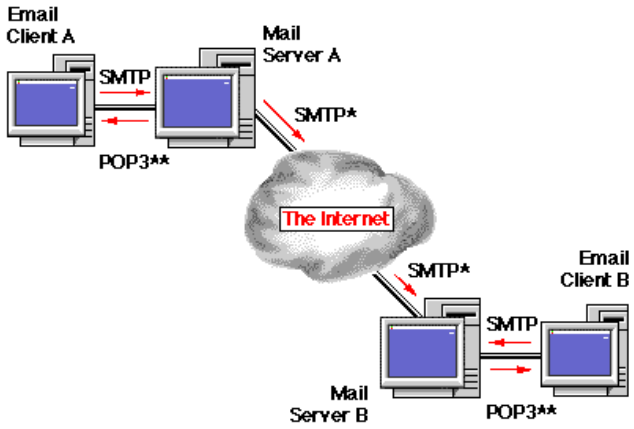
Unit 3

Protocols and Client/Server Applications

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

Simple Mail Transfer Protocol (SMTP)

Simple Mail Transfer Protocol (SMTP) is an Internet standard for electronic mail (e-mail) transmission across Internet Protocol (IP) networks. It's a set of communication guidelines that allow software to transmit email over the Internet. SMTP provides a set of codes that simplify the communication of email messages between servers. It's a kind of shorthand that allows a server to break up different parts of a message into categories the other server can understand. Any email message has a sender, a recipient - or sometimes multiple recipients - a message body, and usually a title heading.



Typically, an e-mail client will connect to an SMTP server and send any mail in the outbox. That SMTP server will then look up the mail servers responsible for each e-mail address that is supposed to receive each message. The server will then contact the destination servers, one at a time, and relay the message using SMTP. The destination server will either pass the message along to another local e-mail server that is assigned to store messages for the person receiving the mail, or store that message itself.

In the process of sending an email, an SMTP client host connects to an SMTP server through a port (typically 25) and provides information about the sender, address of the recipient and body text, through a standard protocol. The SMTP client host breaks up an email address by analyzing the recipient name and the recipient domain name which determines where to deliver the message. The SMTP client host uses a domain name server (DNS) to identify the IP address of a remote SMTP server. Then the SMTP client host (the sender) connects to that of the remote host and delivers the message to the recipient. Alternatively, if the message is temporarily unable to be delivered it goes into a queue and the SMTP client host will later attempt to deliver the message at a specified interval.

Post Office Protocol (POP)

Post Office Protocol (POP) is an application-layer Internet standard protocol used by local e-mail clients to retrieve e-mail from a remote server over a TCP/IP connection.

POP was designed for dial-up users whose Internet connections were intermittent and not reliable. For this reason, POP downloads the entire email message to your computer, so that you don't have to be connected to the Internet to view it. Once the email message is downloaded, it is deleted from the server. The problem with this approach is that you can then no longer access the message from a different computer. If your email is downloaded to a public computer, you may not be able to access it later and it could be viewed by others.

POP3 is the version 3 (the latest version) of the Post Office Protocol.

Advantages of POP/ POP3

- Email is available when you are offline
- Email is not stored on the server, so your disk usage on the server is less
- Just about any email client (software) supports POP

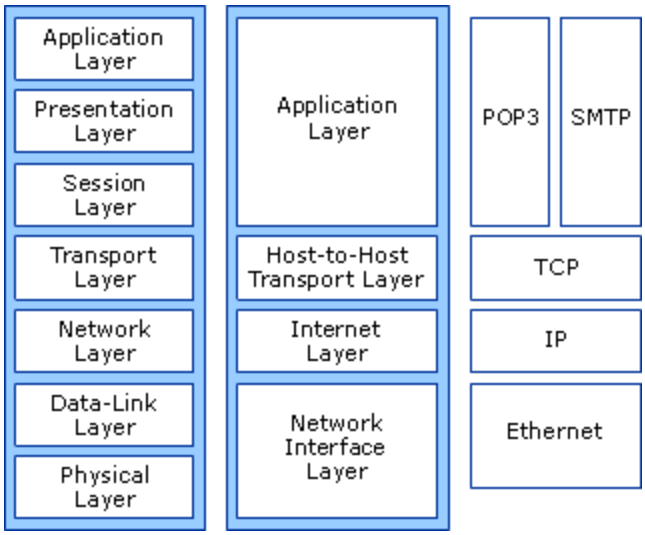
Disadvantages of POP/ POP3

- Can be much slower to check mail
- Much harder to do server-side filtering
- Mail is inaccessible from other machines

