Lecture #9: Networks (Protocols)

CS106E Spring 2018, Young

In this lecture we take a look at protocols. A protocol is an agreement between two or more computers for how they will carry out a task on a network. We take a look at what sort of information is provided in a protocol and what the relationship between protocols and programs is.

Protocols on the Internet are layered. We explore what this means and then look at the four different layers of protocols. We study the Internet Protocol (IP) in some depth, seeing how it uses Packets for transmitting information and study some of the implications of using these Packets. We also see how the Transmission Control Protocol (TCP) builds on top of the Internet Protocol, providing a more robust platform on which we can build applications.

Protocols

We now have our computers physically connected to one another and we have a means of uniquely identifying computers within the network or internetwork. What else do we need to do for our network?

We need to define protocols that will be used to carry out tasks. A ***Protocol*** is an agreement between computers that specifies how the computers will work together.

- We've actually already seen examples of where we might need protocols on the Internet. The Domain Name System (DNS), for example, has a corresponding protocol telling us how, given a hostname, we can convert it to an IP Number. Similarly, the DHCP system specifies how a computer can hook to a network and request an IP Number.

- In general when we define protocol, we need to specify:
  - what sort of requests different computers can make of each other,
  - what sort of information those requests will contain and what is the exact type and format of that information,
  - what to do when something goes wrong.

- Let's take a closer look at these using the HTTP, HyperText Transport Protocol, as an example. HTTP is the protocol that underlies the web.

  - Using HTTP we can send a number of different types of requests to a web server. These include a GET request for a particular resource (e.g., an image or the text of a webpage), a HEAD request asking for information about a resource (but not asking that the resource itself be sent), and a DELETE request, which as the name suggests asks that a resource be deleted.

  - When sending a request or when replying to a request, we are able to specify additional information in the form of a series of attribute value pairs.

A request, for example, might specify what character encodings the requester is able to use (e.g., UTF-8 Unicode) or it might specify that the requester doesn't want to be tracked.

A response might include not only the resource requested, but might give an indication of how long the information will be good for and when it will expire.

- Finally, we have a series of codes that indicate if the request has been successfully carried out or 404 the resource was not found, 403 the resource is there, but the requester doesn't have permission to access it, 501 the server is hosed, etc.

- Anytime we want two or more computers to carry out a task on a network we need to define some sort of protocol, formally specifying how the computers will communicate in order to carry out the task.

Protocols vs. Programs
What is the relationship between a protocol and a program? A protocol formally specifies the type and form of communication that will take place between computers. It partly dictates how a program using the protocol must work, but it is not a program itself.

Looking again at the HTTP protocol used on the web, Google's Chrome, Apple's Safari, and Microsoft's Edge all follow the rules of HTTP when communicating with a web server. However, they are not the same program; they are three distinct programs and in fact may run on different platforms (e.g., Macintosh vs. PC vs. Android).

In addition, traditional web browsers aren't the only programs that follow the rules laid out by the HTTP protocol. Audio web browsers for the visually impaired follow the HTTP protocol. Even more esoteric, the Google Search Bot uses HTTP to request web pages and uses the results to index the web.

Layered Protocols
Internet Protocols are layered. This means that one set of protocols builds on top of another set of protocols.

I find it works best to explain how this works by analogy.
- Suppose after I retire from Stanford, I move to Maine and start making wooden toys. I'm pretty good at working with my hands so these toys get quite popular and there are requests to ship the toys to different people around the world.
- How am I going to ship my toys? Suppose I had a request to ship a toy to California.
    - I could hire a truck and a truck driver and have the truck driver take my toys to the railroad station.
    - I could then hire a train and work out the railroad schedule to make sure that my train was properly routed to get the train to California.
    - I could then hire another driver to pick it up from the train and take it to its destination

  or

    - I could drop it off at the FedEx office

- Notice how dropping it off at the FedEx office doesn't imply all those other steps don't take place.
    - Someone still has to hire the trucks and truck drivers.
    - It's just not something that I personally am doing.
    - I'm assuming all that other stuff is being handled by someone else.

