

Computer Networks
Prof. Sujoy Ghosh
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture No. # 40
HTTP

Good day! This is the last lecture and here we will discuss about the most widely used application layer protocol HTTP. This protocol is at the base of your world wide web.

(Refer Slide Time: 01:08 - 02:55)



HTTP stands for Hypertext Transfer Protocol. It is the network protocol used to deliver virtually all files and other data (collectively called resources) on the World Wide Web whether they are HTML files, image files, query results or anything else. Usually HTTP takes place through TCP/IP sockets. So the first thing is that the majority of resources available today on World Wide Web is of course some files which are usually HTML files. There may be images also. But also increasingly we have a lot of multimedia content on the web which may be audio files, video files etc. And a large number of applications are also being developed based on web because the web is vastly spread and many service providers are beginning to give their services through the web so that the service becomes available 24 hours a day anywhere in the world. So, apart from files there may also be an output of other programs which are also passed through this protocol namely the hyper text transfer protocol. So in this hypertext the word text actually came from the original content which was mostly textually based with hyperlinks but now-a-days a lot of it has changed.

(Refer Slide Time: 02:55 - 03:29)



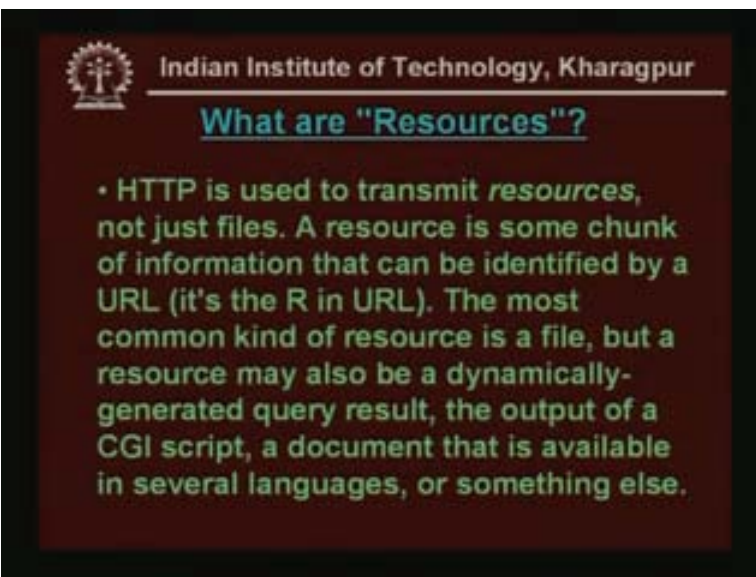
Indian Institute of Technology, Kharagpur

What is HTTP?

- A browser is an *HTTP client* because it sends requests to an *HTTP server* (Web server), which then sends responses back to the client. The standard (and default) port for HTTP servers to listen on is 80, though they can use any port.

So, this is the HTTP hyper text transfer protocol. The HTTP client is our browser therefore this internet explorer, netscape, Mozilla are all HTTP clients. It sends a request to an HTTP server also known as web server which then sends responses back to the client, the standard and default port for HTTP server to listen on is 80 though they can use any port. So, the well known port for HTTP is 80. So the first request which the client is requesting for in any HTTP server or any web server goes to the port number 80. Later on they may negotiate another port number.

(Refer Slide Time: 03:46 - 04:47)



Indian Institute of Technology, Kharagpur

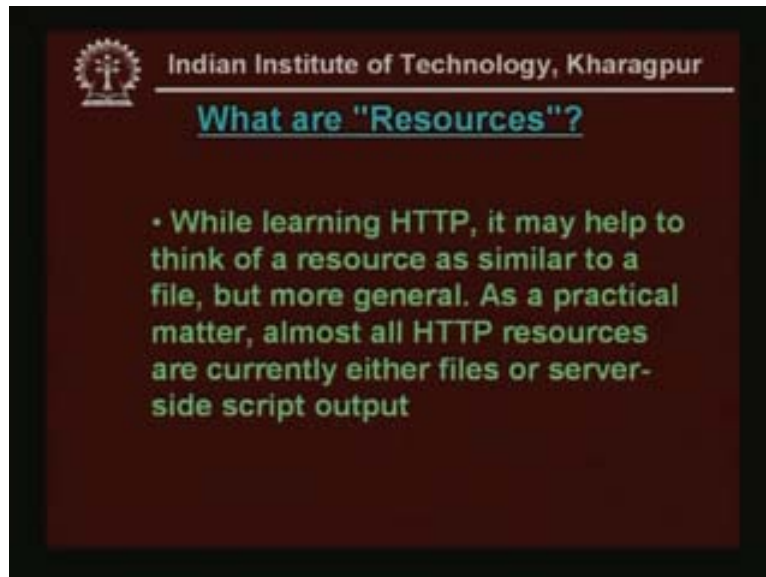
What are "Resources"?

- HTTP is used to transmit *resources*, not just files. A resource is some chunk of information that can be identified by a URL (it's the R in URL). The most common kind of resource is a file, but a resource may also be a dynamically-generated query result, the output of a CGI script, a document that is available in several languages, or something else.

HTTP is used to transmit resources not just files, that is another point because require we are talking about resources. A resource is some chunk of information that can be identified by a

URL. This is R in URL and URL is Universal Resource Locator, it is some kind of an address for these resources which is universally valid. Any resource which has an URL can be accessed through the web and of course it has to be served by the web server. The most common kind of resource is a file. But a resource may also be a dynamically generated query result, the output of a CGI script, a document that is available in several languages or something else. Now-a-days we have dynamic web pages or a web page may change and somehow they are sort of outputs of some programs and they are also resources or rather static files.

(Refer Slide Time: 04:48 - 05:11)



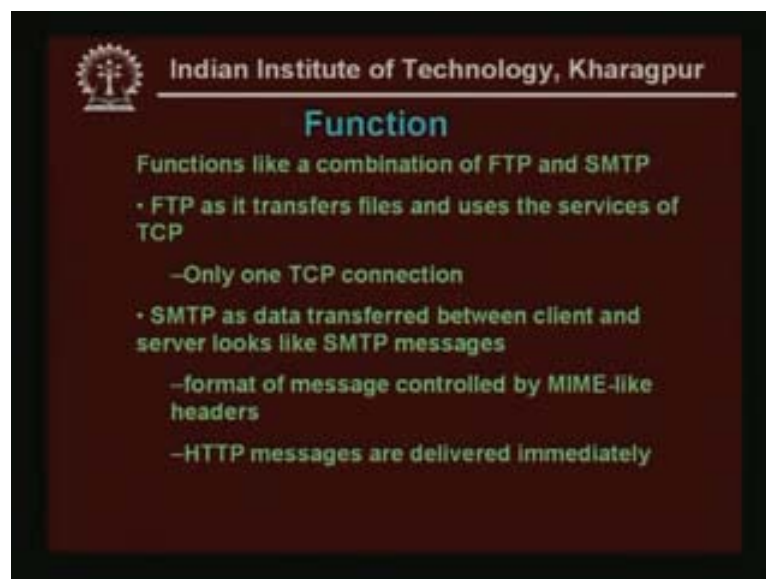
While learning HTTP it may help to think of a resource similar to a file, but more general. As a practical matter, almost all HTTP resources are currently either files or sever side script output that means some script running on the server. What you get to see when you surfing web, sometimes some of the effects may also be the output of programs which have been downloaded along with the page which is running on your machine that means which is running on the client machine through a java, applets etc. But anyway that program actually is coming from the server.

(Refer Slide Time: 05:36 - 06:42)



Used to access data on the www plain text, hyper text, and hyper text means text with hyperlinks that means links to other pages embedded in them. Actually these hyperlinks are so called links to other pages, as soon as you click on a link what happens is that a request is automatically sent to a particular page which the URL is pointing. Therefore it may be a text, it may be an image which can be clicked and whatever be the manifestation of this hyper link there is a URL embedded in it and when you click on it a request is automatically generated. When such hyperlinks are embedded in a text it is called a hypertext. Apart from plain text and hyper text we have audio, video or other things.

(Refer Slide Time: 06:43 - 07:32)



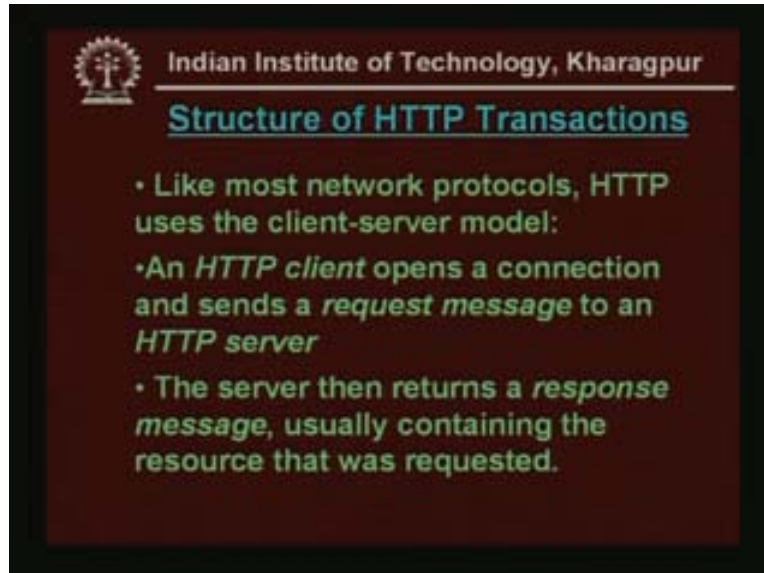
HTTP functions like a combination of FTP and SMTP in some way. FTP as it transfers files and uses the services of TCP. Just like FTP this is also using TCP and has only one TCP connection unlike FTP which uses two connections; one for control and one for data but here only one connection is used. And it is like SMTP. As data transfers between client and server it looks like SMTP messages. So they have format of message controlled by MIME like headers. HTTP messages are delivered immediately unlike mail, unlike SMTP. Like FTP, in HTTP the messages are also delivered immediately.

(Refer Slide Time: 07:32 - 07:49)



HTTP is actually very simple. Client sends a request and server sends a response, similar to mail request and reply, data in the form of a letter with MIME like format.

(Refer Slide Time: 07:50 - 08:06)



Indian Institute of Technology, Kharagpur

Structure of HTTP Transactions

- 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.

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.

(Refer Slide Time: 08:07 - 08:25)



Indian Institute of Technology, Kharagpur

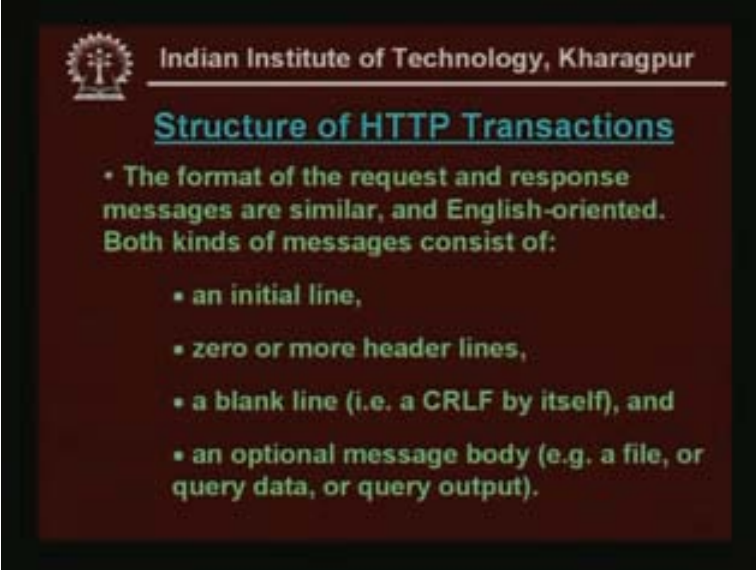
Stateless Protocol


- After delivering the response, the server closes the connection (making HTTP a *stateless* protocol, i.e. not maintaining any connection information between transactions).
- Cookies are maintained in the client machine

After delivering the response, the server closes the connection making HTTP a stateless protocol. That is not maintaining any connection information between transactions. This is one important point to remember that basic HTTP is a stateless protocol. A request has come, it responds and forgets and closes the connection and it forgets everything about it immediately. Sometimes some web servers would really like to keep some information and there are various things like

access logs etc which may be maintain on the server side. Another common way in which some state information is kept are through cookies. So, cookies are a simple text file kind of a thing which is stored at the client's machine. What the server does is, it writes something about this clients request in a text file in the client machine itself so that next time if the that same client connects again then the HTTP server can leave that file and can find out what is the last visit or some information about it. Cookies too sometimes breaks down the privacy in some sense and somebody else is writing things in my machine so that may be a source of breach of privacy or security so some people would like to block the cookies also. It is also possible to block the cookies. But anyway the point is, basic HTTP protocol is stateless. That means when one a request comes and once it is serviced the machine forgets about it.

