Routing in Switched Networks

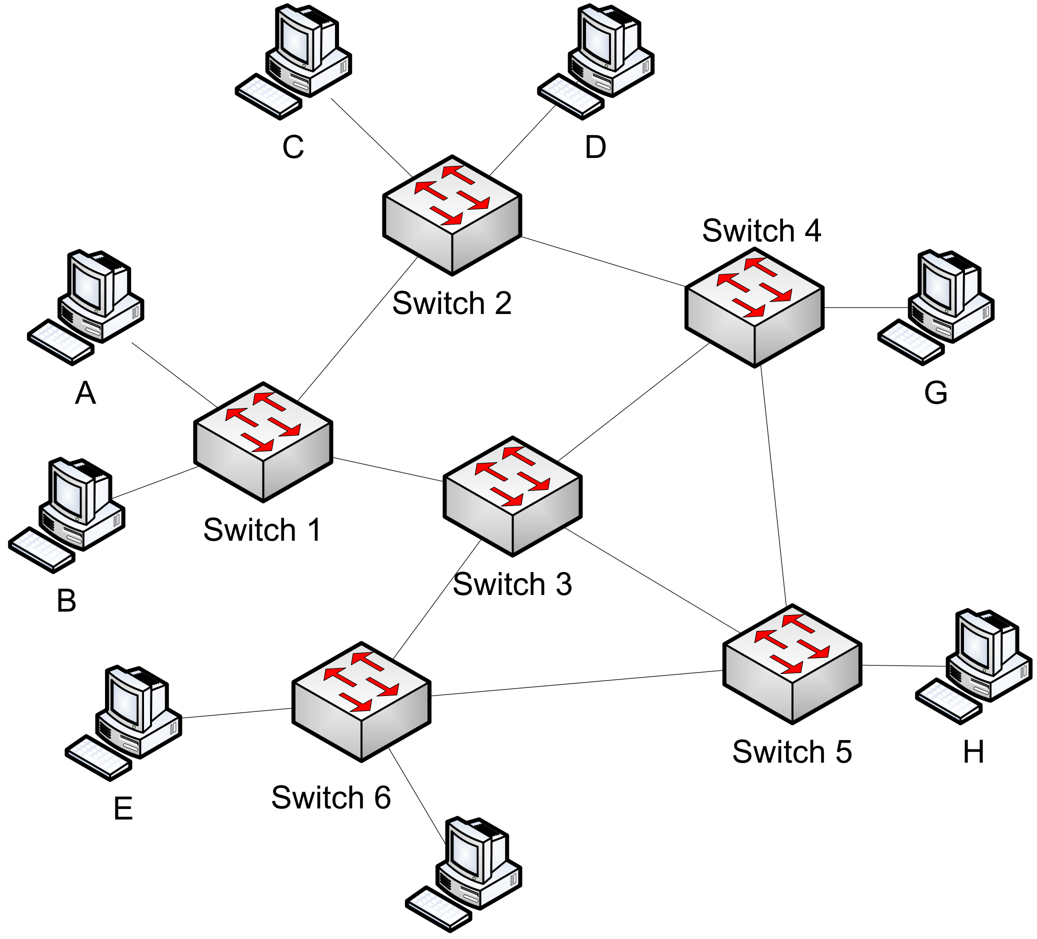
Dr Steve Gordon
ICT, SIIT

Contents

- Routing in Switched Networks — Concept
  - Requirements of Routing
  - Design Elements
- Routing Strategies — Concept
  - Fixed
  - Flooding
  - Random
  - Adaptive
- Routing Protocols and Algorithms — Example
  - An example of Link State Routing
  - Dijsktra’s shortest path tree algorithm

Routing in Switched Networks

- A needs to send data to H – which path does it take?



