Fundamental Network Design Issues

Routing

CS2520/TELCOM2321
Wide Area Network
Spring Term, 2019

Prof. Taieb Znati
Department Computer Science
Telecommunication Program

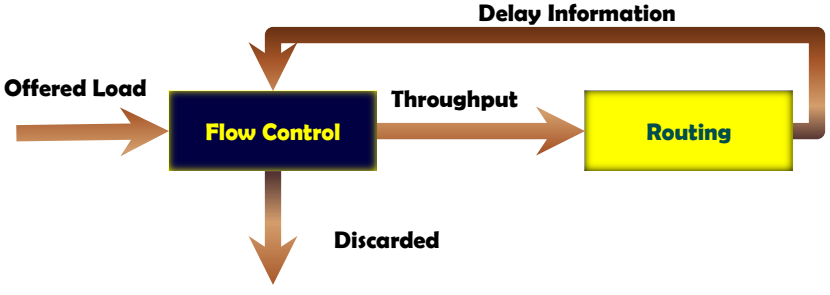
Outline

- Networking Routing Protocols
  - Routing Protocol Design Issues
  - Distance Vector Protocols
  - Link State Protocols

PART 2 – ROUTING

Routing and Flow Control

- Routing determines paths for data flowing between source and destination pairs
- Flow control regulates traffic to avoid congestion
  - Explicit Congestion Indication
  - Implicit Congestion Indication



Routing

- The principal task of the network layer is to accept packets from the source node and deliver them to the destination node
- Generally, more than one route is possible
  - A routing function must be performed
    - Routing algorithm
  - A number of desirable attributes for routing
    - Correctness, simplicity, robustness, stability, fairness, and optimality

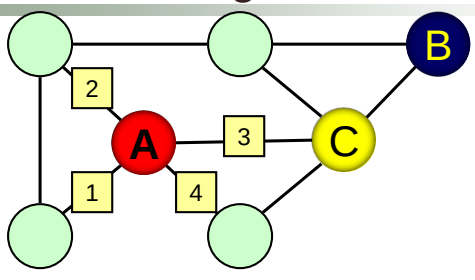
Dimensions of a Routing Task

- In connection-oriented service, routing decisions are only made when a new virtual circuit is being set up
- In datagram service, the decision is made anew at the arrival of every new packet

Routing with Datagrams

- Each node maintains a routing table
  - Entries in the routing table may be based on estimates of congestion in the network
  - Entries may contain alternate links for routing
    - Alternate links can be selected randomly, in a round robin fashion, or used to balance the traffic load

Datagram Routing Tables

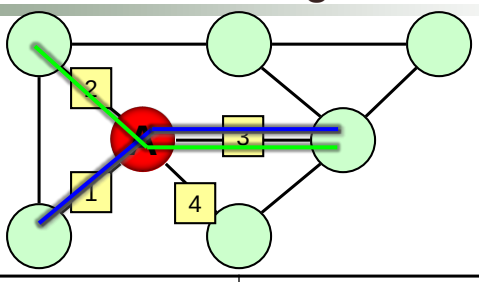


Destination	Link	Fraction
B	2	0.6
	3	0.4
C	3	0.8
	4	0.2

Routing with Virtual Circuits

- Routing tables for virtual circuits indentify circuit numbers instead of destinations
  - Entries in the routing table specify for each incoming link and each incoming virtual circuit number, the corresponding outgoing link and outgoing virtual circuit number
    - Nodes assign numbers independently, so numbers may change along the path

Virtual Circuit Routing Tables



IN		OUT	
Link	Circuit	Link	Circuit
2	1	3	2
3	2	2	1
1	1	3	1
3	1	1	1

Routing Algorithms

- The main design goals of a routing algorithm are:
  - Simplicity and low overhead
  - Robustness and stability
  - Rapid convergence (for dynamic routing algorithms)
  - Flexibility
  - Optimality

Simplicity

- Routing algorithms must offer their functionalities Efficiency
  - Minimum of software and utilization overhead
- Efficiency is important, particularly when the software is implemented to run over a computer with limited resource capabilities