(Refer Slide Time: 10:19 - 11:06)



 Indian Institute of Technology, Kharagpur

Structure of HTTP Transactions

- The format of the request and response messages are similar, and English-oriented. Both kinds of messages consist of:
 - an initial line,
 - zero or more header lines,
 - a blank line (i.e. a CRLF by itself), and
 - an optional message body (e.g. a file, or query data, or query output).

Now, let us look more into the details of the structure of HTTP transactions. The formats of the request and response messages are similar and English oriented. Both kinds of messages consist of an initial line, zero or more header lines, a blank line that is a CRLF Carriage Return and Line feed by itself and an optional message body for example, a file or query data or query output etc so this is the structure. **We will see what each of these areas really contains.** The blank line is just to delineate between the header and the message body.

(Refer Slide Time: 11:07 - 11:28)

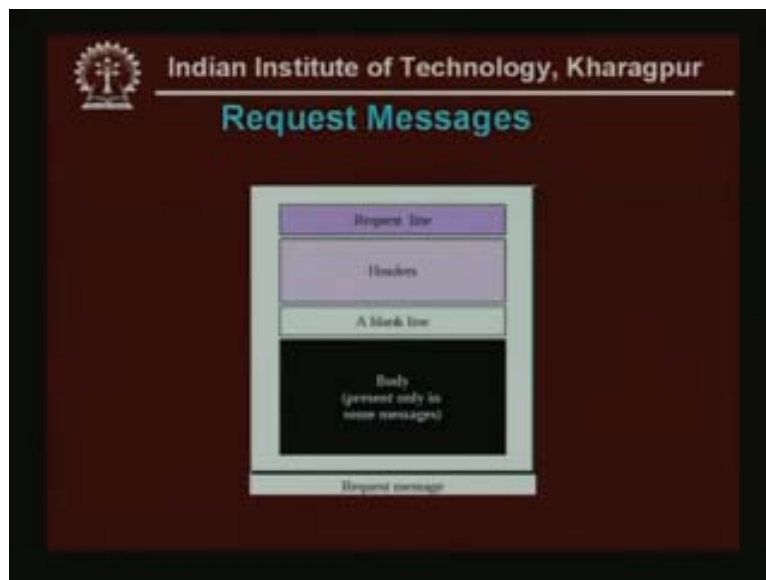
Indian Institute of Technology, Kharagpur

Structure of HTTP Transactions

- Put another way, the format of an HTTP message is:
- <initial line, different for request vs. response>
- Header1: value1
- Header2: value2
- Header3: value3

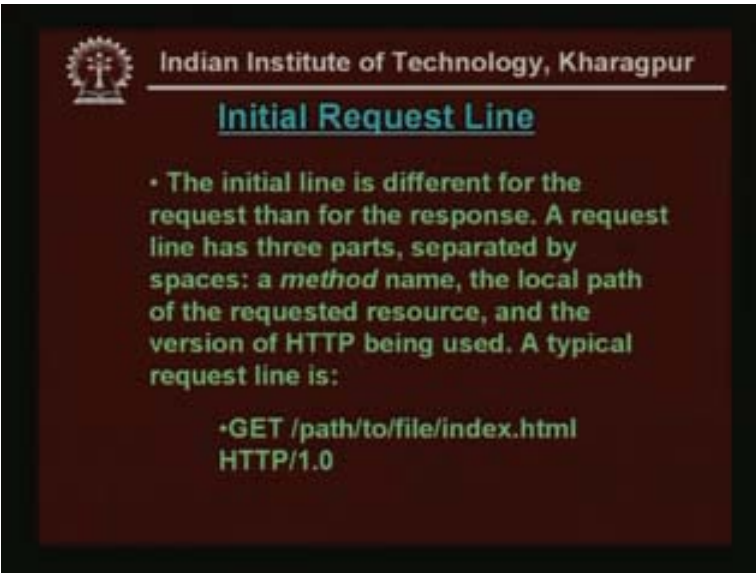
Put another way, the format of an HTTP message is initial line, different for request versus response, then header1: value1, header2: value2 so the header name: value of that header, and that is how the headers go.

(Refer Slide Time: 11:28 - 11:35)



Then you have the blank line and then the body. So you have initial line, headers, blank line and the body.

(Refer Slide Time: 11:36 - 12:52)



Indian Institute of Technology, Kharagpur

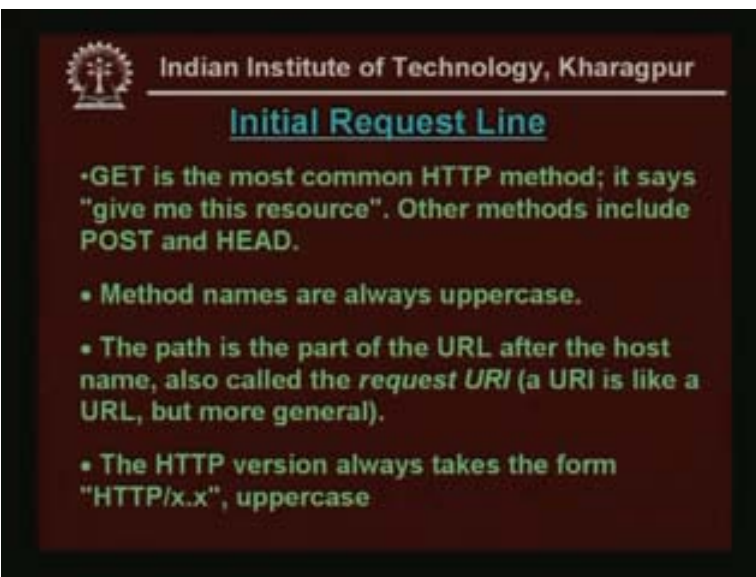
Initial Request Line

- The initial line is different for the request than for the response. A request line has three parts, separated by spaces: a *method* name, the local path of the requested resource, and the version of HTTP being used. A typical request line is:

•GET /path/to/file/index.html
HTTP/1.0

Initial Request Line: This is different for the request than for the response. A request line has three parts separated by spaces: a method name, the local path of the requested resource and the version of HTTP being used. A typical request line is Get /path/to/file/index.html. So this is an example, this is the path name, path name to this index dot HTML file. So, starting from the route it goes down the directory and subdirectory etc to finally locate the resource which is now here in index. HTML so this is really a request line. So GET is a method, we will come to that later on. And finally it mentions the version of HTTP that is being used, HTTP 1.0. This is what is going to go from the client to the server.

(Refer Slide Time: 12:53 - 13:36)



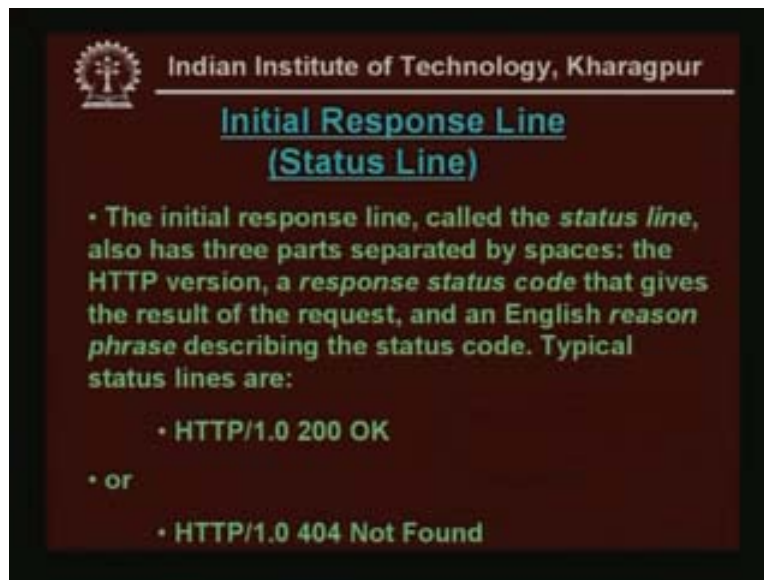
Indian Institute of Technology, Kharagpur

Initial Request Line

- GET is the most common HTTP method; it says "give me this resource". Other methods include POST and HEAD.
- Method names are always uppercase.
- The path is the part of the URL after the host name, also called the *request URI* (a URI is like a URL, but more general).
- The HTTP version always takes the form "HTTP/x.x", uppercase

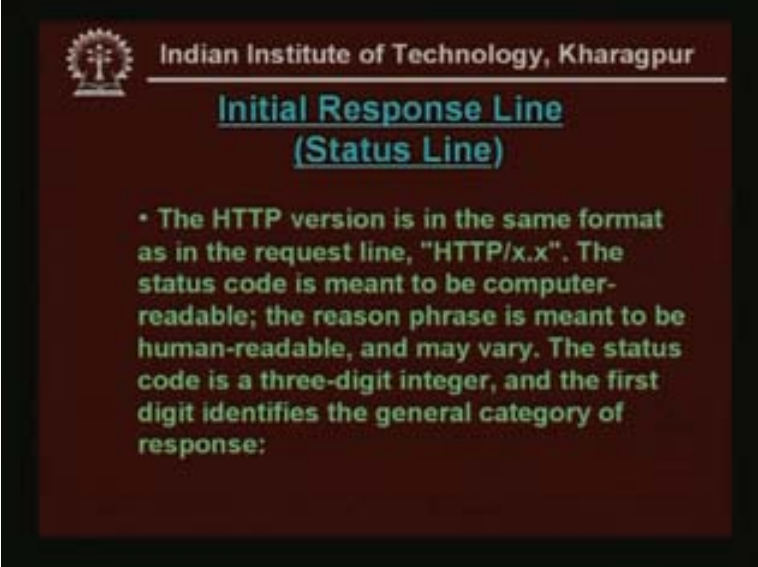
GET is the most common HTTP method. It says “give me this resource”. We will go into little bit more into the details of GET and other methods like, POST and HEAD etc later on. So method names are always upper case. The path is the part of the URL after the host name also called the request URI. So a URI is like a URL but more general. And HTTP version always takes the form HTTP/x.x a 1.0 etc. What we are trying to get is some file located in some site that is in some web server. Some how you should be able to contact that particular machine, some how we have to indicate the machine then we have to say what is it we are doing, what kind of protocol we are using. And finally in that machine we have to give a path name to that file. That path name is necessary because there may be files with the same name in different parts of the directory so always the path name is necessary. Therefore all these together constitute the URL.

(Refer Slide Time: 14:27 - 15:42)



Initial Response line (Status Line): Initial response line, we have seen initial request line now, we see the initial response line. The initial response line called the status line also has three parts separated by spaces; the HTTP version, a response status code that gives the result of the request and an English reason phrase describing the status code. Typical status lines are HTTP/1.0 200 OK that means a request has been received and its response is that it is OK, 200 is the code for it and it is OK. That means whatever has been requested would also be forthcoming or HTTP/1.0 404 Not found. This means, the particular file we were looking for in the request may not be available and therefore it says file not found. Now, what your client or browser would do is that it will read this and then display it may be in a separate window or a separate page that the file has not been found or the source has not been found.