Routing in Switched Networks

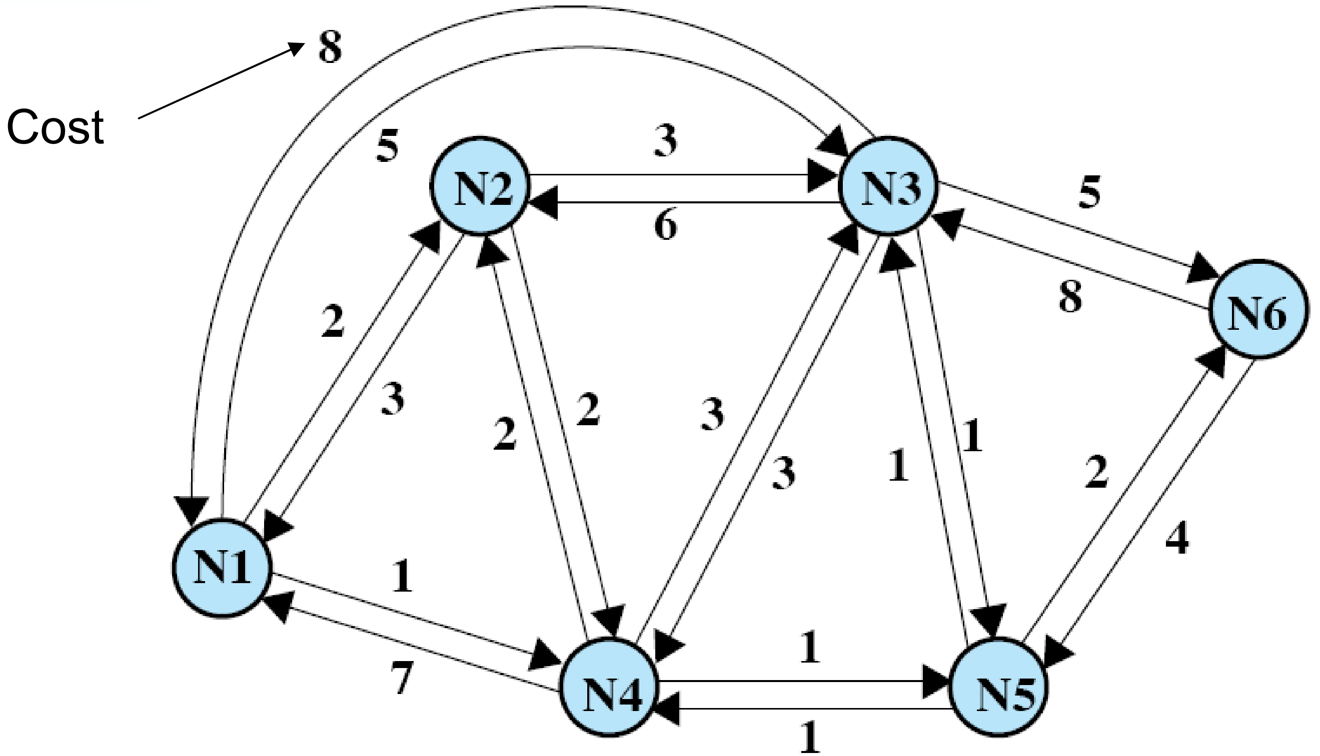
- Routing is a key design issue in switched networks
  - Question: What path (route) should be taken from source to destination?
    - Since there is often more than one path possible
  - Answer: Choose the “best” path!
    - What is “best”, and how to choose it?
    - Remember, real networks may have 100’s to 100,000+ nodes, and many possible paths

- We will look at general routing strategies
  - Routing is applicable to circuit-switched and packet-switched networks
  - We will focus on packet-switched networks in our examples

Requirements of Routing Algorithms

- A routing algorithm/protocol must meet many requirements:
  - Correctness
    - Must choose a path from intended source to intended destination
  - Simplicity
    - Easy and cheap to implement
  - Robustness
    - In presence of errors or overload in some parts of network, network should react to reduce load and find alternate paths
  - Stability
    - New paths (e.g. to avoid overload) should not be too frequent
  - Optimality
    - Choose best paths
  - Fairness
    - Some algorithms may give higher priority to stations close together (shorter paths) than further away (longer paths) – maybe efficient, but unfair to stations that want to communicate over long paths
  - Efficiency
    - Minimise the amount of processing and transmission overhead

Example Network Configuration



Link: a direct connection between two nodes (such as via cable or wireless). E.g. link from N1 to N2

Path (or route): a track or way between two nodes, via one or more links. E.g. a path from N1 to N6 is N1 – N2 – N3 – N6

Hop: traverse a link. E.g. there are 3 hops between N1 and N6 on path N1 – N2 – N3 – N6

Cost: some value assigned to links to allow Least Cost Routing algorithms to find a path

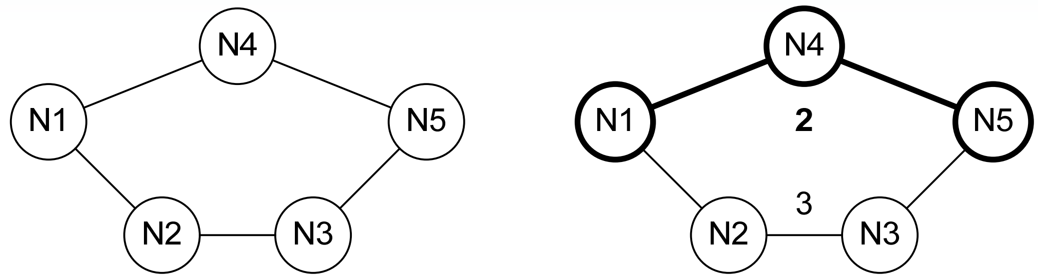
Topology: the arrangement of nodes, and their links, in a network

Elements of Routing Techniques

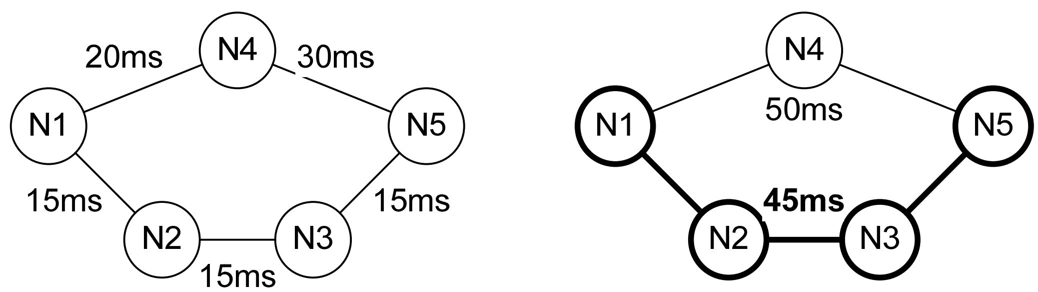
- A route is normally selected based on performance criteria
- A simple criteria: number of hops (select the path with least number of hops)
- In general, *least-cost routing* can be used:
  - Costs are assigned to links, and a routing algorithm selects a path with least cost
  - Cost can be different criteria or metrics:
    - Number of hops
    - Delay
    - Link capacities
    - Throughput (current utilisation)
    - Queue length
    - Error rates
    - Financial cost
    - Security levels
    - Political/legal matters
    - …