How POP3 Works

When you check your e-mail, your e-mail client connects to the POP3 server using **port 110**. The POP3 server requires an account name and a password. Once you've logged in, the POP3 server opens your text file and allows you to access it. Like the SMTP server, the POP3 server understands a very simple set of text commands. Here are the most common commands:

- **USER** - enter your user ID
- **PASS** - enter your password
- **QUIT** - quit the POP3 server
- **LIST** - list the messages and their size
- **RETR** - retrieve a message, pass it a message number
- **DELE** - delete a message, pass it a message number
- **TOP** - show the top x lines of a message, pass it a message number and the number of lines



POP3 Architecture

Your e-mail client connects to the POP3 server and issues a series of commands to bring copies of your e-mail messages to your local machine. Generally, it will then delete the messages from the server (unless you've told the e-mail client not to).

Internet Mail Access Protocol (IMAP)

As its name implies, IMAP allows you to access your email messages wherever you are; much of the time, it is accessed via the Internet. Basically, email messages are stored on servers. Whenever you check your inbox, your email client contacts the server to connect you with your messages. When you read an email message using IMAP, you aren't actually downloading or storing it on your computer; instead, you are reading it off of the server. As a result, it's possible to check your email from several different devices without missing a thing.

There are several advantages to using IMAP. First, it allows you to access your email messages from anywhere, via as many different devices as you want. Second, it only downloads a message when you click on it. As a result, you do not have to wait for all of your new messages to download from the server before you can read them. Third, attachments are not automatically downloaded with IMAP. As a result, you're able to check your messages a lot more quickly and have greater control over which attachments are opened. Finally, IMAP can be used offline just like POP - you can basically enjoy the benefits of both protocols in one.

When using an IMAP connection, users can create multiple folders and mailboxes on the remote server to better organize their email transmissions. IMAP requires continual server access while users work with their mail interface

Pretty Good Privacy (PGP)

Pretty Good Privacy (PGP) is a data encryption and decryption computer program that provides cryptographic privacy and authentication for data communication. PGP is often used for signing, encrypting and decrypting texts, e-mails, files, directories and whole disk partitions to increase the security of e-mail communications.

PGP provides two main functions, encryption and digital signatures. PGP uses a standard public key encryption scheme, wherein it uses encoding and decoding algorithms to create a public key and a private key. The public key is used by other people to encrypt messages that they send to you, and the private key is used by you to decrypt messages that were encrypted with your public key. The idea is that you are the only person with access to your private key, so you are the only person that can decrypt messages that were encrypted using your public key.

How PGP Works?

PGP uses a combination of algorithms to perform encryption. The first step in PGP's encryption process is to compress the text that is to be encrypted, called plaintext. Next, the International Data Encryption Algorithm is used to generate a random session key, which is used to encrypt the compressed file, producing what is called the cipher text. Continuing, the well-known RSA (Rivest, Shamir, and Adleman) public key encryption algorithm is used to encrypt the session key using the recipient's public key. This encrypted session key is then placed at the front of the cipher text file, which is now ready for sending. To decrypt messages, this process is essentially reversed using the private key though, instead of the public key.

To authenticate messages, PGP uses digital signatures. The concept of a digital signature or key fingerprint is to ensure that the message has not been altered in any way from the original and that the message is, in fact, from the sender that is claimed. The key fingerprint is a string of hexadecimal numbers that is unique to the message.

File Transfer Protocol (FTP)

File Transfer Protocol (FTP) is a standard Internet protocol for transmitting files between computers on the Internet. Like the Hypertext Transfer Protocol (HTTP), which transfers displayable Web pages and related files, and the Simple Mail Transfer Protocol (SMTP), which transfers e-mail, FTP is an application protocol that uses the Internet's TCP/IPprotocols. FTP is commonly used to transfer Web page files from their creator to the computer that acts as their server for everyone on the Internet. It's also commonly used to download programs and other files to your computer from other servers.

FTP requires a TCP/IP network to function, and relies on the use of a dedicated FTP server and one or more FTP clients. The FTP server is typically left on (running as a service or daemon) at all times to receive connections from FTP clients. Generally FTP needs server address, username and password for authentication.