(Refer Slide Time: 15:43 - 16:03)



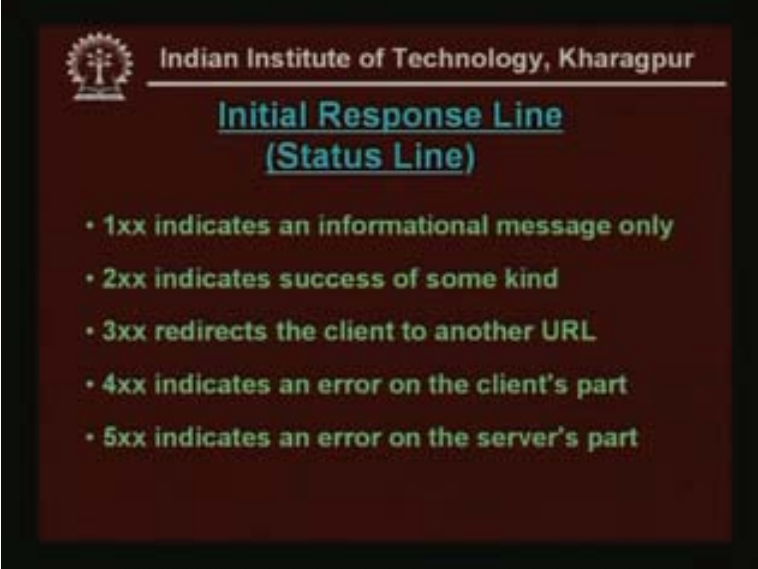
Indian Institute of Technology, Kharagpur

Initial Response Line (Status Line)

- The HTTP version is in the same format as in the request line, "HTTP/x.x". The status code is meant to be computer-readable; the reason phrase is meant to be human-readable, and may vary. The status code is a three-digit integer, and the first digit identifies the general category of response:

The HTTP version is in the same format as in the request line that HTTP/x.x. The status code is meant to be computer readable. The reason phrase is meant to be human readable and may vary. The status code is a three-digit integer and the first digit identifies the general category of response.

(Refer Slide Time: 16:04 - 17:06)



Indian Institute of Technology, Kharagpur

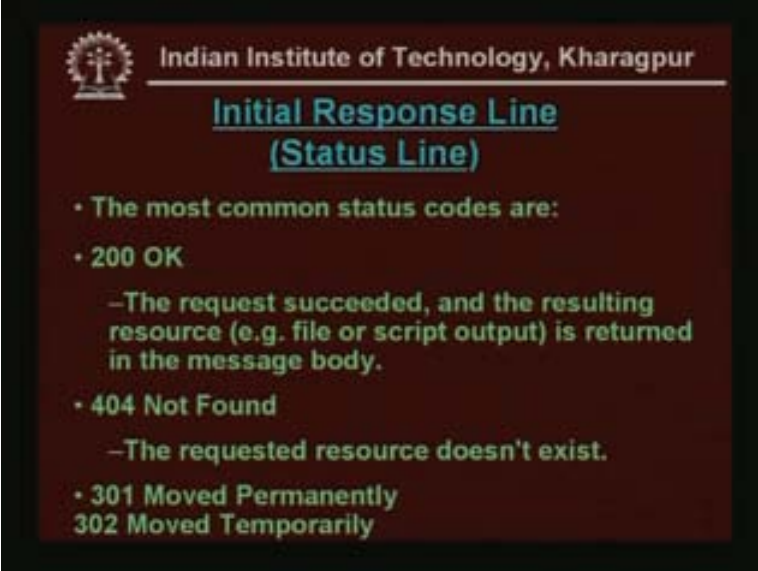
Initial Response Line (Status Line)


- 1xx indicates an informational message only
- 2xx indicates success of some kind
- 3xx redirects the client to another URL
- 4xx indicates an error on the client's part
- 5xx indicates an error on the server's part

1xx indicates an information message only, 2xx indicates success of some kind, 3xx redirects the client to another URL. Sometimes somebody may have moved from his old site to some new site may be to some new URL, so what he would do is that, he would leave a redirection message in the old site, so those clients who have the old URL land up there unknowingly they are

automatically redirected to the new URL. So 3xx redirects the client to another URL, 4xx indicates an error on the clients part and 5xx indicates an error on the server's part. So, whatever resource name it has given it does not exist so that is 404. Therefore this is the status line.

(Refer Slide Time: 17:06 - 17:29)



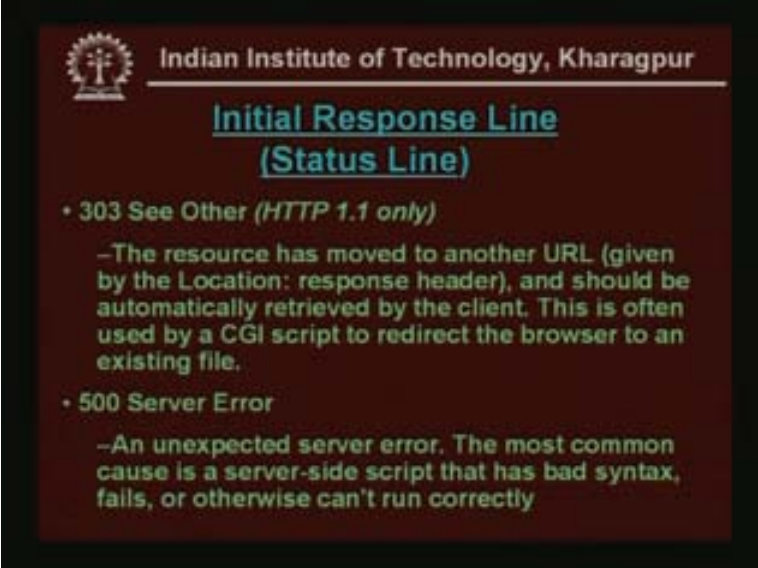
 Indian Institute of Technology, Kharagpur


Initial Response Line
(Status Line)

- The most common status codes are:
- **200 OK**
 - The request succeeded, and the resulting resource (e.g. file or script output) is returned in the message body.
- **404 Not Found**
 - The requested resource doesn't exist.
- **301 Moved Permanently**
302 Moved Temporarily

The most common status codes are of course 200 OK and 404 not found. OK means the request is succeeded and the resulting resource is returned in the message body. And 404 means not found, 301 that it has moved permanently, 302 means it has moved temporarily and so on.

(Refer Slide Time: 17:30 - 18:00)



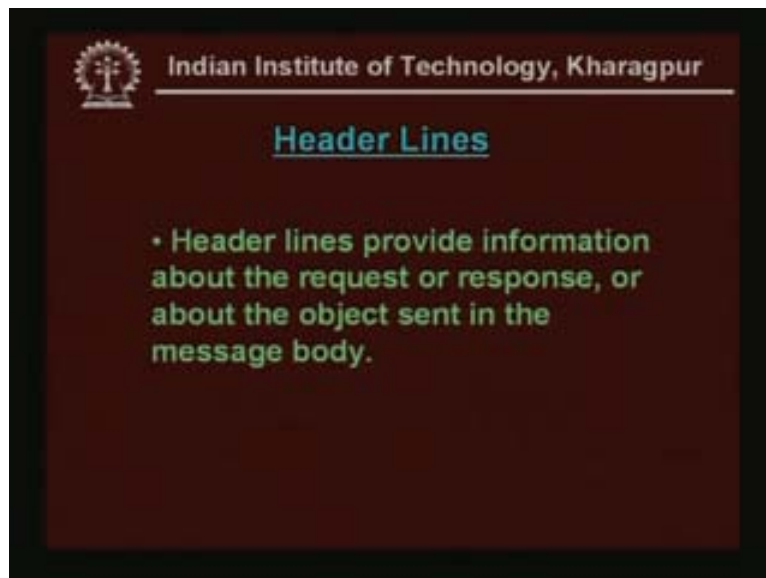
 Indian Institute of Technology, Kharagpur

Initial Response Line
(Status Line)

- **303 See Other (HTTP 1.1 only)**
 - The resource has moved to another URL (given by the Location: response header), and should be automatically retrieved by the client. This is often used by a CGI script to redirect the browser to an existing file.
- **500 Server Error**
 - An unexpected server error. The most common cause is a server-side script that has bad syntax, fails, or otherwise can't run correctly

So 303 means the other the resources moved to another URL given by the location response header and should be automatically retrieved by the client. This is often used by a CGI script to redirect the browser to an existing file. The 500 is the server error. It is an unexpected server error, the most common cause is a server side script that has a bad syntax or fails or otherwise cannot run correctly.

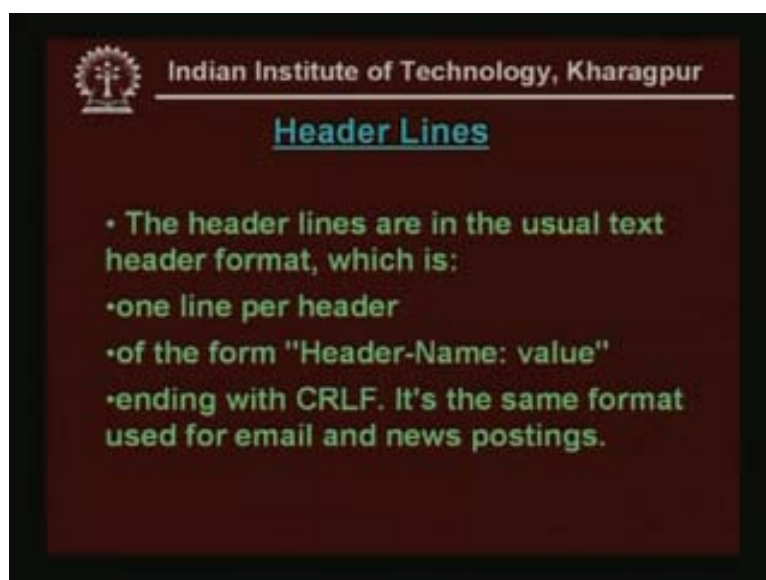
(Refer Slide Time: 18:01 - 18:14)



Now, after the initial line there are the header lines.

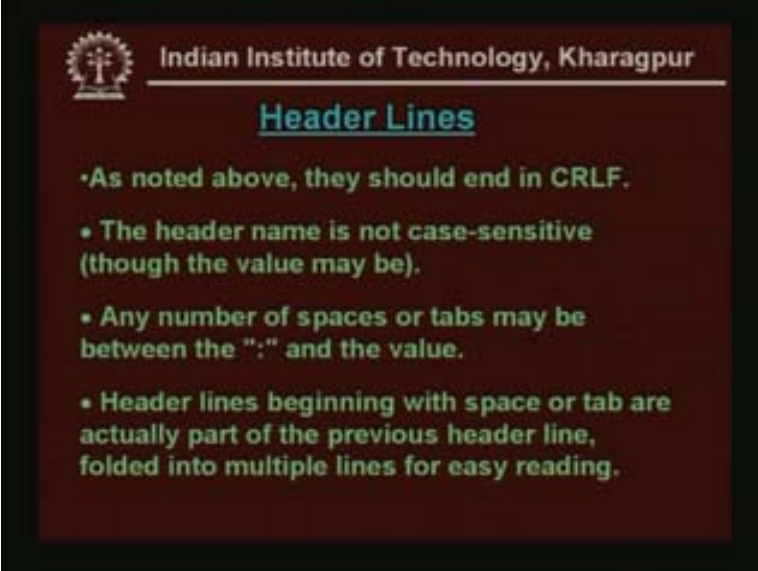
Header Lines: Header lines provide information about the request or response or about the object sent in the message body.

(Refer Slide Time: 18:15 - 18:32)



The header lines are in the usual text header format which is one line per header, of the form “Header - Name: value”, ending with a CRLF. It is the same format used for email and news postings.