Examples of "Best" Paths

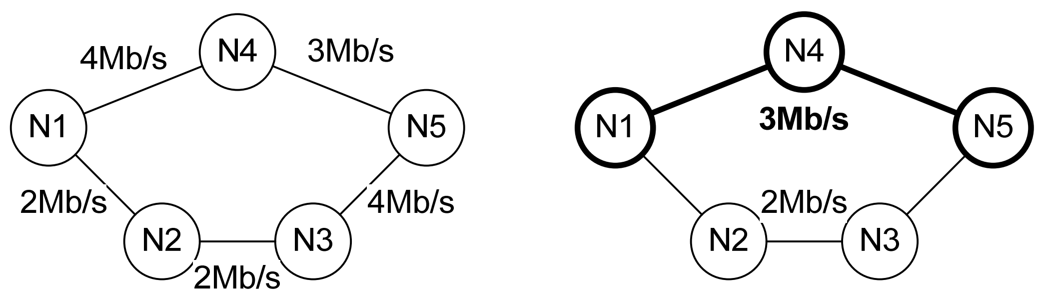
Minimum hops



Minimum delay



Maximum
data rate



Elements of Routing Techniques

- When and where do you make the routing decision?
  - When to make a decision for a route?
    1. For all packets, decision is made for the duration of network
    2. For each packet a decision is made (datagram packet switching)
    3. When setting up a virtual circuit, a route is chosen (virtual circuit packet switching)
  - Which nodes chooses the route?
    - Centralised: A single central node, e.g. in a Network Control Centre, decides on the routes used by all nodes
      - If central node fails, entire network fails!
    - Distributed: each node selects an output link for incoming packets (robust, but complex)
      - E.g. To send to H, A decides to send to Switch 1; Switch 1 decides to send to Switch 3; Switch 3 decides to send to Switch 5; Switch 5 sends to H
      - Most common approach today
    - Source Routing: source station decides a route
      - E.g. To send to H, A decides the path should be Switch 1 – Switch 3 – Switch 5

Elements of Routing Techniques

- To make routing decisions, nodes need to know information about the network
  - Topology, link costs, current usage of links and routers, …
  - How do nodes collect this information and how often is it updated?
    - More frequent updates give more reliable information and hence better routing decisions
    - But increases overhead (consumes network resources)
- Where does the information come from? The source of information may be:
  - None: no information about the network is used. E.g. Switch 1 makes a decision without knowing about other links or nodes in network
  - Local: use the information only known to the current node. E.g. Switch 1 makes a decision based on its 4 input/output links
  - Adjacent node: your neighbour nodes send you information. E.g. Switch 1 makes a decision based on its input/output links, as well as input/output links of Switches 2 and 3
  - Nodes along route: the nodes along the path/route send you information. E.g. Switch 1 makes a decision based on information from Switches 3 and 5.
  - All nodes: all nodes send you information.

Elements of Routing Techniques

- When does the information come? Updating may be:
  - Never: if no information about the network is used, you never update!
  - Continuous: if using local information, essentially you have access all the time. E.g. Switch 1 has continuous knowledge of its own link rates and usage
  - Periodic: at regular intervals. E.g. Switch 3 tells Switch 1 about the current utilisation of links every 1 minute
  - Major load change: in increase/decrease of traffic on a link. E.g. if utilisation of link from Switch 3 to 5 goes above 70%, the Switch 3 tells Switch 1
  - Topology change: a new node is added to or removed from network, or links are changed. E.g. the link from Switch 3 to 4 fails.

Obtaining information about network

- We said nodes often need to know information about the network
  - Topology, link costs, current usage, …
- And this information may be updated during the network operation
- What are the practical methods for obtaining this information?
  - The network administrator maintains the information:
    - When a network is built, the network administrator knows some of this information
    - When changes are made to the network (e.g. a new link added), the network administrator will be aware of this information
  - In moderate sized to large networks (more than 10's of nodes), it is not practical for a person to manage the network manually
    - Nodes involved in routing must automatically exchange information
    - That is, the network is used to exchange information about the network
- A trade-off in routing algorithms and protocols is:
  - Increasing the amount of routing information (in terms of size and frequency) that is exchanged, results in:
    - Increased accuracy in routing decisions (GOOD)
    - Decreased efficiency, since we use the network for exchanging routing information, instead of sending real data (BAD)

