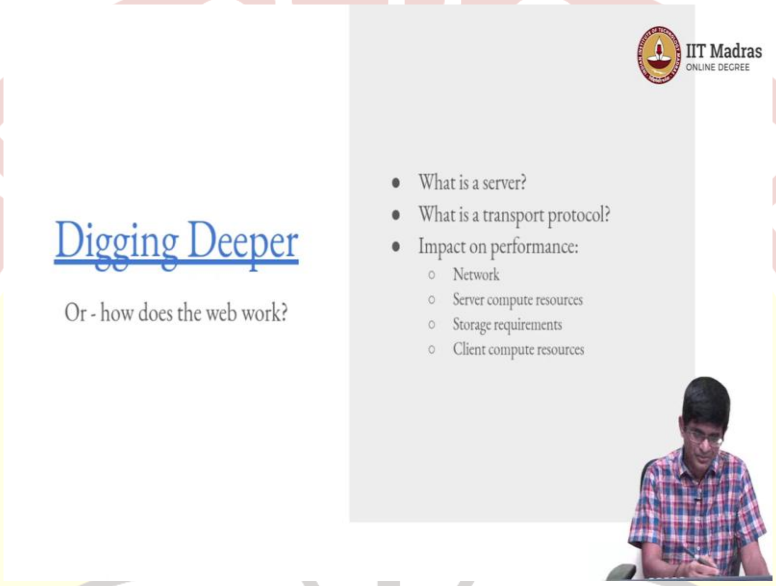
INDIAN INSTITUTE OF TECHNOLOGY MADRAS • सिद्धिर्भवति कर्मजा •

IIT Madras
ONLINE DEGREE

Modern Application Development – 1
Professor Nithin Chandrachoodan
Department of Electrical Engineering
Indian Institute of Technology, Madras
How does the Web works?

Hello, everyone, and welcome to this course on Modern Application Development.

(Refer Slide Time: 0:16)



Digging Deeper

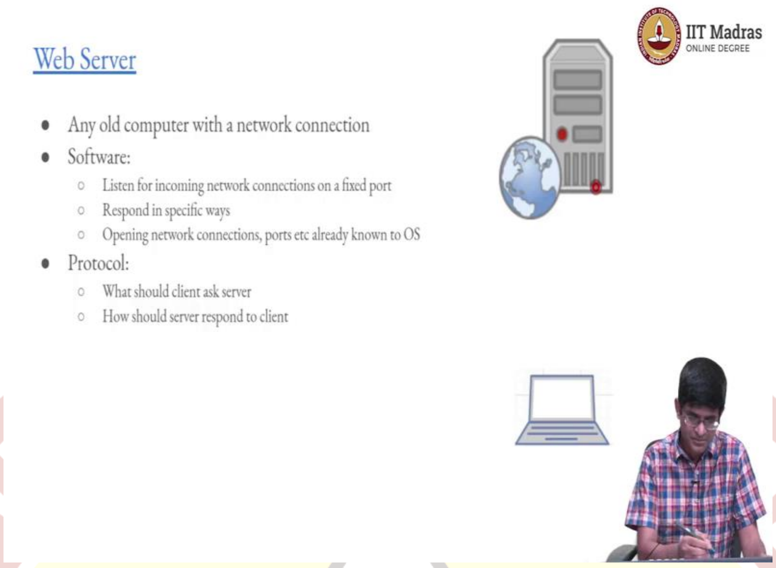
Or - how does the web work?

- What is a server?
- What is a transport protocol?
- Impact on performance:
  - Network
  - Server compute resources
  - Storage requirements
  - Client compute resources

So now, let us dig a little deeper or try to understand a little bit about how the web works. So, what we are going to do over here is to take up a few topics, one of them is to understand what exactly do we mean by a server? What is the transport protocol and then we are going to look a little bit into certain aspects of the performance.

So, there are several factors that determine how well a server can operate. One of them is the nature of the network, the computational resources that the server has, the storage requirements, that sort of depends on the application that you have, but is also constrained by what kind of server you are using. And also, to some extent, what kind of computation can be done on the client end itself. We will take a few examples here and there to sort of get a flavour for what kind of things the designer of an application needs to keep in mind.

(Refer Slide Time: 1:22)



Web Server

- Any old computer with a network connection
- Software:
  - Listen for incoming network connections on a fixed port
  - Respond in specific ways
  - Opening network connections, ports etc already known to OS
- Protocol:
  - What should client ask server
  - How should server respond to client

So, first things first, what is a web server and at its most basic level, any old computer with a network connection can act as a web server. So this, whatever machine you are currently viewing this video on, has enough capabilities to act as a web server of its own. In fact, you might find that, there are some applications running in the background that are actually effectively using the HTTP protocol without you are even knowing about them.

Because it is such a nice and simple protocol that many applications sort of tend to build on top of HTTP. And you might have multiple such servers running there without you are even knowing that they are. But the point is that if you wanted too, you could also create your own web server and it is as simple as basically running a small piece of software on your machine.

Now, there are certain pieces of terminology that I will be using over here, for example, network, connections, ports, sockets, and so on which, ideally, in order to understand that you need to know a little bit more about how computer networks work. We cannot really go into too much detail on that in this set of videos. But for the most part, in order to understand what is happening at a high enough level, you do not need to know the full detail of how the network functions.

So, when I say for example that software listens for incoming network connections on a fixed port. All that you really need to think about it is there is a piece of software executing on the processor, on the server that you have and which has some connectivity with the network equipment attached to that server. Now, what happens is the moment you have network equipment, it means that you could potentially get bytes, some packets of information coming in from somewhere else.