Robustness

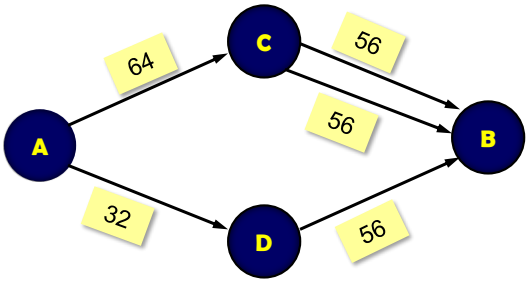
- Routing algorithm must perform correctly despite unforeseen circumstances
  - Hardware failures
  - High load conditions
  - Incorrect implementations
- Unstable routing algorithms can cause considerable problems when they fail at the network junction points

Robustness

- Robustness requires three tasks:
  - Monitoring the network status
  - Update routing decission to reflect new changes
  - Enforce new decisions to react as fast as possible to network changes

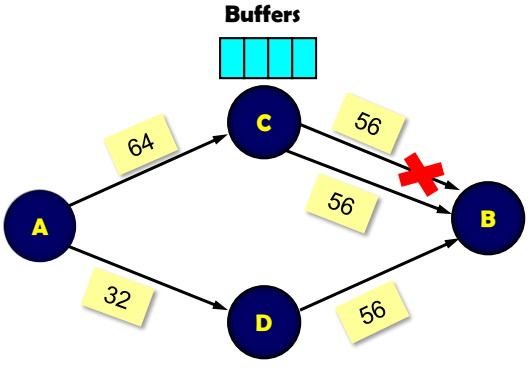
Robustness Example

- Buffers do not accumulate bits, since C and D rate of transmission is faster than input bit rates



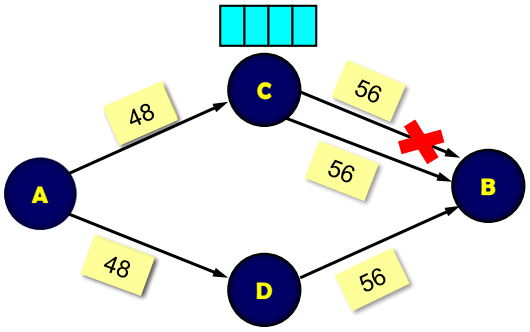
Robustness Example – Link Failure

- One of the dual links fails
  - Buffering occurs at node C, since the arrival rate is higher than the transmission rate



Robustness Example
Routing Decision Adjustment

- After *S* units of time, A learns of the failure and updates routing decisions to take the failure into consideration
  - Buffers at C begin to empty



Robustness

- The time *S* to learn about the failure has two implications
  - Buffers at C must be able to store the traffic exceeding the output link capacity for the period *S*
  - Bits are delayed at the node C during the failure

Stability and Rapid Convergence

- Convergence is the process of agreement by the routers on the optimal routes
  - Dialogue among routers is carried by update messages
  - Routing updates stimulate recalculations of optimal routes
- Routing algorithm that converges slowly can cause loops or network outages

Slow Convergence and Routing Loops

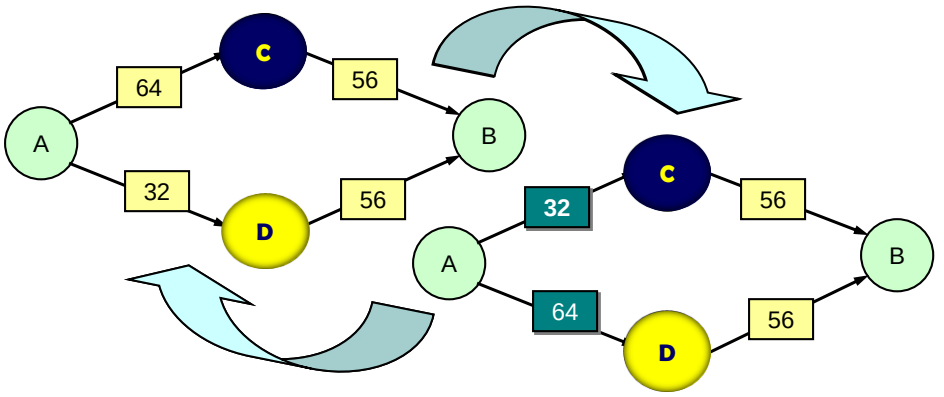
- Some algorithms exhibit slow convergence of routing information
  - Information about possible routes is computed on the basis of notification messages coming from router's neighbors
  - In a case of failure of a distant link, the decision about the best route might be based on an outdated information
  - The process of detecting that there is no valid route to a destination may take a long time
  - Switching to another route may create temporary routing loops (until all routers' tables converge to a steady state)