Routing Strategies

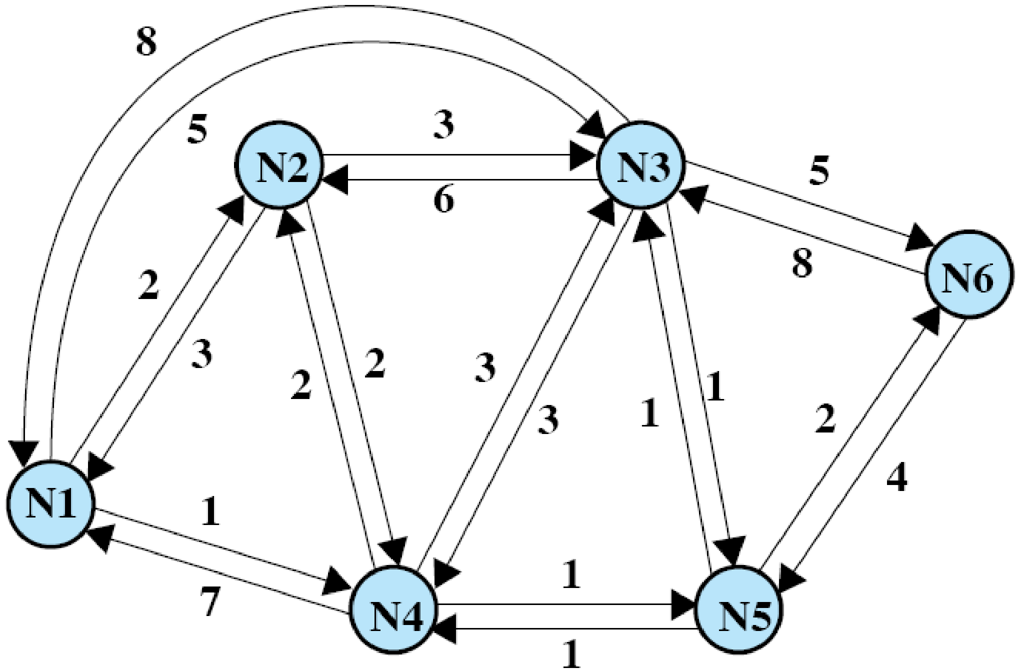
Fixed
Flooding
Random
Adaptive

Strategy 1: Fixed Routing

- Use a single permanent route for each source to destination pair
  - E.g. the route from A to H is ALWAYS via Switches 1, 3 and 5
  - The routes are determined using a least cost algorithm
    - E.g. Dijkstra and Bellman-Ford algorithms for shortest path in a graph
    - The metric can be anything. For example: monetary cost, delay, capacity, hops or a combination
  - Route is fixed
    - At least until a change in network topology (node/link added/deleted)
    - Hence cannot respond to traffic changes (e.g. overload in one portion of the network)
  - No difference between routing for datagrams and virtual circuits
  - Advantage is simplicity
    - You assign the routes at the start, and then nothing to do
  - Disadvantage is lack of flexibility
    - When the network is operating, changes in load may mean better routes than initially selected should be used

Fixed Routing Example

- Consider the example network. You can manually determine least cost paths from any source to any destination (there are 30 paths in total)
  - N1 to N6: 1 – 4 – 5 – 6 (cost = 4)
  - N2 to N5: 2 – 4 – 5 (cost = 3)
  - N4 to N6: 4 – 5 – 6 (cost = 3)
  - N6 to N1: 6 – 5 – 4 – 2 – 1 (cost = 10)
  - …



Fixed Routing Example

- Storing the least cost routes:
  - A route is a path from one node to another
    - E.g. Least cost route from N1 to N6 is: N1 – N4 – N5 – N6
  - A node should have least cost routes to all other nodes in network
  - However, there is no need to store the entire route
  - Each node keeps track of the next node in the least cost route from itself to a destination
    - E.g. N1 knows the next node in the path to destination N6 is N4
      - And N4 knows the next node in the path to destination N6 is N5
- Routing Tables (or Directories)
  - A node S stores a routing table containing two columns:
    - Destination Node, D
    - Next Node in path, N
  - "From source S, in order to reach destination D, send to next node N"
  - (Alternatively, if using centralised routing, a central node may store all this information)

Fixed Routing Example

Routing Table (Directory) stored on one centralised node

CENTRAL ROUTING DIRECTORY

To Node \ From Node	1	2	3	4	5	6
1	—	1	5	2	4	5
2	2	—	5	2	4	5
3	4	3	—	5	3	5
4	4	4	5	—	4	5
5	4	4	5	5	—	5
6	4	4	5	5	6	—

Fixed Routing Example

Routing Table (Directories) distributed amongst nodes

Node 1 Directory

Destination	Next Node
2	2
3	4
4	4
5	4
6	4

Node 2 Directory

Destination	Next Node
1	1
3	3
4	4
5	4
6	4

Node 3 Directory

Destination	Next Node
1	5
2	5
4	5
5	5
6	5

Node 4 Directory

Destination	Next Node
1	2
2	2
3	5
5	5
6	5

Node 5 Directory

Destination	Next Node
1	4
2	4
3	3
4	4
6	6

Node 6 Directory

Destination	Next Node
1	5
2	5
3	5
4	5
5	5

Fixed Routing Summary

- When is a decision made for a route?
  - At network startup
- Which node chooses the route?
  - Centralised or distributed
- Where does the network information come from?
  - All nodes
- When is the network information updated?
  - Never

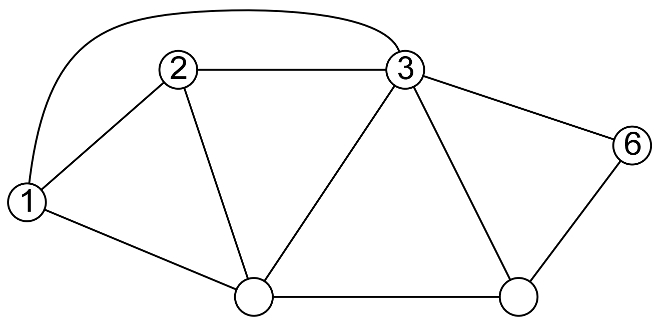
- In practice, only used for small, stable networks (10's of nodes)