(Refer Slide Time: 18:33 - 18:53)



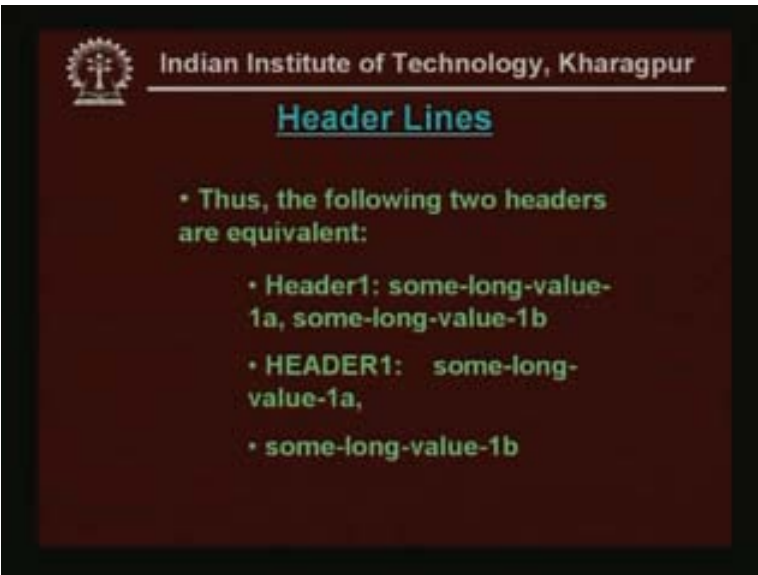
Indian Institute of Technology, Kharagpur

Header Lines

- As noted above, they should end in CRLF.
- The header name is not case-sensitive (though the value may be).
- Any number of spaces or tabs may be between the ":" and the value.
- Header lines beginning with space or tab are actually part of the previous header line, folded into multiple lines for easy reading.

As noted above they should end in CRLF. The header name is not case sensitive though the value may be. Any number of spaces or tabs may be between the colon and the value. Header lines beginning with space or tab are actually part of the previous header line folded into multiple lines for easy reading.

(Refer Slide Time: 18:54 - 19:14)



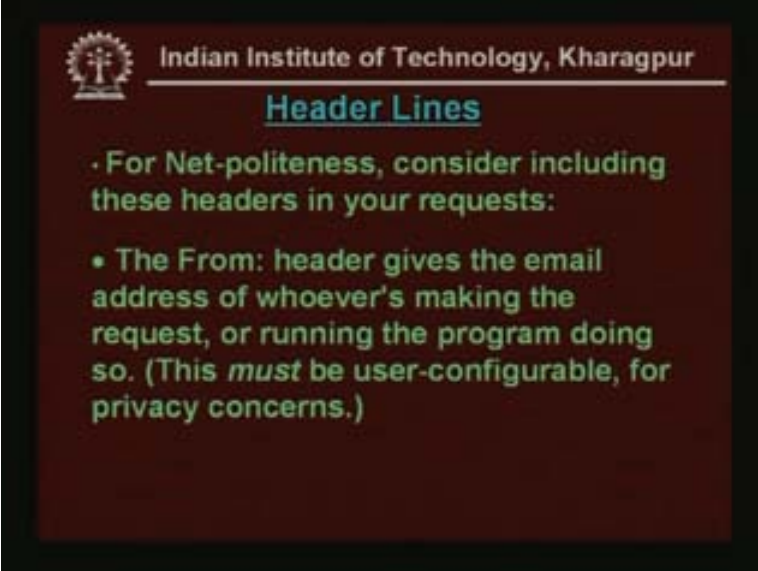
Indian Institute of Technology, Kharagpur

Header Lines

- Thus, the following two headers are equivalent:
 - Header1: some-long-value-1a, some-long-value-1b
 - HEADER1: some-long-value-1a,
some-long-value-1b

The following two header lines are actually equivalent. You may say header 1: some-long-value-1a some-long-value-1b or header1: some-long-value-1a, then some-long-value-1b, this is just folded.

(Refer Slide Time: 19:14 - 19:44)



The slide features the IIT Kharagpur logo and name at the top. The title 'Header Lines' is centered. Below it, a green bullet point states: 'For Net-politeness, consider including these headers in your requests:'. A second green bullet point explains the 'From' header: 'The From: header gives the email address of whoever's making the request, or running the program doing so. (This *must* be user-configurable, for privacy concerns.)'.

Indian Institute of Technology, Kharagpur

Header Lines

- For Net-politeness, consider including these headers in your requests:
- The From: header gives the email address of whoever's making the request, or running the program doing so. (This *must* be user-configurable, for privacy concerns.)

And for net-politeness when you send in a request, the 'from' part should be included. That means header gives the email address of whoever is making the request. You may or may not because this might lead to a lot of unsolicited mails coming to you also or running the program doing so. This must be user configurable for privacy concerns.

(Refer Slide Time: 19:45 - 20:16)



The slide features the IIT Kharagpur logo and name at the top. The title 'Header Lines' is centered. Below it, a green bullet point explains the 'User-Agent' header: 'The User-Agent: header identifies the program that's making the request, in the form "Program-name/x.xx", where x.xx is the (mostly) alphanumeric version of the program. For example, Netscape 3.0 sends the header "User-agent: Mozilla/3.0Gold".'

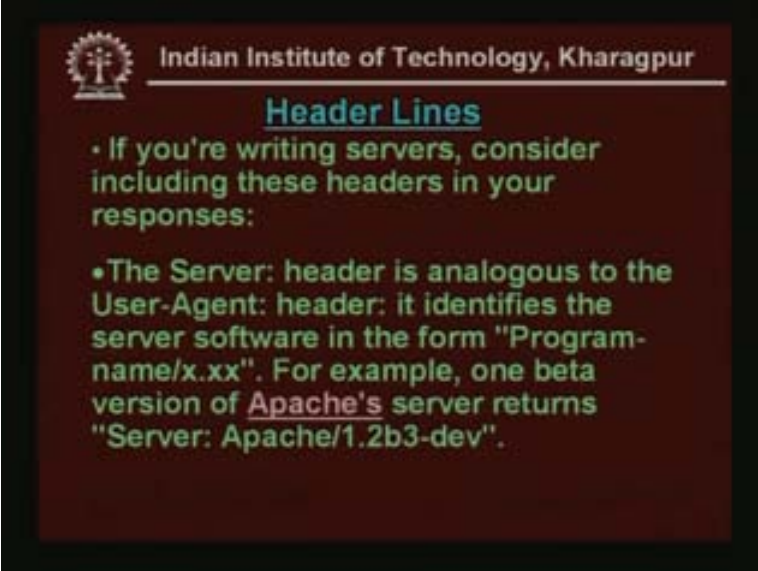
Indian Institute of Technology, Kharagpur

Header Lines

- The User-Agent: header identifies the program that's making the request, in the form "Program-name/x.xx", where x.xx is the (mostly) alphanumeric version of the program. For example, Netscape 3.0 sends the header "User-agent: Mozilla/3.0Gold".

The User Agent: header identifies the program that is making the request in the form program name/x.xx where x.xx is the mostly alphanumeric version of the program. For example, Netscape 3.0 send the header "User-agent: Mozilla/3.0 Gold". So that is the program version.

(Refer Slide Time: 20:16 - 20:37)



Indian Institute of Technology, Kharagpur

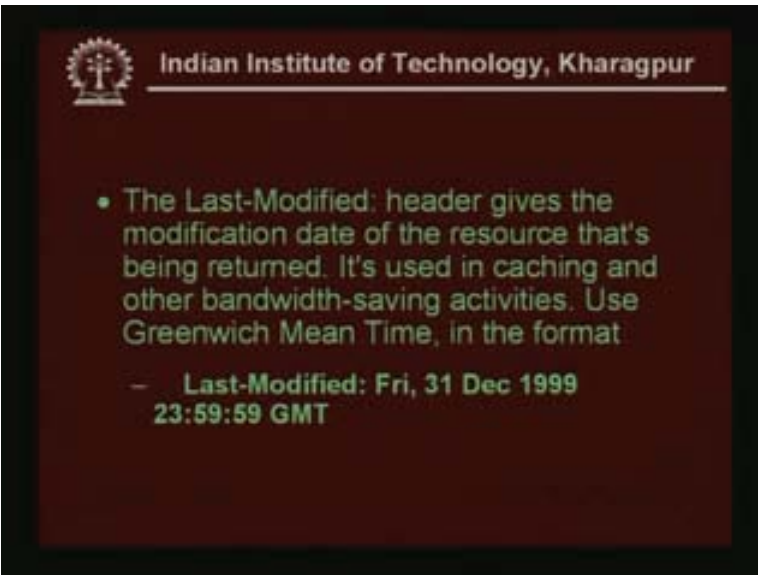
Header Lines

- If you're writing servers, consider including these headers in your responses:
- The Server: header is analogous to the User-Agent: header: it identifies the server software in the form "Program-name/x.xx". For example, one beta version of Apache's server returns "Server: Apache/1.2b3-dev".

If you are writing servers consider including these headers in your responses.

The server: the header is analogous to the user agent header; it identifies the server software in the form of program name. For example, one beta version of apache is server returns server apache such and such.

(Refer Slide Time: 20:38 - 22:05)



Indian Institute of Technology, Kharagpur

- The Last-Modified: header gives the modification date of the resource that's being returned. It's used in caching and other bandwidth-saving activities. Use Greenwich Mean Time, in the format
 - Last-Modified: Fri, 31 Dec 1999 23:59:59 GMT

The last modified: the header gives the modification date of the resource that is being returned. it is used in caching and other bandwidth-saving activities. Used GMT in the format last modified such as day, date and time and this last modified is important. The point is that, sometimes there are some web pages which are seen by a lot of people and if they are seen by a lot of people in the same organization one way to conserve bandwidth is through the user proxies. Once the page is brought in, that means it is put in that proxies cache automatically so that the next person who is under that same proxy makes the request and he gets it straight away from the cache without having to go to the distant location. But then this cache has to be updated regularly. If there is a modification on the other side, if the last modified value has changed then you should get a new copy of that, then you should refresh that page.

(Refer Slide Time: 22:06 - 22:13)



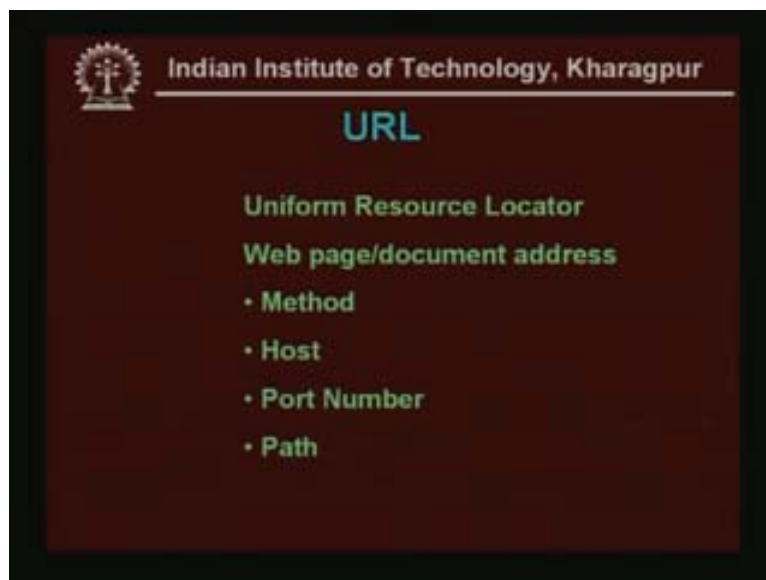
Request Line consists of the request type, URL and version.

(Refer Slide Time: 22:14 - 22:17)



Categorize request message into several methods.

(Refer Slide Time: 22:17 - 22:34)



URL: This is the uniform resource locator which is the webpage or document address. it indicates the method, the host, the port number and the path.

