Routing Algorithms

Routing in Packet Switched Networks

- The process of finding a path from a source to a destination through intermediate nodes
- A route – a path from the source to the destination
- Router – the intermediate nodes that forward the packet towards its destination
- Routing Table
  - Kept at each node
  - Contains the route to each destination node reachable from that node
  - A routing protocol fills up and maintains the routing table at the nodes
    - Updates routes if existing routes change or new better routes found

How is a route stored?

- Basic fields of a route
  - Destination
    - Route to which destination?
  - Next hop
    - To reach that destination, what is the next node the packet should be forwarded to
      - The next node will then use its next node field to go one more hop and so on till the destination is reached
  - Cost
    - What is the cost of this route?
      - Definition of cost can vary. One common metric is the number of hops (links) to the destination
- There are usually other fields, we will see, but these three are the most important and what you need now to understand routing protocols

Routing vs. Forwarding

- Routing is the process of finding the routes
  - Routing protocols can run even if no packets need to be sent
    - Keeps the table ready to forward a packet
  - There are also on-demand routing protocols in other types of networks that finds a route only when a packet has to be sent (we will not study)
- Forwarding is the process of using a route to send a packet towards the destination
  - Done only when a packet arrives at the node whose final destination is not this node
  - Done in real time
- Sometimes, routing is loosely used to refer to both
  - "route a packet"

Routing Protocol Goals

- Correctness – should find a correct route
- Optimal - compute "best" route
- Efficiency/Low overhead - should not send too many messages to find route
- Robust - can handle failures like link failure, node crashes
  - Routes should be automatically updated
- Rapid convergence when network conditions change
  - Updated routes on network change should be found fast

Performance Metrics

- Used for selection of a route among multiple possible routes
- Some examples
  - Minimum number of hops
  - Least "cost"
    - cost can be based on different things like traffic carried, types of links, available bandwidth etc.
- Delay, throughput

When/Where is Routing Decision Taken?

- When?
  - Periodically, per packet or per virtual circuit establishment?
- Where?
  - Distributed – all nodes decide cooperatively
    - Internet routing protocols are all distributed
  - Centralized – one node decides routes for everybody
    - Not good for large networks
  - Source – decided by source of packet, complete path to destination put in packet (***source routing***)
    - IP routing supports this for special applications

Routing Strategies

- Fixed/Static
- Flooding
- Random
- Adaptive/Dynamic

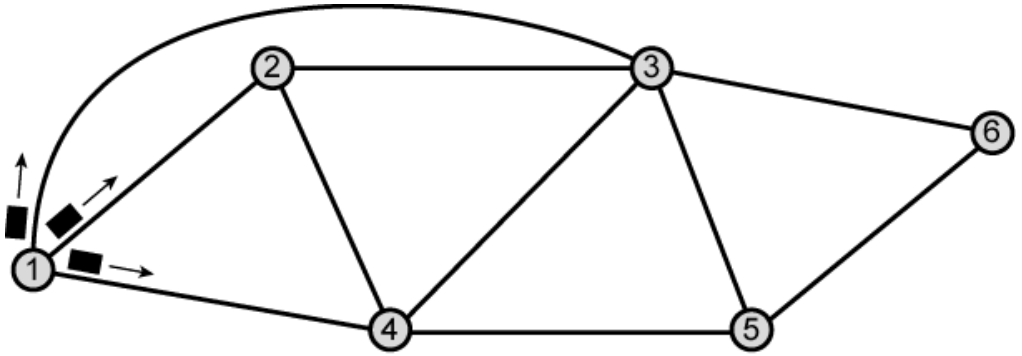
Fixed/Static Routing

- Single permanent route for each source to destination pair
- Routing table created & updated manually
- Table is fixed unless manually changed again
- No dynamic update when network conditions change (for ex., a link goes down, or a shorter path comes up, or some link becomes congested)
- Fine for very small networks
- Not good for large networks
- But static routing is used for fixed, small networks