Strategy 2: Flooding

- Instead of choosing a route *before* sending the data, just send the data to everyone!
  - It will eventually reach the intended destination
- Flooding:
  - A copy of the original packet is sent to all neighbours of the source (that is, all nodes that have a link to the source)
  - Each node that receives the packet, forwards a copy of the packet to all of its neighbours
- Some simple extensions to improve basic flooding:
  - Don't send back to the node that just sent you the packet
  - *Only forward packet once*: nodes remember which packets they have forwarded (based on sequence number and source/destination addresses); do not forward a packet if you have previously forwarded that same packet
  - *Duplicate detection*: each packet has a sequence number, so if destination receives multiple copies of the same packet, it can discard the duplicates
  - *Limit the number of hops*: include a "hop counter" in the packet; decrement the counter each time the packet is forwarded – if it is 0, then discard the packet
    - Limits the number of packets sent and useful if errors in the network (a packet will not be sent forever)

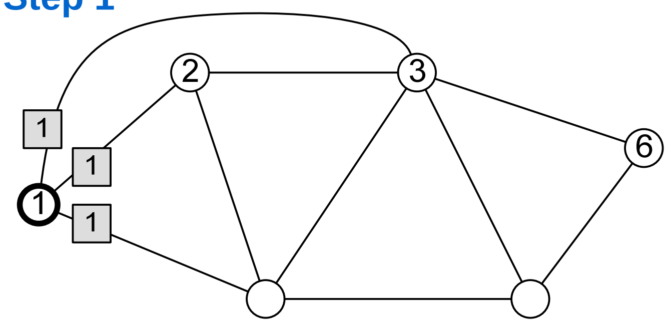
Flooding Example

Network topology



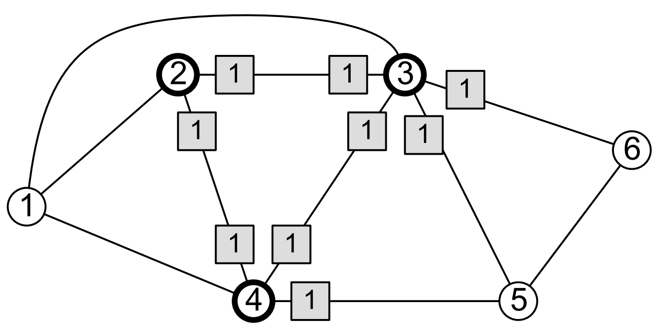
Node 1 has a packet to send to Node 6. Assume all links are full-duplex.

Step 1



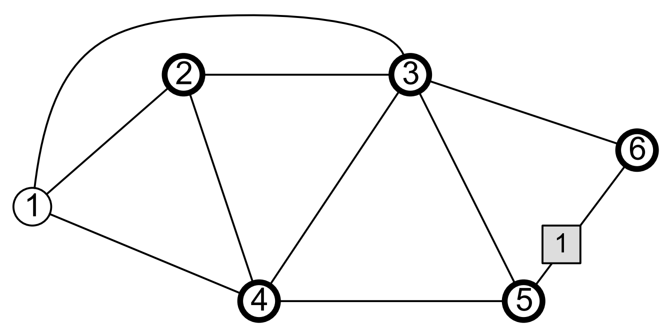
Node 1 sends a copy of packet to all neighbours (2, 3 and 4). Packet contains sequence number 1.

Step 2



Nodes 2, 3 and 4 send copy to their neighbours, except they don't send back to Node 1. Also, they only send 1 copy on each link.

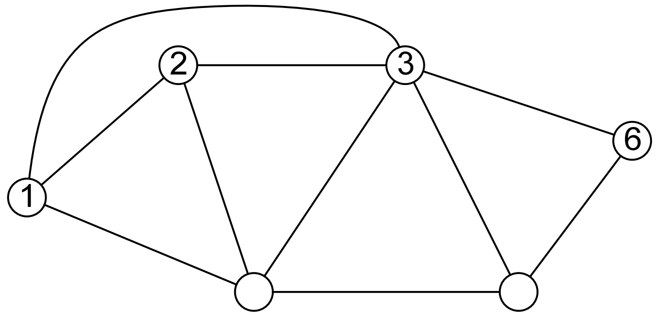
Step 3



Node 6 is the destination, so does not forward the packet. Nodes 2, 3, and 4 have already sent the packet. Node 5 sends the packet to 6 (because 5 does not know 6 has already received packet). Node 6 will discard this packet.