Flexibility

- Flexibility refers to the capacity of the routing algorithm to adapt quickly and accurately to a variety of network circumstances
- Flexible routing algorithms must adapt easily to:
  - Changes in the network bandwidth
  - Increase of the router queue size
  - Network delay
  - Link failures

Stability

- The existence of different routes to a destination means possible stability problems, if routing decisions are based on the network delay or the queue size



Routing Desirable Characteristics – Fairness

- Routing must accomodate traffic with different quality of service requirements (QoS)
- Packets within the same class should suffer similar delays

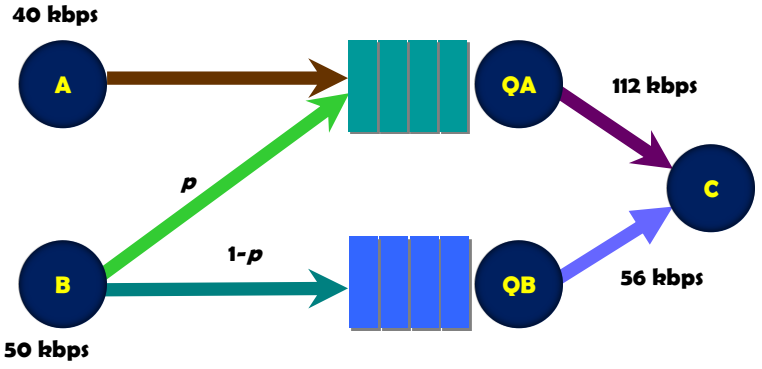
Fairness

- A commonly used formula to compute the delay, *D*, a packet suffers at a node is:

$$D = \frac{1}{\mu - \lambda}$$

  - μ - the transmission rate
  - λ - the arrival rate

Fairness - Example



Fairness – Example

- Using the delay formula, we can write for the stream incoming to the queue A:

$$D_A = \frac{1}{112 - \lambda_A}$$

$$\lambda_A = 40 + 50p$$

$$D_A = \frac{1}{72 - 50p}$$

Fairness – Example

- Similarly, using the delay expression, we can write for the stream incoming to the queue B:

$$D_B = \frac{1}{56 - \lambda_B}$$

$$\lambda_B = 50 - 50p$$

$$D_B = \frac{1}{6 + 50p}$$

Fairness – Example

- A fair routing algorithm must select *p* such that:

$$D_A = D_B$$

- Solving for *p* results in:

$$72 - 50p = 6 + 50p$$

$$66 = 100p$$

$$p = 0.66$$

Optimality

- Optimality refers to the capability of the routing algorithm to find the “best route”
  - It is strictly defined with respect to the routing metrics and metric weightings
- May be difficult to achieve

Routing Metrics

- Path length, sum of the costs associated with links
  - Frequently reduced to hop count
- Reliability, link bit error rate, usually assigned by a network administrator
- Delay, as the amount of time required to move packets from a source to a destination
  - Delay depends on the link bandwidth, router queues and the network load
- Communication costs to reduce expenditures

Optimality

- Consider the previous example and assume that the objective is to minimize the average packet delay:

$$D = \frac{40}{40+50} D_a + \frac{50}{40+50} D_b$$
$$D_a = D_A$$
$$D_b = pD_A + (1-p)D_B$$

Optimality

- Taking the derivative of the delay expression *D* with respect to *p* and setting it to 0 produces the value of *p* that minimizes the average network delay
  - Solving the equation results in:

$$D = \frac{40}{90}\frac{1}{72-50p} + \frac{50}{90}\left(\frac{p}{72-50p} + \frac{1-p}{6+50p}\right)$$

$$\frac{70}{9}\left(\frac{2}{(36-25p)^2} - \frac{1}{(3+25p)^2}\right) = 0 \qquad p \approx 0.526$$

ROUTING ALGORITHMS AND PROTOCOLS

Routing Algorithm Classification

- Routing algorithms can be classified as:
  - Static or dynamic
  - Single path or multipath
  - Flat or hierarchical
  - Intradomain or interdomain
  - Link state or distance vector

Static Routing

- Static routing algorithms use a mapping predefined by the network administrator to determine routes
  - Routing table mappings do not change unless the administrator changes them
  - Static routing algorithms are simple to design
  - They only work well in environments where the network traffic is predictible and the network design is simple

