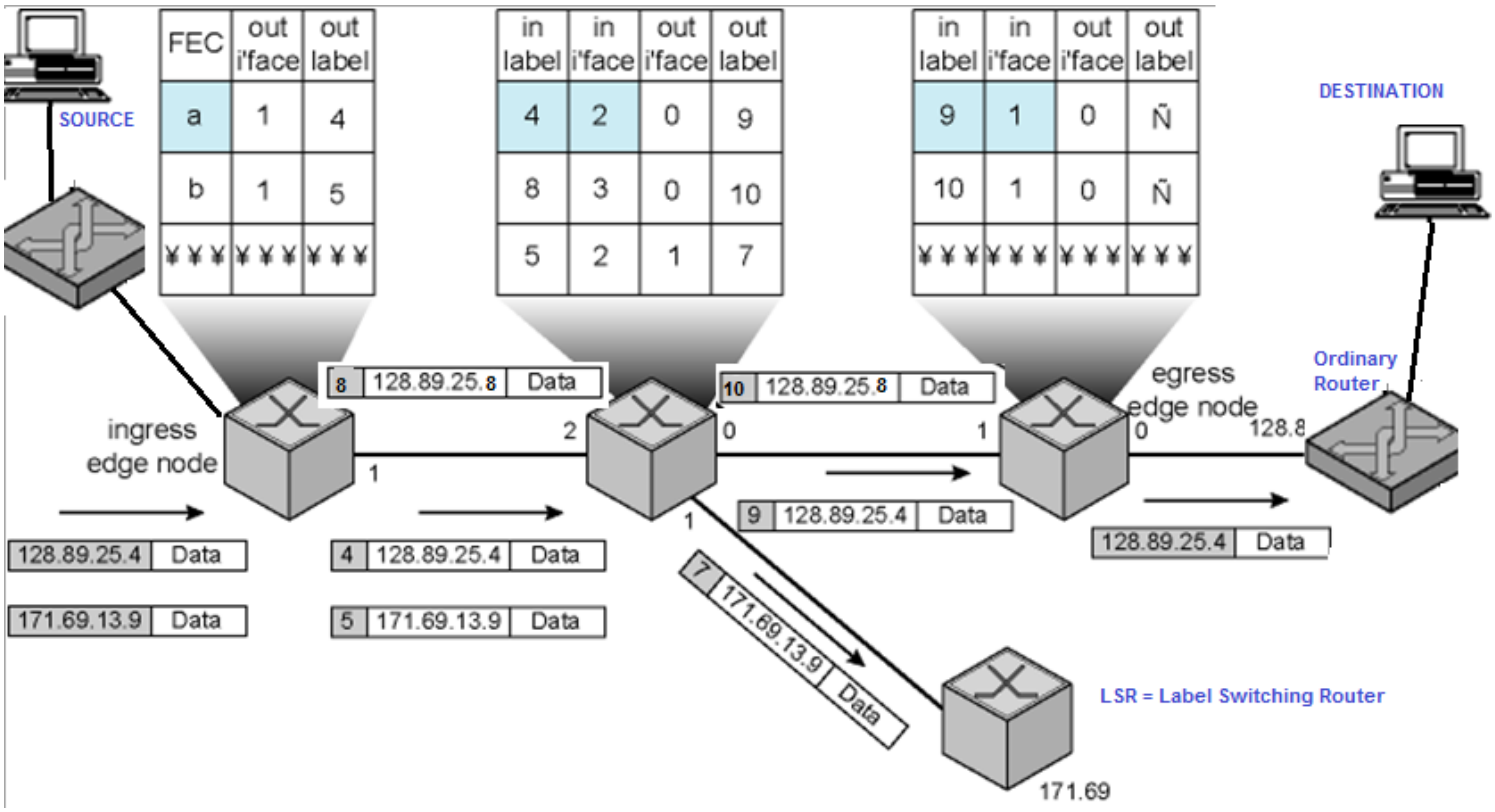


Fig. MPLS Packet Forwarding

MPLS Benefits

The initial goal of label based switching was to bring the speed of Layer 2 switching to Layer 3. Label based switching methods allow routers to make forwarding decisions based on the contents of a simple label, rather than by performing a complex route lookup based on destination IP address. This initial justification for technologies such as MPLS is no longer perceived as the main benefit, since Layer 3 switches are able to perform route lookups at sufficient speeds to support most interface types.

However, MPLS brings many other benefits to IP-based networks. Forwarding packets based on labels rather than routing them based on headers results in several important advantages:

- `Faster Speed:` Due to the labeling technology, the speed of performing lookups for destinations and routing is much faster and without overloading the CPU than the standard IP table lookups (concerned with longest prefix match) non-MPLS routers have to perform, due to simpler label lookup.
- `QoS:` This is a big one. MPLS networks achieve greater Quality of Service for their customers. Quality of Service (QoS) means exactly that – you can expect a higher standard of service such as reliability, speed, and voice quality. This is for a few reasons, one already mentioned above.
  In addition, MPLS networks are able to assign priorities to the different packets based on what the labels say about that packet. Packets with greater priority, voice over data for example, are given more bandwidth allocation. A packet that which is not deemed as high priority is given less. Obviously sending documents online don't need to be assured of the same bandwidth required for someone who is wanting to have a conversation.
- `Faster Restoration:` MPLS networks are also able to restore interrupted connections at a faster speed than typical networks.
- `Security`: MPLS offers greater security and are often required for companies e.g. telecoms which need enhanced privacy and security for their network needs. It's also very popular with organizations that need a scalable WAN that can carry both voice (phone calls) and data.

In addition to all the above advantages, one of the most important advantages of MPLS is that it is independent of the layer 2 and layer 3 technologies and hence allows integration of networks with different layer 2 and layer 3 protocols.

Comparison between MPLS and IP Routing

Parameters	MPLS	IP Routing
Switching/Routing principle	Switching traffic based on labels advertised by LDP	Routing based on the destination address for entries in the routing table.
Switching/Routing path	Establishes LSP (dedicated path) before data can flow.	No dedicated path is established, packet is routed based on IP addresses.
Tables usage	Builds LFIB (Label Forwarding Information Base) table using LDP protocol.	Stores IP routing table.
Layer of functioning	Labels inserted between layer 2 and Layer 3 (hence layer 2.5)	Performed at Layer 3
Overlapping IP address	MPLS can allow communication across overlapping IP addresses of multiple customers	Does not allow communication across overlapping address of different customers
Related terms	LSP, LDP/TDP, VRF, LFIB, Push, Swap and Pop.	Route Lookup, IP protocol
Traffic Latency	Lower latency than traditional IP routing	Incurs higher latency than MPLS
Topology and services	With MPLS, providers can create (with use of different labels and label stacks) different topologies & services (MPLS-TE, MPLS VPNs).	Single topology can be created per IP routing domain.
Traffic Engineering	Scalable and proficient in service	Partially possible but not scalable solution
Separate Routing table	In MPLS network , each customer has separate routing network	Traditional IP routing can only have 1 Routing table for all customers
Scalability	Medium	High
Target scope	Service provider domain, Large & Multitenant Data Centers.	Home, Office, Service PTP/Underlay links, Data centers etc.
Traffic type	Allows non-IP traffic forwarding in addition to IP traffic	Allows forwarding of IP traffic