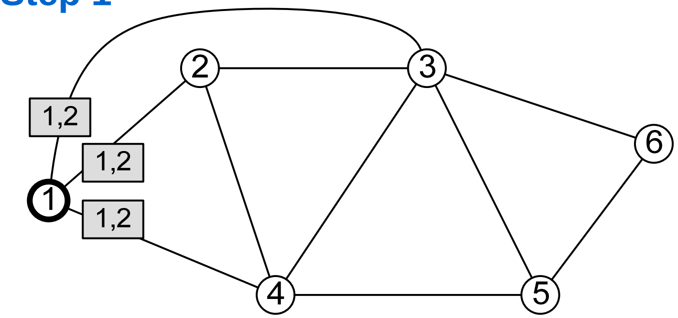
Flooding Example with Hop Count

Network topology



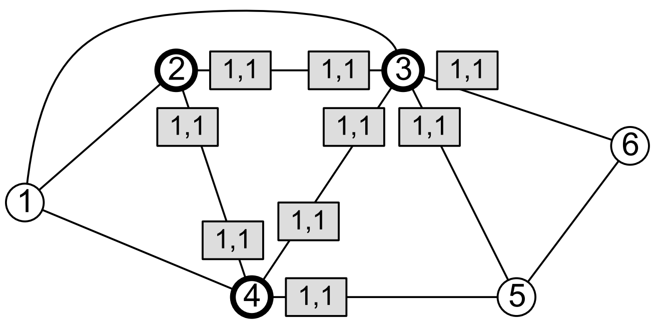
Same as previous example, except a hop count is included in the packet. Initial value is 2.

Step 1



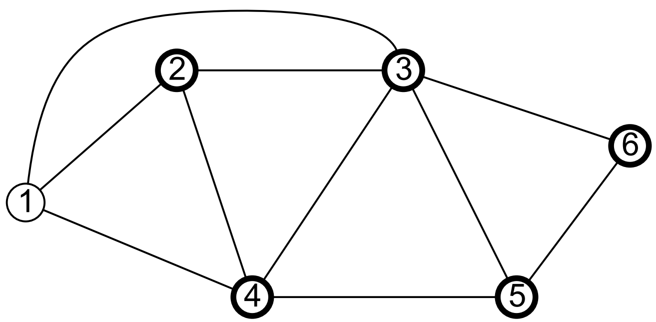
Node 1 sends a copy of packet to all neighbours (2, 3 and 4). Packet contains sequence number 1 and hop count of 2.

Step 2



After receiving the packet, nodes 2, 3 and 4 decrement the hop count to 1, check it is not 0, and send to neighbours.

Step 3



Node 6 receives the packet (it’s the destination). Nodes 2, 3 and 4 will not forward (already sent). Node 5 will received, decrement the hop count, and since it is 0, discard the packet.

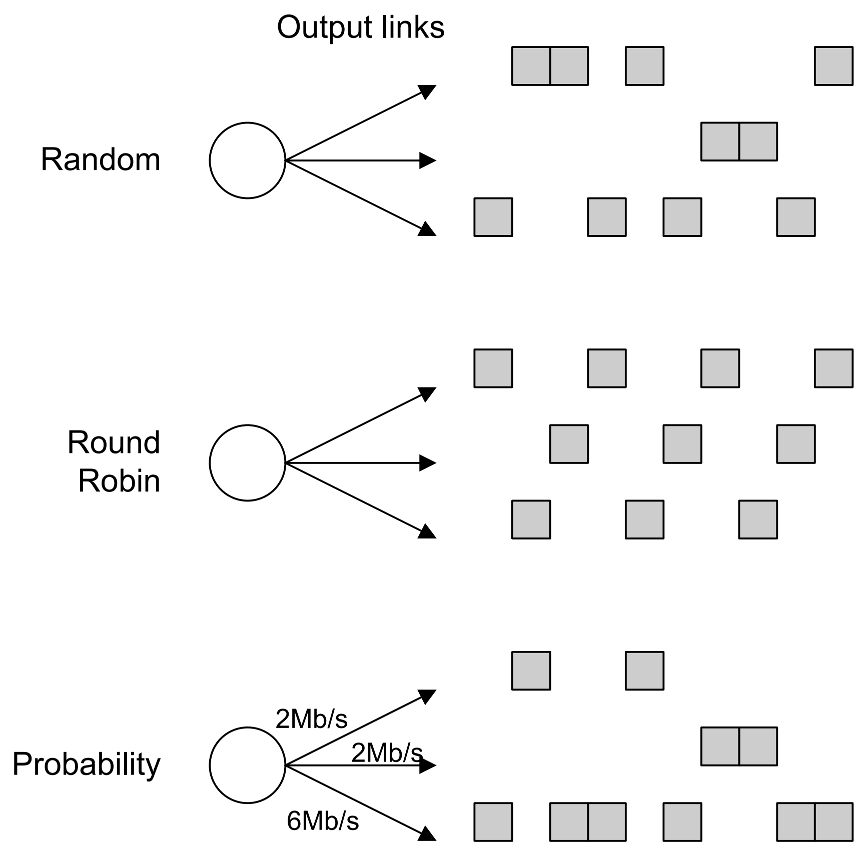
Characteristics of Flooding

- With flooding, all possible routes are tried
  - (Assuming hop count limit is not used and no failures)
  - At least one packet will take the minimum hop route from source to destination
    - Example: can be used to setup a virtual circuit. Source sends connect request using flooding, and the switches that forward the connect request record their addresses in the connect request. The destination sends a connect response back along the shortest (minimum hop) path
  - All nodes are visited
    - Useful for distributing topology information. Example: Switch 1 sends information about its links using flooding. All nodes in the network will eventually learn about the links of Switch 1.
- Flooding is very inefficient
  - Need to send many copies of a packet to get 1 packet from source to destination
    - Example with hop count: 12 packets transmitted in network
    - The best case using minimum hop path is 2 packets (plus any packets needed to determine this path)
  - When using hop count limit, the packet may not reach destination!