Dynamic Routing

- Dynamic routing algorithms adjust, in the real time, to the current network condition
- Dynamic changes are conveyed by routing update messages
  - New routes may result upon exchange of update messages
- Dynamic routing may be supplemented by static default routes, as a “last resort”

Single Path or Multipath

- Routing algorithms may support multipaths to the same destination
  - Multipath permits traffic multiplexing over multiple links
- Multipath algorithms may improve the network throughput

Hierarchical Routing

- A group of routers form what amouts to a Routing Backbone
- Hierarchical routing is usually defined based on the concept of Autonomous Systems (AS's) or Areas
  - Backbone routers can communicate between domains, while other routers are restricted to deal with intradomain routing
- Hierarchical routing mimics the organization of most companies
  - Easier to implement and manage for large networks
  - Easier to connect different parts of network managed by different operators

Flat Routing

- In a flat routing algorithms, all routers are peers
  - Routing algorithm operates on a flat name space
- Generally, flat routing is more efficient than hierarchical routing, allowing more direct paths, thus reducing the average delay and increasing the network capacity

Host **OR** Router Intelligent

- Source routing assumes that the source node determines the entire route to the destination
  - Routers merely act as store-and-forward nodes
- Router intelligent algorithms rely entirely on the router to determine the path to the destination

Host or Router Intelligent – Trade-Off

- Trade-off – route optimality versus traffic overhead
- Source routing can achieve optimality, since the end node computes all paths to the destination
  - Optimality may vary to reflect a particular system's definition
- Discovering all routes, however, may be costly
  - Typically requires broadcasting by the source to discover all paths to destination
    - Destination collects information about all possible paths and sends back the result to source
    - Source selects the best path to destination, based on its own meteric

Intradomain or Interdomain

- Intradomain algorithms are optimized to work within a domain: managed by a single operator
- Interdomain algorithms are optimized to work between domains
  - The nature of these algorithms are different
    - Political regulations
    - Security issues
    - Quality of service problems
  - An "optimal" intradomain routing algorithm may not necessarily be optimal or even perform well between domains

Routing Techniques
Path Optimality Principle

- "If a router J is in the optimal path from I to K, then the optimal path from J to K also falls along the same route"
- As a direct consequence, the set of optimal routes from all sources to a given destination from a tree rooted at the destination
  - The tree is referred to as "a sink tree"

Routing Techniques
Bellman-Ford-Moore Algorithm

- The network is represented as a set of nodes with N elements
- D_i represents the minimum distance for reaching a node i from a specified origin
- C_{ij} represents cost (distance) of the link between a node i and a node j
  - $C_{ii} = 0$
  - $C_{ij} = \infty$, if the node i and the node j are not adjacent

Shortest Path Algorithms
Bellman-Ford-Moore Algorithm

- The objective is to find optimum routes from a given source, say node 1, to all other nodes in the network
- The principle of optimality can be expressed as:

$$D_i = \min_j \{D_j + C_{ji}\}; \quad \text{for all } i \neq 1$$

$$D_1 = 0$$

Bellman-Ford-Moore Algorithm

- The algorithm iterates on the number of "hops" in the path
  - First find optimal paths from a prescribed source node to all other nodes subject to the constraint no path contains more than one hop, then finding optimal paths subject to the constraint no path contains more than two hops, etc.
- The shortest path subject to the constraint no path contains more than *h* hops is called the shortest ($\leq h$) path

