CPSC 471: Computer Communications

Routing

Figures from [Computer Networks: A Systems Approach](), version 6.02dev (Larry L. Peterson and Bruce S. Davie)

Forwarding vs. Routing

- Take a packet on an input and send it out on the appropriate output

- Build up the forwarding tables

Routing

- Issue for every packet in datagram networks
- Issue only for connection request packets in VC networks
- We'll examine routing in small and medium-sized networks first
  - Intradomain routing protocols

Forwarding Table vs. Routing Table

Forwarding Table

- Map network prefix to outgoing interface and MAC information (e.g., Ethernet address)
- Optimize for address lookup for forwarding packet

Prefix/Length	Interface	MAC Address
18/8	if0	08:00:2B:E4:0B:01:02

Table 15

Routing Table

- Precursor to building the forwarding table
- Map network prefix to next hop
- Optimize for calculating changes in topology

Prefix/Length	Next Hop
18/8	171.69.245.10

Table 14

Network as a Graph

- Routing is a graph theory problem
- Nodes (vertices) represent routers
- Edges are network links
  - Have costs that typically represent delays or distances
- Find the lowest cost path between two nodes

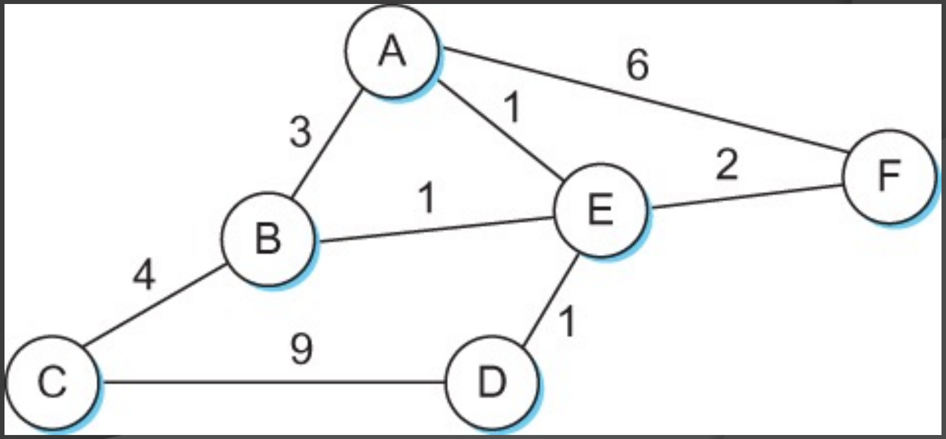

Figure 84

Edge Costs

- Could assign these statically
  - Link or node failures?
  - Addition of new links or nodes?
  - Dynamic edge costs?
- Need dynamic, <u>distributed</u> routing protocols
  - Distributed algorithm more scalable
    - May introduce consistency errors

Distance-Vector Routing

- Each node creates a vector of the distances/costs to all other nodes
  - Distributes this vector to all of its neighbors
- Step 1: create tables for initial distances & routing
    - Assumes each node knows costs to each directly connected neighbor
  - Initial distance table
    - Currently believed distances from node to all other nodes
  - Routing table
    - Contains distance to destination and next hop
  - Down link → infinite cost/distance

Distance-Vector Routing continued

- Step 2:
  - Every node sends message of its personal list of distances to its directly connected neighbors
  - Updates its distances to reflect new info
- Takes a few exchanges of info before each node has a complete routing table
  - Convergence