Strategy 3: Selective Flooding

- A specific case of flooding, with constraints on which neighbours to send to
- (Normal) Flooding: send to all neighbours
- Selective Flooding: send to a selection of neighbours
- How do you select the neighbours?
  - Random flooding (or random routing): randomly select 1 or more output links to send to
  - Round Robin: select link 1, then next time link 2, then next time link 3, and so on until all links selected, then return to link 1
  - Probability (based on metric): e.g. if link 1 is 1Mb/s and link 2 is 2Mb/s, select link 1 with probability 1/3 and select link 2 with probability 2/3
- Characteristics of Selective Flooding
  - Similar to flooding, there are no overheads in distributing updates
  - More efficient than normal flooding
  - But a random route is typically not least cost
    - E.g. you may choose a route that has many hops
    - Therefore network carries higher than optimal load (but not nearly as high as flooding)

Selective Flooding Examples



(Selective) Flooding Summary

- When is a decision made for a route?
  - For each packet sent
- Which node chooses the route?
  - Distributed (although a route isn't really chosen)
- Where does the network information come from?
  - No network information needed
  - For probabilistic selective flooding, local information is typically used
- When is the network information updated?
  - Never (continuous for probabilistic selective flooding)

- In practice, used for distributing topology information and other network management information to nodes in network (e.g. broadcast to everyone)
  - Used by routing protocols to learn about other nodes in the network

Strategy 4: Adaptive Routing

- Use a least-cost routing algorithm to determine a route, and adapt the route as network conditions change
  - At one point in time, route X may be the least-cost route from A to B, and then some time later, route Y may be the least-cost route from A to B
  - Adaptive routing is used by almost all packet switching networks, e.g. the Internet
- Adaptive routing requires information about network
  - E.g. regular updates on the current amount of traffic and delay of links and routers
  - Where does the information come from?
    - Option 1: Local to node
      - Route to output link that has shortest queue. This is rarely used in practice
    - Option 2: Adjacent Nodes
      - Adjacent nodes provide information about their delay/link status; node uses a Least-Cost Routing algorithm to determine the best route
    - Option 3: All Nodes
      - Like Option 2, but now collect information from all nodes in network

Adaptive Routing

- Advantages:
  - Improved performance
    - Potentially can select the most suited paths
  - Can help with congestion control, because tends to balance amount of traffic across the network
- Disadvantages:
  - Decisions more complex (complex algorithms needed to select the best path)
  - Tradeoff between quality of network information and overhead
    - The more information required for routing decisions, and the more often updates are delivered, then the more overhead in the network
    - Reacting too quickly can cause oscillation
      - First select path A for packet 1, then path B packet 2, then path A, and so on: varying delays (jitter) and bandwidth; varying loads
    - Reacting too slowly means information may be irrelevant
      - Select path A based on outdated information, when in fact path B is better