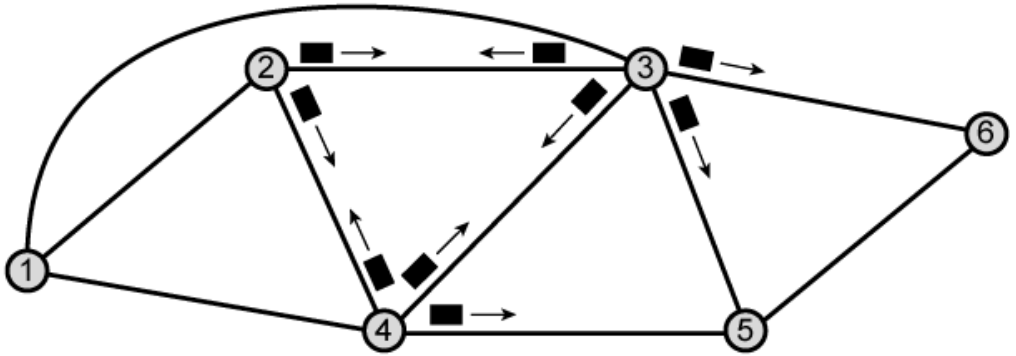
Flooding

- No network info required
- Packet sent by node to every neighbor
- Incoming packets retransmitted on every link except incoming link
- Eventually a number of copies will arrive at destination
- Each packet is uniquely numbered so duplicates can be discarded
- Nodes can remember packets already forwarded to keep network load in bounds
- Can work around failed links/nodes

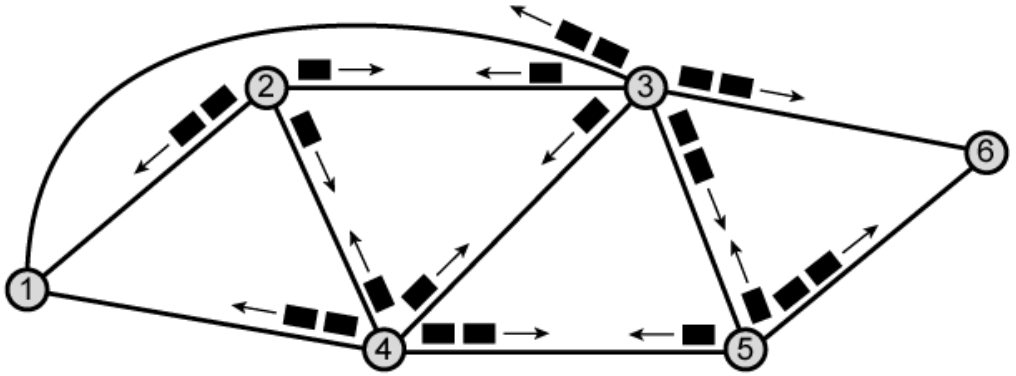
Flooding Example



(a) First hop



(b) Second hop



(c) Third hop

Properties of Flooding

- All possible routes are tried
  - Very robust
- At least one packet will have taken minimum "cost" route (for ex., min hop count)
  - Can be used to set up virtual circuit
- All nodes are visited
  - Useful to distribute information
- But no routes remembered (no routing table)
- Too many copies of a packet may be sent

Random Routing

- Node selects one outgoing path for retransmission of incoming packet
- Selection can be random or round robin
- Can select outgoing path based on probability calculation
- No network info needed
- Route followed is typically not least cost nor minimum hop
- No routes remembered

Adaptive/Dynamic Routing

- Used by almost all packet switching networks
- Routing decisions change as conditions on the network change
  - Failure
  - Congestion
- Requires info about network
- Decisions more complex
- Tradeoff between quality of network info and overhead
- Reacting too quickly can cause oscillation
- Reacting too slowly can make routes obsolete

- Routes saved in routing tables
- Routers communicate among themselves to update routing tables dynamically when network conditions change
- No manual intervention needed normally

Types of Adaptive Routing Strategies

- Distance Vector Routing
- Link State Routing
- Path Vector Routing (basic idea similar to distance vector routing with some changes)

