Slide 1

Hello, this is Raphael Portnoy. Welcome to our second chapter of the computer networking class. Today, we're going to look at the application layer. We're going to look at things like web and HTTP, different applications such as FTP SMTP.

We're going to look at DNS, we're going to discuss principles of networking applications, and we're going to look at the p2p applications something that may be of real interest to you

Slide 2

Our goals today to discuss transport layer service model. We're going to look at the client-server paradigm in a peer-to-peer paradigm and as I said before we're going to learn about different application level protocols. Things that are really acting in network and really have effect on computer network. Here are some of the examples of the network and apps. Email, web, remote login, file sharing, YouTube, Netflix anything that you can imagine today is probably networking app the way I started in computer networking is by playing a game doom Many many many years ago and when I was done with a standalone version I wanted to play with my friends we learned how to run it in a network and that was the beginning of a very successful career let's hope that you're going to enjoy this class very much

Slide 3

So, what is a network app? It's an application that communicates with other applications that are part of the same ecosystem over the network. We have different browsers. We have different devices and all of those different entities have to communicate together across the global network. so, we don't want to write a specific piece of code for an iPhone or for a personal computer or for a car. we want to be able to write an application that runs across different pieces of hardware it also allows us to rapidly deploy this application throughout the ecosystem we can develop different sets of hardware and then develop a network app that communicates over the network. Things like web servers and browsers Emails of different types we'll talk about them later today and other applications There are two types of application architectures client-server and peer-to-peer. Let's look at them in more detail.

Slide 4

Client-server architecture consists of two entities a server and multiple clients. A server is an always-on host that has a permanent IP address and now they sit somewhere in a big data center Microsoft, Google or your company's data center. Clients communicate with the server. They could be intermittently connected. Think about your laptop or your iPhone, they may have dynamic IP address and they do not communicate with each other.

Slide 5

Alternatively, p2p architecture is completely different from client-server we do not have an always-on server. These disparate systems serve as clients and as servers. The peers connect and request information from each other and they can come and go. It requires complex management, but it allows really great self-scalability. You may have heard of different peer-to-peer systems, such as file sharing or

movie sharing. That architecture is used also widely for software distribution and many different things as well.

Slide 6

let's get tactical now we have processes communicating between clients and servers client processes are the ones that initiate the communication server processes are the ones that are sitting and waiting to be spoken to you could think about your browser when it opens up and you go to yahoo.com yahoo.com is a server that has a process running waiting for your browser to connect to your browser when it opens up makes different requests and that's how you start loading the web pages process is run on across all the hosts within the same host we can have multiple processes for example a host such as yahoo.com or google.com or any other website that you know runs multiple processes so multiple clients can connect to it aside from that applications that are part of the p2p architecture have both client processes and server processes

Slide 7

Brings us to the next technical issue of the conversation which is sockets. Processes send and receive messages to and from sockets. Socket is something analogous to the door. The sending process shoves the message out of the door. It goes through the network. On the receiving end, it gets received through the proper door and get served up to the application.

Slide 8

To receive messages, a process must have an identifier. As you probably already know, each host device has a unique IP address. Identifier consists of the IP address of the receiving host and the port numbers that are associated with the process on the host. For example, http usually goes through the port 80. Our mail usually goes through port number 25. To send a http message or web request to specific web host, we would type in the host IP address and the port number. You may think about why don't we do that? Why don't we put port number 80? Well, our browsers are now smart enough to recognize http traffic by default goes through port 80. This also means you can configure web server to receive http traffic through different port. You will then have to specify but we mostly don't do that. We will look at in a little bit more specific further down the line.

Slide 9

So by and large, an application layer protocol defines the types of messages that are exchanged. Message syntax, what fuels the messages, and how fields are delineated. Message semantics. Rules. There are some open protocols that are defined in request for comments such as HTTP and SMTP and they're used to ensure interoperability between different systems. There are also proprietary protocols such as Skype that can only use within the ecosystem of the specific app.

Slide 10

Now let's take a look at what type of transport services applications need? We do have to ensure data integrity to, some level, some applications like, your email or file transfer or web transaction require 100% reliable data. Others, such as audio applications, streaming audio, Spotify, YouTube, they can tolerate some loss. we also have requirements on timing, some applications like telephony require

specific low delay where are others you can incorporate some of the delay into the application delivery. At the end of the day, you don't really care, whether your mail gets delivered 30 seconds sooner or later. We have to look at things like throughput, where some applications require minimum bandwidth allocation to be effective and then of course security things, like encryption and data integrity play a huge role in what type of transport service we select.