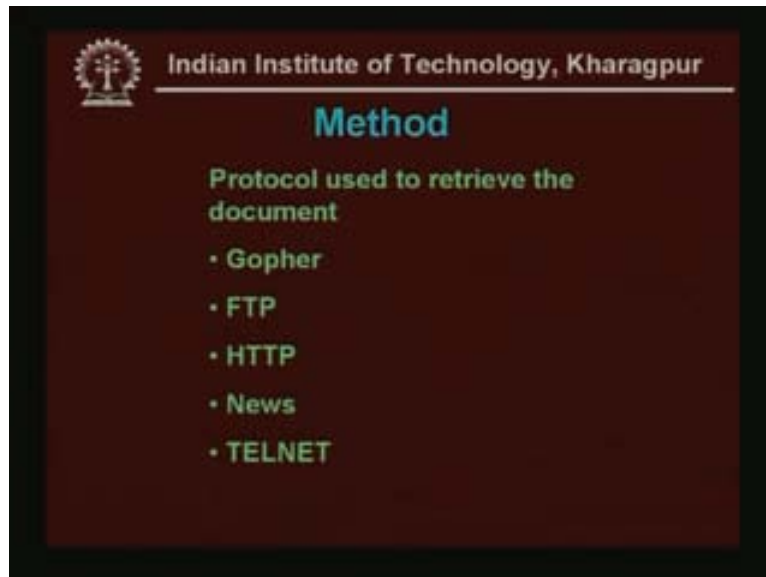
(Refer Slide Time: 22:35 - 25:24)



First is the method that is used and this method could be something like HTTP or even FTP etc, `://` then the host and for this host we usually like to give the domain name because it is much easier for us to remember the domain name. Secondly, domain name remaining the same, its underlying IP address may have changed. If you give the domain name what will happen is that first it will go through the resolver and get the IP address. The domain name can be given over here. Alternatively an IP address can also be given over here. If IP address is given then straight away the IP address is followed by a: followed by the port so that immediately becomes the socket. The IP address and the port number together form the socket so we give the method and the socket and when you have done up to this much the port for the initial request is port number 80. If you give the host IP address and the port number, you have given the socket which means that we have indicated or we have specified where to go, first to the domain name and subsequently to the IP address of the host.

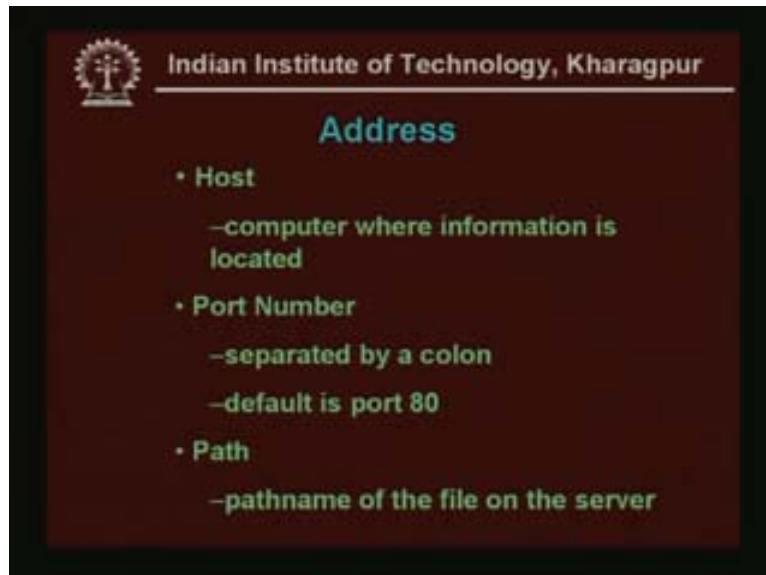
By the way this host must be on a public IP naturally and they cannot have any private IP if they have to be accessed over the internet, so the entire socket. Now we have reached the machine and we also know the method to use and then in that machine we have to locate the resource which is the path, slash means you start at the root then you go right down to the subdirectory where that particular file is present and then you give the name of the file, this is the entire URL. This way all the resources located in any of these machines connected to the internet having a public proper IP address can be uniquely specified. This is basically the addressing scheme for HTTP.

(Refer Slide Time: 25:25 - 25:40)



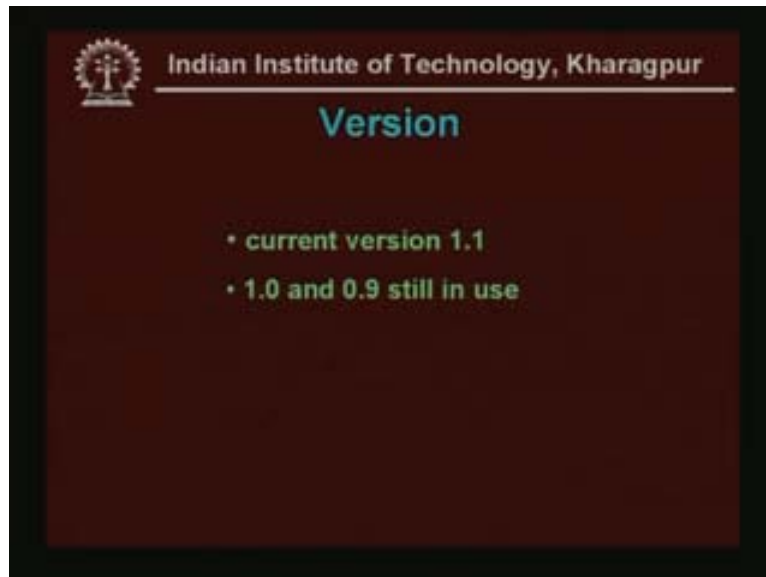
Protocol used to retrieve the document, as I said HTTP is the most common, you can also use FTP or Gopher or news or telnet. These could be the different methods used in the method part.

(Refer Slide Time: 25:41 - 25:57)



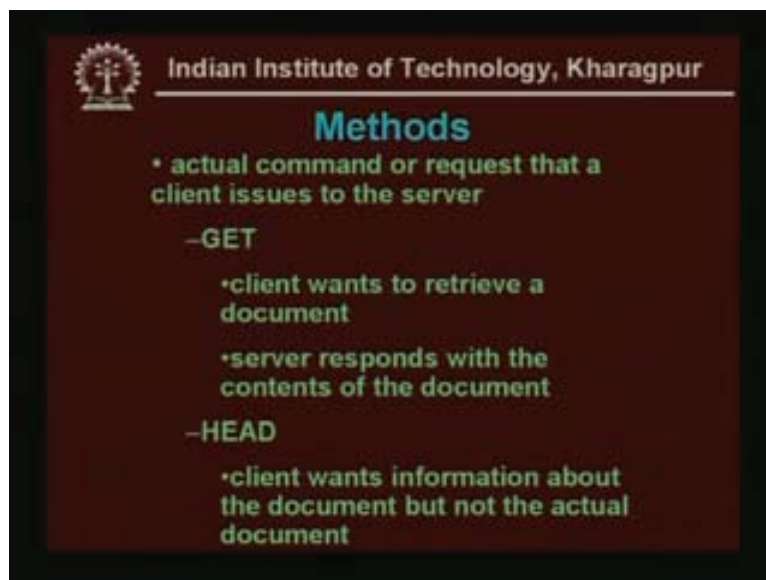
Host part is the computer where the information is located given by the domain name or may be the IP address directly. Port number is separated by the colon and as I said the default port is 80 and path is path name of the file on the server.

(Refer Slide Time: 25:58 – 26:04)



Version: current version 1.1 and 1.0, 0.9 are still in use.

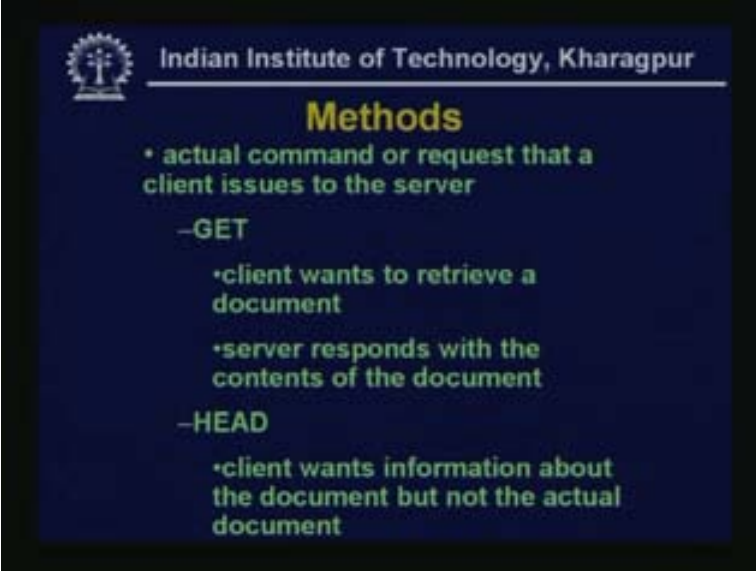
(Refer Slide Time: 26:05 - 26:20)



Now let us look into the methods in more detail. HTTP is an important protocol and moreover many applications based on web use these methods directly so that you can give some program output or can interact with your client through the web. Originally HTTP was not envisioned or envisaged just for locating some resource download it and see text and later on may be hypertext and some multimedia content. But increasingly the web had become so popular and so many people are hooked onto the web that people started giving services through the web. Naturally all

these services go through java script and a browser client etc so actually it is somehow communicating in HTTP at some level.

(Refer Slide Time: 27:28-27:49)



Indian Institute of Technology, Kharagpur

Methods

- actual command or request that a client issues to the server

–GET

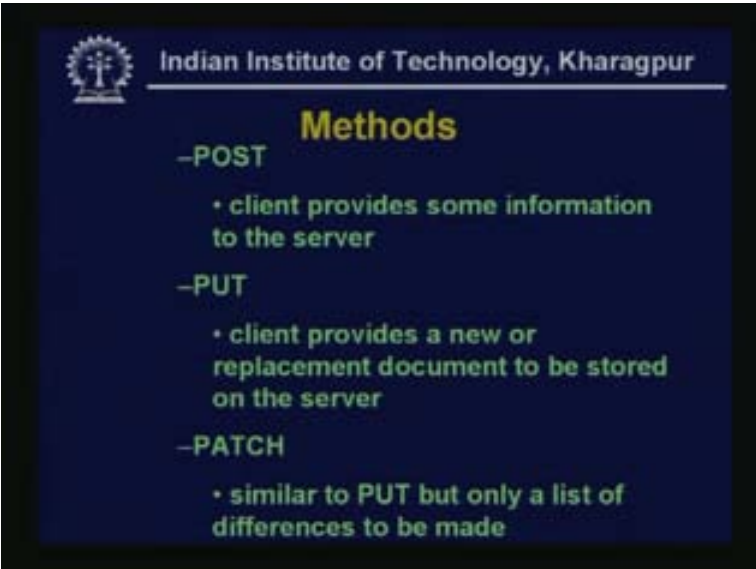
- client wants to retrieve a document
- server responds with the contents of the document

–HEAD

- client wants information about the document but not the actual document

Methods: The actual command or request that a client issues to the server. One is GET: client wants to retrieve the document, server responds with the contents of the document. HEAD: client wants information about the document but not the actual document so just the header part of it is what is needed.