- All internet routing protocols are based on variations of these

Distance Vector Routing

- Based on Bellman-Ford Shortest Path algorithm to find the least cost route between nodes
- Each node x has a routing table with entries of the form <p, q, c>, which says that x has a path to node p that goes through its neighbor node q (the next hop) and the cost of the path is c
- Each node periodically sends its routing table to all its neighbors
- If a node does not receive a routing table from one of its neighbors y for some predefined time, the link (x,y) is assumed to be down
- On receipt of a routing table at x from y, update the information for any node p at y if
  - the path through y is of lower cost than the path x knows currently, OR
  - y is the next hop on the current path to p from x

The update rule

Let d be the cost of the link (x, y)

Node x, on receive of a routing table from y, does the following:

For each entry <p, q, c> in the routing table of y

if there is no entry for p in routing table of x

add < p, y, c+d> to the table of x

else

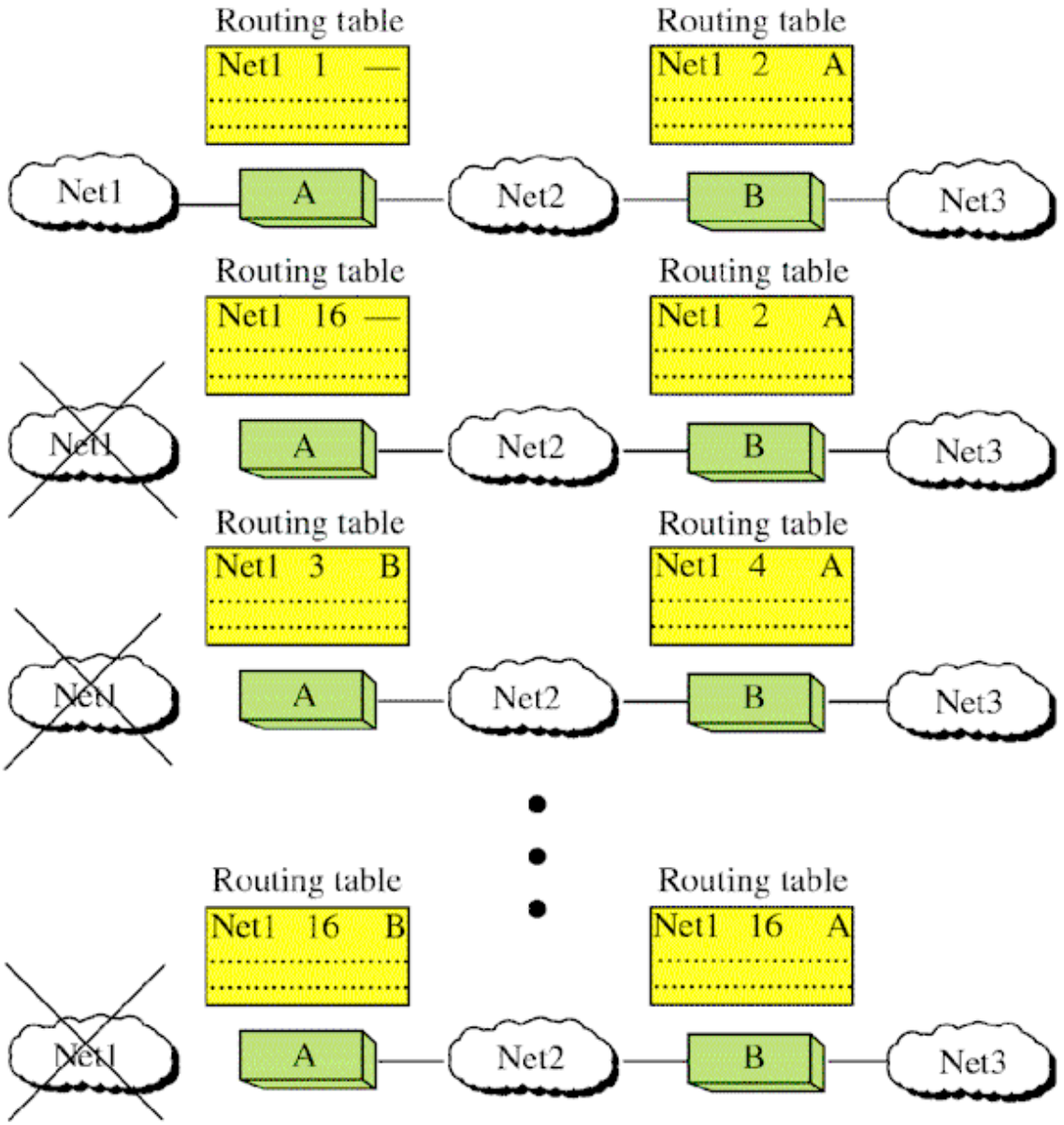
let the current entry be <p, r, c'>

if (c+d) < c' or if y = r

remove <p, r, c'> from table of x

add <p, y, c+d> to table of x

Counting to Infinity Problem



- Possible solutions
  - Triggered Update: When a link failure is detected, propagate that information immediately (without waiting for the next update time)
    - Ensures that the infinity value is propagated before it can be overwritten in most cases
  - Split Horizon: do not advertise a route to a node from which the route is learnt in the first place (the next hop)
    - Breaks all 2-node loops
    - Verify that 3-node loops can still occur
  - Poisoned Reverse: if y is the next hop for z to get to destination x, then z will advertise to y that its distance to x is infinity
    - Avoids y trying to reach x via z
- None of these (or combinations) solve the counting to infinity problem fully, just reduces the chance
  - Verify this claim by forming counterexamples

- Advantages of distance vector routing
  - Simple to implement
  - Low memory requirement at each node
- Disadvantages
  - Slow convergence
    - If a network becomes inaccessible or a link cost changes, it may take a long time for all other routing tables to know this
      - Proportional to diameter of the network
  - High overhead – updates sent even if no change in routing table
