49202 Communication Protocols

Introduction and Overview

Daniel R. Franklin

Faculty of Engineering & IT

February 26, 2024

Telecommunications Networks

- The goal of a telecommunications network is the transfer of information from one place to another.
- Different types of information may have different requirements - for example, accessing a web page is very different to making a voice call or performing remote robotic surgery!
  - We may have different requirements in terms of (주파수)대역폭 *bandwidth* (throughput), *latency* (delay) or *reliability* (data loss/damage)
- And many different technologies are used for implementing telecommunications networks:
  - Ethernet, WiFi
  - Cellular networks (4G/5G etc.)
  - Satellite networks - e.g. SpaceX's Starlink, various geostationary satellite networks and links
- How can we make them all work together?
- In fact, how can we *not care at all* about the type of network that your data flows through? **I just want it to work!!**

The Internet

- What is the Internet? 인터넷이 위의 고민을 해결하는 방법이다
  - In short, a universal network of diverse networks - the name *Internet* comes from the term *inter-network*.
- It physically consists of many interconnected networks - local area networks (LANs) and wide area networks (WANs) - which are joined together via network links
- It is *distributed* - it is not run by a single organisation, but rather by a vast number of local administrative domains
  - These are known as *autonomous systems* (ASs) - examples include the UTS network, the networks of Internet service providers such as Telstra or TPG, many large businesses
  - See here for a list of ASs in Australia: `https://ipinfo.io/countries/au`
- An AS is a single administrative domain, and typically manages many internal, interconnected local area networks.
- These may all be of different types and capacities

A layered approach

- What we would really like is for our applications - the things that we want to *use* over this internetwork - to not have to worry *at all* about what sort of network they are sitting on top of
- A web browser is a web browser. We don't care if it is being used over an Ethernet LAN or a cellular network or whatever else.
- The solution to this problem is to *abstract* away the 기초를 이루는 underlying network by creating a stack of *functional layers*
- There are various ways to do this - the International Standards Organisation (ISO) created a model called the Open Systems Interconnect (OSI) protocol stack, which is often used as a sort of reference model. It has 7 layers.
- But as often happens, what a committee decides often gets simplified in the real world. So the Internet, as it exists today, uses a somewhat simpler model with five layers

A layered approach

- These layers are:
  1. The physical layer - which represents bits (numbers - zeros and ones) as physical signals which can be transmitted
  2. The data link layer - which organises these bits into some sort of logical structure for delivery within a single local network
  3. The network layer - which transfers information between different local networks - this is what we call the internetworking protocol layer - IP
  4. The transport layer - which provides additional functions such as reliability and application multiplexing - most commonly either TCP or UDP, depending on application requirements
  5. The application layer - i.e what you actually interact with - web browsers/servers, Zoom, network games etc.
- This suite of 5 layers is known as **TCP/IP** (although it also includes UDP, and IP may be IPv4 or IPv6).

Physical communications

- Every end-user device - phones, laptops, Internet-connected appliances etc. - accesses the Internet by some sort of physical interface
- This interface may be a wire (for example, Ethernet), but it is often wireless - e.g. a cellular network connection or WiFi interface
- A network interface is fundamentally a device which is capable of transferring *information* (a stream of binary digits or bits) from one place to another
- It does this by creating some sort of **physical representation** of the information - for example:
  - A voltage across a pair of copper wires, where a positive voltage represents a 1 and a negative voltage represents a 0
  - A frequency-modulated (FM) radio pulse, where a signal with frequency $f_c + f_x$ represents a 1, and $f_c - f_x$ represents a 0
  - An optical pulse, where different wavelengths (colours) represents 0 or 1

Physical communications

- This *physical* representation of information (numbers) allows information to be transmitted as a signal of some sort from a sender to a receiver
- Provided that both ends agree on what the physical representations *mean*, the receiver can interpret the physical signal as information
- The physical layer refers the agreed representation of information as a signal, and include definitions for things like the voltages, frequencies, timing sequences etc. that are used - without that agreement, information cannot be transferred.
- This solves part of the problem of shifting information from one place to another.
- Of course, this 'information' is only a string of zeros and ones. If you want to do something useful with this stream you need some additional features!

The data link layer

- What else do we need? Remember, we want to *communicate*.
- The stream of bits might include *errors* - missing or flipped bits
- When the transmitter is silent, the receiver may still pick up and amplify noise - which would look like random bits. So how do we know where an actual message starts and ends?
- If we are doing two-way communication over a *shared medium* (e.g. a radio channel) we may need to decide how to take turns, or what to do if two transmitters send a message at the same time
- And if there are more than one transmitter/receiver present, we probably want to be able to *identify* who is sending and who is receiving.

The data link layer

- All of these problems can be solved by adding *structure* to a bit stream. This structure is defined in the *data link layer* and typically includes:
  - *A start sequence* which can unambiguously be interpreted as the start of a block of data
  - *Addresses* for both sender and receiver
  - *Error detection codes* to ensure that corrupted data can be discarded
  - *Some sort of length indicator or end-of-data flag* so we know where this data block ends
  - **Most importantly**: *the payload* - this is the whole point of doing all this work!
- We may also have things like protocol version identifiers, packet type identifiers and other interesting metadata.
- By adding this structure - known as a *frame* - we can now send a payload to a given destination, and that destination can detect the message, extract the payload and check it for errors.