And the operating system then takes care of taking those bytes, those packets and interpreting them. So, it basically looks at the text, the headers that are present in those bytes and says it looks as though this packet of information is meant as a connection from such and such machine to this particular port on this machine. Now, what is a port, at that point a port just becomes a number.

Let us say I choose a number between 0 to usually 65,536 that is 65,535 that is 2 to the power of 16 minus 1. So, in other words, ports are usually given a 16 bit value. So, all that happens is there are some bytes which are sent from computer A. So, for example, I have my client, which is a laptop, I have a server out here, which is sitting somewhere else. And in between them, I have a network.

So, what happens on this network is that the client, the laptop, sends a set of bytes. Within those bytes, it has some header information. As part of the header information, it puts in extra detail saying I want to connect to this port on the server. The server receives those bytes, the packet of information, it unwraps it and starts interpreting it, and it says, this is an incoming packet from this machine, and it wants to talk to some port number x on my software, and it goes and looks up which of this different applications running on this server is listening to port number, let us say 80.

And when it finds out that there is one particular program running, which is listening on port 80, it gives that information to that program and says, now you know what to do. And hopefully, the programmer knows what to do. So, all that we really need in order to take a computer and convert it into a server is to run the appropriate software. It has to be something which can listen for connections, incoming connections on a fixed port, some port, which the other side knows, and to respond in specific ways.

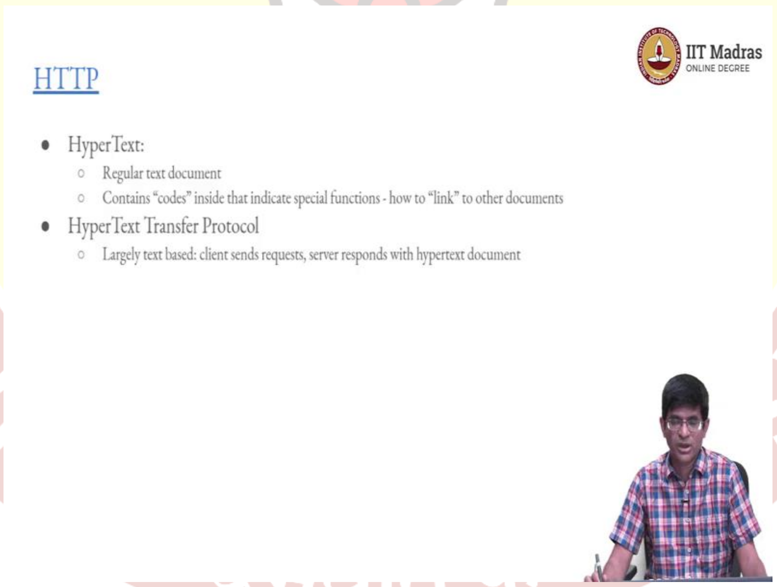
At the most basic, all that it needs to do is take the request, find out what is being requested, is it a file? Hopefully, it is? If it is something else, then you might have to do some other work. But if it is a file, then basically take that file, send it back. Things like how to open a network connection, how to handle ports, all of those are not part of the web server, they are already known to the operating system.

So, this is where again, the separation of functionality comes into the picture, the operating system knows how to handle the most basic level; the ports, the web server knows how to handle things at one higher level than that. And now once we have that, all that we need to do is, remember protocols, what does a protocol say? It says how to communicate between two devices.

A network protocol might say, what kind of formatting and headers should be used when you are trying to communicate between two network devices. Over here, we are going to say, you are sending requests from a client that is a laptop or whatever, to a server. This is one step above the network, because I have already specified things like which is the target address, which is the port and so on. But I still have some more information, which is part of my protocol.

And now this information further says, what is the request? Do I want to get a file? If so, which file? What kind of languages am I willing to accept in response? What kind of responses am I going to accept, all of that information is sent as part of the header information. That defines the protocol and it also tells the server how it should respond to the client. And this standardized part, the part which says how the client should talk to the server, and how the server should respond, is what is called a Hypertext Transport Protocol, HTTP.

(Refer Slide Time: 7:32)



HTTP

- HyperText:
  - Regular text document
  - Contains "codes" inside that indicate special functions - how to "link" to other documents
- HyperText Transfer Protocol
  - Largely text based: client sends requests, server responds with hypertext document

So, HTTP is essentially a way of transporting data. But what is the 'HT'? To start with, it is hypertext. Any regular text document can be thought of as a form of hypertext. The main point being that in a hypertext document, there are specific codes that are embedded inside the document. These codes are not arbitrary numbers, they could be text by themselves. We will look at how exactly it is done later, but the important point is, it is just something that is interpreted in a certain way.

So, ultimately, if I look at a hypertext document, and just print out all the characters, it just looks like text to me, it becomes special only when there is a client or a browser or some application that knows how to interpret those codes. The moment it knows how to interpret the codes, it knows that when it sees, for example, a specific thing which indicates a link, it has to change the formatting of that text and make it something which is shown to the user or something you can click on.

So, this allows you in turn to link to other documents. So, a click by itself does not do anything with the document, it tells the browser, go request a new document from the server, either the same server or from another server. And in that way, you create links to multiple different documents. So, the Hypertext Transfer Protocol, the neat thing about it is it is a very simple protocol. It is largely text based.

In fact, we will look at some of the requests and responses that go back and forth. You can actually read them. They are all based on text. So, the client sends requests, the server responds usually with a hypertext document. In other cases, it may be with some other data corresponding to whatever was required to be sent.