ICS 451: Today's plan

• Distance-Vector Routing (continued)
• Link-State Routing
• IP
Managing Routing Tables: 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
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 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, \(i \neq i'\) 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
Link-State Routing Overview

- In distance-vector routing, information from the entire routing table is sent to neighbors.
- In link-state routing, information about the neighbors is sent to all routers in the network.
- Information about neighbors is the state of the links, or *link state*.
- The Hello protocol establishes which neighbors are reachable:
  - Hello message sent every $N$ seconds.
  - Neighbor times out after $k \times N$ seconds.
Link-State Routing

• By collecting information sent from every router, each router can build a graph of the entire network

• each router uses the graph to compute a route from itself to each destination
  – typically, using Dijkstra's shortest path algorithm

• when routes go down:
  – a new Link-State Packet (LSP) reports it, or
  – a saved LSP times out
Sending to Every Router

- each LSP has
  - the address of the originating router, and
  - a sequence number
- every router getting a new LSP forwards it to (almost) all its neighbors
  - LSP from a new originating router
  - LSP from a known router, but higher sequence number
- this is a *flooding* protocol
Flooding Game

- establish point-to-point links with neighbors
  - through a hello protocol
- when you get a message:
  - record the message
  - if it is new, forward to all neighbors
    - except the neighbor from which the message was received
Link-State Game

• establish point-to-point links with neighbors
  – through a hello protocol
  – make sure you both agree on the link cost m

• flood your link state to every router
  – (D, m) for each neighbor
  – put your name and a sequence number
    (0, 1, 2, …) in each packet

• build a graph of the network
Internet Protocol

- IP version 4 or version 6
  - overall operation the same, details different
  - IPv4: 32-bit addresses, IPv6: 128-bit addresses
- best-effort datagram communication
IPv4 addresses

- 32 bits or 4 bytes
- each byte written in decimal: 128.171.224.100
- 127.0.0.1 is 0x7F000001 or 01111111000000000000000000000001
- One address for each interface
  - and for the loopback interface (127.0.0.1)
IPv4 addresses and routing

- 32 bits means 4Gi = 4,294,967,296 addresses
- If assigned at random, routers would need to have 4Gi routing table entries
  - and routing information for 4Gi routes would have to be distributed to each router
- To summarize information, adjacent addresses are grouped into networks
  - So routers only need a route to each network
- The network part of the address is the first $n$ bits
  - $n$ varies
Netmasks

- each packet carries an IP destination address
- routing protocols exchange IP destination addresses together with a netmask that indicates the value of $n$ for that destination
- two representations for the netmask:
  - actual bits, such as 255.255.0.0 or 255.255.192.0
    - the first $n$ bits are 1, the remainder are 0
  - number of bits, such as /16 or /18
- first representation used in the computer
Network Sizes

• with netmasks, all networks have sizes that are powers of 2
  – but two addresses are reserved:
    • all 0's is the network number
    • all 1's is the network broadcast address
• every network with at least two addresses can be split into smaller subnetworks
  – each will have a size a power of two
Routing Table Lookup

- each packet has an IP destination address $A$
- given an entry with destination network $D$ and network mask $M$, the packet matches the entry if
  \[ D \& M == A \& M \]
- that is, if the first $n$ bits of the destination matches the first $n$ bits of the address
Multiple Matches

- there may be more than one routing table entry that matches a destination IP address
- if so, the one with the longest mask is used
- this lets us have generic routes, with shorter masks, and specific routes, with longer masks
- a route with a 0-bit netmask is a default route
  - used only if nothing else matches
- a route with a 32-bit (128-bit) netmask is a host route