CS 240: HTTP Overview Transcript

[00:00:00] Now let's do a little overview of HTTP and talk about how it works at a high level.

[00:00:06] So on this slide, you see a diagram where it shows the chess client and server, these don't necessarily have to be chess either.

Start visual description. The professor demonstrates a diagram showing a generic client-server system. The diagram illustrates the communication between the client and server programs, initiated by the client. End visual description.

[00:00:15] Uh This is just a generic client server diagram.

[00:00:18] So in a client server system, communication between the client and server programs is initiated by the client.

[00:00:27] So whenever the client program wants to interact with the server in some way, the first thing it does is it establishes a connection with the server.

[00:00:36] It's kind of like when you call somebody on the telephone, you have to type in their number and that establishes a, a phone connection between you and them.

[00:00:44] So the same thing goes in a client server system, the client establishes a connection with the server.

[00:00:49] And then the next thing that happens is the client sends what's called an HTTP request to the server that HTTP request contains all the details of what the client is asking the server to do.

Start visual description. The professor explains the process of sending an HTTP request from the client to the server. The example given is a chess application where the client sends a login request containing the username and password to the server. End visual description.

[00:01:03] So for example, in the chess application, the client might be asking the server to log in the user. So, the user types in their username and password, then the client would create an http request message that contains the username and password.

[00:01:19] And then it would, would send that that request over to the server and the server would receive that log in request and then it would process uh the login request. It would validate the username and password. And if they're, if everything checks out, ok, then it would return a message back to the client that says yes, the login succeeded.

[00:01:37] Now, if the username and password are in Val, the server would instead return a message to the client that says no, the login failed.

[00:01:46] And then once that interaction that that request response interaction has completed, then the client would go ahead and close the connection to the server.

[00:01:55] So, so typically the client would have to connect to the server each time it wants to send a message to the server and get back a response.

Start visual description. The professor describes the client-server interaction, emphasizing that the client must connect to the server each time it wants to send a message and receive a response. End visual description.

[00:02:03] So the client doesn't hold open the connection forever.

[00:02:05] It just opens it when it needs it.

[00:02:08] So that's basically um how a client server system works.

[00:02:12] Now, you could build lots of different applications using that, that same basic idea.

[00:02:16] But of course, we’re going to, we’re going to implement the multi-user chess game um with this technology.

[00:02:23] Now, one thing I want to point out is when we talk about server here.

[00:02:27] Um People use the term server in different ways.

[00:02:31] One way people use that word is that in reference to a computer hardware.

[00:02:36] So somebody might say I’m going to buy a server, which means they’re going to buy a computer which, which um operates as a server.

[00:02:44] Another way people use the word server is to refer to a server program.

[00:02:48] And that’s the way we’re going to use the word in, in this class.

[00:02:51] So when I talk about a server, I’m not really talking about a machine, I’m talking about a program.

[00:02:58] So the chess server that you’re going to write is just a regular java program.

[00:03:02] It has a class that has a main method on it.

[00:03:05] And um you just run it like a normal java program.

[00:03:08] The thing that makes that program a server is that uh when you run it, it sits there and listens for incoming client connections on the, on the network and whenever a connection is, is established and then a request comes in, then the server would process that request and return a response.

Start visual description. The professor explains the concept of a server program, using the example of a chess server written in Java. The server listens for incoming client connections on the network and processes requests. End visual description.

[00:03:25] So um that’s, that’s the, the, the sense in which we’re using the term server.

[00:03:33] OK.

[00:03:34] Now, let’s talk about this idea of a client connecting to the server.

[00:03:38] Um We already said that establishing a connection between the client and the server is very similar to making a phone call.

[00:03:44] And so the client would establish a connection with the server.

[00:03:49] And once that connection has been established, what that really provides is it provides a bidirectional capability for the client program and the server program to, to exchange data back and forth.

[00:04:01] Now, when I say data, I’m just talking about bytes.

[00:04:05] So, the connection essentially establishes a pipeline through which the client can send any bytes of data that it wants to the server.

[00:04:13] And then in the other direction, the server can send any bytes of data that it wants to the client.

[00:04:20] And so it’s just a bidirectional pipeline. Now, what’s in the bytes that are being transferred depends on what networking protocol that we’re using.

[00:04:29] Now, in, in our case, we’re using the HTTP protocol.

[00:04:33] And so the http protocol would define the contents of the various messages that are being passed back and forth between the client and the server.

[00:04:41] Uh In the case of HTTP, the protocol is primarily text based. So, um these messages that are being passed back and forth are made up of readable text streams that that that you can actually look at and, and, and read um other protocols uh would just send binary data that’s not human readable.

[00:04:59] So there’s lots of different networking protocols in the world.