- Also notice that my shipping company isn't necessarily handling everything on their own either.

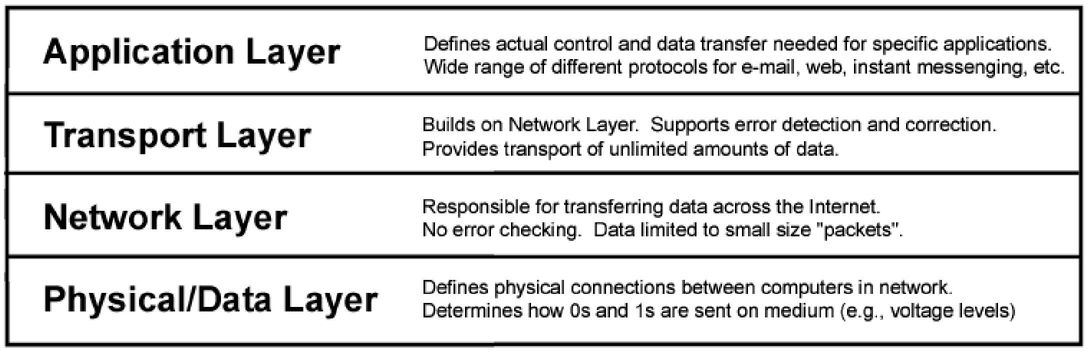
- FedEx probably does, since at this point it's quite large. However, a smaller shipping company might hand off their shipping trailer to a railroad company and assume that the trailer will arrive on the other side of the company.
- Again just as I don't have to worry about the details because the shipping company is dealing with it, the small shipping company doesn't have to worry about some details because the railroad company is dealing with them.
- The details still need to be dealt with, the question is who is dealing with them.

- Okay, how does this relate to our layered protocols?
  - When I'm writing a new application for the Internet, I don't want to have to worry about whether my data is being transferred via a fiber optic cable or a twisted wire pair. I don't want to have to worry about what signal frequency on a network represents a 1 or which represents a 0. I just want the data to get where it needs to go.
  - Now that doesn't mean that someone doesn't have to worry about the network type or how it represents 1s and 0s, it just means I don't want to worry about it.
  - I want the equivalent of dropping the data off at FedEx, and having it magically appear at its destination.

  - As it turns out, this is exactly what layered protocols provide.

The Internet Protocol Stack

The Internet uses four different layers of protocols to form what is referred to as ***The Internet Protocol Stack*** or sometimes ***The Internet Protocol Suite***. We'll explore each of these layers, starting at the bottom layer.



Application Layer	Defines actual control and data transfer needed for specific applications. Wide range of different protocols for e-mail, web, instant messenging, etc.
Transport Layer	Builds on Network Layer. Supports error detection and correction. Provides transport of unlimited amounts of data.
Network Layer	Responsible for transferring data across the Internet. No error checking. Data limited to small size "packets".
Physical/Data Layer	Defines physical connections between computers in network. Determines how 0s and 1s are sent on medium (e.g., voltage levels)

Physical Layer

Ultimately, a network must define how computers in the network are connected together and how 0s and 1s are transmitted between them. For example, on old modems that transmitting digital data over voice telephone lines, the protocol specified that the originating modem would send a 0 by making a 1070 Hz tone and a 1 with a 1270 Hz tone whereas the responding modem would send a 0 by making a 2025 Hz tone and a 1 with a 2225 Hz tone.

This layer is necessary for any new network and by its nature will be different for each new type of network. How information is sent in a fiber optic ring network is very different from how it is sent in a WiFi star network. If I decide the best way to transmit information across the Pacific is to use sharks with frickin laser beams on their heads, I need to figure out how transmit a 0 and a 1 with those lasers, and how to ensure that the sharks are going to be properly aligned to transmit and receive bits.