FTP Vs HTTP

1. HTTP is used to view websites while FTP is used to access and transfer files. FTP's file transfer purpose is more or less for website maintenance and batch uploads, while HTTP is for client-end work and for end users to upload things such as movies, pictures and other files to the server.

2. HTTP and FTP clients: The common HTTP client is the browser while FTP can be accessed via the command line or a graphical client of its own.

3. HTTP Headers: HTTP Headers contains metadata such as last modified date, character encoding, server name and version and more which is absent in FTP.

4. Age Difference: FTP is about 10 years older than HTTP.

5. Data Formats: FTP can send data both in ASCII and Binary Format but HTTP only uses Binary Format.

6. Pipelining in HTTP: HTTP supports pipelining. It means that a client can ask for the next transfer already before the previous one has ended, which thus allows multiple documents to get sent without a round-trip delay between the documents, but this pipelining is missing in FTP.

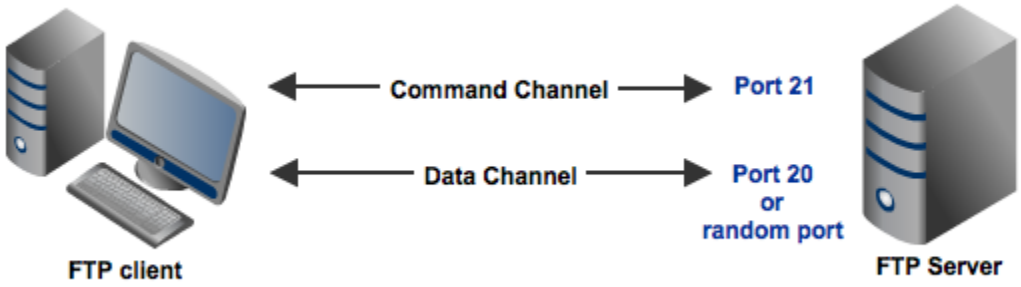
7. Dynamic Port Numbers in HTTP: One of the biggest hurdles about FTP in real life is its use of two connections. It uses a first primary connection to send control commands on, and when it sends or receives data, it opens a second TCP stream for that purpose. HTTP uses dynamic port numbers and can go in either direction,

8. Persistent Connection in HTTP: For HTTP communication, a client can maintain a single connection to a server and just keep using that for any amount of transfers. FTP must create a

new one for each new data transfer. Repeatedly making new connections are bad for performance due to having to do new handshakes/connections all the time.

9. One area in which FTP stands out somewhat is that it is a protocol that is directly on file level. It means that FTP has for example commands for listing dir contents of the remote server, while HTTP has no such concept.

A typical FTP session operates using two channels: **a command (or control) channel** and a **data channel**. As their names imply, the command channel is used for transmitting commands as well as replies to those commands, while the data channel is used for transferring data.



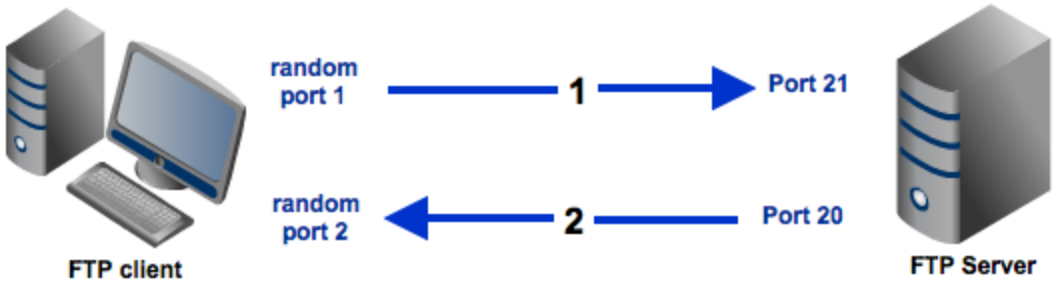
Unless you configure your FTP server differently, you will normally set your command channel to use port 21. The port you'll use for the data channel, on the other hand, can differ depending on which data transfer mode you choose. If you choose active mode, then the data channel will normally be port 20. But if you choose passive mode, then the port that will be used will be a random port.

Active mode FTP

Among the two modes, Active mode is the older one. It was the mode introduced in the early days of computing when mainframes were more common and attacks to information security were not as prevalent.

Here's a simplified explanation on how an active mode connection is carried out, summarized in two steps.