[00:05:01] And http is just one that we’re using, but it turns out to be perhaps uh one of the most important protocols in, in the world.

[00:05:09] So the client opens a connection to the server now to make that possible, any computer that’s going to participate on the internet has to have something called an IP address.

Start visual description. The professor discusses the importance of IP addresses for computers participating on the internet, comparing it to having a phone number for making calls. End visual description.

[00:05:22] So just like if, if you want people to be able to call you on the phone, you have to have a phone number.

[00:05:27] Well, the same thing holds on the internet and in the web, any computer that wants to send and receive messages over the internet has to have an IP address.

[00:05:37] Now, this is something you’re probably familiar with an example IP address uh would be uh 1 28.1 87.80 0.20.

[00:05:46] So you can see an IP address is really just four numbers uh separated by dots or periods.

[00:05:53] And these numbers are in the range of 0 to 255.

[00:05:57] So that’s what an IP address looks like.

[00:05:59] And every computer on the internet has to have a unique IP address.

[00:06:02] And that’s the address at which uh clients will connect to the server.

[00:06:09] So in this case, the machine that the server programs running on has an IP address.

[00:06:16] Now, the client needs to know the server's IP address in order to connect to it.

[00:06:22] But it's kind of hard to remember these IP addresses like you really don't want to have to memorize those or even type them in.

[00:06:29] And so typically we don't use IP addresses directly instead what we use are strings called domain names.

[00:06:37] So typically, for example, if you wanted to um do a Google search, you could fire up your web browser and in the URL BAR, you could type in www.google.com and then you could hit return and somehow your, your web browser would connect to uh Google's web server and, and send it a request.

[00:06:57] So you don't type in the IP address of Google server rather you type in www.google.com.

[00:07:04] So www.google.com is a domain name.

Start visual description. The professor explains the use of domain names instead of IP addresses for easier human readability, using the example of typing [www.google.com] (http://www.google.com) instead of an IP address. End visual description.

[00:07:07] It's a human readable name that we use instead of IP addresses.

[00:07:12] So the thing to understand is that a, a domain name is just the human readable form of an IP address.

[00:07:18] So the first thing your browser does when you type in a domain name of a website, first thing it does is it uses uh internet service called DNS, which stands for domain name system.

[00:07:29] And what it does is it sends the domain name that the user types in and it sends it to the DNS service and says, hey, could you translate this domain name into an

IP address? So basically, giving back the IP address that corresponds to this domain name.

[00:07:43] And so DNS would return to the, the web browser.

[00:07:45] In this case, the IP address of the Google um server and then the browser or the client would use the IP address, it gets back from DNS to create that connection to the server.

[00:07:59] So we do have IP addresses underneath, but mostly we work in terms of human readable domain names like byu.edu or, or what have you.

[00:08:09] Now, let’s assume that every computer that’s connected to the internet has a unique IP address.

[00:08:14] So, so clients can connect to it.

[00:08:16] The other challenge we have is that if you think about a computer, there’s lots of programming programs running on every computer.

[00:08:25] For example, if you have a laptop, um, at any moment, there’s dozens, if not hundreds of programs running on your laptop at any moment.

[00:08:33] And any number of those programs that are running on your computer could be using the internet.

[00:08:40] And so just having an IP address is not enough to establish a connection to the, the program that you’re trying to connect to.

[00:08:49] Remember a server in our uh case at least is a program.

[00:08:55] So if you look at this slide here, you can see that on this server machine, which is represented by this, this green box, there’s a whole bunch of different programs running on that, that server machine that are using the internet.

[00:09:08] And so when a client wants to connect uh to one of these programs that are running on the server, just providing the IP address isn't enough.

[00:09:16] If you give me the IP address of the server machine, that tells me what machine the program you're connecting to is running on.

[00:09:22] But it doesn't tell me which program on that machine you're trying to connect to.

[00:09:26] And so for this reason, we also have something called a port number or a TCP port number.

[00:09:32] And these port numbers are in the range 0 to 65,535.

[00:09:37] And whenever a program is using the internet, it runs on a particular port number.

[00:09:44] So for example, when I start my chess server program, I'm going to have to tell what port number, what TCP port should it listen for connections on? Now, in this case, uh for example, this, this program in the middle is listening on port 10,000.

[00:10:00] So let's assume that I ran my chess server and, and I told it to use port 10,000.

[00:10:05] So that means when the chest server comes up, it's going to initialize and then it's going to just sit there and listen on port 10,000 for incoming connections.

[00:10:16] So what that means is that when a client program wants to connect to my server, it has to specify the IP address of the server machine.

[00:10:25] But it's also got to specify the port number that the program is, is listening for connections on.

[00:10:31] So the, the combination of an IP address and a port number provides all the information that is needed for the client to establish that connection to the server program that it is trying to connect to.