(Refer Slide Time: 27:50 - 28:07)



Indian Institute of Technology, Kharagpur

Methods

–POST

- client provides some information to the server

–PUT

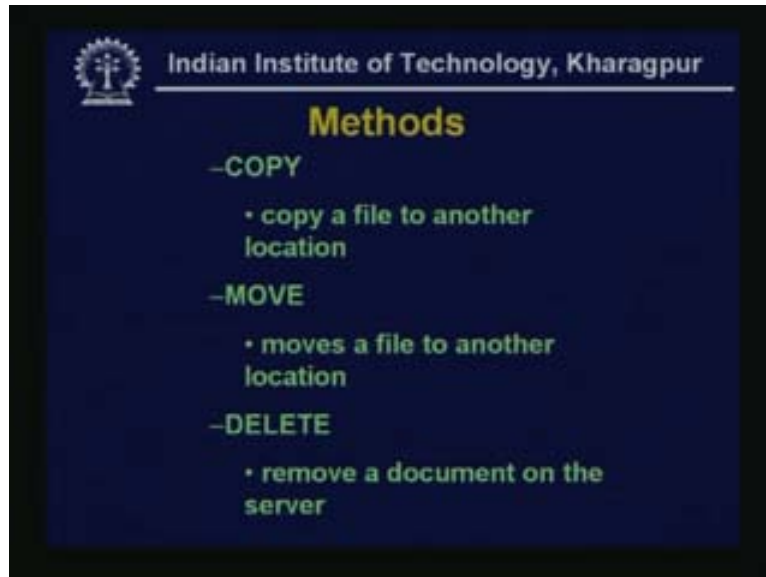
- client provides a new or replacement document to be stored on the server

–PATCH

- similar to PUT but only a list of differences to be made

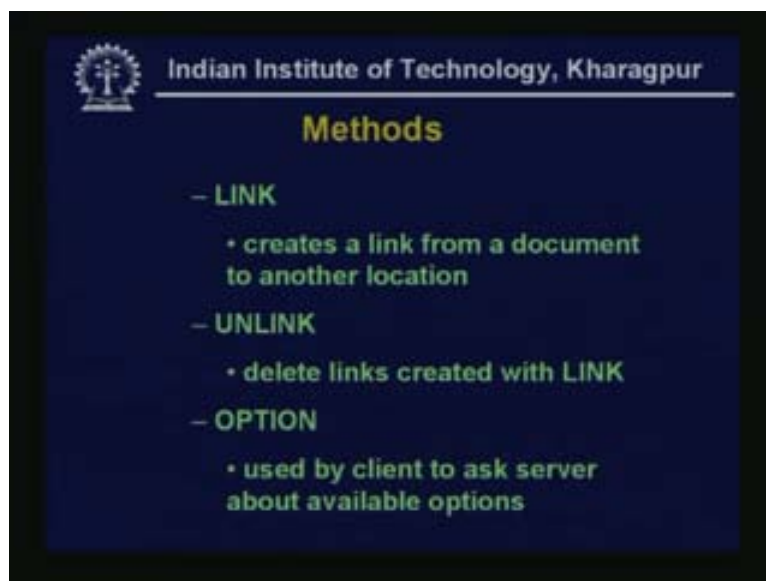
POST: client provides some information to the server. PUT: client provides a new or replacement document to be stored on the server. PATCH: it is similar to PUT but only a list of differences to be made rather than the actual new document.

(Refer Slide Time: 28:08 - 28:20)



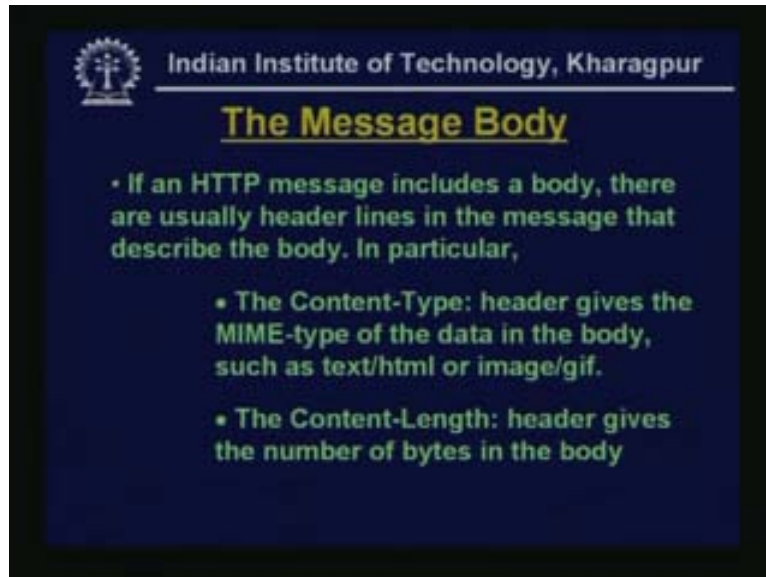
COPY: copy a file to another location. MOVE: moves a file to another location. DELETE: remove a document on the server.

(Refer Slide Time: 28:21- 28:37)



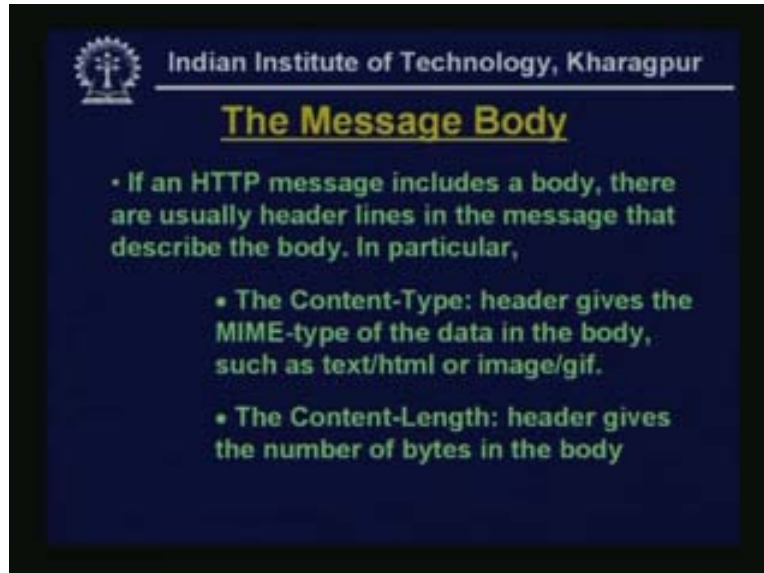
LINK: creates a link from a document to another location. UNLINK: delete links created with LINK. OPTION: used by client to ask server about available options. These are all the different methods.

(Refer Slide Time: 28:38 - 29:20)



The Message Body: An HTTP message may have a body of data sent after the header lines. In a response this is where the requested resource is returned to the client, the most common use of the message body. That means you have asked for a web page and that page is here, that content of that page is here or perhaps explanatory text if there is an error. In a request this is where user entered data or uploaded files are sent to the server. So this is the message body, the main content which is coming in the message.

(Refer Slide Time: 29:20 - 30:25)



Indian Institute of Technology, Kharagpur

The Message Body

- If an HTTP message includes a body, there are usually header lines in the message that describe the body. In particular,
 - The Content-Type: header gives the MIME-type of the data in the body, such as text/html or image/gif.
 - The Content-Length: header gives the number of bytes in the body

If an HTTP message includes a body there are usually header lines in the message that describes the body. In particular the content type, header gives the MIME type of the data in the body such as text/or HTML or image/gif. Suppose some image is being sent on request then on other side the client would know that what kind of resource it is, it is being told that this is an image and a gif type of image, so that on the client side that particular application or program to display gif files would be activated. The content length: header gives the number of bytes in the body.

(Refer Slide Time: 30:26 – 31:07)



Indian Institute of Technology, Kharagpur

Sample HTTP Exchange

- To retrieve the file at the URL
 - `http://www.somehost.com/path/file.html`
 - first open a socket to the host `www.somehost.com`, port 80 (use the default port of 80 because none is specified in the URL)

To retrieve the file at the URL we write something like `http://www.somehost.com` this is the domain name `path/` and of course since over here we will see that in this request the port number

has been left out so by default it is port number 80/path/file.html. First open a socket to the host www.somehost.com at port 80 and then get that file.

(Refer Slide Time: 31:08 – 31:52)



The slide is from the Indian Institute of Technology, Kharagpur. It features a dark blue background with a white logo on the top left. The title 'Sample HTTP Exchange' is in yellow. The text is in white and green. It describes a client request sent through a socket.

Indian Institute of Technology, Kharagpur

Sample HTTP Exchange

- Then, send something like the following through the socket:
 - GET /path/file.html HTTP/1.0
 - From: someuser@jmarshall.com
 - User-Agent: HTTPTool/1.0
 - [blank line here]

Then, send something like the following through the socket: GET/path/file.html HTTP/1.0, this is a request. You see the sequence; first you open a socket to the host which is the web server which is in this case some host. com at port 80, then you put another request the get request with the file name and the HTTP version. From some user from somewhere and user agent you mention and then you put the blank line, this is the header part.

(Refer Slide Time: 31:53 – 32:16)



The slide is from the Indian Institute of Technology, Kharagpur. It features a dark blue background with a white logo on the top left. The title 'Sample HTTP Exchange' is in yellow. The text is in white and green. It describes a server response sent back through the same socket.

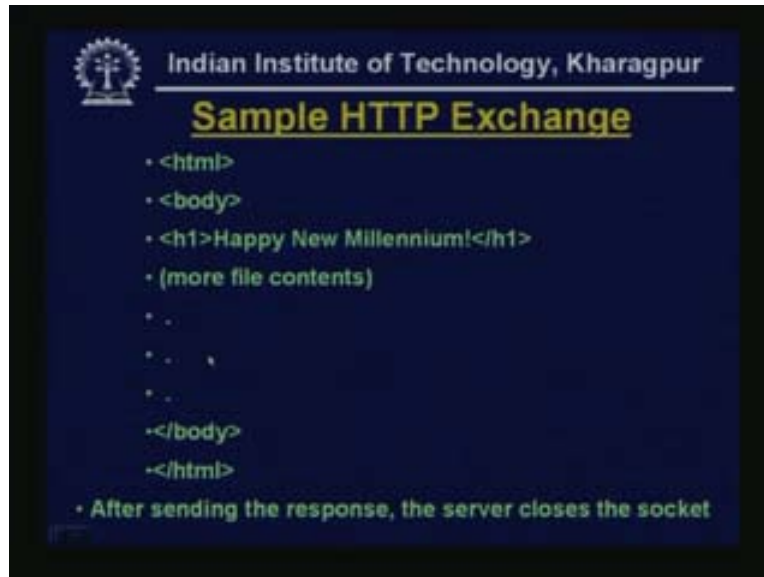
Indian Institute of Technology, Kharagpur

Sample HTTP Exchange

- The server should respond with something like the following, sent back through the same socket:
 - HTTP/1.0 200 OK
 - Date: Fri, 31 Dec 1999 23:59:59 GMT
 - Content-Type: text/html
 - Content-Length: 1354

The server should respond with something like the following: sent back through the same socket: HTTP / 1.0 200 OK that means the request is granted and then request is OK. Then the date that means when it was last modified, content type is text/HTML and content length is something.

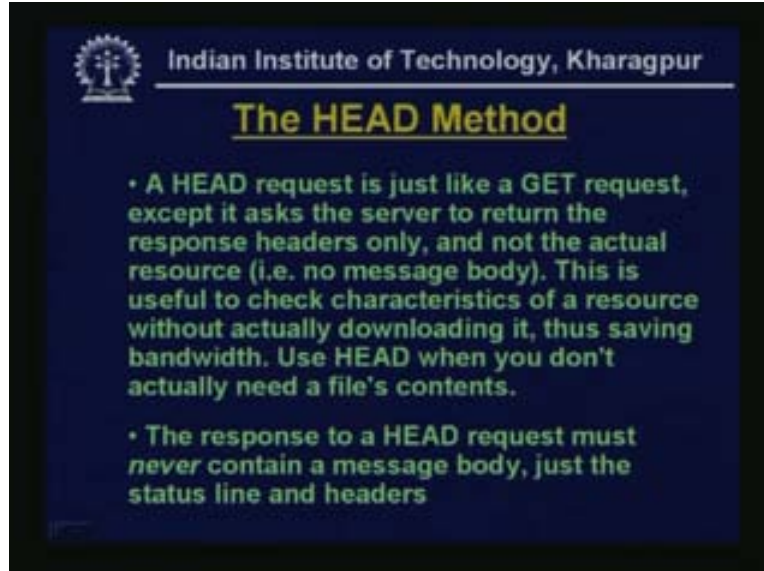
(Refer Slide Time: 32:17 – 34:41)



And HTTP in the body the body will contain may be in this particular case a text document which is in HTML. HTML stands for hypertext mark up language. Once again we will not have time to go into HTML what you can do is that you can open any web page and look at the source you will find that in your browser if you look at the top menu then there is a way of rather than looking at the final made up version you can look at the original source and there you can see the HTML file.