1. A user connects from a random port on a file transfer client to port 21 on the server. It sends the PORT command, specifying what client-side port the server should connect to. This port will be used later on for the data channel and is different from the port used in this step for the command channel.
2. The server connects from port 20 to the client port designated for the data channel. Once connection is established, file transfers are then made through these client and server ports.

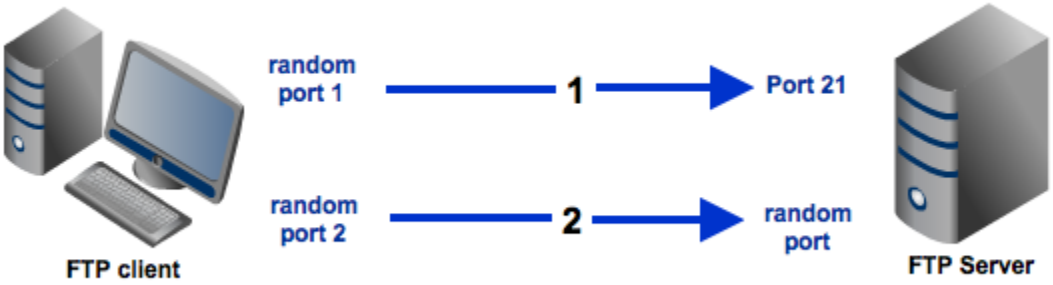


Passive mode FTP

In passive mode, the client still initiates a command channel connection to the server. However, instead of sending the PORT command, it sends the PASV command, which is basically a request for a server port to connect to for data transmission. When the FTP server replies, it indicates what port number it has opened for the ensuing data transfer.

Here's how passive mode works in a nutshell:

1. The client connects from a random port to port 21 on the server and issues the PASV command. The server replies, indicating which (random) port it has opened for data transfer.
2. The client connects from another random port to the random port specified in the server's response. Once connection is established, data transfers are made through these client and server ports.

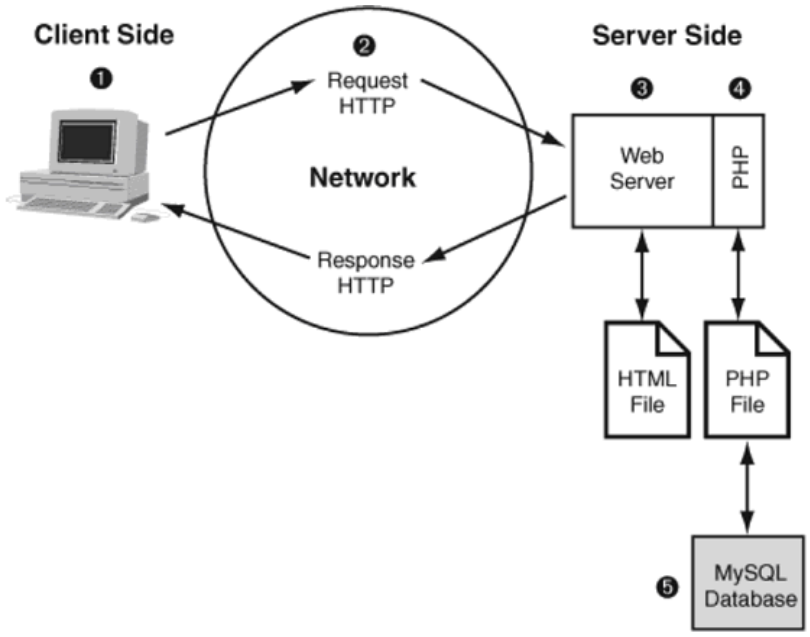


Hypertext Transfer Protocol

Short for Hypertext Transfer Protocol, HTTP is a set of standards that allow users of the World Wide Web to exchange information found on web pages. 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 primary function of a **Web/ HTTP server** is to deliver web pages on the request of clients using the Hypertext Transfer Protocol (HTTP). This means delivery of HTML documents and any additional content that may be included by a document, such as images, style sheets and scripts.