Distance-Vector Routing Example

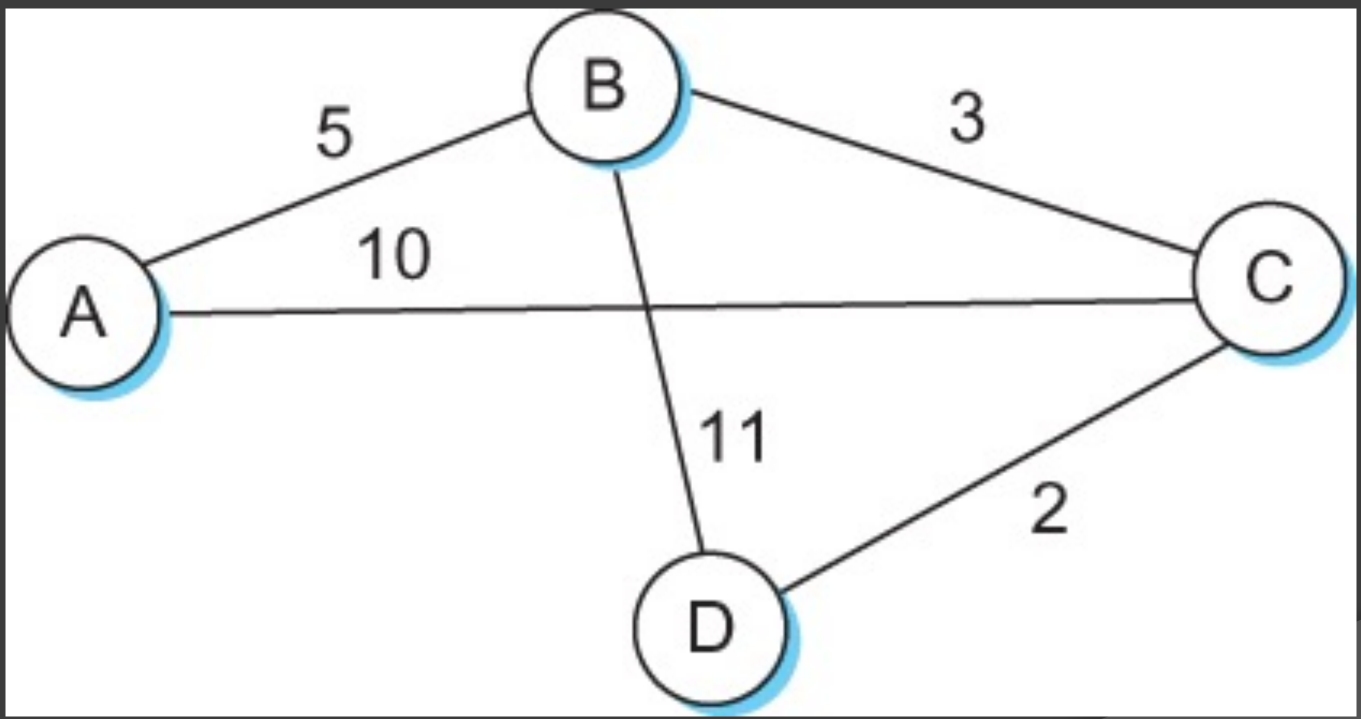

Figure 89

Routing Updates

- Periodic Update
  - Lets neighbors know node is still running
  - Allows changes in distance/cost to be known
- Triggered Update
  - Happens when a node notices a link failure
  - Node receives an update from neighbor and changes a route in its routing table

Count to Infinity Problem

- Consider what happens when F detects the link from F to G fails
- Now consider what happens when A detects the link from A to E fails
  - This is the count to infinity problem

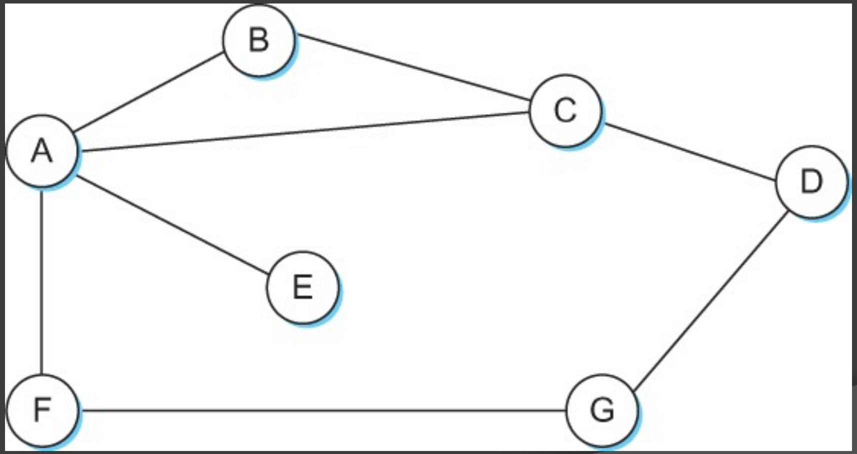

Figure 85

Solutions

- Use a relatively small number to approximate infinity (e.g., 16 hops)
- For routing loops with two nodes
  - Split Horizon
    - Do not send routes learned from a neighbor back to that neighbor
  - Split Horizon with Poison
    - Send negative info about routes learned from a neighbor back to that neighbor