The HTML file has a number of tags and these tags help in specifying the format in which the text is to be displayed. Essentially there are other things but the major part of it is the format. There may be format, there may be tags like header sort of shown in bold, there may be tags to show the back ground color, the foreground color, the font colors, shape, size etc. All kinds of formatting information are there in HTML documents. Apart from that it is also possible to embed links. This is one example of what an HTML document might look like. First of all it is the tag HTML, whenever you have the tag you must have the beginning of the tag you must have the corresponding ending of the tag so HTML here and HTML here. Similarly body, then it says something happy new millennium or something which is the header, it says h1 which is a header then there may be more file contents and etc. After sending the response the server closes the socket. This is a stateless protocol and the server forgets about it. This is an example of what a simple text HTML document looks like.

(Refer Slide Time: 34:42 - 35:41)



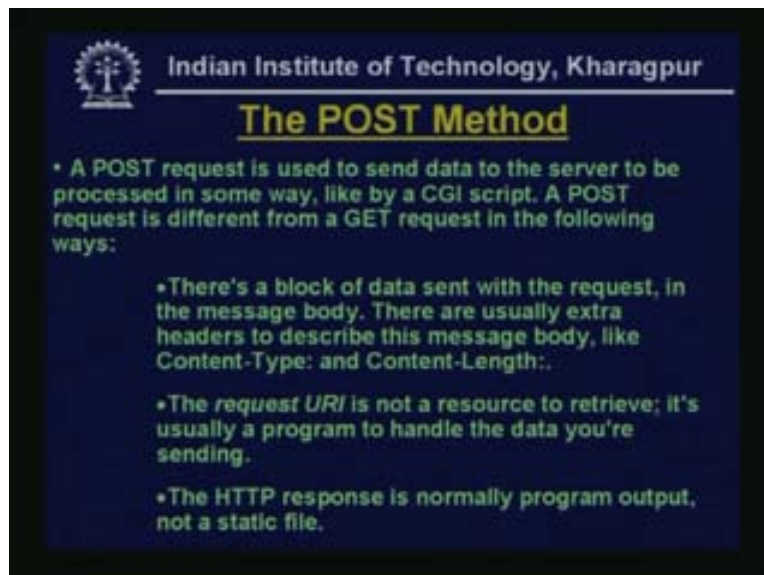
Indian Institute of Technology, Kharagpur

The HEAD Method

- A HEAD request is just like a GET request, except it asks the server to return the response headers only, and not the actual resource (i.e. no message body). This is useful to check characteristics of a resource without actually downloading it, thus saving bandwidth. Use HEAD when you don't actually need a file's contents.
- The response to a HEAD request must *never* contain a message body, just the status line and headers

Now let us look a bit more deeply into the methods. The Head Method: A HEAD request is just like a GET request except it asks the server to return the response headers only and not the actual resource, that is no message body. So this will work very fast. If you just want to know the header and then take some decision whether or not you want to look at it then you can ask just for the header and not the rest of it so it will come down very fast. This is useful to check characteristics of a resource without actually downloading it thus saving bandwidth and also a lot of time. Use head when you do not actually need files content. The response to a head request must never contain a message body, just the status line and header. That is the head method, just get the header.

(Refer Slide Time: 35:42- 36:23)



Indian Institute of Technology, Kharagpur

The POST Method

- A POST request is used to send data to the server to be processed in some way, like by a CGI script. A POST request is different from a GET request in the following ways:
 - There's a block of data sent with the request, in the message body. There are usually extra headers to describe this message body, like Content-Type: and Content-Length:.
 - The *request URI* is not a resource to retrieve; it's usually a program to handle the data you're sending.
 - The HTTP response is normally program output, not a static file.

The POST Method: A POST request is used to send data to the server to be processed in some way like by a CGI script. A POST request is different from a GET request in the following ways: there is a block of data sent with the request in the message body, there are usually extra headers to describe this message body like content type, content length etc., The request URI is not resource to retrieve, it is usually a program to handle the data you are sending. The HTTP response is normally program output, not a static file. This is quite important, for example, now-a-days for your electric utility or for your telephone, suppose the bill has not reached what you can do is that you can see your bill on the net. So what happens is, suppose you are a BSNL subscriber, the first thing you would like to do is to go to the BSNL's home page. If you know the address then it is very fine or otherwise you can use any searching in like Google or something to get the address of BSNL and then click on that address which is usually a link to the web page.

A link to the web page, meaning, that usually any website would contain an initial or main page or index page as it is called and if you simply make an HTTP request to the correct server the main page or the first page or the index page would be displayed first. Actually, that is an important point because while surfing we are accessing so many pages in so many machines. In the URL as we have seen, in that last part the located part you have to give the entire path name as well as the file name. Unless you are very well aware and very conversant with one particular sight or may be a few sites other than that you will not know either the path name or the file name, so how do you locate it and that is not how people serve. What they do is that may be they know that in this particular example we were giving so we are trying to find our own telephone bill which is missing so I want to see my bill. So I will connect to BSNL and I will somehow find the main address of BSNL which may be BSNL.co.in or something and that is where I will go.

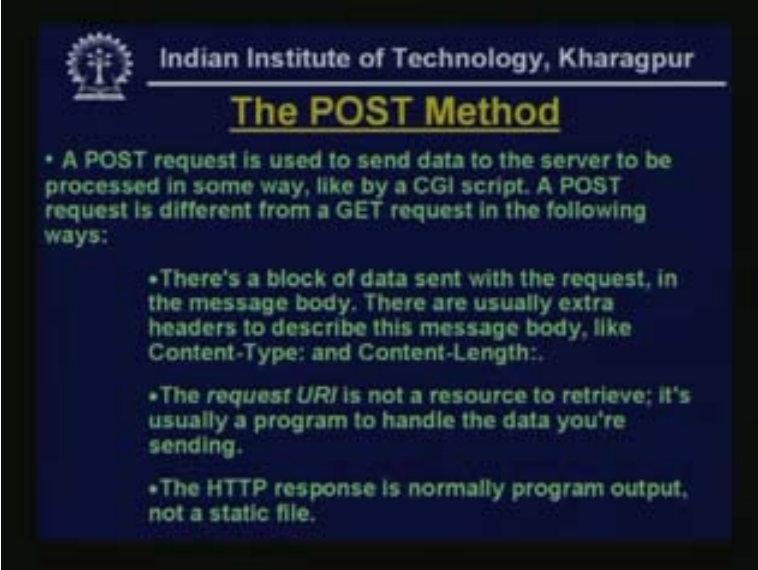
Now, I have just mentioned the domain name and nothing else which actually means a server, which actually means this domain name would stand for some web server. The domain name of course can be translated via the DNS system into the proper IP address. And since it is an HTTP protocol that is under default port is 80 so up to that much that means up to the socket part it is fixed. Now what about the rest? The rest is taken for granted that if you have not specified something you will definitely be given only that first page or index page or main page of that site. That comes as a response in the message body in the form of an HTML document and that gets displayed in that nice format usually by the browser on your stream.


Now of course, that is a main page it may contain several things amongst all that I find out that if I am looking for a duplicate bill or something where do I go? Therefore something would be written over there like duplicate bill where if you take your cursor on that you will see that the cursor has changed showing that the color has changed or something has got underlined somehow to indicate that this is actually a link when you click on that link. Now what does that contain? It says duplicate bill and that is how it is communicating to the user but underline this text, there is a hyperlink and this hyperlink contains the path name and the file name which will take you to the page where you can get information about duplicate bills.

Let us say, I click it, as soon as I click it, actually I generate a fresh request where the server socket etc is already known. And now this new path name is added on to it so this is a new

request. HTTP of course being a stateless protocol has shown you the first page and has forgotten that you actually exist. But then now you have made another request which now brings a second page into zero. Now in that page may be you will find that there is a form over there. Who would process the form on the other side, may be some CGI script or some other program you are not going into that part in the server, but before that whatever information you have may be your name, may be your telephone number, something you have to enter, there will be a form you enter that and then you submit. Actually submit sounds are very nice and user-friendly but then your machine which was the client so long now has to upload some information, the information which you have just typed in it has to upload that information on to the server. This uploading of information may be done via this post, this is the opposite of that.

(Refer Slide Time: 42:02- 42:11)

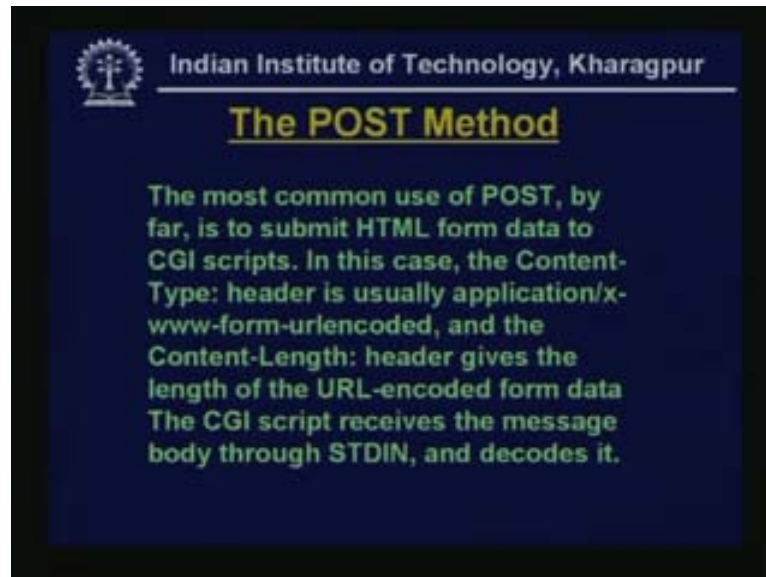


 Indian Institute of Technology, Kharagpur

The POST Method

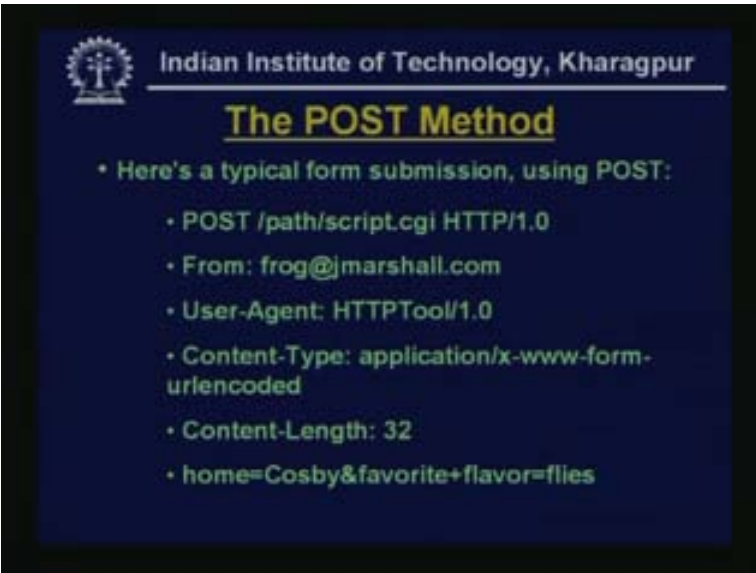
- A POST request is used to send data to the server to be processed in some way, like by a CGI script. A POST request is different from a GET request in the following ways:
 - There's a block of data sent with the request, in the message body. There are usually extra headers to describe this message body, like Content-Type: and Content-Length:.
 - The *request URI* is not a resource to retrieve; it's usually a program to handle the data you're sending.
 - The HTTP response is normally program output, not a static file.