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

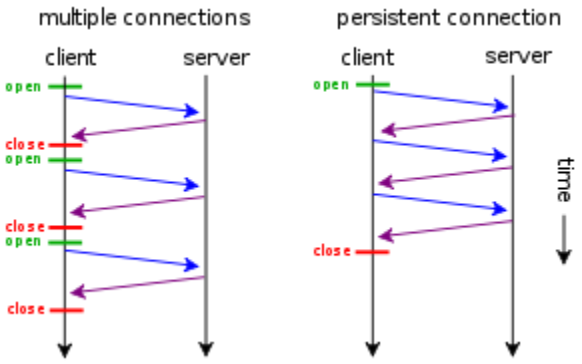


There are three important things about HTTP of which you should be aware:

- **HTTP is connectionless:** After a request is made, the client disconnects from the server and waits for a response. The server must re-establish the connection after it process the request.
- **HTTP is media independent**: Any type of data can be sent by HTTP as long as both the client and server know how to handle the data content. How content is handled is determined by the MIME specification.
- **HTTP is stateless**: This is a direct result of HTTP's being connectionless. The server and client are aware of each other only during a request. Afterwards, each forgets the other. For this reason neither the client nor the browser can retain information between different requests across the web pages.

Non Persistent and Persistent HTTP Connection

The HTTP client initiates a TCP connection to the server on the HTTP defaultport number 80. The HTTP client sends HTTP request message to the server via its socket associated with the TCP connection. The request message includes the path name.The HTTP server process receives the request message via its socket associated with the connections and retrieves the object from its storage, encapsulates the object in an HTTP response message and sends the response message to the client via its socket.



The HTTP server process tells TCP to close the TCP connection; however, the connectionis not actually terminated until the TCP knows for sure the client has received the responsemessage intact. The HTTP client receives the response message and the TCP connection terminates. The message indicates that the encapsulated object is an HTML file. The client extracts the file from the response message, examines the HTML file, and finds referencesto the objects.

Consider what happens when a user clicks a hyperlink. This causes the browser to initiate a TCP connection between the browser and the Web server; this involves a three-way “handshake”—the client sends a small TCP segment to the server, the server acknowledges and responds with a small TCP segment, and, finally, the client acknowledges back to the server. After the first two parts are completed, the client sends the HTTP request message combined with the third part of the three way handshake into the TCP connection.

Once the request message arrives at the server, the server sends the HTML file into the TCP connection back to the client

HTTP can use both **persistent and non-persistent connections**; although, persistent connections is the HTTP default mode. HTTP clients and servers can also be configured to use non persistent connections instead.

With **persistent connections**, the server leaves the TCP connection open after sending a response. Subsequent requests and responses between the same client and server can be sent over the same connections. An entire Web page or multiple Web pages residing on the same server can be sent from the server to the same client over a single persistent TCP connection. There are two versions of persistent connections: **without pipelining and with pipelining**. Without pipelining, the client issues a new request only when the previous response has been received.

Another disadvantage of **no pipelining** isthat after the server sends an object over the persistent TCP connection, the connection idleswhile it waits for another request to arrive. This idling wastes server resources.

The default mode of HTTP uses **persistent connections** with **pipelining**. The HTTP client issues a requestas soon as it encounters a reference, thus, the HTTP client can make back to back requests for the referenced objects; in other words, it can make a new request before receiving a response to a previous request. When the server receives the back to back requests, it sendsthe objects back to back

HTTP Methods

The GET Method

The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the? character.The data passed using the GET method would be visible to the user of the website in the browser address bar but when we pass the information using the POST method the data is not visible to the user directly.

http://www.test.com/index.htm?name1=value1&name2=value2

The POST Method

The POST request method is designed to request that a web server accept the data enclosed in the request message's body for processing. It is often used when uploading a file or submitting a completed web form.

GET vs. POST

The following table compares the two HTTP methods: GET and 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

HTTP Message Format

Like most network protocols, HTTP uses the client-server model: An HTTP client opens a connection and sends a request message to an HTTP server; the server then returns a response message, usually containing the resource that was requested. After delivering the response, the server closes the connection.

The life-cycle of an HTTP request commonly looks like this:

1. A user visits the URL of a website.
2. This creates a request which is routed to a web server via the internet (a network of DNS's, routers and switches) over HTTP (Hypertext Transfer Protocol).
3. The web server receives the HTTP request and responds to the user with the web page (or content) which was requested.

Hypertext Transfer Protocol (HTTP) uses Transmission Control Protocol (TCP) as the Transport Layer Protocol at Well Known port number 80. Once the TCP connection is established, the two steps in Hypertext Transfer Protocol (HTTP) communication are:

a. HTTP Request Message

The client initiates an HTTP session by opening a TCP connection to the HTTP server with which it wishes to communicate. It then sends request messages to the server, each of which specifies a particular type of action that the user of the HTTP client would like the server to take. Requests can be generated either by specific user action (such as clicking a hyperlink in a Web browser) or indirectly as a result of a prior action (such as a reference to an inline image in an HTML document leading to a request for that image.)