RIP: Routing Information Protocol

- Routers advertise the costs of reaching networks

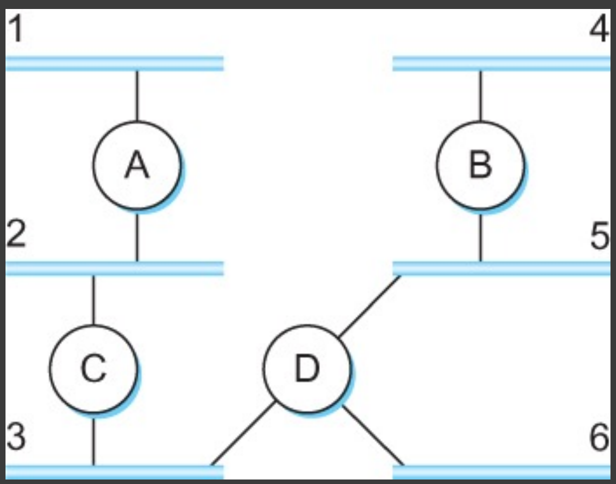

Figure 86

0	8	16 31
Command	Version	Must be zero
Family of net 1		Route Tags
Address prefix of net 1		
Mask of net 1		
Distance to net 1		
Family of net 2		Route Tags
Address prefix of net 2		
Mask of net 2		
Distance to net 2		

Figure 87

Link-State Routing

- Relies on
  - Reliable broadcast of about how to reach neighbors
  - Sum of all of the accumulated link-state info
- Reliable flooding
  - All nodes participating in routing need a copy of link-state info from all other nodes

Link-State Packet (LSP) Contains

- The ID of the node that created the LSP
- A list of the node's neighbors
  - And the cost of the link to each one
- A sequence number
- A time-to-live (TTL) for the LSP
  - Make sure you have the most recent copy of info

Link-State Flooding

- If X receives a LSP from Y
- X sees if it has a stored copy of that LSP
- If not, stores this LSP and forwards to neighbors
- If it does, then it check the sequence numbers
- Does the new LSP have a higher sequence number?
  - If yes, then this LSP is newer
    - X stores the LSP and forwards to neighbors
  - If not, then this LSP is not newer
    - X discards the packet (and does not forward it)

Link-State Flooding Example

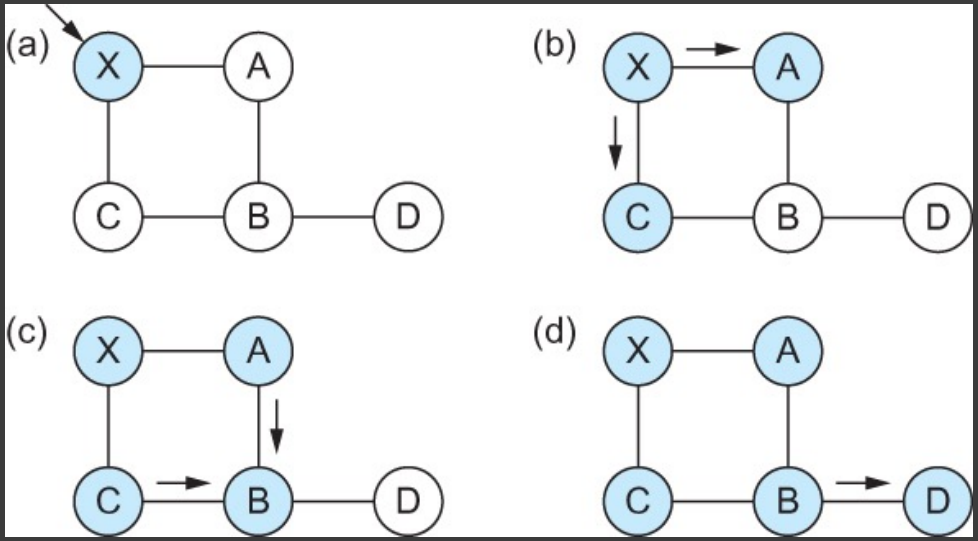

Figure 88

Generation of LSPs

- A node will generate a LSP if
  - Periodic timer expires
  - Change in network topology is detected
    - Neighbor goes down
    - Link to neighbor goes down
- Similar to RIP (for Distance-Vector Routing)