These protocols are very low level and they are below the level almost all programmers would want to work at. Essentially the company creating the new type of network would specify the Physical Layer protocols, and everyone else would work at a higher level.

Network Layer

The Network Layer provides for a uniform model for getting information across the Internet, but it does so with some very severe limitations. On the Internet the network protocol used is the ***Internet Protocol*** or IP. When we talked about IP Addresses in the last lecture, those were defined by this protocol.

The Internet Protocol specifies that:

- It will make best effort to get information from one point of the network to another point in the network.
    - However, best effort does not mean that the information will definitely get there. Information delivery at this level is unreliable. We'll see the next layer up will do a better job of this.
- At the Network Layer, all information is sent in small size chunks called ***IP Packets***.
    - Data in each IP Packet is limited to 64 kbytes.
    - In addition to the actual data the IP Packet contains other information including the sender and recipient's IP Addresses, a checksum for error detection (something we'll explore during our Security lectures), and an indication of how many bytes are actually being sent.
    - If you're sending something larger than 64 kbytes, it will have to be broken down into multiple IP Packets.
    - Packets are <u>not</u> guaranteed to arrive in the order in which they are sent, and as we've previously seen, in fact they are not guaranteed to arrive at all.

- IP provides for unreliable transmission of very limited amounts of data. This doesn't seem particularly useful. Why is this protocol so primitive?
    - As with the Physical Layer, every new network needs to have its own implementation of the IP Protocol.
    - My network of sharks with firkin laser beams needs its own implementation of the IP Protocol. I'll need to figure out how to get IP Packets to each shark with a given IP Address. Keeping the IP Protocol simple simplifies my task of getting my shark network working.

- As with the Physical Layer, this is still not a level that most programmers will want to work at.
- We'll study some of the implications of the use of IP Packets after first finishing our overview of the remaining two layers in the Internet Protocol Stack.

Transport Layer

Protocols at the Transport Layer allow us to transfer unlimited information from one device on the Internet to another device on the Internet with error handling taken care of.

- There are several different protocols at this level, but the most well-known is ***Transmission Control Protocol*** or ***TCP***.
    - TCP allows us to send data from one IP address to another IP address.
    - It breaks down our data into appropriate IP Packet sized chunks for us.
    - If the packets arrive in the wrong order, it propely reorders them for us.
    - If packets get lost by the underlying IP Protocol, it sends a request to the sender to send a new copy of the lost packet.

- Note that TCP and other Protocols at this level are built on top of the underlying IP Protocol.

    - Their implementation assumes that if they break things down into IP Packet sized chunks, there will be a best-effort attempt to get the data to the destination IP Address.

- TCP is so widely used, that you will often hear that the underlying protocols of the Internet are TCP/IP – meaning that the combination of the network-level Internet Protocol (IP) and the transport-level Transmission Control Protocol (TCP) is what really makes the Internet work.

- Here with TCP we are finally reaching a level that most programmers will work at.
    - If I'm developing a brand new application, using the Internet TCP provides me with a reasonable set of features to build my application on top of.

Application Level

The application level is where all the protocols consumers care about reside. For example:
- There are several protocols associated with email. These include:
    - SMTP (Simple Mail Transfer Protocol) is the protocol that sends email through the Internet.
    - Once your mail server receives an email message via SMTP, it sits on the mail server. When you use an email program to read your messages from the server, the email program probably either uses POP (Post Office Protocol) or IMAP (Internet Message Access Protocol) to get the messages from the mail server to your device.
- As we've previously seen the HTTP HyperText Transport Protocol is used to request and send webpages.
- Transferring files is often done with FTP (File Transfer Protocol) or SFTP (Secure File Transfer Protocol).