    - Not sending it will be interpreted as link failure
    - Can optimize by sending full routing table only periodically and sending only incremental changes every period
  - Routing loops may take a long time to be detected (counting to infinity problem)
    - Needs to fix a value of "infinity" (a route cost that cannot occur normally)
    - Limits the size of the network in which it can run

Link State Routing

- Each node floods information about only its neighbors (who and cost) to all nodes in the network
  - Link State Advertisement (LSA) packets
- All nodes receive and store complete information about the network, then run some least cost algorithm (ex. Dijkstra's) to compute least cost route to all other nodes and stores in routing table
- When are LSA packets sent?
  - At the beginning, and periodically with full information
  - On any local topology change detected by a node (link/neighbor down, link cost change)

- Some issues that protocols need to handle:
  - What if an LSA packet is lost?
    - A change information may be lost until the next periodic update
  - What if an LSA packet arrives out of order?
    - Link (x, y) goes down, node x sends LSA with this information. This reaches all nodes, but is delayed to some node z
    - Link (x. y) comes up again, node x sends LSA with this information. Reaches all nodes, including z
    - Now the LSA packet with the link down information arrives at z
    - Node z thinks (wrongly) link (x.y) is down, while others think (rightly) it is up
    - How can you solve it?
      - Use sequence numbers in LSA packets to distinguish old, stale LSA packets

- Advantages
  - Faster convergence
  - No counting to infinity problem
    - But can routing loops still occur at all?
  - Less overhead than distance vector
    - Periodic sends can be much less frequent. Mostly incremental changes flooded
- Disadvantages
  - More information needs to be stored and processed at each node

Metrics Used

- What is a good cost metric to use for each link?
- Static metrics
  - Number of hops
  - Link bandwidth
- Dynamic metrics
  - Delay on a link
  - Load (traffic) on a link
- Dynamic metrics seem better, but are harder to implement, also can cause oscillations
  - Think when all traffic goes to a low delay path for example because it is low cost
- Static metrics are mostly used
- We will look at some actual routing protocols next