Design Goals of Link-State Routing

- Newest link-state info must be flooded to all nodes as quickly as possible
- Old info must be removed from the network and not allowed to circulate
- Try to minimize amount of routing traffic on the network

Methods to Achieve Design Goals

- Reduce overhead by only generating LSPs when absolutely necessary
- LSPs carry sequence numbers
  - Node increments # when generating an LSP
  - Old info is replaced by new info
- LSPs carry a TTL
  - Node decrements TTL when forwarding an LSP
  - LSP also ages when stored in a node
  - Old info is eventually removed from the network

Dijkstra’s Algorithm

- When a node has copies of the LSPs from all other nodes, it has a complete map of the network
- How to calculate routes using this info?
- Uses Dijkstra’s shortest-path algorithm from graph theory

Dijkstra's Algorithm Example

- Calculate the routing table using forward search algorithm
- Two lists: **Tentative** and **Confirmed**
- Each list contains:
  (Destination, Cost, NextHop)
- Example seems to have false info (in Tentative) but then corrects itself

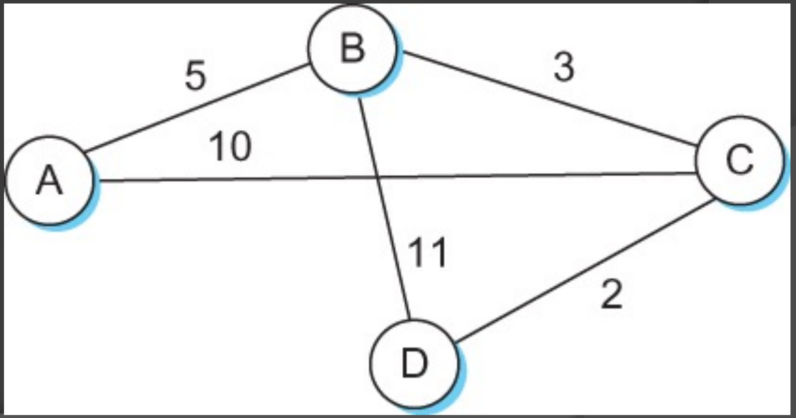

Figure 89

Pros/Cons to Link-State Routing Algorithm

- Stabilizes quickly
- Doesn’t generate much traffic
- Responds quickly to topography changes
- Each node must store a large amount of info

OSPF Protocol

- Open Shortest Path First
- Adds features to basic link-state routing
  - Password authentication of routing messages
  - Additional hierarchy
    - A domain may be partitioned into areas
  - Load balancing

OSPF Packets

0	8	16	31
Version	Type	Message length	
SourceAddr			
AreaId			
Checksum		Authentication type	
Authentication			

Figure 90. OSPF header format

Figure 91. OSPF link-state advertisement

LS Age			Options	Type=1
Link-state ID				
Advertising router				
LS sequence number				
LS checksum			Length	
0	Flags	0	Number of links	
Link ID				
Link data				
Link type		Num_TOS	Metric	
Optional TOS information				
More links				

Distance-Vector vs. Link-State

- Distance-Vector Routing
  - Each node only communicates with directly-connected neighbors
  - Tells them everything it has learned
- Link-State Routing
  - Each node communicates with all other nodes
  - Tells them only the state of directly-connected links
- ARPANET demonstrated better stability for Link-State Routing

How to Calculate Link Costs?

- Use cost of “1” per link?
  - Does not distinguish links using latency, bandwidth, or current load
- Use queue length?
  - Does not consider link bandwidth or latency
- Use delay as a measure of load?
  - Works well for light loads, but unstable for heavy loads
  - Range of link values were much too large

Metrics in Reality

- Static metrics are the actual norm
  - Metrics controlled by network admin
  - Dynamic metrics are too unstable
- Differences in link speeds/latencies not as great
- Often set as: *constant* * 1/*link_bandwidth*

How do you build a router or a switch?