- In fact, anytime two or more computers carry out a task on the Internet, there needs to be a protocol associated with it.
    - This protocol might be published, allowing other programmers to take advantage of it, and write programs using the protocol.
    - On the other hand, the protocol might be kept private, allowing only people within a company to understand the details of how an Internet application works and how computers using that application communicate.
    - However ultimately, there must be some sort of protocol defined.

- As you probably already guessed, applications build on top of the underlying transport layer protocols such as TCP in order to carry out their application-level protocols.
    - So for example, if I were to write a new mail server, I would use TCP in order to implement the SMTP, POP, and IMAP protocols my server needed to support.

Implications of the Internet Protocol IP Layer

Let's take a look at a few implications of using IP Packets. Some of these effects might be reproducible without packets, depending on the techniques used, but they fall very naturally out of using packets.

- Notice that dividing all data into small sized packets makes it very easy to share a network.
    - If I'm in the middle of downloading a copy of the movie "The Fellowship of the Ring" and my roommate is trying to look up something on the web for his Chemistry research, he doesn't have to wait until I've downloaded all 6.96 Gigabytes of the movie. The movie will be broken into small sized packets and his Chemistry webpages can be interspersed between the many 64-kilobyte packets needed for my movie.

- The packets make it easy to store up some of the movie in advance, make an estimate on how fast the movie is downloading, and start playing the movie, before all the packets arrive.

- In a similar vein, if I'm using the Internet to video conference with someone in Stanford's Florence program, the voice and video is getting shipped in packets. As we've seen, some of those packets may get lost, or they may take longer to get there than other packets. The video conferencing program may at some point decide that waiting for those additional packets to arrive will disrupt the conversation more than playing the incomplete packets currently available.
    - I'll get garbled speech, missing video frames, or both.

Packet Switching and Circuit Switching

Let's take a closer look at that last example. Traditionally making phone calls overseas is quite expensive. In fact, I recall paying $1/minute to talk to a friend when she went overseas to Korea to teach. How are technologies like FaceTime or Skype free?

- Traditionally phone calls use a technology called *Circuit Switching*. In Circuit Switching a complete telephone circuit is reserved exclusively for each person's phone call. When I'm talking to my friend in Korea, there are physical wires between here and Korea reserved entirely for our conversation.
    - Whether we're talking away taking full advantage of the line, or upset and not really saying anything to each other, either way, the line is completely ours.
- Skype and FaceTime use ***Packet Switching***. The conversation is broken up into IP Packets. This means that the line is shared with everyone else's packets.
- Because the line is shared, it's much less resource intense and therefore much cheaper.

- Sharing lines is so much more efficient, that many companies (including Stanford) have switched their internal phone lines to use packet switching instead of circuit switching. Telephones in Stanford offices are now IP Phones rather than traditional phones. These IP Phones break our voice signal into IP packets and send them through lines that are shared with packets from other phone conversations.

- In fact, telephone companies are also changing from Circuit Switching to Packet Switching, so that traditional phone call you make, might actually be carried by packets, just like Skype.

Additional Network Topics

Here are a few more network topics you may run into.

Internet vs. Intranet

You may hear the name Intranet, particularly used within the term Corporate Intranet. So what is an Intranet?

An Intranet is a network that uses Internet protocols such as TCP/IP in order to communicate between the computers in the Intranet. A Corporate Intranet is a network that uses Internet protocols but is entirely internal to the company. Intranets may be connected to the wider Internet, typically through a set of firewalls computers, which are used for security.

Transport Layer Security (TLS)

TLS is a protocol built on top of transport-layer protocols (typically TCP). It provides for authentication of the party we are communicating with and encrypts the messages sent. We'll talk more about authentication and encryption in our lecture series on computer security later in the quarter.

Application level protocols can be implemented on top of TLS. For example, HTTPS – a secure version of the HTTP protocol used on the web – will run on top of TLS allowing us to communicate with our bank without fear of eavesdropping.

You may also hear of ***SSL (Secure Socket Layer)*** which is the predecessor to TLS.