Slide 11

Here are some of the examples of the transfer server service requirements for some of the common applications. File transfer, for example cannot tolerate data loss. The throughput is elastic which means we could experience some cubed and still deliver the file and it's not time-sensitive. Real-time audio and video and stored audio and video are very similar. There are less tolerant we can lose a little bit of information without really affecting the quality of the app but there is time sensitive you don't want to sit on the phone and hear the delay in jitter. So, you could see that different apps and this is very very important require different network architecture and different design to accommodate all of our user requirements. Let's take a look at two fundamental transport layer services. Transmission control protocol service provides reliable transport between sending and receiving processes. It provides flow control which allows the sender to gauge the conversation with the receiver. It provides congestion control which allows the sender to throttle when the network is overloaded. But it does not provide timing minimum throughput guarantee or security. It does however require connection setup which means that we have to agree on the parameters of the conversation before we start sending in receiving information. User Datagram protocol does not provide any of the services that TCP provides, however It's 40% less in terms of bandwidth requirements, so it makes it really attractive for those networks that are bandwidth constrained.

Slide 13

And here are some of the examples of the applications and the underlined transport services. Email uses TCP. If you recall TCP provides guarantee that the traffic is not going to get lost. Web uses TCP. Of course, we need to see the web page that we want to exceed to get loaded. Things like Internet telephony and streaming multimedia can use either TCP or UDP. Primarily it uses UDP because we want to be able to pump the information out onto the network as much as possible.

Slide 14

Now we're going to take a look at web and HTTP. Well, Let's review how the web page is constructed. We have different objects that can be HTML files, JPEG images, Java applets, or audio files. The web page consists of a base HTML file that includes several reference objects in it and each object has an addressable URL. So, you could see it in an example that we have a host name and then a corresponding path to a particular object.

Slide 15

Hypertext transfer protocol is the web's application layer protocol. It follows the clients' server model, where different hardware devices such as personal computers, laptops, iPhones, use client, a browser that sends and receives requests using http protocol and displays web objects. The server, on the other side, allow us to communicate across all of the different hardware devices independently. It runs the

software that operates independently of the hardware of the client and still display the object in its intended way.

Slide 16

HTTP is a stateless protocol server maintains no information of the past client request. HTTP uses TCP. Our client initiates TCP connection creates a socket to server on a port 80. Server then accepts that request and then processes the information, so it gets send the entire web page back to the client. After that the connection is closed.

Slide 17

There are two types of http connections. Non- persistent http and persistent http. The difference is, in a non-persistent http, at most, there is only one object that can be sent over TCP connection and then the connection is closed, which means it would require multiple connections to send through multiple objects. Persistent http, on the other hand, allows us to send multiple objects through single TCP connection and then close it.

Slide 18

Let's take a look at a non-persistent HTTP connection. Let's imagine that we're trying to access a URL which has a home that Index.html file that contains text and references and JPEG images a very simple page. HTTP client will initiate TCP connection to the server. The server at that host is open. It listens for requests and when the request comes in and port 80 it will accept the connection. The client will then send a request message to the server indicating that we want to get the information about the base HTML object. The server will receive it form the response message containing the requested object and then send it back into the socket. The server will then close the TCP connection. The client will receive it. The client will understand that there is reference to 10 JPEG objects within that file and then repeat steps one through five for every object that's part of that a base HTML.

Slide 19

We're now going to take a look at the RTT. Round-trip time. It's a time that's required for a small packet to travel from a client to the server and back. If we look at the HTTP response time, we need one RTT to initiate the TCP connection. We need another RTT for the HTTP request and the first few bites of the response. Then we have file transmission time incorporated. So, if we pull all of those different parameters together, we can define the non-persistent HTTP response time as two RTT's plus the file transmission. Take a look at the diagram

Slide 20

Persistent HTTP another hand leaves the connection opened so subsequent HTTP messages between the same client server can be sent over the same connection. As little as one RTT is required for all reference objects; whereas non-persistent HTTP requires to RTT per object. Why do we have two different ones? Well it really depends on what type of network we run and if the network is stable, we can use persistent HTTP. If, however the network is choppy, then we would use non-persistent HTTP which would allow us to transfer large number of smaller objects and pull the file together.