- Similar designs
- Router has a few more complexities
- We'll start by looking at a switch design

A Simple Switch

- General-purpose processor used as a switch
- Throughput shared by all users of the switch
- All packets pass through a single point of contention
  - Bottleneck
- Well designed switch moves data from inputs to outputs in parallel (if possible)

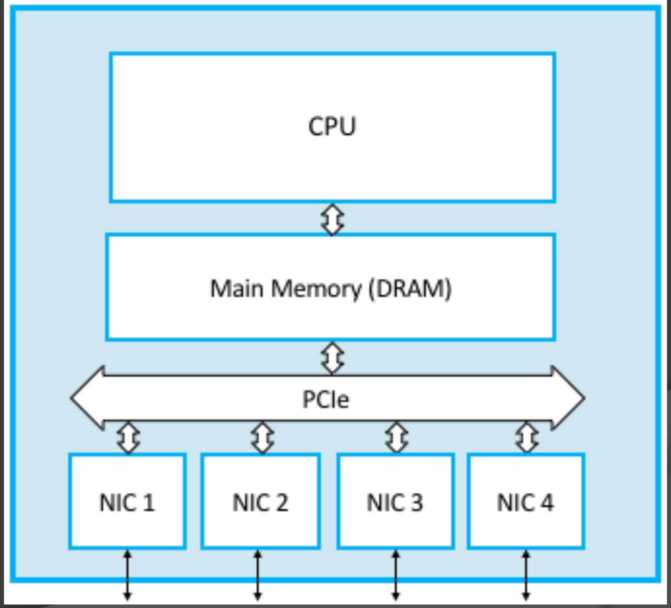

Figure 93

Input Ports

- Input port usually figures out where packet needs to go
- Port sets up fabric to deliver packet to output port
  - Or attaches info to the packet for the fabric to read
    - Self-Routing Fabric

Input Ports continued

- Cause performance bottlenecks
  - Receives a steady stream of packets
  - Analyze header and determine which output port(s) the packet must be sent to
  - Pass the packet onto the fabric
- Must handle packets at a high rate
  - Input link rate
- What's tougher?
  - High data rate link transmitting small packets
  - High data rate link transmitting large packets

Buffering

- Can happen at:
  - Input port
  - Output port
  - In the fabric (internal buffering)
- Main source of delay in a switch
  - Packets can be dropped
- Input buffering has serious limitations
  - Head-of-line blocking
- Most switches use output buffering or mix of internal and output buffering

Fabrics

- Try to achieve high levels of parallelism
- Some common types
  - Shared bus
    - Bus throughput is a bottleneck
  - Shared memory
    - Packets are written/read from memory locations by input/output ports
    - Memory bandwidth is a bottleneck

Crossbar Fabric

- Can be configured to connect any input port to any output port
- Requires each output to accept packets from all inputs at once
  - Need a lot of memory bandwidth at the outputs
- In reality, more complex designs are used

Self-Routing Fabric

- Often use many simple, interconnected 2x2 switching elements
- Among most scalable fabric designs

Router Implementation

- Can be built similar to a switch
  - But there are a few differences
- Routers must handle variable-length packets

Router Implementation continued

- Harder to characterize performance
- Forwards a certain number of packets per second
  - Must decide what packet size to support at line rate
- Likely to sustain rate for larger packets, but not for smaller packets
- Choose minimum IP packet size (40 bytes)?
- Choose average IP packet size (300 bytes)?

Router Implementation continued

- Centralized Forwarding Model
  - One processing engine for IP forwarding algorithm on all ports
- Distributed Forwarding Model
  - Several processing engines
    - One per port
    - More often, one engine serves one or more physical ports
  - Each forwarding engine needs a forwarding table
    - All forwarding tables need to be up to date

Router Implementation continued

- IP forwarding algorithm more complex than fixed-length MAC address or VCI lookup in a table
  - E.g., subnetting, CIDR

- Use a network processor
  - Optimized for network tasks
    - Lookups of addresses, checksum calculation, etc.