CSC458

Computer Networking Systems

Last updated October 8, 2025

Contents

1 Delivering Packets

Sending large data over the internet breaks them in packets, which are sent over the internet to the recipient (well, at least the admin of the recipient as the recipient is not supposed to know that this happens).

When you send something, you can get an estimate when a packet is supposed to be delivered, however

- There are no guarantees that packages will be delivered
- If so, packets are not guaranteed to be in order

Since packet sends can fail:

- Receiver needs to acknowledge receives (positive – got it, negative – didn't get it)
- Needs to keep local copies
- What if an acknowledgement is lost?

Moreover, with the internet:

- Data integrity is not guaranteed. Random bits can flip.
- Packets can be fragmented. This is a serious possibility. What if a packet reaches a network with a smaller packet size limitation?
  - E.g. you send a 9KB packet through a network that can't have packets that large, it results in fragmentation
- Packets may be duplicated. The network may end up duplicating the packets you are sending.

1.1 Layering in the internet

- Application layer
  - You or the application you run do this

- Transport layer
  - Admin / Network / Operating system
  - On behalf of the application, carries the responsibility of providing reliable, in-sequence, E2E delivery of data.
- Network layer
  - Provide the best effort service (analogue to the postal service). No sequencing and reliability, just sends the packages from one side of the globe to the other side of the globe
  - Packets may go through multiple network nodes
- Link layer
  - Packet is injected on one side of the link, ejected from the other side of the link. Analogue to actual postal office trucks.

1.2 How to transfer a file

1. FTP application asks through an **API** (this is socket programming), you make certain calls to the OS that you want to send data to someone else in the network. A desire to send something to host B through a system call.
2. The **TCP** (transmission control protocol) – A TCP connection setup packet is set up. It's a packet to say I want to send something. No content, header is a connection setup header.
3. TCP sends packet to the **IP** layer. The IP receives the TCP packet. It encapsulates the TCP packet in an IP packet. The data portion of the IP packet is the TCP packet, with a newer header: destination address, source address, and protocol.
4. The IP packet is delivered to the link layer. It then puts the packet in the data portion of an ethernet (**MAC**) packet. A new header is created.
5. A router receives the package. It opens up the ethernet packet, and looks at the destination of the IP packet. It then re-encapsulates the IP packet in another ethernet layer and sends it to the next router.

6. This process repeats.
7. ...
8. When the ethernet packet reaches the final router, it is opened up. The receiving host verifies that it is for itself. The transport layer opens the packet. It sees that there is a connection setup request. And then it sends back an acknowledgement.

Each layer takes something, puts it in another envelope, then sends it. Does it belongs to me? Open it up. Does not? Keep sending it.

Layers talk to each other. It's just layers below it does the heavy lifting.

1.3 Packet Switching and Circuit switching

Early on, for phone systems, we had circuit-switched systems. We have a network, with a source A who wants to communicate with a destination B. It's the method used by the telephone network. Circuit switching dedicates a path for you. That's too old and therefore expensive, since the path is dedicated to you.

Packet switching is used by the internet. Each packet is individually routed, every piece is put in a packet so it can find its way to the destination. It is the method that makes the internet possible.

2 Link Layer

Media is what is used to send bytes from one place to another. Can either be:

- Wire
- Fiber
- Wireless

In the physical layer, you encode bytes, and to send bytes from one adapter to another, you'll have to send them over and generate an analogue waveform from digital data at a transmitter and the receiver must be able to recover it. There are different ways of mapping bits to signals.

2.1 Framing

We can't just send garbage bits, since the receiver gets bits all the time. We need to signal the receiver that they are getting useful information. This is done by framing.

When you send a packet, it is put into a frame before the packet is sent off. The frame consists of:

- The sentinel 0x7E
- The actual frame content
- The ending sentinel 0x7F

To prevent the ending sentinel from being injected, replace all instances of 0x7F with 0x7FX... there will always be a way.

```
7E ... 7F
```

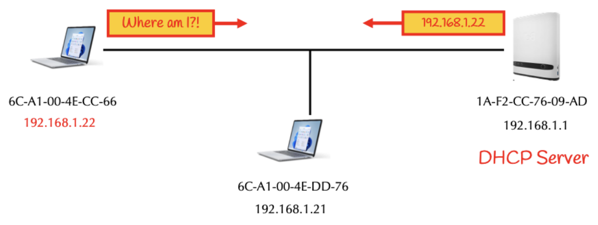
2.2 Mac Address

All your adaptors are built with unique mac addresses, which for the purposes of this course, assume that they cannot be changed. It is a flat name space of 48 bits, in the form AA-BB-CC-DD-EE-FF in hex. It works like a social insurance number.

The first 24 bits of a mac address are usually assigned by vendors (e.g. Dell) by the IEEE. The last 24 bits are assigned by the vendor from its block. There is a broadcast mac address, which requests that a frame be sent to **all** adapters.

2.3 IP Addresses

IP addresses can change. Let's assume, for now, that your only method of connection is through a wire and these are the only three devices connected by that wire.



If on the same network, there is an ARP table that maps an IP address to the mac address. That's how you know someone's mac address.

2.4 Model of a Link / Calculating Delays

You want to send:

- An M Bit Message

It reaches the recipient

- At a rate of R bits per second
- At a delay of D seconds

So, how long does it take to send a message?

Firstly, we need to look at the delays:

- **Propagation delay (D)**
  - Propagation delay = $\frac{\text{Distance}}{\text{speed of light in media}}$
- **Transmission delay (M/R)**
  - $\frac{M}{R}$ (message size / rate)
  - How quickly can you inject the message on the wire. In other words, it takes this long for the message to come out on the other side of the wire.
  - Unit analysis: $\frac{[\text{bits}]}{[\text{bits}]\left[\text{seconds}^{-1}\right]}$ = seconds
- **Queueing delay**

So therefore, we get:

$$\text{LATENCY} = \text{Propagation delay} + \text{Transmit delay} + \text{Queue delay}$$

2.4.1 Example Calculations

If, for now, we neglect queueing delay, then as long as we can calculate the propagation and transmission delay, then we can calculate the latency. For example:

If we have $D = 0.010$ s, $R = 56000$ b, $M = 8000$ b, then latency $= 0.010 + \underbrace{\left(\frac{8000}{56000}\right)}_{\text{transmission delay}} =$ 0.153 s

If we have a T3 (45Mbps) line, we have $D = 0.050, R = 45 \times 10^6,\ M = 8000$, then latency $= 0.050 + \frac{8000}{45 \times 10^6} = 0.050$ milliseconds – in this case, since the rate is so high then we only suffer the propagation delay D.

2.5 Round Trip Time

Latency is typically the one-way delay over a link. A round trip is the time that holds sending a signal and getting it back. Therefore, the round-trip time (RTT) is **twice the one-way delay.**

2.6 Throughput

The measure of the system's ability to "pump out" data. This is *not* the same as bandwidth.

$$\text{Throughput} = \frac{\text{Transfer size}}{\text{Transfer time}}$$

This is **not** the same as bandwidth. It's usually lower than bandwidth, despite having the same units, since it has to account for more things. Note that:

- Transfer time = sum of
  - Time to get "started shipping the bits" (new thing)
  - Time to ship the bits (your typical bandwidth)
  - Time to get "stopped shipping the bits" (new)

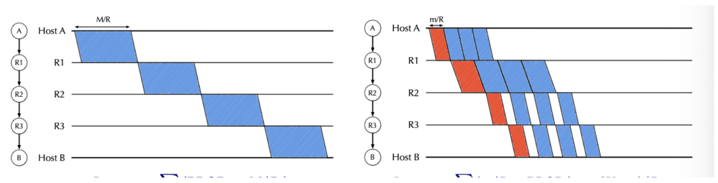
2.7 Data-In-Flight

Consider a 1 bit per second network. Suppose that latency is 16 seconds. Therefore, the network, or moreover "data in flight" is up to **1 b/s × 16s = 16b**. This is known as the bandwidth-delay product. Therefore, we need to have sent 16 bytes (if we keep sending) before we can hear back from the receiver.

2.8 Packet Switching

When data is sent from a source to a destination, it might go through multiple routers. It takes time for data to be given to a router before the router can transmit the data to the next one.