Bellman-Ford-Moore Algorithm

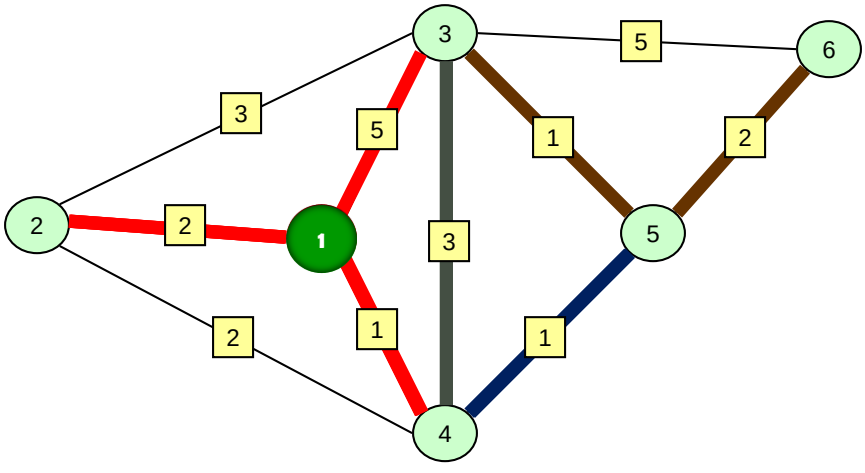
- Based on the shortest ($\leq h$) path concept, the algorithm can be expressed as follows:
  - Let $D_i^{(h)}$ be the shortest ($\leq h$) path from 1 to i
  - For each successive $h \geq 0$, compute

$$D_i^{(h+1)} = \min_j \left\{ D_j^{(h)} + C_{ji} \right\}; \quad \text{for all } i \neq 1$$

$$D_i^{(0)} = \begin{cases} 0 & i = 1 \\ \infty & i \neq 1 \end{cases}$$

  - Connect i with the predecessor node, j, for which the minimum is obtained, possibly eliminating any connection formed during an earlier iteration
  - Continue until no more changes occur in the next iteration

BFM Algorithm – Example



Bellman-Ford-Moore Algorithm Complexity Analysis

- Worst case computational complexity?
  - For a network of N nodes, a path can contain at most N-1 links
    - Hence, N-1 iterations, over the number of hops, before convergence
  - For each iteration, the minimization is performed for N-1 nodes
  - Fore each node, the iteration involves at most N-1 alternatives
- Thus, the worst case computational complexity is $O(N^3)$
  - More careful analysis results in $O(mL)$, where m is the number of interations required to converge, and L is the number of links

Shortest Path Algorithms Dijkstra Algorithm

- The objective is to find shortest paths from a specified source, say 1, to all other destinations
- The approach is similar to BFM, but iterates on the length of path
  - Let N_c be the set of nodes currently connected to the destination, D_j the current estimate of the distance to the node j from the destination
    - Initially: $N_c = \{1\}$, $D_1 = 0$, and $D_j = \infty$, for $j \neq 1$

Dijkstra Algorithm

- Step 1: Update labels
  - For all $j \notin N_c$, set

$$D_j = \min_{i \notin N_c} \{D_j, D_i + C_{ij}\}$$

    - If the minimum is the same as previous D_j, leave the tentative connection of j to a successor unchanged
      - Other wise – i.e., new minimum – connect j to the successor i, for which the minimum is obtained

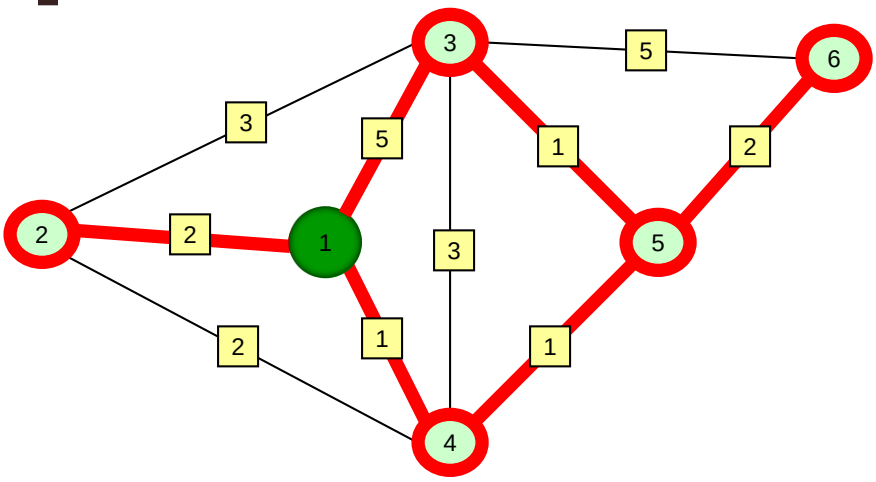
Dijkstra Algorithm

- Step 2: Find the next closest node
  - Find $i \notin N_c$, for which

$$D_i = \min_{j \notin N_c} \{D_j\}$$

  - Set $N_c = N_c \cup \{i\}$, and premanently connect i to the appropriate successor node
  - If $N_c = N$, then stop, else return to Step 1