Request Message Structure:

The first line of a request (the message line) contains the HTTP method and target. For example, a message line for a GET request contains the keyword GET and a string that represents the object that is to be fetched, as shown in the following example:

GET /mysite/mydirectory/index.html HTTP/1.1\r\n

The rest of the request contains HTTP headers, including a required Host header and, if applicable, a message body.

The request ends with a bank line (an extra <CR><LF> or \r\n).

Following is an example of a request:

Get /mysite/index.html HTTP/1.1\r\n
Host: 10.101.101.10\r\n
Accept: */*\r\n
\r\n

b. HTTP Response Message

HTTP Response Codes indicate whether a specific HTTP requests has been successfully completed or not. The response header contains the date, size and type of file that the server is sending back to the client and also data about the server itself. The header is attached to the files being sent back to the client.

The HTTP response contains the requested resource (the HTML content in this case), as well as other information about the response. The first line is especially important and contains the HTTP response status code (200, 400, and 404). The status code communicates the overall outcome of the request back to the client. Was the request successful? Was there an error? Different status codes exist that indicate success, an error, or that the client needs to do something (e.g. redirect to another page).

Like the request, an HTTP response contains additional pieces of information known as HTTP headers. For example, one important HTTP response header is Content-Type. The body of the same resource could be returned in multiple different formats like HTML, XML, or JSON and the Content-Type header uses Internet Media Types like text/html to tell the client which format is being returned.

N-Tier Architecture

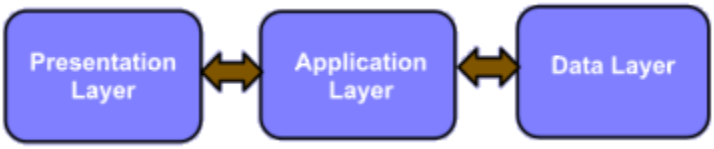
N-tier architecture is a client-server architecture in which, the presentation, the application processing and the data management are logically separate processes. N-Tier architecture is an industry-proved software architecture model, suitable to support enterprise-level client/server applications by resolving issues like scalability, security, fault tolerance and etc.

By breaking up an application into tiers, developers only have to modify or add a specific layer, rather than have to rewrite the entire application over, if they decide to change technologies or scale up.

Most of the applications used nowadays are 3- Tier application which is categorized as:

1. **Presentation layer**: a layer that users can access directly, such as desktop UI, web page and etc. Also called client.

2. **Application layer**: this layer encapsulates the business logic (such as business rules and data validation), domain concept, data access logic and etc. Also called middle layer.



3. **Data layer**: the external data source to store the application data, such as database server, CRM system, ERP system, mainframe or other legacy systems and etc. The one we meet often today is database server.

Advantages of N-Tier Architecture:

1. **Scalable**: Data tier can be scaled up by database clustering without other tiers involving. The web client side can be scaled up by load-balancer easily without affecting other tiers. Windows server can be clustered easily for load balancing and failover. In addition, business tier server can also be clustered to scale up the application, such as Weblogic cluster in J2EE.

2. **Better and finer security control to the whole system**: we can enforce the security differently for each tier if the security requirement is different for each tier. For example, business tier and data tier usually need higher security level than presentation tier does, then we can put these two high security tiers behind firewall for protection. 1 or 2 tiers architecture cannot fully achieve this purpose because of a limited number of tiers. Also, for N-Tier architecture, users cannot access business layer and data layer directly; all requests from users are routed by client

presenter layer to business layer, then to data layer. Therefore, client presenter layer also serves as a proxy-like layer for business layer, and business layer serves as a proxy-like layer for data layer. These proxies-like layers provide further protection for their layers below.

3. **Better fault tolerance ability**: for example, the databases in data layer can be clustered for failover or load balance purpose without affecting other layers.

4. **Friendly and efficient for development**: the decoupled layers are logic software component groups mainly by functionality; they are very software development friendly and efficient. Each layer can be assigned individually to a team who specializes in the specific functional area; a specialized team can handle the relevant task better and more efficiently.

5. **Friendly for maintenance**: N-Tier architecture groups different things together mainly by functionality and then makes things clear, easily understandable and manageable.

6. **Better reusability**: this is due to the logically grouped components and the loose couplings among layers. Loosely-coupled component groups are usually implemented in more general ways, so they can be reused by more other applications.