If R_i is the rate of which router i seconds data, take a look at this graph ($R_{\text{min}} = \min_i R_i$):



Without parallelizing the sends, the latency is

$$\sum_i \left(\underbrace{\text{PROP}_i}_{\text{prop delay of router I}} + \frac{M}{R_i} \right)$$

However, with parallelizing the seconds, it ends up being

$$\sum_i \left(\frac{m}{R_i} + \text{PROP}_i \right) + \frac{m(K-1)}{R_{\text{min}}}$$

$$= \left(m \sum_i \frac{1}{R_i} \right) + \left(\sum_i \text{PROP}_i \right) + \left(\frac{m(K-1)}{\min_i R_i} \right)$$

Where M is the message size, m is the packet size $\frac{M}{\text{packet count}}$, and K is the number of packets in that message.

Breaking messages into packets allow for parallel transmission across all links, reducing end-to-end latency. It prevents a link from being "hogged" for a long time by one message.

If you look at this equation:

$$\underbrace{\left(m \sum_i \frac{1}{R_i} \right)}_{1} + \underbrace{\left(\sum_i \text{PROP}_i \right)}_{2} + \underbrace{\left(\frac{m(K-1)}{\min_i R_i} \right)}_{3}$$

- Term 1 is the width of each individual trapezoid.
- Term 2 is the skew of each individual trapezoid.
- Term 3 accounts for the bottleneck.

2.9 Store-And-Forward vs. Cut Through Switching

You'll know that for store and forward, it's going to be

$$\sum \text{PROP} + \sum \text{TRANSIMIT}$$

Through all hops.

For cut-through switching, it's

$$\sum \text{PROP} + \max(\text{TRANSMIT})$$

Focus on the bottleneck. And the things that can't be made any faster.

2.10 Queueing Delay

Because the egress link is not always free when a packet arrives, it might be queued in a buffer. If the network is busy, packets might take forever.

And you have no way of computing the queueing delay, because it depends on user behavior.

$$\text{E2E latency} = \sum_i (\text{TRANSPORT}_i + \text{PROP}_i + Q_i)$$

2.11 Increasing Link Bandwidth

- Increasing wire count
- Increasing bits per second

3 Error Detection and Correction

How do we detect and correct messages that don't make it past transmission? The responsibility for making these cuts across the different layers. What happens when a bit gets flipped?

3.1 Ways to Correct Errors

- Detect and retransmit (ARQ)

  - Less expensive unless too many packets have errors
  - (Low overhead, less and simpler processing)
- Error correcting codes
  - Transmitting these are more expensive, but cheaper if errors do occur
  - Reduces the number of retransmissions

Most data networks today use error detection, not error correction. Being said:

- If errors are frequent but small, go for correction.
- If a single message tends to get corrupted by a lot, go for error detection.

3.2 Attempts to Fix or Detect Errors

We can send two copies of the data.

If each "copy" is different, we know there's an error.

But we don't know how to fix it.

Being said, if we send three copies of the data, as long as the bit flips are not too much, we can correct errors.

3.3 Encoding to Detect Errors

Of course, this requires understanding from both the sender and the receiver. You can re-encode your message with redundancy such that your message can be checked and potentially corrected for errors. The simplest way, as stated before, is by duplicating the message. That might not be the most effective. At least, here are some new concepts.

Errors must not turn one valid codeword into another valid codeword, or we cannot detect them

3.3.1 Hamming Distance

The hamming distance of a code is the smallest number of bit differences that turn any one codeword to another

If a code consists of a bunch of codewords, it's the hamming distance of the codeword with the lowest hamming distance.

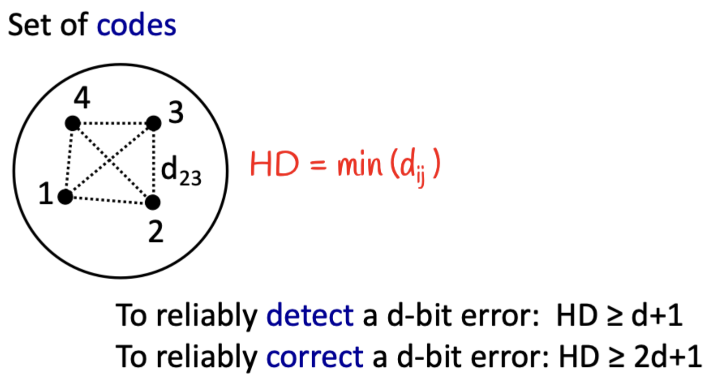
3.3.2 Hamming Distance Theorems

For a code with distance at least $d + 1$, d errors can be detected.

Proof: $c_1 \xrightarrow{d} c_1' \xrightarrow{1} c_2' \xrightarrow{d} c_2$ in the worst case.

For a code with distance at least $2d + 1$, d errors can be corrected.

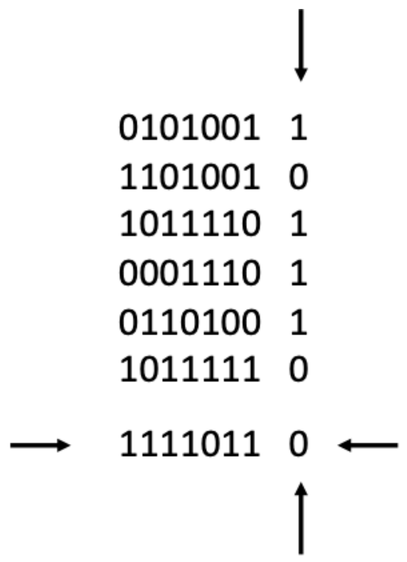
Proof: $c_1 \xrightarrow{d} c_1' \xrightarrow{\text{at least } d+1} c_2$ in the worst case.



To reliably detect a d-bit error: HD ≥ d+1
To reliably correct a d-bit error: HD ≥ 2d+1

3.3.3 Parity Bits

Start with n bits and add another so that the total number of 1s is even. This can be done consistently. E.g. 0110010 → 01100101. This will detect an odd number of bit errors, but not an even one. This does not correct any errors. Can this be improved? By having 2D parity.



This one detects all 1, 2, 3 bit errors and many errors with > 3 bits, and corrects all 1-bit errors. The new numbers ensure that each column and row have an even number of 1s.

3.4 Checksums

No, your checksum does not need to be explicit. The idea of checksums is for you to add up all your data, then send the negative of it along with your data. Explicitly:

- The **checksum** is the 1s complement (negative) of the 1s complement sum of the data

Where:

- 1s complement flip the bits
- 1s complement sum: regular bitwise sum, but if there's an overflow carry-on, wrap it to the back (to the LSB).
  - I'm presuming this halts.

When your checksum is included, when the recipient sums up all the data, it should sum to 0.

4 Interconnecting LANs

Instead of having two devices, what if we have many devices sharing the same medium. The most dominant technology is ethernet. It has been one of the most successful protocols that was designed to connect machines in a local area network. It has both wireless and wired connections, supports a range of speeds (from 10Mbps to 400Gbps), and maybe way more over one connection.

There are multiple devices connecting over a shared medium. That creates some problems.

4.1 Ethernet uses CSMA/CD

So, we have the same medium used across different devices. Let's call the space the **collision domain**. If two devices try to talk at the same time, there will be a collision in the communication. In a shared medium, only once device can talk at a time. What does ethernet do?

- Carrier sense: Wait until link to be idle.
  - You can start talking as the link is idle.
  - Channel idle: start transmitting
  - Channel busy: wait until idle
  - This method gets to live due to its simplicity.
- Collision detection
  - What happens when two devices talk at the same time?
  - While you're talking, you will listen to the medium as well. If you detect a collision, pause, stop, wait for a random time, and then send a jam signal (some noise that ensures the other side knows a collision occurs).
  - If no collisions occur, then the transmission can complete as normal.
- After a collision

  - After a collision, wait for a random time before trying again.
  - After the mth collision, choose K randomly between 0 to $2^m - 1$ and wait for $K \times 512$ bit times (multiply by the time it takes to transmit 512 bits) before trying again.
  - m get to reset to 0 after transmitting something without a collision

This is a **simple** but very effective solution. You don't have to add complex systems. This runs on the assumption that none of the nodes are malicious.

There are reasons behind every single decision that has been made. The fact that you have to wait for $K \times 512$, 512 is the minimum frame size for ethernet. You don't want to go below that. But that creates some interesting problems.

4.2 Limitations on Ethernet Length

Suppose A sends a message to B. How long does it take? The latency of this link is D seconds. How long does it take for the message to get from A to B:

- At time 0, A sends the message.
- If B is silent at time D, then B hears that A is talking.
  - If, at time $D - \varepsilon$, B has not heard from A, but B starts talking. ε seconds later, B realizes there is a collision. A doesn't know. That's why B is going to send a jam signal (noise) to make A realize that there was a collision.
  - The Jam signal returns to A at time $2D$. At the time $2D$, A realizes when there is a collision.
- A's transmission is successful after time $2D$ when the message was sent.
- There is a cap on the size of ethernet connection. If the minimum size of the packet is 512 bits, 25000 meters is the maximum length of the wire.