(Refer Slide Time: 42:12 - 44:16)



The most common use of post by far is to submit HTML form data to CGI script. CGI stands for Common Gateway Interface so when you send something, this goes in some format and the CGI is a language where it is easy to pass whatever you have sent and then get the appropriate values. Therefore what CGI script will do is that, with all these information, it will form some kind of query may be a data base query to some database, in our example we were talking about telephone bills, so in that BSNL server where all the bills or may be at least the current bills are stored. So it will go with the current request information, form a query, get that kept your bill and then your bill was of course in a data base. So now that information has come, that has to be put together in the form of a dynamically generated web page and sent back. This is a very common way of using post. In this case, the content type: header is usually application/x-www-form-URL encoded. And content length header gives the length of the URL encoded form data, the CGI script receives the message body through STDIN and decodes it. CGI is specially designed so that such data which is coming in can be easily **passed** and all the values retrieved. This is how post method is used. So this is how applications over the internet are generally implemented.

(Refer Slide Time: 44:17 – 44:23)



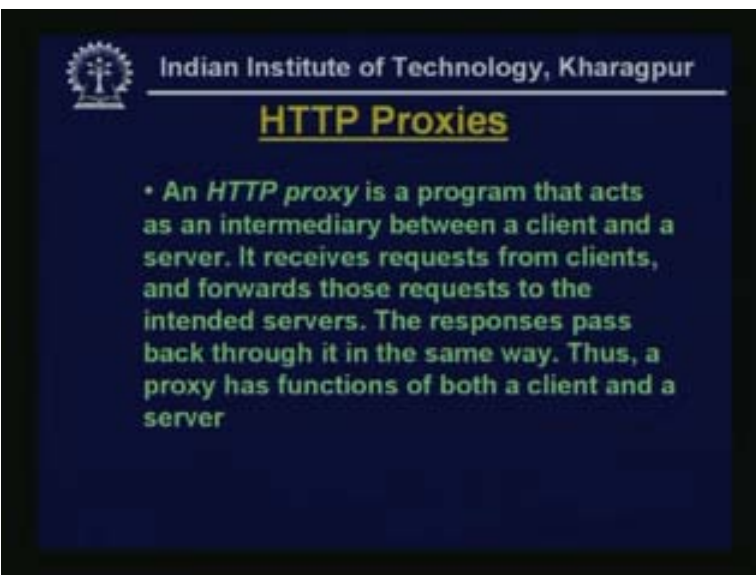
Indian Institute of Technology, Kharagpur

The POST Method

- Here's a typical form submission, using POST:
 - POST /path/script.cgi HTTP/1.0
 - From: frog@jmarshall.com
 - User-Agent: HTTPTool/1.0
 - Content-Type: application/x-www-form-urlencoded
 - Content-Length: 32
 - home=Cosby&favorite+flavor=flies

Here is a typical form submission using post, let say POST /path/script.cgi HTTP/1.0. So, basically we are actually invoking some CGI script and we have given that address of the URL of the script and we have mentioned the method and the HTTP version. From: is of course somewhere. User Agent: HTTP tool/1.0, Content-Type may be application/x www form URLencoded, Content Length: 32, Home: = something. So, this is a typical form submission, so, on the other side this will be taken and processed. And finally a dynamic web page would be generated and shown.

(Refer Slide Time: 45:24 – 46:57)



Indian Institute of Technology, Kharagpur

HTTP Proxies

- An *HTTP proxy* is a program that acts as an intermediary between a client and a server. It receives requests from clients, and forwards those requests to the intended servers. The responses pass back through it in the same way. Thus, a proxy has functions of both a client and a server

HTTP proxies: Are important for various reasons. One reason we have already discussed is that, suppose you are in an organization where a large number of people access the web outside, now-a-days almost everybody wants to access the web and sometimes the same page. Of course the WAN bandwidth is always costly and WAN bandwidth is always much lower than the internal band width. If some page is already been brought in and that page is fresh enough then you can cache it in the proxy, this is one major use of HTTP proxy, there are also other uses of HTTP proxy. An HTTP proxy is a program that acts as an intermediary between a client and a server. Instead of a client directly calling the server or directly sending a request to the server it may only send it to the HTTP proxy. It receives request from clients and forwards those requests to the intended servers. The responses passes back through it in the same way. Thus a proxy has functions of both a client and the server. It comes in between the client and the server.

(Refer Slide Time: 46:58 - 51:24)



When a client uses a proxy it typically sends all requests to the proxy instead of to the servers in the URLs. Requests to a proxy differ from normal requests in one way. In the first line they use the complete URL of the resource being requested instead of just the path, this is of course important. Actually if you are sending a request to the original server then the server has got the request that means you have reached up to that server. Now, you need to simply give the path from the root to the actual file name.

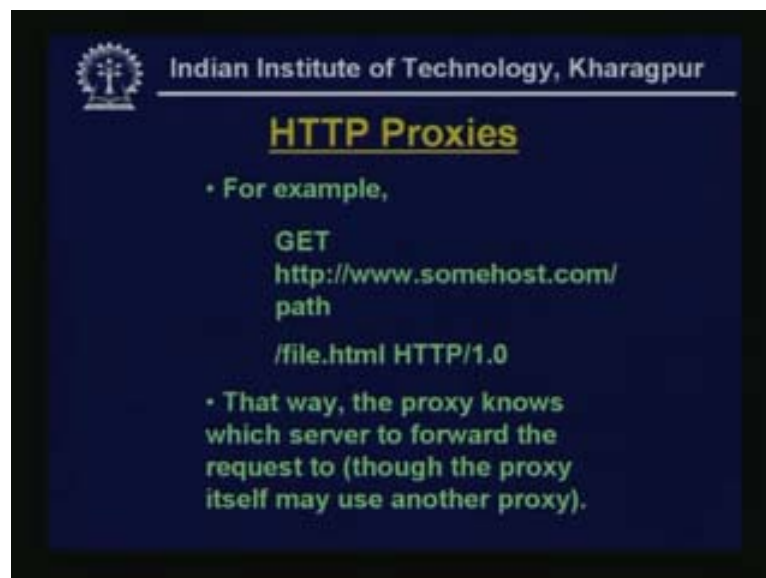
Now, instead of sending the request directly to the web server, the client, it sends its request to the local proximation. Therefore in the proximation it has to supply not only the path but also the domain name or IP address of the server that it is sending its request to. Apart from caching and bandwidth conservation there is another use sometimes that a proxy is a put to. You may do it in proxy, you may do it in some other places also like fire walls etc. In many organizations now-a-days what is happening is that the network is becoming so big but the IP address space is small. You cannot give a proper public IP address to all the machines in the network, may be the organization does not have such a big network for example, class B addresses have become very

few or almost non-existent these days. Therefore many organizations will not have a class B address. If that is so you cannot give proper IP address.

Now, if you cannot give the proper IP address to the client, may be you can still send the request to the server but how will the servers respond back? For responding back the server has to communicate with you, it has to open the TCP connection. For that on the other end there must be a machine with the valid IP address. One way to solve this problem is to do network address translation and although network address translation may be done at the fire wall, may be done at the router, it may be also be done at the proxy. that means what we do is that, if it is a big organization may be 5000 machines or something and has got just one class C address may be 1250 addresses. What it might do is that, it might allot some number say 50 valid IP addresses to the proxy so that whenever there is a request which is being sent by the client. Now the client need not be given a public IP.

If the client may be given a private IP which is just private to that network that does not work outside the boundary of the organization, from that it sends a request to the proxy, the proxy takes up this request and proxies that means it sends the request out to the world through the intended web server giving one of the valid IP addresses and when it gets the response, it changes the IP back again to the actual private IP of the machine and gives it back. That is another way, proxy may be used. Proxy may also be used to keep track of what kind of data or what kind of sites people in the organization are generally looking at, they may feel that this site may be good, this may be bad but anyway the point is that this may be done if you are routing all your HTTP requests through the proxies. Naturally it has to give the entire URL when it sends a request through a proxy.

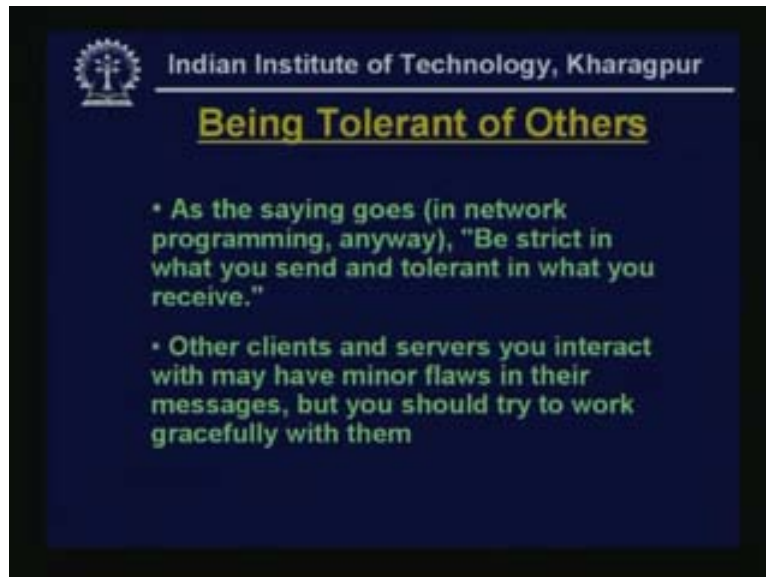
(Refer Slide Time: 51:25 - 51:55)



For example, GET HTTP then you give the domain name, then the path, then you give the path name, then the HTTP and its version number. That way the proxy knows which server to forward

the request to though the proxy itself may use another proxy. There may be hierarchy of proxies etc, you are not bothered about that.

(Refer Slide Time: 51:56 – 52:15)



As the saying goes in network programming anyway be strict in what you send and tolerant in what you receive. Other clients and servers you interact with may have minor flaws in their message but you should try to work gracefully with them.

(Refer Slide Time: 52:16 -52:51)



HTTP specification suggests the following: Even though header lines should end with CRLF that is a carriage return and line feed someone might use a single line feed instead, accept either

CRLF or LF. The three fields in the initial message line should be separated by a single space but might instead use several spaces or tabs, accept any number of spaces or tabs between these fields etc. Usually on the other side is tolerant of these. With this we come to the end of this lecture.

Actually, HTTP and this entire gamut of web services have been developed in such a manner that these demands codes by itself. Actually this whole field of networking has grown at a tremendous rate. People usually talk about Moore's law for computing power and memory capacity etc. But if you look at the network traffic and the speed at which network has been growing, you will find that this is actually much steeper than the Moore's law. Moore's law by itself is exponential so being steeper than Moore's law means that you have a higher base, steeper rate or growth which is a very phenomenal rate of growth. And this phenomenon rate of growth has to respond to a large number of technologies, a large number of protocols etc., and it is expanding all the time.

Actually we have tried to encompass the basic technologies, first of all the lower layers and just we have touched upon core ideas in all the seven layers. We talked about web today and you can have an entire course on the way web services are given and all web applications can be developed etc. We have given only one lecture on a network security, you can have an entire course and networks security. It has become so important these days because transactions are being done over the network. What we expect is that as time goes on, this ubiquitous-ness of network is only going to grow and now-a-days we are not always care full to may be carry a pen because may be wherever we go we will find a pen.

Similarly, the expectation now would be that wherever you go may be you can find the network outlet where you may be having a laptop or palm top or whatever and through the DICP get yourself configured and be on the network. And of course people are thinking of advertising intelligent machines like your microwave oven or your heater or other things also on the network. So this is going to be a universally networked world. So with this we come to the conclusion of this lecture. Thank you.