Dijkstra Algorithm – Example



Dijkstra Algorithm – Complexity Analysis

- Each step of the algorithm requires an amount of computation proportional to the number of nodes N
  - Each step is repeated N-1 times
- Worst case Computational Complexity is
  - $O(N^2)$, better than $O(N^3)$
    - A significant improvement over BFM, if the network is small and dense, with a high ratio of the number of links to the number of nodes
  - Usually, however, $L \ll N(N-1)$, resulting in $m \ll N$, and $mL \ll N^3$

Distributed Asynchronous BFM Routing Algorithm

- One of the main reasons of the popularity of BFM is its suitability for implementation in a distributed manner
- A precise implementation of the algorithm requires synchronization among the nodes
  - All nodes must start at the same time
  - All nodes must abort and restart if a link status changes while the computation is running

Distributed Asynchronous BFM Routing Algorithm

- Practical implementation uses an asynchronous approach, which eliminates the need for the synchronization, by computing distances at regular intervals
  - Convergence occurs to correct distances if no changes occur during route update time

Routing Algorithms Practical Implementations

- Modern computer networks use dynamic routing algorithms rather than static ones
- Two implementations of dynamic algorithms are frequently used:
  - Distance Vector Protocol
  - Link State Protocol

Distance Vector and Link State Protocol

- Distance Vector algorithms (BFM) call for each router to send all or some portion of its routing table, but only to its neighbors
- Link State algorithms (Dijkstra) flood link state information to all nodes
  - Each router sends only portion of the routing tables that describes the state of its own links
    - Link State algorithms converge more rapidly, so they are less prone to routing loops
    - Link State algorithms may be computationally expensive

Distance Vector Routing

- Basic idea:
  - Each router starts out with a distance 0 for those networks it is attached to
  - Each router transmits its distance vector to each of its neighbor periodically or whenever the information changes
  - Each router calculates its own distance vector based on the information received from each neighbor

Distance Vector Routing

- Router Z, attached to Net 1 and Net 2

Destination	Distance	Route
Net 1	0	Direct
Net 2	0	Direct

- At a given time

Destination	Distance	Route
Net 1	0	Direct
Net 2	0	Direct
Net 3	9	Router A
Net 4	5	Router B
Net 5	7	Router X
Net 6	3	Router D
Net 7	3	Router X

Distance Vector Routing

- Z received an update message from X

Destination	Distance
Net 1	2
Net 3	5
Net 4	7
Net 5	6
Net 6	10
Net 7	4
Net 8	4

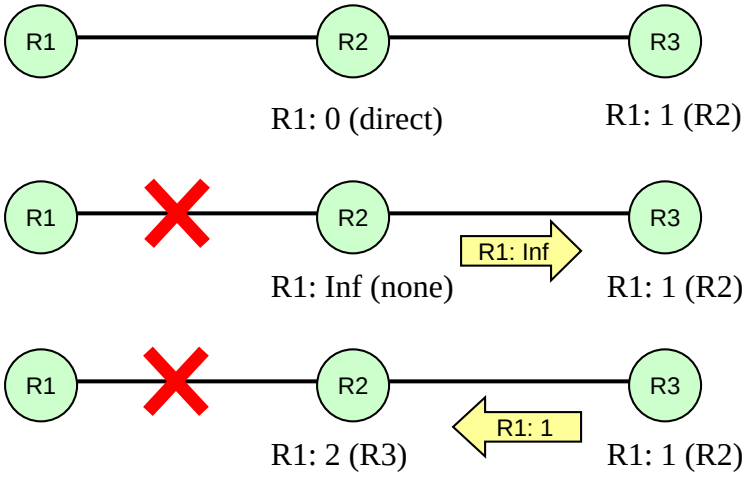
Destination	Distance	Route
Net 1	0	Direct
Net 2	0	Direct
Net 3	9	Router A
Net 4	5	Router B
Net 5	7	Router X
Net 6	3	Router D
Net 7	3	Router X

Distance Vector Routing

- Z updates its table

Destination	Distance	Route
Net 1	0	Direct
Net 2	0	Direct
Net 3	6	Router X
Net 4	5	Router B
Net 5	7	Router X
Net 6	3	Router D
Net 7	5	Router X
Net 8	5	Router X