Slide 21

Http protocol maintains a variety of status calls. They appear in the first line of the server to client message. Some examples are: 200 when the requests are successful, 404, if the file was not found on the server, or 505, if http version is not supported.

Slide 22

Let's talk about cookies. We use cookies to maintain a user information on the server side. For example, Susan always access the internet from her PC. She visits specific ecommerce site for the first time that's when the information gets stored right on the server. There are four components of the cookie. There is a cookie in the header line of the HTTP response message. There is a cookie header line in a request message. Cookie file is kept on the user's host and then there is a copy on the backend database. You probably have used them before and you have seen them. When you go to Amazon website all of a sudden you see information in your cart. That is there because of the usage of the cookie. They are also used to access and your privacy information and a lot of times you may have seen them in an Internet ad that appear on your computer screens.

Slide 23

Here is an example of keeping the state or how the cookies are used. On the client side, there is a cookie file that sits in the browser specific to the website that you have visited. There is a different one for eBay or Amazon. If than a week later you come to the same website, the cookie information on your computer is going to match with the backend database of the site you have visited. It's going to pull up the information about your usage, your cart, or anything else that was stored in the cookie file.

Slide 24

Cookies can be used for variety of reasons. Things like authorization, shopping carts, recommendations, user session state for the web email, and there is a huge issue about cookies and privacy which basically allow other entities to collect information about the user. Probably one of the top 10 problems on the internet today.

Slide 25

Now let's take a look at web caches also known as a proxy server. It's an intermediary that acts as a client and a server. It stores some of the information from the web server, and then it serves it to the clients when they have to access it, usually through the web browser or other means. Browser sends all of these HTTP requests to the specific cache so rather than going to the destination server, we ask somebody locally. Do you have the information? And then if they do, we get it. If they don't, they have to go and obtain that information from the originating server. Cache as I said before as both as a client and a server. It is typically installed by the ISP or a university and it allows us to reduce response time. It allows us to reduce the traffic on institutional access link and in the internet dense environment poor content providers get access to a lot of the information.

Slide 26

Let's look at the cache as an example. Let's imagine that we have 1.54 megabit per second access link. We have an institutional network, which is really fast and then we have some origin servers out in the internet. If we imagine that that link is overly saturated with traffic, we may have a problem in terms of the utilization. One way to solve this problem is to buy a larger link. We can actually do that and then our link utilization is going to go down from 99 percent to nine-point nine percent. That's what the ice piece would love us to do, but usually it's a lot of money and it has to be maintained on an ongoing basis. Another way to solve that problem is to install local cache, and that's where the information about some of the content on the internet or inside our corporate network may reside. Imagine if you're a global company and you have a lot of content throughout the internet you then need to access it from many different workstations. By installing a local cache and keeping the information locally we can actually make access a lot more efficient.

Slide 27

So how do we make sure that we always have up-to-date information? Well, HTTP application layer protocol has a conditional gate command that's part of it. We use their command to check on whether or not the content is up-to-date. We can specify the date of the cache copy inside our HTTP request. The server then will check with the origin and if the content has not changed, it'll serve us what it has. If the content in fact has changed, it'll go out grab the latest information and then serve it to us. It will also store it for future use.

Slide 28

Next in our agenda is file transfer protocol, also known as FTP. We use FTP to move files from one host to the next. Its primarily use in web environment now when we have to upload newly develop pages. It operates over port 21 and it has been defined in an RFC 959.

Slide 29

Now we're going to take a look at the electronic mail in different protocols that allow us to support that function. Things like SMTP, pop3, or an IBM. Electronic mail has three major components. There are user agents. There are mail servers, and then there's a protocol itself, that's called SMTP. Simple Mail Transfer Protocol. User agent also known as a mail reader is used on our devices. You may be familiar with different user agents, such as Outlook, Thunderbird, or a native iPhone app. Outgoing and incoming messages are stored on a server.

Slide 30

Mail servers house the mailboxes of all of our different users. They also maintain message queues for the outgoing messages to be sent in the use SMTP protocol between themselves to send and receive client messages.

Slide 31

SMTP uses TCP as an underlying transfer protocol to transfer the email messages reliably from client to the server on port 25. It's a direct transfer from the sending server to the receiving. There are three

phases of the transfer. There is handshaking also known as greeting. There is a transfer itself, and then there is closure.