4.3 The Ethernet Frame Structure

Destination comes before source because the destination is more important than the source. The structure is as follows:

- Preamble
- Destination address
- Source address
- Type
  - Indicates the higher layer protocol (usually IP)
- Data
- CRC
  - Cyclic redundancy test

Ethernet is a connectionless service. No handshaking is required. You just send the message.

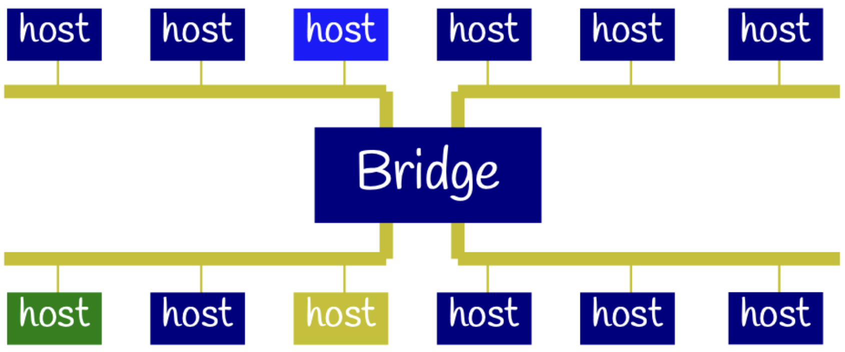
Ethernet is unreliable. Receiving adapter doesn't sand ACKs or NACKs.

Packets passed to the network layer may have gaps. Gaps will be filled if application is using TCP. Otherwise, application will see the gaps.

4.4 Link Layers: Bridges

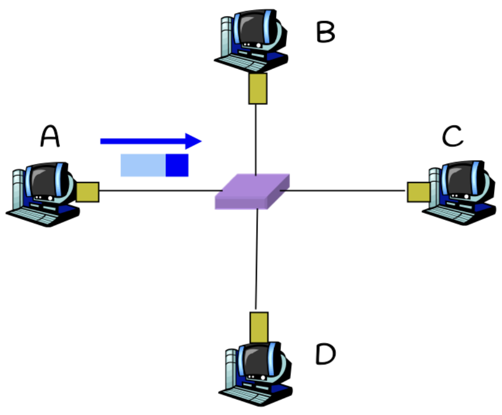
Ethernet medium can't be too long. A single ethernet network is a collision domain, where two devices cannot talk at the same time.

When you have a **bridge**, the job of the bridge is to transfer packets to another network if the receiver happens to be on the other side of the network. The bridge can be over more than one ethernet network. Each segment is in its own collision domain, and the bridge only delivers messages if the other side is idle.



This breaks up collision domains, allowing the network to be expanded.

4.5 Link Layer: Switches



Switches connect individual computers. Essentially the same as a bridge... not quite.

- Dedicated access: Host has a direct connection to the switch rather than a shared LAN connection.
- Full duplex
  - Each connection can be sent in both directions
- Completely avoids collisions.

Buffering the data stored in switches takes time. Buffering delay can be a high fraction of the total delay. To solve this problem, use cut-through switching. As soon as header received, keep transmitting when the outgoing link is idle.

But what happens if links don't have the same rate? You can't transmit something you haven't received, so if the in link has the same or faster rate than the out rate.

The ethernet packet has a CRC field that checks for errors. These are issues to be dealt with, but if you fix these problems using cut-through switching, it solves a lot of the problems.

4.6 Self-Learning

The switch stores a table mapping ports to device. That table starts of empty, so the switch does not know what every port maps to.

Whenever a switch receives a packet:

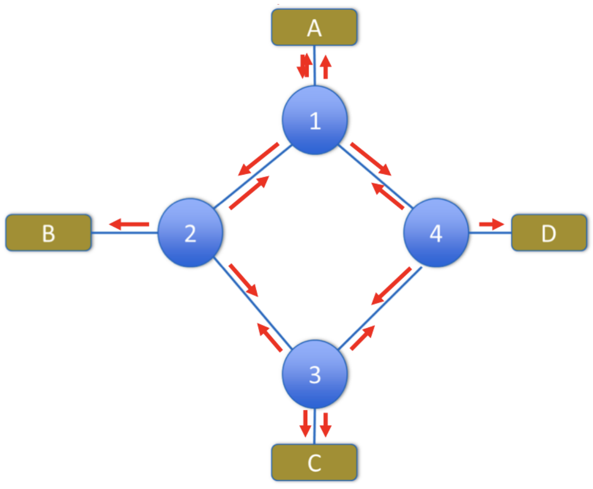
- When *A* sends a packet to the switch, the switch learns where *A* is.
- If the switch has to forward a packet to a destination it does not know, if has to flood by sending it to everyone.
- **ABSOLUTELY NOTHING MORE, NOT EVEN THE RECIPIENT.** If the recipient doesn't do anything, well that's the most wasteful thing that could happen.

Precisely:

- When a switch receives a frame
- If entry found for destination then
  - If destination is on the segment from which frame arrived
    - Then drop the frame
    - Else forward the frame on interface indicated
- Else flood

Flooding does not make me know anything more. Which means:

4.6.1 Flooding can create loops



You get the picture.

4.6.2 How do we fix the looping problem? (By making Spanning Trees)

- Even though we have a network, we should remove some of the links. Since a network can be represented as a graph, if the graph has cycles, packets can rotate forever.
- **So convert the graph into a spanning tree.**

There are multiple spanning trees that can be composed from a graph (there are different ways).

The spanning tree protocol is a protocol by which switches and ridges construct a spanning tree. It has:

- Zero configuration
- Self-healing

Should have some constraints for backwards compatibility.

4.6.3 Constructing a Spanning Tree

We need a distributed algorithm. Switches cooperate to build the spanning tree and adapt automatically when failures occur.

Key ingredients of the algorithm:

- The root. One of the nodes in the graph needs to be the root.
- Each switch must identify the interfaces that are on the shortest path to the root
- Exclude all other interfaces

Nope, not a good algorithm.

4.6.4 The Spanning Tree Algorithm

Each node broadcasts a message $(Y,\ d,\ X)$.

- Proposes Y as a root
- From node X
- Saying that the distance from the root (between Y and X is d)

And this keeps happening:

- Initially, each switch proposes itself as the root
  - For all $X \in$ Switches, X announces $\left(\underset{\text{root}}{X}, \underset{\text{dist}}{0}, \underset{\text{self}}{X}\right)$
- Switches update their view of the root and their distance to the root, where if they hear someone else say the root is different, they listen if the broadcasted root's ID is lower.
  - At each switch Z, whenever a message (Y, d, X) is received from X:
    * If Y's ID < current root:
      - Root = Y
      - Shortest distance to root = $d + 1$

    * If Y's ID = current root and shortest distance to root > $\underbrace{d}_{\text{from message}} + 1$
        · Shortest distance to root = $d + 1$
  - If root changed or shortest distance to root changed:
    * Send all neighbors message $(Y,$ shortest dist to root, $Z)$

Repeat until no change. Identify interfaces (links) not on a shortest path to the root and exclude them from the spanning tree.

Sometimes, there is a tie in the algorithm. When there are multiple shortest paths to the root (e.g. node 4 and node 5 have the same distance to the root), choose the path to the neighbor that has the smallest identifier.

This algorithm converges, but there are potential things we need to resolve.

- What if the root node or any other switch in the node fails?
  - Need to elect a new root. This time, use the robust spanning tree algorithm.

The algorithm must react to react to failures.

4.7 Robust Spanning Tree Algorithm

Root switch continues sending messages, periodically reannouncing itself as the root. Other switches continue forwarding messages.

Errors are detected through timeout (soft state). Switches wait to hear from others, and eventually times out and claims to be the root.

This works if a link or any other part of the network goes down. This message gets sent to all of the neighbors of the node, to all edges regardless of whether it's active or not.

30 seconds is an arbitrary number. It can be 30 seconds, 3 seconds, 0.3 seconds. The overhead and the reliability changes have impacts. If you do this more often, you'll get faster convergence but more overhead.

