ICS 451: Today's plan

- Network Layer Protocols: virtual circuits
- Static Routing
- Distance-Vector Routing

Virtual Circuits: Motivation

- Implementing the routing table in hardware can be expensive
- to make it simpler and cheaper, simplify
- each packet header includes a bit-string called *virtual circuit identifier (VCI)* or *label*
- the combination of VCI and input interface is used as an index into the forwarding table
- the table stores output interface and VCI
  - so the VCI in the header changes at each hop

Virtual Circuits

- A virtual circuit forwarding table can be built in hardware and operate very quickly
- To build the table, must:
  - decide the route in advance
  - select a VCI for each segment of the route
    - that is available on that interface!
  - configure the forwarding tables of all the routers
- Then, the packet is sent with VCI 1
- forwarded with VCI 2
- then again on VCI 3, etc

Virtual Circuit Setup

- Before a host can send any packets to a destination, it must set up a virtual circuit
  - to that destination
- it does so by sending a special *signaling* packet to its router R
- R signals other routers along the path
  - each router sets up its forwarding tables
- after signaling is complete, R returns to the host the VCI to use for the new Virtual Circuit

Advantages of Virtual Circuits

- each virtual circuit can be assigned a set of resources (e.g. bandwidth)
  - Quality of Service, or QoS
- every forwarding can check whether the resources are exceeded
  - if so, the packet is marked “loss priority”, i.e. to be dropped before other packets in the queue
- forwarding table implemented in hardware can be very fast
  - low latency

Real-Life Virtual Circuits

- Multi-Protocol Label Switching (MPLS)
  - used in the Internet backbone
  - each router has a virtual circuit to every other router within a relatively small network
- Asynchronous Transfer Mode (ATM)
  - evolved from Synchronous Optical Network (SONET) used to carry digital voice data
  - telephone companies wanted to use ATM for data as well as voice
  - QoS allowed real-time voice traffic as well as less delay-sensitive data traffic

Managing RoutingTables: Static Routing

- Static routing means manually adding and deleting routes from a routing table
  - sometimes can be done on the command line
- works well for small unvarying networks
- works poorly in large, time-varying networks

Managing RoutingTables: Distance-Vector Routing Example

- Suppose Alice and Bob are connected by a link with a delay of .3s
- Alice has a route to Charlie, with a delay of .5s
- Bob tells Alice that he can reach Charlie, with a delay of 0.15s
- Should Alice use her existing route to Charlie, or change her routing table to use Bob as the next hop?

Elements of Distance-Vector

- every link has a cost > 0
  - the cost is also known as the *metric*
- we wish to route over the shortest (lowest cost) path to each destination
  - that is, the path with the lowest total metric
- the computation should be completely distributed, with no central point of failure
- the result should be a consistent set of routing tables

Distance-Vector

- the vector to a destination is the interface and next hop used to reach that destination
- the distance is the sum of the metrics (costs) of all the links on the shortest path to that destination
- a distance-vector message has the distance to each destination in the routing table
- the vector is the sender of the message

Distance-Vector Routing Example

- Suppose Alice and Bob are connected by a link with a delay of .3s
- Alice has a route to Charlie, with a delay of .5s
- Bob tells Alice that he can reach Charlie, with a delay of 0.15s
  - Bob's distance to Charlie is 0.15s
  - Alice's vector to Charlie, if she uses Bob's route, is Bob himself
- The cost of the route via Bob is less than the cost of the other route, so Alice sends via Bob

Distance-Vector Algorithm

- periodically and when routing table changes:
  - build a distance-vector message with the information from the routing table
    - a set of values (D, m)
  - send it to all the neighbors (over all interfaces)
- when receiving a message on interface i with cost c, for each (D, m)
  - if D is not in the routing table, add (D, i, m+c)
  - if (D, i, m') is in the routing table, replace it
  - if (D, i', m') is in the routing table, i ≠ i' and m + c < m', replace this entry with (D, i, m+c)

Distance-Vector Game

- establish point-to-point links with neighbors
  - through a hello protocol
  - make sure you both agree on the link cost m
- build your routing table
  - initially only has your name at distance 0
- run the distance-vector algorithm, recording new and better routes

Distance-Vector Game, part II

- add a point-to-point link with a new neighbor
  - make sure you both agree on the link cost m
- exchange your routing table with your new neighbor
- see if you get any new routes
  - if you do, distribute your new routing table to all your neighbors again
  - until no new routes are created

Distance-Vector: removing links

- Alice has a route to Charlie via Bob with m_{abc}
- the link between Bob and Charlie goes down
  - Bob no longer has a route to Charlie
  - until he gets the next routing update from Alice!
  - now Bob has a route through Alice of cost m_{babc}
  - but this route is a routing loop!
- eventually Alice times out and deletes her route
  - then she gets a new route from Bob!!!
- The distance keeps increasing
  - this is called “counting to infinity”

Dealing with “counting to infinity”

- make “infinity” a small number, e.g. 16
  - reasonable when each link has m=1
- if Bob sends a worse route, update if existing route has Bob as next hop
- split horizon: Alice does not send to Bob routes for which the next hop is on the same interface as Bob
- split horizon with poisoned reverse: Alice does send such routes to Bob, but with a cost of infinity