Slide 32

Let's look at our scenario. Alice wants to send a message to Bob. Alice is going to use her user agent to compose a message to a specific email address. Alice is then going to send a message to her mail server and message is going to be placed in the queue. The client side of SNTP is going to open a TCP connection to Bob's email server. SNTP client will then send Alice's message over TCP connection. Bob's mail server will receive it, place it in his mailbox, and when he involves his user agent, he will be able to access and read that message.

Slide 33

SMTP is used for delivery and storage of the messages on the receiver side. There are also different protocols that are used for the retrieval of the message. Pop protocol is used for the authorization and download. We have a more sophisticated protocol called IMAP. Internet mail access protocol that has more features such as manipulation of store messages, creation of folders, and you may have used that and seen it before. Finally, we could also use HTTP to access our mailboxes. The most prominent one is the Gmail client that allows us to access our messages from Gmail.

Slide 34

A little bit more about pop3 and IMAP. Pop3 was originally designed to download messages from the servers and then erase them on the servers. That was during the times when storage was very very very very expensive. Pop3 was then modified to allow download and keep function, so this way we can download messages in our user agent and there is still stored on servers themselves. We have to remember that pop3 is stateless across different sessions. If you have multiple devices using pop3 protocol you may have seen messages that you have read on one device showing up as unread on another device. IMAP allows us to keep all messages in one place on the server itself. It allows us to organize messages in different folders, and it keeps user state information across different sessions

Slide 35

We're now going to take a look at the domain name system. We as people have many different identifiers, Social Security numbers, names, passport numbers, but on the internet the only way to identify a host is through its IP address. We use names, www.yahoo.com, to map that name to the IP address. It's primarily a system that is used by humans and for humans. The main name system is a distributed database implemented on an IRR key, and it's an application layer protocol that allows hosts and name servers to communicate to resolve names, such as address to a name translation. The interesting observation is that it's a core Internet function that's implemented as an application layer protocol which means that we have moved complexity at the network's edge.

Slide 36

The DNS services, as we just discussed, are host name to IP address translations. We also used it for mail server LSN, and there is a lot of low distribution, which pushes us to replicate web servers to many different IP addresses corresponded to a single name. We don't centralize DNS because we don't want

to have a single point of failure. We don't want to increase traffic volume. It's a lot of maintenance. In short, it doesn't really scale.

Slide 37

Let's look at how it really works. DNS is a distributed hierarchical database. At its core, we have root DNS servers that support dot-com, dot-org, or the new DNS servers, which then subsequently support specific host DNS servers. So as an example, if the client wants to obtain an IP address for the amazon.com, it will have to query the root server to find the dot-com DNS server. It will then have to query the dot-com DNS server to get the Amazon dot-com DNS server and eventually it will get the IP address to a name nothing.

Slide 38

There are thirteen root name servers worldwide. It gets contacted by a local name server that cannot resolve the name. The root name server contacts the authority name server if the name mapping is not known. It gets the mapping and returns the mapping to the local name server.

Slide 39

There are some top-level domain servers that are responsible for the dot-com, dot-org, dot-net, dot-edu, dot-jobs or any other domain that you can imagine. Network solutions maintain servers for dot-com. EDUCAUSE maintains servers for the dot edu top-level domain. Top-level countries maintain their own domain servers. Authoritative DNS servers are owned by the organization, and they provide what's called authoritative host name to IP mapping. Sometimes they reside on organizational premises and sometimes at different providers such as GoDaddy.

Slide 40

Local DNS server does not strictly belong to the IRR queue. Many times, an organization or an ISP decides to maintain its own local domain server to expedite the resolution time of the IP addresses two names for its own users. What happens is when the host makes a DNS query it gets sent locally to a local server and similar to a web cache if it exists it gets searched right away and if it doesn't then the local DNS server will contact the authoritative server to get the right name to IP address resolution and serve it to the client.

Slide 41

Let's take a look at a couple of examples of the resolution. If we wanted to resolve an IP address for a specific host in iterated query would go through each individual example of the iterative process and contact the local DNS first, then the root DNS top-level domain and finally get to the authoritative domain server. Alternatively, we could use something called recursive query which will go through the entire chain, resolve the name and then come back. As you can see there are different ways to obtain the name, and then serve it to the client.