4.8 Spanning Tree Protocol Strengths and Weaknesses

Strengths

- Plug n' play: zero configuration
- Simple
- Cheap

Weaknesses:

- Wasted bandwidth (especially the default 30 second delay of the root broadcasting)
- Delay in reestablishing spanning tree
- Slow to react to host movement, need timeout
- Poor predictability: depending on location of root and traffic pattern

This algorithm is not optimal in any sense. It can give longer paths depending on node IDs and how the algorithm works.

5 The IP Layer

The internet protocol, in the layer above the link layer: the network layer.

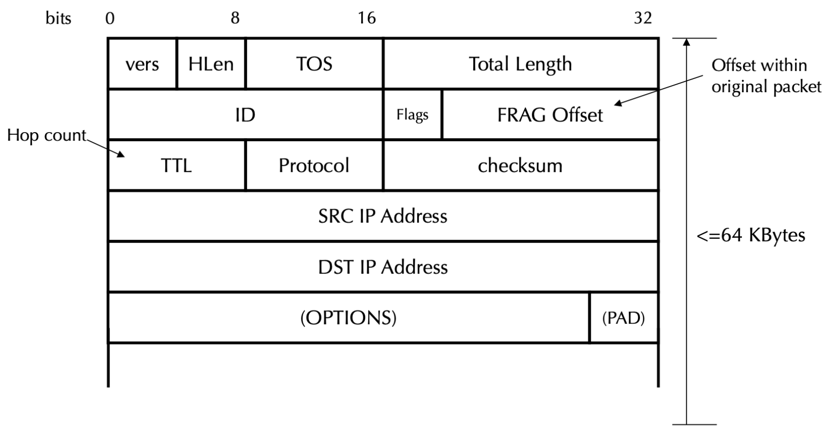
5.1 Characteristics

Like ethernet, IP is connectionless. You are not establishing connection between sender and receiver. You just send it. As long as you have packets, you send them. So:

- Connectionless: there can be mis-sequencing
  - It receives individual packets and delivers packets, but it never is the one that establishes the path between the source and destination.
- Unreliable: may drop packets

- Best efforts: but only if necessary. Only drops if it absolutely has to.
- Datagram: individually routed
  - Going to put everything inside the envelope

5.2 The IP Datagram



Each ward (row) is 32 bits. HLen is the row count. Max. is 60 bytes for the header length.

The maximum, IP packet size is 2^{16} = 65536

5.3 IP Addresses

IPV4 addresses are 32 bits long, usually represented in dotted decimal notation. Every interface has a unique IP address. IP addresses are not linked to devices.

Hence:

- A computer might have two or more IP addresses. A computer can connect to more than one network.
- A router has many IP addresses.

IP addresses are assigned statically or dynamically.

IPV6 addresses are 128 bits long.

5.4 Sending Messages

If we want to send packets from A to B:

- Same network:
  - No need to route, *A* can directly send the packet over the link layer to *B*
- Different network:
  - *A* must send the packet to a router (which has a different IP) that can forward it towards *B*

These scenarios are different!!

5.5 Are Hosts on the network

Every IP address is divided into two parts:

- The network ID
- The host ID

The network ID can be 8 bits, and the host ID can be 24 bits. Or they can be equally split.

If I know my network ID is 1.2.3, then if I have the destination ID, I can compare the start of the address. Therefore I can know that the person I'm talking to is in my network.

How do I know which part of my IP address is the network ID or not? I have to be told that. When I configure my network or get my IP address from DHCP, I get a network mask. It tells me how many bits is in the ID. For example:

- If my IP address is 1.2.3.4/8, it means that the first 8 bits in my IP address is the network. I could also be given a bitwise mask. Therefore taking the & of this, with my IP address, I get the network ID.
- This means I can easily check if someone else's IP address matches my network ID.

This means I can have anything from 1 to 31 bits of network ID. My network mask can change this.

5.5.1 Historical Aside

In early days of the internet, instead of doing this, they didn't want to share the network mask, instead they want everything to be self-contained. The way they did it, is to define different classes of addresses.

- Any IP address that starts with 0 is considered class "A", and these addresses have 8 bits of network ID followed by 24 bits of HostID. This means you can only have 128 class A network… that does not sound like a lot. You'd run out of them in an instant.
- And so on for class B… the entire country of China would have less IP addresses than MIT.

Then people figured out that this didn't work. So, instead of storing everything in the IP address, how about let the network be identified by the network mask?

5.6 Subnetting and Supernetting

A network can allocate more bits at the front to make new sub-networks. If UofT has 8 bits for its network ID, it can break its network into sub networks:

- Network ID + 00 is the CS department
- Network + 01 is the ECE department
- And so on

We can also Supernet. This can be the reverse. Nobody else needs to care about the subnetting that occurs in UofT addresses. You only need to know UofT's network ID.

Subnetting and supernetting has been very helpful – they allow hiding details of internal network from the rest of the world.

5.7 DNS

Domain names are hierarchal.

TLDs are assigned by the internet corporation for assigned names and number (ICANN).

If you have an application and you want the IP address of www.eecs.berkeley.edu, your operating system is going to connect to a local DNS server, asking for the local IP address associated with the URL. If the local DNS does not have that information, it will forward you to the root domain name server (.edu), which then forwards you to berkeley.edu, which then forwards you to eecs.berkeley.edu

Berkely's DNS is responsible for anything inside Berkely.

5.8 ICMP (Internet's error reporting)

When something goes wrong, the router can provide some information. If a packet gets dropped, information can be provided to the end user. We use that information to do something very interesting. About time to live: if a packet's TTL reaches 0, usually, routers that drop the packet saying TTL reaches 0 sends the message back.

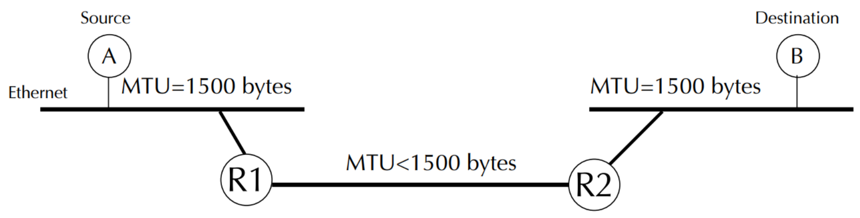
So, if you want to send a packet to Google: sending it to the first router is, that router will drop the packet and send it back. By doing this, you can learn all the router your messages go through. This is exactly what **traceroute** does. Sends a packet w/ TTL1, TTL2, and so on.

An ICMP message is an IP datagram and sends it back to the message sender.

5.9 Fragmentation

On the IP side:

Obviously data you send has to pass through multiple routers.



Links between routers have a maximum transmission unit **(MTU)**. This limits the size of packets. Can't send individual IP packets larger than the limit. Though in practice, these tend to be $\geq$ 1500 bytes.

So, if you need to send something from A to B, well, it might be routed as follows:

$$A \xrightarrow{\text{MTU=1500}} R_1 \xrightarrow{\text{MTU=500}} R_2 \xrightarrow{\text{MTU=1500}} B$$

So, let's send a packet of size 1500 bytes (for the purposes of simplicity, let's ignore the fact that packets need to be a size that is a multiple of 8, and let's assume header doesn't take any space). It can be sent to R_1 without the need to be fragmented. But the link from R_1 to R_2 can only support packets of 500 bytes.

What do you do? **Fragment it.** Split the IP datagram. Your datagram is initially 1500, it can be split into 3. Under the assumption that headers don't take space (this assumption is wrong), you can get the data across. But this also means having to put more information in the headers so that the fragmented datagrams can be reconstructed.

These two pieces of information are enough for each piece:

- **Offset:** satisfies `data[offset:offset+frag_len]` is what was sent. (`frag_len` is the size of the fragment, after all).
- **More frag:** `1` if this isn't the end of the fragment. `0` if it is. Hence, `1` for all but the last.

> There couldn't be anything but the offset. Trying to store the fragment number brings issues if you need to fragment the data again, which **will happen** if somehow your next link can only serve packets even smaller.

In practice, offsets must be a multiple of 8 (during a test, just say you're ignoring this – no need to focus on the hard detail). While the example of 1500 $\to$ 500 has nice numbers, in the case where headers do take up space, splitting up the data just becomes a bit messier just to me.

5.10 MTU Discovery

If you know the destination (can be a URL), you can figure out, across all links in the path, the largest size you can make a packet without causing any fragmentation. That's because you can include a flag which disallows fragmenting, having the routers drop the packet when it fragmented it otherwise. Try the command:

```
traceroute -F www.uwaterloo.ca 1500
traceroute -F www.uwaterloo.ca 1501
```

If the first command succeeds but the second one fails, you know that 1500 is the maximum size of a packet you can send without any fragmentation. On fail, you'll get an ICMP error message.

