Lec 22: Routing & Transport

- Routing
  - Source vs. path routing, BGP

- Transport - layer
  - UDP & TCP

application

presentation

session

transport

network

link

phygs
Reality: routing table entries correspond to subnetworks of different sizes.

Subnetwork is identified by IP address + mask

\[ \text{e.g.} \quad 1.2.x.x \quad \text{represented as} \quad 1.2.0.0 / \frac{\text{30}}{255.256.0.0} \]

To see if address is in a network: (logical) and address w/ netmask, compare to network IP.
How to find path from A to E:

1. Computed shortest path through graph.
2. Each node checks if packet is next.
3. Node 3 receives packet, sends it to next node.
4. Node 6 receives packet, sends it to next node.
5. Node 7 receives packet, sends it to next node.
6. Node 8 receives packet, sends it to next node.
7. Node 9 receives packet, sends it to next node.
8. Node 10 receives packet, sends it to next node.

P4o: Shortest path.

Note: Data can be dropped if topology changes.
network IDs, not host IDs.

Routing H1 for 2:
- A, B, C, D:
- E: 4
- F: 5

Routing H1 for 3:
- A, B, C, D:
- E: 4
- F: 5
- G: 1

Path: each node in path has a route table telling it how to forward packets (based on next hop).

Pro: ISPs can dynamically adjust to respond to traffic.
- Topology can be private.

Con: each ISP needs routing H1.
- A lot of storage?
- Need to update routing tables when network changes.

Other concerns:
- What about traffic on each link?
IP addresses grouped into "subnets"
Border Gateway Protocol (BGP)

- maintain "distance vector" to each end network
- exchanger with neighbors

Directed Shortest Path Algorithm

- if longer path, charge eventually
- B: ISP 3, best path
- C: ISP 1, obvious for

ISP 1 to ISP 2, using it to ship to ISP 2
ISP 1 reads to ISP 2, who redial it as needed
Transmission Control protocol (TCP)

- Multiplexing

- "Stream" abstraction.
  - Two-way stream of data, each endpoint can write into stream, read from stream.

- In-order, guaranteed delivery.
TCP
- segments represent:
  - 0, 1, 2, 3, ...
- label segments with
  - sequence numbers
A send data, with sequence
  - B reply with acknowledgement (ACK)
  - when B receives ACK, linearly receive data
  - if it doesn't, discard data

A → B
  - hello
  - 4/0
  - DATA2
  - DATA3

B → A
  - goodbye
  - 4/0
  - ACK2
  - 4/0
  - ACK2

Can combine DATA & ACK messages:
  - send 1 packet instead of 2.

"piggybacking"