Slide 42

Once a name server learns about the mapping it caches it and then there is a specific timer for how long we maintain that information before it disappears. Top level the main servers typically cached in local

name servers. So, we don't have to go every time to a yahoo.com entity that houses that IP name. We can record the IP address to a name resolution locally and then use it for our purposes. Cache entries may become out-of-date and if that happens our local domain server will send a request to the top-level domain server for an up-to-date information.

Slide 43

There are different types of the DNS records. Type A, which is a direct resolution of the name to an IP address. Type NS, it stands for the name inside the domain. The value is the authority name of the server for that domain There is a Canonical Name Record which indicates the real name of the server. You can go to IBM.com, but in reality, when the server was built, it could have been called something else. Finally, there is a MX record that is part of the DNS record system that is used for the mail server resolution.

Slide 44

How do we insert the record into the DNS? If we have a startup for example that's coordinate called Networkutopia.com. We can contact one of the registered DNS registrars, such as Network Solutions or GoDaddy. Once we register the name within the registrar, we will be provided the actual name itself networkutopia.com. An IP address of the authoritative nameserver and eventually we can set up our name records, so the server name can resolve to the IP addresses as well as our MX records for the email resolution.

Slide 45

Well, if you can imagine DNS plays an integral role in our network operations; therefore, we have to do everything we can to protect it. There are different ways the in DNS can be attacked. There is distributed denial of service attacks where the root servers or top-level domain service can be bombarded with bogus traffic. There could be read the racket acts such as man in the middle where the DNS queries can be intercepted, and our browsers can be redirected to completely different servers. There is DNS poisoning where a bogus request can be sent to a DNS server and then get cached. Different ways to attack the DNS, and we have to make sure we're really protected for the integrity of our network.

Slide 46

Finally, peer-to-peer applications. Well, as we discussed before pure peer-to-peer architecture has no server that's always on. There are arbitrary systems that come and go, and they communicate directly with each other peers are intimately connected in change IP addresses. Some of the examples of the PDP architecture, BitTorrent for file distribution, Skype for web, or others.

Slide 47

Well if we think about how much time does it take to distribute a specific file from one server to different number of peers. In a client-server environment that file distribution is going to take place sequentially. The server is going to connect to a single client give chunks of a file connect to the next client give file and so on and so forth. Even in the network with abundant resources that time

distribution is going to be depending on a number of peers in the network. If we look at the specific transmission times, it's going to take specific time for each individual client to download the entire file.

Slide 48

However, in the p2p architecture, we can actually start distributing file chunks among our clients without waiting for the entire file copy download. Here's an example of time it takes to distribute a file in a client-server architecture and p2p architecture. You could see as the number of clients increases in a p2p architecture the time to distribute the client does not increase as much as in a client-server architecture. Why? Because we're able to distribute the file chunks amongst each other without actually distributing the file itself.

Slide 49

Let's take a look at BitTorrent to better understand how the p2p file distribution works. We're going to take a file, a movie, or a song we're going to divide it into 256 kilobyte chunks. We then going to have peers in the torrent that are going to send and receive this file chunk. As an example, when Alice arrives, there is a tracker that doesn't have the file itself but keeps a list of participating clients in a torrent. The torrent group of peers is currently exchanging different chunks. When Alice arrives, he or she are going to start receiving chunks, and at the same time giving chunks that she saved to other clients participating in a torrent.

Slide 50

When the peer arrives, it has no chunks. Once it's registered with a tracker, it’s going to accumulate file chunks by connecting to local peers. As the number of the chunks grows, the peer itself is going to be able to distribute those chunks to its neighbors. Once we get the full file, we can selfishly leave, or we can continue staying in a torrent and continue the file distribution. As a side note, BitTorrent can also be used to distribute software. Currently many companies who write software use that technology to quicker deliver next releases to its developers.

Slide 51

The distributed hash table is a database that's part of the p2p architecture that is used to keep records of the pairs corresponding to the key and a value. Similar to us humans where we can have a name and a social security number, there is a key in the tracker database that identifies for example a movie title and an IP address. That is the key that's used to distribute the information to our peers arriving to torrent and also how we distribute our information on our computer to other peers that are part of our torrent. All of the IP addresses are known so if you're distributing the software everybody knows your AP address.

Slide 52

In summary, we covered a lot of materials today. We looked at two different architectures for applications, client-server and p2p. We looked at different requirements such as reliability, bandwidth, delay. We discussed TCP and UDP transport services. We looked at variety of different protocols, such as HTTP, SNTP, DNS, and BitTorrent. Thank you and goodbye.