6 Forwarding

Done by routers, if IP datagrams need to be hopped.

6.1 Switches vs. Routers

Switches have nice properties. If you have many switches and you connect them to each other, they just work. Is it optimal? No. Depending on how you connect it, it can hurt performance, but it works.

They can do fast filtering and forwarding of frames, no matter how you connect them.

However, disadvantages of switches over routers:

- Topology restricted to a spanning tree, which is sub-optimal

- Large networks require large ARP tables. The ARP table grows with the size of the nodes. In data centers, there can be millions of devices, meaning the ARP table can be extremely large. Occasionally, you will have a broadcast. If your network is 1M nodes, you will have 1M broadcast messages throughout the network.
- Instead of doing link layer switching, you do some network layer routing. Routing happens at the network layer, not the link layer. If you are going beyond your local network, you will be using network layer routers.

6.2 Longest Prefix Match

There are too many IP addresses. IPV4 can only have 2^{32} distinct IP addresses. This is why we have network IDs and host IDs. You care more about the network. All devices in one network can be treated as the same, and therefore, if a router knows what network is being targeted they can already know the link to send it from.

Based on the router's routing table, which can be extremely large, the destination IP address must be matched against entries there.

It is typical to consider forwarding to an egress port that has a corresponding IP address entry in the IPv4 table that happens to be the prefix of the IP address we want to target (In form 123.123.123.123/SUBNET_MASK, only consider it w/ the subnet max). What if there are multiple? Match the one with the longest subnet mask. I mean the one that has the most specificity. Too vague? Don't

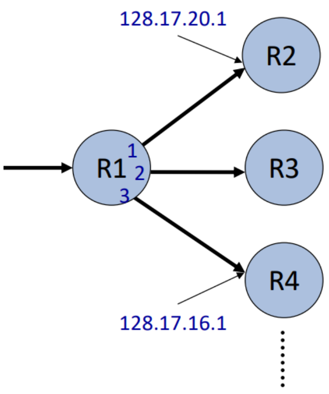
6.3 Routing Techniques

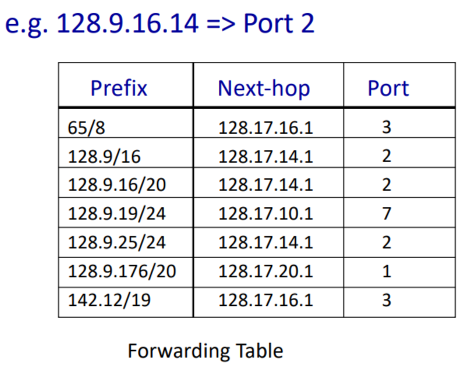
Say we have a bunch of routers in a network with a label (maybe $A, B, C, \ldots$), in a network, representable by a graph. Routers are short-sighted, so they can't see all at once. If there isn't a link between A and C, where should A route the data to? If B happens to have a link with A and C, then A can send the data to B so that B can send it to C. Therefore, A should record that whenever it wants to send something to C, it should hop the packet over to B. This will be stored as a table for all possible other routers. How do we build that table?

6.4 Forwarding in an IP Router

Look up packet destination in forwarding table. If known, forward to the correct port. If unknown, drop the packet. Decrement TTL, update header checksum. Forward packet to outgoing interface. Transmit packet onto link.

In a real router, it has to be read through an address book, which probably looks like




e.g. 128.9.16.14 => Port 2

Prefix	Next-hop	Port
65/8	128.17.16.1	3
128.9/16	128.17.14.1	2
128.9.16/20	128.17.14.1	2
128.9.19/24	128.17.10.1	7
128.9.25/24	128.17.14.1	2
128.9.176/20	128.17.20.1	1
142.12/19	128.17.16.1	3

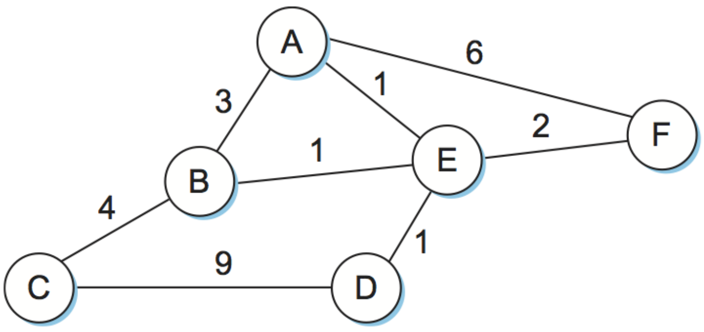
Forwarding Table

Each router has a forwarding table. It maps destination addresses to outgoing interfaces.

When a packet is received:

- Destination IP address is inspected in the header.
- The table is read, and the outgoing interface is identified
- Packet is forward out of that interface
- Next router takes over

6.5 Distance-Vector Algorithm (Bellman-Ford)



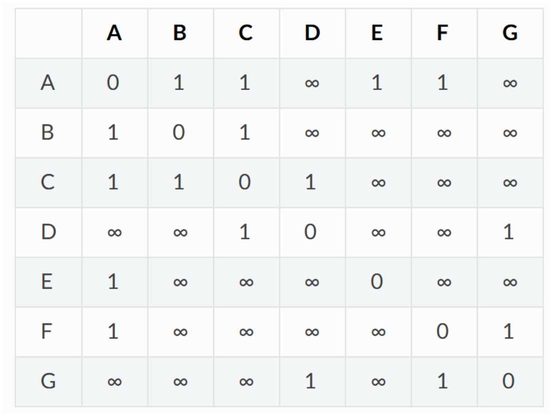
You have a network of routers.

How does each router build up the forwarding table?

The idea is:

- Each node constructs a 1D array containing the distances to all other nodes
- Then each node shares it with its immediate neighbors

Initially each node only knows the cost of its immediate neighbors: for example, C knows that to B, it costs 4, and to D, it costs 9.



	A	B	C	D	E	F	G
A	0	1	1	∞	1	1	∞
B	1	0	1	∞	∞	∞	∞
C	1	1	0	1	∞	∞	∞
D	∞	∞	1	0	∞	∞	1
E	1	∞	∞	∞	0	∞	∞
F	1	∞	∞	∞	∞	0	1
G	∞	∞	∞	1	∞	1	0

Initially, if not as neighbors, cost is set to ∞.

And then every node sends a message to its directly connected neighbors this list.

Nodes receiving this information update their tables accordingly. For example, *B* tells *C* that it costs itself 3 to send to *A* and 1 to send to *E*. Thus *C* learns it costs 4 + 3 to send to *A* from itself and should hop to *B*, and it costs 4 + 1 to send to *E* and should hop to *B*. Obviously, only update if a cheaper path has been found.

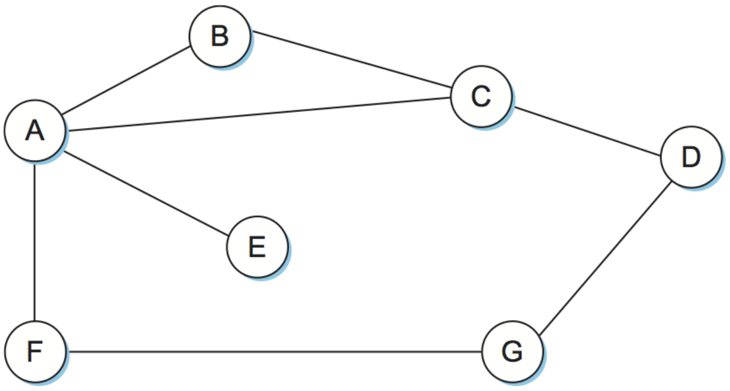
Repeat until convergence.

It is provable that after *K* iterations, all connections that require *K* hops or less will be optimal if nothing fails. This is done by proof by induction.

6.5.1 Distance-Vector Algorithm Pitfalls

Fails can happen.

Recall this.



If *A* – *E* fails, then *A* advertises a distance of infinity to *E*.

But *C* cannot pick that up. It advertises a distance of 2 to *E*.

Therefore *A* thinks that it can use *C* to get to *E*, and claims a distance of 3 to *E*.

This results in the distances constantly increasing until it hits infinity.

So, how to mitigate:

1. Make infinity a small number (16)
2. If *C* learns from *A* about *E*, then *C* will not teach *A* (split horizon)
3. If *C* learns from *A* about *E*, then *C* tells *A* not to send to it for the purposes of sending to *E* by advertising a distance of ∞ (split horizon with poison reverse)

6.6 Link-State Packet

Again, we assume that each node is able to find out the state of the link to its neighbors and the cost of each link. Well, we'll have to do flooding, but this time, it's reliable.