Distance Vector Routing – Link Failure Bouncing Effect – Counting to Infinity

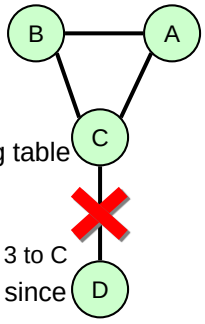


Infinite Loop Fixes

- Split horizons
  - Never send the information about a route back in the direction from which it came
  - Helps prevent two-node routing loops
  - The solution may fail in larger size networks

Split Horizon Failure

- Initially A and B have a distance 2 to D via C
- The link C-D goes down
  - C marks D unreachable and reports that to A and B
- Suppose A learns first that D is unreachable via C
  - A sets D's routing table entry to infinity
- B, who did not hear from C yet, advertises its routing table to A
  - A thinks that the best path to D is through B.
  - A updates its routing table and reports a route of cost 3 to C
    - There is no violation of the split horizon rule, since A learned the information from B
  - C thinks D is reachable via A at cost 4 and reports that to B
  - B reports a cost of 5 to A who reports new cost to C , etc ….



Sequence of Events

Split Horizon with Poison Reverse

- Split horizon with poison reverse improves distance vector convergence over simple split horizon by advertising all network IDs to all neighbors
  - But those network IDs learned from a given direction are advertised to the same direction with a metric of ∞, indicating that the network is unavailable
- Poison revers has no benefit beyond split horizon in a single path internetwork
- In a multipath internetwork, split horizon with poison reverse greatly reduces count to infinity and routing loops

Infinite Loop Fixes
Poison Reverse Updates and Hold Down

- Increases in routing metrics generally indicate routing loops
- When increase is observed, send “Poison Reverse” updates to remove route and place it in “Hold Down”
  - “Hold Down” causes discarding of any changes affecting recently removed routes for a period of time
- Prevents larger routing loops, but may slow down the convergence

Distance Vector Shortcomings

- Slow convergence when routes change frequently
  - Information cannot propagate through a node until that node recalculates its routing information
  - Many routers may become nonfunctional as a result of the delay
- May cause Routers to have wrong information
- Does not scale properly to large networks
- May require large message exchange before stability

Link State Routing

- The algorithm requires that each participant has complete network topology information
- Basic steps include
  - Discover neighbors and learn their network addresses
  - Measure the delay or cost to each of the neighbors
  - Construct a packet containing the information just learned
  - Send the packet to all other routers
  - Compute the shortest path to every router

Link State Routing

- Each participant actively tests the status of all links to neighbor Routers
  - This is accomplished by periodically exchanging “alive” messages with neighbors
- Each participant forms and propagates the link status information to all other participants using a Link State Packet
  - Periodically,
  - When a neighbor becomes active,
  - When a cost of a link changes (after a link failure e.g.)
- Armed with a complete map of the topology, participants compute routes to each destination

Link Cost

- Routers are expected to have a reasonable estimate of the delay to each of the neighbors
  - Can be achieved by measuring the round trip time of a special “ECHO” request packet sent over the link to the neighbor
    - In a response, the neighbor is expected to send an “ECHO” reply immediately
    - For better results, the test can be conducted several times and averaged out

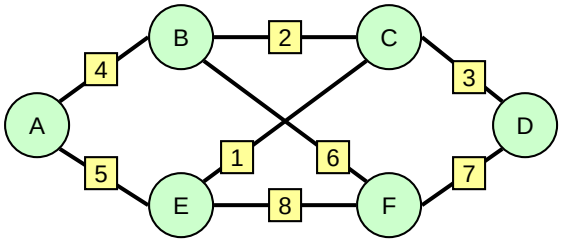
Link Cost Measurement – Load Factor

- Load can be factored in by starting the timer when the “ECHO” packet is queued, not when it reaches the front of the queue
- Should the load be taken into consideration?
  - Including traffic-induced delays in the measurement helps the router select the unloaded links over the loaded ones with the same bandwidth
  - Factoring the load in the delay measurements may cause routing table ocsillations which lead to undesirable erratic routing behaviour (inefficiency, loops)

Link State Protocol Packets

- Link State Packet contains the following information
  - Identity of the router
  - Sequence Number, incremented for each new packet sent
  - Age of the packet
  - List of the neighbors and the delay to reach them