Internetworking - the network layer

- So far so good - we have local connectivity!
- But now we have a problem. How can I get data from a device in a WiFi network to a server in an Ethernet network?
- We need to provide a common language that hosts in all different types of networks can speak
- The addresses that we were using before to identify a destination or source in a local network are no good now - how does an Ethernet network recognise a WiFi address?
- So, we need a *second* set of addresses. These are the **network layer addresses**, known as **IP (internet protocol) addresses**.
- The IP address (more or less) uniquely (not really) identifies a host *globally* (its complicated)

The network layer

발생하다

- There are a few other complications which can arise in moving data between networks
- Sometimes the maximum size that a chunk of data can be in one network is different to the next
  - We may need to split up a packet into smaller chunks
- Maybe some errors occurred - we don't necessarily trust all the LANs to do perfect error detection! What if the Layer 3 header has been corrupted?
  - So we might need to do some error detection just for the layer 3 header, to make sure that at least we are sending the packet to the right destination

The network layer

- And how do we choose the path that a packet takes if we have to travel through many other networks to our destination?
  - Wouldn't it be nice if this could be figured out automatically?
  - If we had a way to do this, it could even repair local damage by finding alternative paths to a given destination
  - In fact, this was one of the original reasons the Internet was invented - routing around the damage which might occur during nuclear war.
- This problem is solved, not by the internet protocol itself, but by a number of additional protocols called *routing protocols* which discover and disseminate information about network connectivity.
- There are many different routing protocols and we will spend a lot of time discussing them!

The transport layer

- Now we can get data from a host in one network to a host in another! Fantastic!
- But we may need some more functionality:
  - Layer three *does not care at all* about reliability of packet delivery - corrupted packets will be *discarded* 오염된, 쓸모없는
  - This might be a problem if you are sending a credit card number or performing telesurgery!
  - On the other hand, some applications (e.g. Zoom, Skype or a network game) might not care at all!
  - But one thing that we *definitely* care about is making sure that you aren't sending e-mails to Zoom or network game packets to Paypal when you're trying to make an online purchase

The transport layer

- So, we have a few alternative transport layers available, depending on what you want to do. They can all do *application multiplexing/demultiplexing* - directing the right packet to the right application.
- Some (such as TCP) also provide reliability, which means:
  - Guaranteed delivery of all of the packets in a stream (it will resend anything which is lost) and
  - *In-order* delivery of all the packets in the stream
  - Behaviour which allows the rate of packet transmission to be controlled so as to avoid *congestion* in network bottlenecks and *data overflow* at the receiver
- If you don't care about any or all of those things, you might just use UDP, which basically only does application multiplexing/demultiplexing.

The application layer

- So - the first four layers let us represent information as bits, add structure so we can reliably exchange data between hosts in a single network, add *more* structure so we can send data from a host in one network to a host in another network, AND let us choose our application, with optional reliability features.
- This brings us to the whole purpose of the Internet - the application layer.
- This is the layer you are most familiar with - things like e-mail, web browsers, Zoom, Teams, online games, instant messaging applications, BitTorrent and many others
- Each of these has its own protocol which builds on either TCP or UDP (depending on the application's requirements) plus IP.
- These applications *do not care* about the underlying data link or physical layer.
- We also have a range of *helper application-layer protocols*, for services such as DNS (for translating symbolic names to IP addresses) and DHCP (to ease deployment of networks by automatically allocating IP addresses).

Traversing the layers - encapsulation and decapsulation

- When an application (such as a web browser) sends data to a web server, a chunk of data is *encapsulated* in a data structure for the next layer down.
- The web browser creates a TCP segment, with its own header (which contains TCP meta-data)
- This then gets passed to the IP layer, which adds its own header as well - the TCP segment (payload + TCP segment header) is the payload of the IP datagram
- This is then passed to the data link layer, where the appropriate frame header (and possibly trailer) are added (e.g. an Ethernet frame header/trailer)
- Finally, the physical layer converts this (which is just a block of zeros and ones) into some sort of physical signal, which can be transmitted

Traversing the layers - encapsulation and decapsulation

- When the transmitted frame arrives at the intended receiver (within the local network), it is decoded and converted back into a Layer 2 (data link layer) frame.
- If there are no errors, the frame header is discarded and the payload (the IP datagram) gets passed on to Layer 3.
- If this is the final destination (which the receiver checks by looking at the destination IP address and comparing it with its own), it will be passed on to the transport layer for any required processing, and then the transport layer will forward it to the appropriate application (in this case, a web server)
- If it is NOT the final destination, we figure out where to send it next (typically some other network)
- We then encapsulate the IP datagram in another (different!) Layer 2 frame, with a different source/destination Layer 2 address, and send it on its way
- The source and destination IP address, however, *do not change* (unless they do, but more on that later!!)