Reliable flooding makes sure all participating nodes in the routing protocol get a copy of the link-state information from all the other nodes. Just send out the link-state information out on all its directly connected links. Then each node forwards that information out of all its link. This information is enough for each node to know the entire network.

When each node receives an update packet (link state packet), they'll get the following information in the packet:

- Which node made it?
- List of directly connected neighbors of that node
- A sequence number
- TTL

The first 2 can help reconstruct the network, the last two enable reliability. It ensures that each router always gets the latest and most recent copy of the information, since there can be multiple, contradictory LSPs from one node traversing the network.

When a packet is received, let's say that node X receives a packet from Y:

- X checks to see if it has already stored a copy of an LSP from Y
- If not, the LSP is stored.
- If already, then only store and use the LSP if it has a larger sequence number.
- Afterwards, if stored, LSP is sent to all its neighbors except for where the LSP was sent.

LSPs carry a TTL. This ensures that old link-state information is eventually removed from the network.

Also, if a link goes up and down way too often wait a bit. Maybe 5 seconds.

6.6.1 Calculating Routes and Building a Table

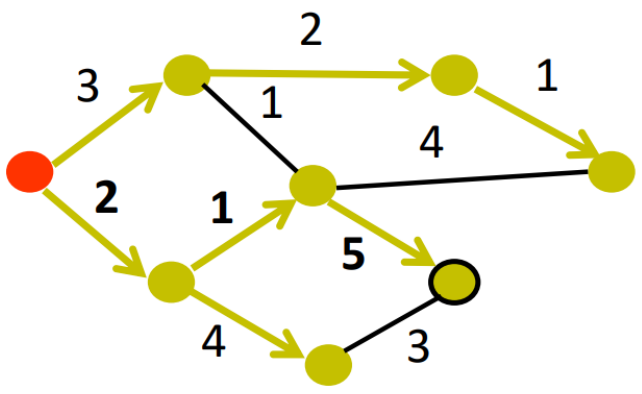
Dijkstra's algorithm is used. For each other node it finds the shortest weighted path.

With that it'll know which node to forward the packet to at first.

This is the algorithm:

```
M = {s}
for each n in N - {s}
    C(n) = l(s,n)
while (N != M)
    M = M + {w} such that C(w) is the minimum for all w in
       (N-M)
    for each n in (N-M)
    C(n) = MIN(C(n), C(w)+l(w,n))
```

You'll see that for each node; it builds a spanning tree.



Hence, forwarding table is:

	link
v	(u,v)
w	(u,w)
x	(u,w)
y	(u,v)
z	(u,v)
s	(u,w)
t	(u,w)

6.7 Comparison

	LS (Dijk)	DV
Message complexity	Since flooding happens, $O(nE)$ messages are sent after all. Each node has a message eventually spread through all links.	Exchange only happens through neighbors. Messages aren't flooded. Convergence time varies.
Speed of convergence	$O\left(n^2\right)$ algorithm requires $O(nE)$ messages	Convergence time varies. There may be routing loops, and the count-to-infinity problem
Robustness	Node can advertise incorrect link cost. Each node computes only with its own table.	DV node can advertise incorrect path cost. Errors can propagate.

7 Network Layer LEGACY

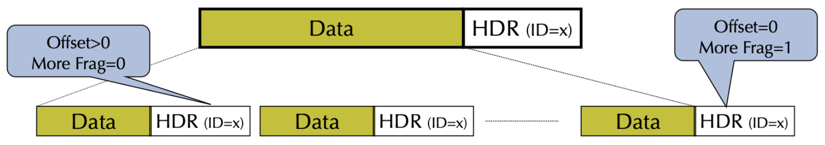
7.1 Fragmentation

A router may receive a packet larger than the maximum transmission unit (MTU) of the outgoing link. Assume that there is a link that has a smaller MTU.

You send a packet, and one of the links have a MTU smaller than your packet. This requires fragmentation. A router will fragment the IP data gram into multiple, self-contained datagrams.

Each fragmented datagram should contain the *offset* (not the fragment count – what if it needs to be fragmented again) and a bit indicated if that is the last fragment.

Solution: R1 fragments the IP datagram into multiple, self-contained datagrams.



Note: offset must be a multiple of 8.

Fragments are re-assembled by the destination host, not by the intermediate routers. If you know the MTU across the path, the host might fragment the data at the start rather than have the routers do the work. This is called **MTU discovery.**

If you are using the same path over and over, sometimes you send different size packets, and see which ones are delivered without fragmentation.

In the IP header, we have a flag called “no fragmentation”. If an IP packet has that bit set to 1, we want to have it not fragment the packet. If the IP router cannot send a packet which wishes not to be fragmented, the packet is dropped. Try:

```
traceroute -F www.uwaterloo.ca 1500
traceroute -F www.uwaterloo.ca 1501
```

DF=1 set in IP header, routers send ICMP error message, which is shown as: !F

Most links use MTU $\geq$ 1500

7.2 Switches vs. Routers

Switches have nice properties. If you have many switches and you connect them to each other, they just work. Is it optimal? No. Depending on how you connect it, it can hurt performance, but it works.

They can do fast filtering and forwarding of frames, no matter how you connect them.

However, disadvantages of switches over routers:

- Topology restricted to a spanning tree, which is sub-optimal
- Large networks require large ARP tables. The ARP table grows with the size of the nodes. In data centers, there can be millions of devices, meaning the ARP table can be extremely large. Occasionally, you will have a broadcast. If your network is 1M nodes, you will have 1M broadcast messages throughout the network.
- Instead of doing link layer switching, you do some network layer routing. Routing happens at the network layer, not the link layer. If you are going beyond your local network, you will be using network layer routers.

7.3 How does a Router Work?

Each router has a forwarding table. It:

- Maps destination addresses
- To outgoing interfaces

Packets. Once they arrive in ingress ports, something in the router makes decision for forwarding based on information for the forwarding table. There is a cross-write fabric, based on the decision of the forward table, transports data from the input port to the correct output port. There is a buffer in the router for each output port, which they leave one-by-one.

7.4 Forwarding in an IP Router

- Look up packet DA in forwarding table
  - If known, forward to correct port
  - If unknown, drop packet
- Decrement TTL, update header checksum

Why did we do that? To avoid loops. Occasionally loops are going to be formed. You are talking about a network as big as the internet, and you have way too many entities trying to control the network.

Once TTL updated, update checksum and then forward the packet. Transmit packet onto link.

Look up IP address in router.

7.5 Organizing routing tables

Have entries for IP address. If packet for 1.2.3.4, it somehow knows what egress port to send the data from. And so on. Somehow, it knows this for every destination address.

This works. But it has a problem. What's the problem?

There are too many IP addresses. IPV4 only has 2^{32} IP addresses. Lots of IP addresses. How can we simplify this? Usually, there is a reason we have network IDs and host IDs. Usually, you care about the network. All the devices in one network can be treated the same. So, you create prefixes. If you are an ISP and there are traffic for any of the machine with a specific prefix, just send them over this link. And so on.

That significantly reduces the size of the table, but that vreates the problem.

Each IP address matches exactly one IP address per entry. When you have prefixes, some IP addresses might belong to mor ethan one provider.

Instead of performing exact matching for IP addresses, we do the longest prefix match.

7.6 Longest Prefix Match

What does that mean?

Start taking from the start of the IP address, see how many bits we can match, which ever matches more of a prefix should be chosen.

Imagine UofT has a big network with a small prefix. UofT devices its network to multiple subnets, lets say CS has a smaller workload with a larger prexfix. If rogers has two rules, one saying, dero any machine in UofT, send it on port 1, or CS, send it fto port 2, f a packet in CS matches both, then we're going to match 2. There isn;'t any reason t use two rules.

More specific route – longest matching prefix.

7.7 Where do Forwarding Tables Come From

What if your network changes? What if a link goes up or down? We need to have a mechanism that is more than a manual entry of IP prefixes In the routing table?

7.8 Packet Routing And Forwarding

How are routing table entries filled up?

7.9 Distance Vector Shortest Path

After K iterations all K-hop distance paths you should know the most optimal ones.

Iteration for the bellman ford: maximum distance in the network

How do we know the algorithm always converges: mathematical induction. For $k = 1$, we get the most optimal path for 1 hop, and then use induction to prove for all k

What happens if the topology changes or the cost changes?