Link State Protocol Packets – Example



A	
Seq #	
Age	
B	4
E	5

B	
Seq #	
Age	
A	4
C	2
F	6

C	
Seq #	
Age	
B	2
D	3
E	1

D	
Seq #	
Age	
C	3
F	7

E	
Seq #	
Age	
A	5
C	1
F	8

F	
Seq #	
Age	
B	6
D	7
E	8

LSP Sequence Numbers and Age Fields

- Routers can get confused if sequence numbers wrap around
  - Use a large sequence number field
    - A 32-bit sequence number would take 137 years to wrap around, assuming one LSP per second

LSP Sequence Numbers and Age Fields

- If a router crashes, it may lose track of its last sequence number
  - If it starts again at 0, the packet will be rejected as a duplicate
- If a sequence nubmer is corrupted, it may make other packets obsolete
  - A 1 bit error causes the sequence number 4 to be received as 65540
    - All packets 5 through 65540 will be considered obsolete
- Include an “Age” field for each packet which is decremented once per second
  - Age field is set to *max_value* (on the order of an hour) by the router that generates the LSP

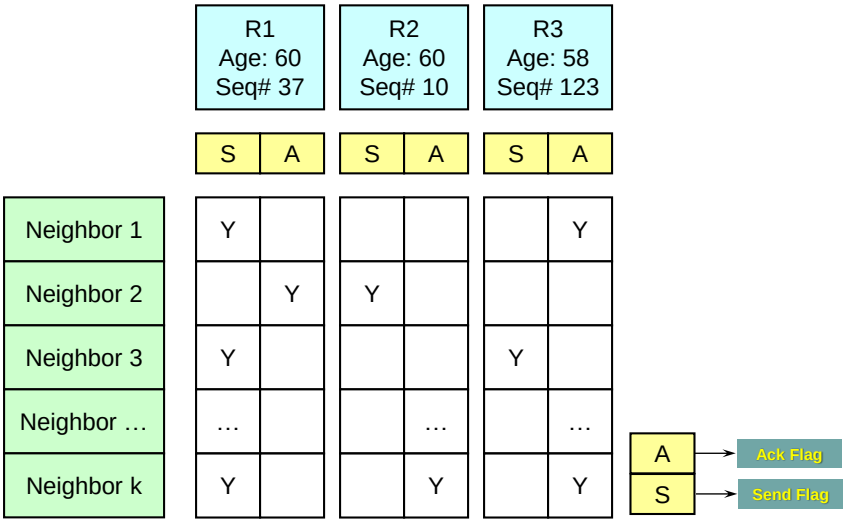
Link State Packets Dissemination Rules

- Only newly generated LSP are forwared to neighbors
  - LPSs are not flooded immediately, but queued in a holding area for a short period of time
- Discard any LSP packet if its “Age” is 0
  - Packet is considered too old
  - Prevents initially lost packets from living for an indefinite period of time
- Discard any received LSP packet from the same source, if the received packet is identical to the one stored in the memory
  - If the two packets are different, discard the older one

Link State Packets Disseminating LSPs to Routers

- To guard against errors on the router-to-router link, LSPs are acknowledged
  - When a line goes idle, the LSP holding area is scanned in a round-robin fashion to select a packet or an acknowledgement to send

Link State Protocol
Route Data Structure



Disseminating LSPs to Routers
LAN Based Interconnection

- Consider the LAN itself as a node
  - A router is elected to be the “designated router”
    - The designated router names the LAN and constructs an LSP on behalf of the LAN
    - Every router, including the designated router, reports only a single link in its LSP for the LAN itself

LAN Based Interconnection Reliable Dissemination

- Provide periodic summaries of the database content
  - A special packet, Sequence Numbers Packet (SNP), which contains the latest sequence numbers of all LSPs is multicast
  - A router can explicitly request any missing LSP from the designated router
  - If a router notices that the designated router missed its LSP, the router will retransmit the LSP
  - This scheme is adopted by ISO in IS-IS and DECnet Phase V

Outline

- Part 1 – Data Link Layer Flow Control
  - Stop-and-Wait Flow Control
  - Window-based Flow Control
  - Error and Flow Control
    - Stop-and-Wait ARQ
    - Selective Repeat ARQ
    - Go-Back-N ARQ
- Part 2 – Networking Routing Protocols
  - Routing Protocol Design Issues
  - Distance Vector Protocols
  - Link State Protocols