Well, bad news travels slowly. If a link fails it could result in an infinite loop. Distance-vector protocols suffer from “counting to infinity” problem. It can take very long to converge to optimal path if that happens.

How to mitigate this issue? We won’t solve it entirely.

Solution:

- Change infinity to some small integer (16). If there are 16 iterations and im still not finding the shortest path just say its gone and ignore that path
- Split horizon: okay but can create awkward problems. If R_2 learns the shortest path requires going to R_3, it will not tell R_3 about anything (not advertising back to R_3)
- Split horizon with poison reverse: Instead of not advertising to R_3 it advertises infinity, to prevent R_3 to sending things back to R_2.

All of them work in some scenarios but there are corner cases in which some of these techniques break.

7.10 Link State

In distance vector, information is only send to neighbors. Neighbors are the only entities I talk to. In link state protocol, that is not the case. Every node in the network send what it knows about the topology to everyone else in the network.

Flooding is not a good idea. In general it is not agood idea. But link state protocls take advantage of flooding in a very limited scale.

If you have a network, and you kno you have 3 links, you flood the network telling everyone in the network you have __ more links. This information is delivered to everyone throughout the network.

Every node gathers this state information from everyone else. It re-creates the topology of the network, locally. It creates a data structure that represents the topology of the network, which comes from those packets that come from all the node in that network. You now have a local graph, and you know how to find the shortest path in a graph.

Use Dijkstra's single source shortest path algorithm. At each step of the algorithm, router adds the next shortest path to the tree. If you want to send traffic to any other node in the network, how do yu route it? Once you run that algorithm, you convert the shortest path information to the routing table.

And by this you make a forwarding table (djk algorithm constructs a spanning tree – see the slides) that contains the first node in the path that goes to the eventual node you want to go to.

7.11 Reliable Flooding of LSP

The link state packet:

- The ID of the router that created the LSP
- List of directly connected neighbors and cost
- Sequence number
  - This makes sure everyone knows the latest version of the state
- TTL
  - Because we are flooding these things can go in the network forever.

Sometimes, if a link goes up and down way too much, maybe don't rely on it.

8 Internet Topology (Autonomous Systems)

The internet is what is divided into autonomous systems. These are parts of the network that are owned and managed by different institution's. They have their own objectives in managing why and how they do things. The collection of routers and links that are managed by a single entity, could be a company or university, has a bunch of these devices and they are the ones deciding how the network is managed.

We have a hierarchy of autonomous systems. We have a:

- Large tier 1 providers
- With a nationwide network
- Could span across multiple countries or continents

And then we have medium-sized regional providers (rogers, bell). They are not the top tier providers which go over multiple continents

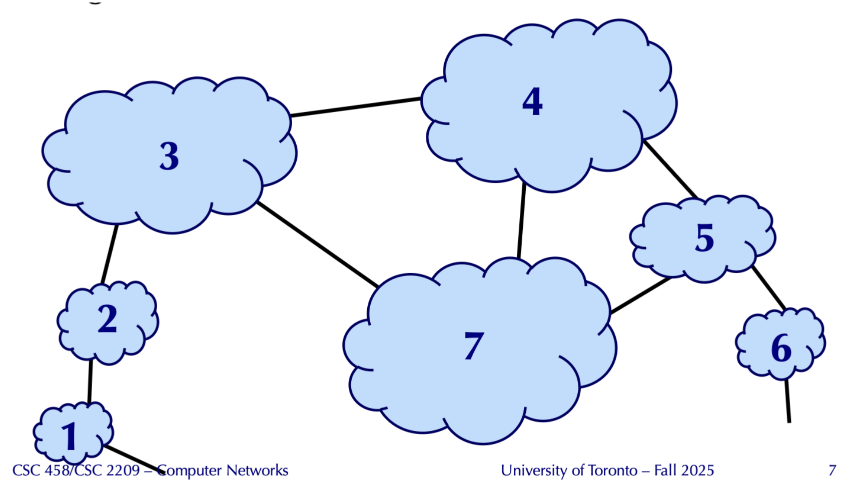
- These medium-sized networks provide service to downstream customers, but they need a provider of their own. This means they typically have a national or regional scope.

And then you have smaller networks managed by individual companies.

These autonomous systems (AS) don't always want to share information to each other. This is what makes life a bit more diffcut when it comes to routers.

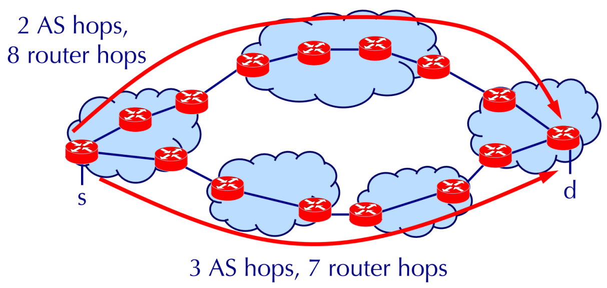
This new system allows us to create a new abstraction: a graph. Each node is an autonomous system (AS). Each link can be a collection of links that connect these autonomous systems.

Two autonomous systems may have multiple physical links that connect to each other. In the abstraction, we show this as a single link. It is a logical link, not a physical link. Sometimes, there is not even a physical limn connecting them. Sometimes, AS starts something called an exchange point, which they all connect to. This creates the opportunity for these AS to connect to each other.


3
4
5
2
7
6
1
CSC 458/CSC 2209 – Computer Networks
University of Toronto – Fall 2025
7

8.1 Characteristics of AS paths

The AS path may be longer than the shortest path, since the router path may be longer than the shortest path.



8.2 Points of Presence

Points of presence should be placed in high-density areas, or near other providers / exchange points. Of course, beware of cost and real-estate. You'll usually find PoPs in major metropolitan areas.

8.3 Identifying AS

Each AS has a number associated to it. AS numbers are 32-bit values, which used to be 16. Currently estimated to be over 90,000 in use.

Certain companies may have multiple AS, if they cover multiple continents. You don't want to send a packet to the other side of the globe and send it back.

Once you have these AS numbers, you can identify paths. At this level of detail, we don't care about what happens inside the AS. There might be 10s, 100s, 1000s of routers in the AS. But when you are talking about path from a client to a server, server to a client, we talk about going through a path of autonomous systems.

We don't think about the actual routers connect the traffic. We are thinking of which AS are in the way when going from the webserver to the client.

The links can carry extra information, depending on which business relationships these AS have.

The business relationship defines how much traffic the link is supposed to carry. Which destinations to reach. How much money one AS pays to another. There are lots of relationships that exist.

Usually, we classify them to:

- Customer provider
  - Customer pays provider, provider provides connection. Princeton is a customer of AT&T
- Peer-peer
  - Two nodes make an agreement to share each other's customers.
  - Because it's simple.
  - Bell and rogers have a similar no. of customers, so they might reach an agreement.

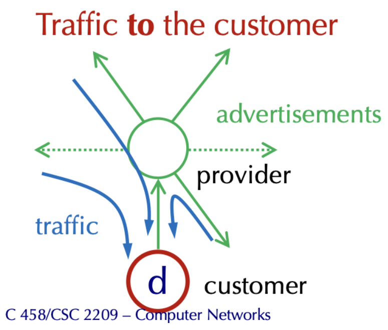
What does it mean to have these kinds of relationships.

8.4 Customer-Provider Relationship

Customer tells the provider what its IP addresses are.

The provider's job is to tell everyone in the world, the IP address in this range are paying me to connect them to the world. When you receive an advertisement, I own, I have a connectivity to the customer *d*. The provider propagates this to everyone else in the world. If you have traffic to *d*, I know how to deliver it. It is going to say to its own providers, it can carry traffic to *d*.

If the customer gets traffic, provider receives the traffic, and the customer gets the traffic.


Traffic **to** the customer
advertisements
provider
traffic
d
customer
C 458/CSC 2209 – Computer Networks

Customer needs to be reachable for other, so customer needs to be reachable from everyone.

Customer does not want to provide transit service. Customer oes not let its provider route through it.

The provider for every IP prefix it knows, it tells the customer it can deliver traffic.

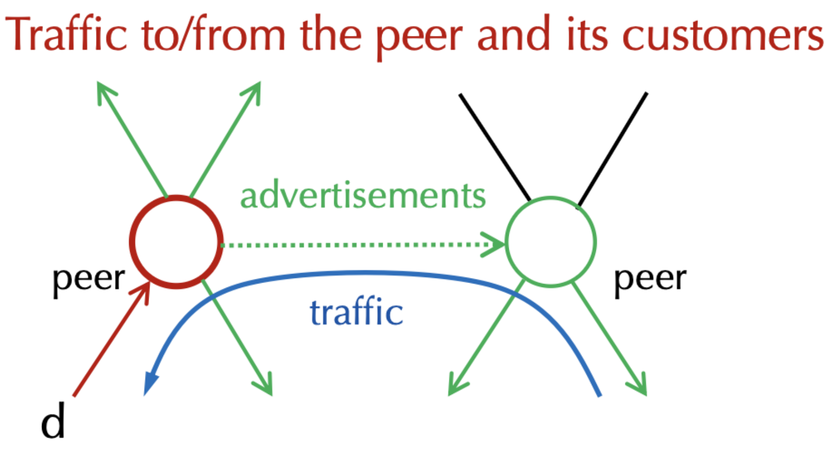
If the customer has traffic to the provider, it will send it to the provider.

The customer will also send to its own customers, if there are any. This means if the customer needs to send information, maybe from someone else, it sends the information.

For internal networks, like UofT, it is their job to provide service for their own customers. It will not provide the link to other providers.

8.5 Peer-Peer Relationship

If Bell has a customer *d*, and peer learns about *d*:

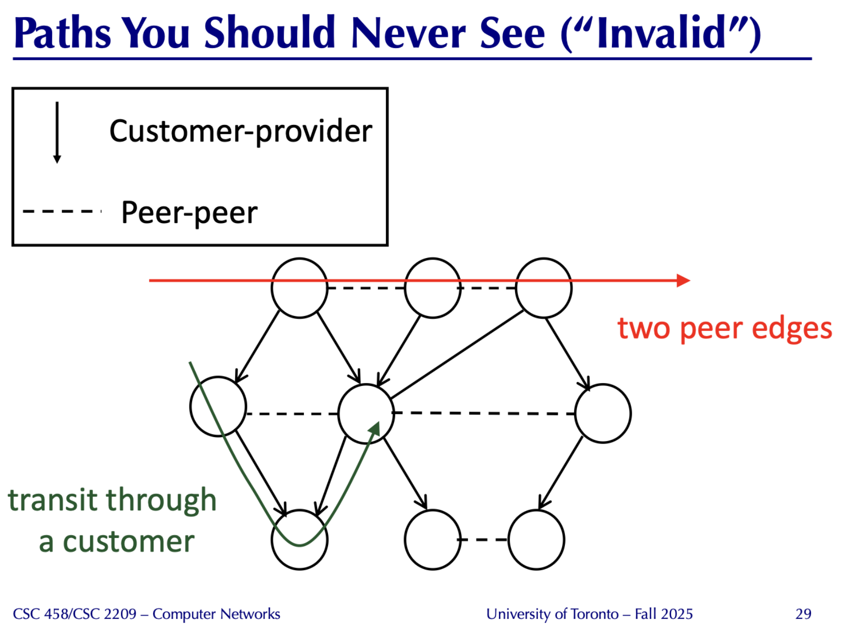


If UofT is a customer of bell, and Waterloo is a customer of rogers, rogers is going to send the advertisemtn to its customers.

The idea, is AS exports only a peer's routes to its customers.

8.6 Things that don't happen

Having a customer forward something


Paths You Should Never See ("Invalid")
Customer-provider
Peer-peer
two peer edges
transit through a customer
CSC 458/CSC 2209 – Computer Networks
University of Toronto – Fall 2025
29

8.7 BGP Vs. IGP

BGP is the border gateway protocol. To send traffic to google, you need to send it to rogers. This should be consistent across all AS.

IGP is the internal gateway protocol. It tells me to go from this node to that border router, the next hop is this. What goes inside is internal to the AS.

When I combine this information: eventually, each router only needs to know where the next hop is. That's it.

8.8 Internal Routing Protocols

- RIP
  - Uses DV
  - Updates sent every 30s
  - No auth
  - Written in BSD
  - Not used
- OSPF
  - Link-state updates sent via flooding
  - Router use Dijkstra's algorithm
  - Authenticated updates
  - AS may be partitioned into areas
  - Widely used

For the AS-level topology:

- Destinations are IP prefixes
- Nodes are AS
- Links are connections and business relationships

8.9 Routing between Autonomous Systems

The path-vector routing between AS is an extension of DV. It supports flexible routing policies and avoids the count-to-infinity problem.

The idea is to advertise the entire path. Hence: to broadcast destination *d*, which router 1 will do, it can say to router 2: `d: path(1)`. And then router 2 can say to router 3: `d: path(2,1)`. And so on.

Nodes can easily detect a loop, such as it seeing itself in the path.

Nodes can simply discard paths with loops.

This is BGP. It is a path-vector routing protocol. BGP advertises complete paths, saying: “the network 176.64/16 can be reached with the path {AS1, AS5, AS13}. Paths with loops are detected and ignored. When a link or router fails, the path is withdrawn.

8.10 BGP Operations

For two AS to start up a BGP session:

- A session is established on some TCP port
- All active routes are exchanged
- Incremental updates are sent over and over
  - While the connection is ALIVE exchange route UPDATE messages

A node learns multiple paths to the destination. Updates are done incrementally, and if an active route is not available it sends a withdrawal message to all other neighbors

Examples of BGP messages:

- Open
- Keep alive
- Notification: shuts down a peering session
- Update: Announcing new routes or withdrawing previously announced routes

BGP announcement = prefix + path attributes

Where attributes include: next hop, AS Path, local preference, multi-exit discriminator

8.11 BGP Path Selection

In the simplest case, take the shortest AS path and break ties arbitrarily. However, BGP is not limited to shortest path routing. It can do policy-based routing.

So, how do we pick routes:

- Routes learned from the customer are preferred over routes learned from the peer, preferred over route learned from the provider
- The customer advertises if they are **willing** to forward traffic

So with this, this is how BGP route selection goes:

1. Highest local preferences. Prefer customer routes over peer routes.
2. Shortest path, lowest MED, I BGP < E BGP, lowest IGP cost to BGP egress
3. And then break ties by choosing the lowest router ID

8.12 BCP Policy

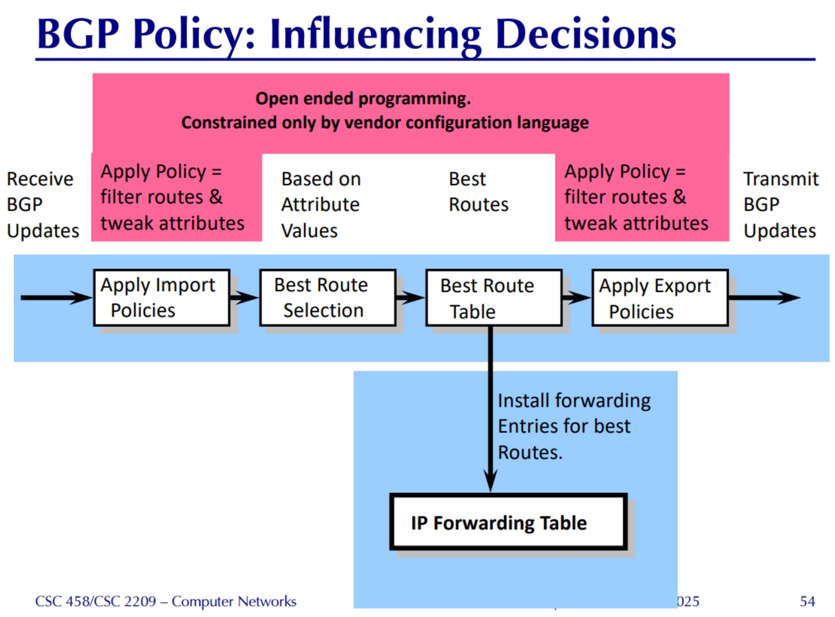
Import policy

- Filter unwanted routes from neighbor.
- Assign local preference to favored routes

Export policy

- Filter routes you don’t want to tell your neighbor
- Manipulate attributes to control what they see, such as making a path artificially longer than it is

If I don’t want another router communicating with me or if I want a router to take a specific path, this is something I can do. Lie to my neighbors.



BGP Policy: Influencing Decisions

Open ended programming.
Constrained only by vendor configuration language
Receive BGP Updates
Apply Policy = filter routes & tweak attributes
Based on Attribute Values
Best Routes
Apply Policy = filter routes & tweak attributes
Transmit BGP Updates
Apply Import Policies
Best Route Selection
Best Route Table
Apply Export Policies
Install forwarding Entries for best Routes.
IP Forwarding Table
CSC 458/CSC 2209 – Computer Networks
025
54