Routing Protocols and Algorithms

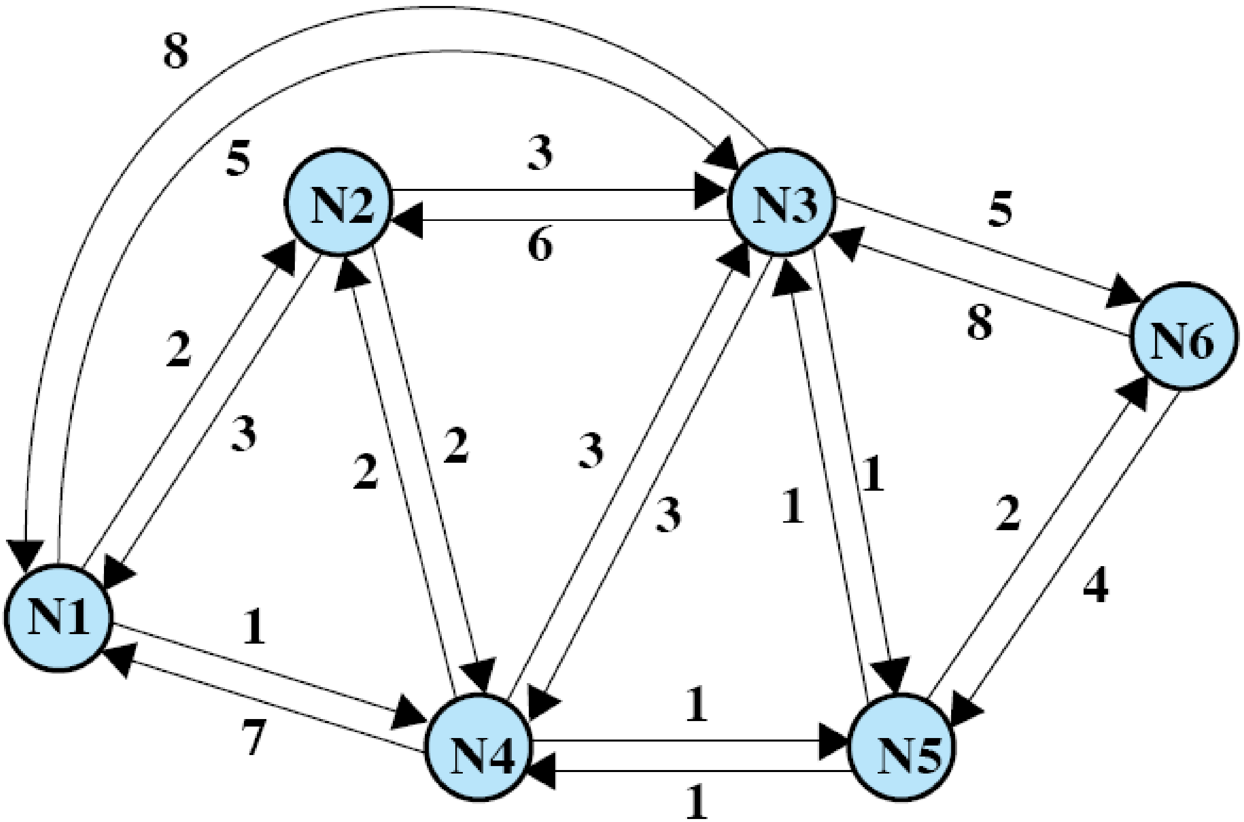
Routing Protocols

- A routing protocol is used by nodes to automatically determine the routes in the network
- A routing protocol specifies:
  - Routing algorithm for determining least-cost routes: e.g. Dijkstra, Bellman-Ford or variants
  - Routing information to be exchanged between nodes
    - May depend on routing algorithm
  - Formats of messages used to deliver routing information
  - Rules as to when to send routing messages and what to do upon receiving them
  - Metrics to be used in routing algorithm (hop count, bandwidth, …)
  - Optionally, default values of specific parameters may be given
    - E.g. time between updates
- Real routing protocols include: OSPF, RIP, BGP, IGRP, EIGRP, PNNI, IS-IS, DSDV, AODV, …
  - You will be introduced to these in later courses

An Example: Link State Routing

- Lets consider a simple routing protocol that uses Dijkstra's algorithm to determine the least-cost routes
  - This is an example of a *link state routing* protocol
    - An alternative type of routing protocol is *distance vector* which uses algorithms like Bellman Ford for determining least-cost routes
    - You may cover details of different routing protocols and algorithms in other courses
- The aim of a link state routing protocol:
  - Each node learns the topology of the network, then calculates the least-cost route from itself to every other node using (for example) Dijkstra's algorithm
- The actions taken at each node in a link state routing protocol are:
  1. Record the state of its own links (e.g. source/destination, metric)
  2. Send the state of its own links to every other node using flooding
  3. Form a shortest path tree from itself to every other node
     - Dijkstra's algorithm is a common approach for calculating this tree
  4. Build a routing table based on the shortest path tree

Link State Routing: Example Network

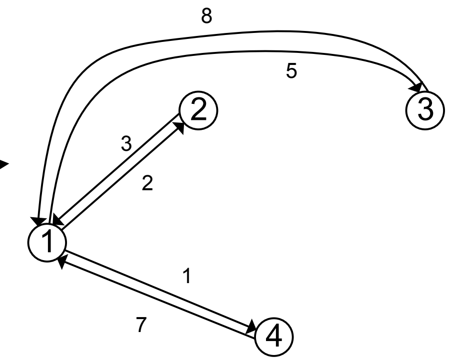


Link State Routing: Step 1

- Each node records the state of its own links
  - As this state is going to be sent to other nodes, we will call it a *Link State Packet* (LSP)
- A Link State Packet typically contains:
  - The identity of the current node
  - List of links that the current node has, including their costs
  - A sequence number (used by the flooding protocol)
  - A hop count or age (used by the flooding protocol)
- Remember, that this information changes over time
  - Link costs change (e.g. queue delay)
  - Links fail or new links are created
  - Nodes join and leave the network

N1	
To N2	2
To N3	5
To N4	1
From N2	3
From N3	8
From N4	7
Seq = 0	
Hop Count = 5	

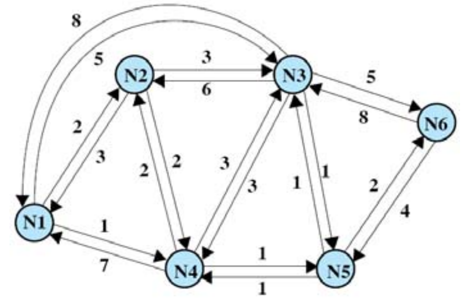
Upon network startup, N1 only knows about its own links



Link State Routing: Step 2

- Each node sends its Link State Packet to all other nodes
  - Flooding is used
- The result of each node flooding its LSP is:
  - Each node knows the current state of links at every other node
    - E.g. N1 will eventually receive LSPs from N2, N3, N4, N5 and N6
  - That is, each node knows the current network topology
- However, since the LSPs may change over time, a node will send a new LSP depending on its update strategy
  - A common approach is to send a new LSP when the topology changes
  - Also, periodic updates are possible: e.g. a new LSP is sent every 1 to 2 hours

After Step 2, N1 knows about the entire network topology



Link State Routing: Step 3

- From the topology, determine the least-cost paths from one node to every other node
  - Dijkstra's algorithm finds the shortest path tree for a graph of links and nodes
    - The source node (the current node performing the calculation) is the root of the tree
    - Shortest path tree is a tree in which the path between the root and every other node is the shortest
  - Each node would use Dijkstra's algorithm to find the least-cost paths to all other nodes
    - In the following slides we give an informal explanation of Dijkstra's algorithm, and then explain through an example of Node N1 applying the algorithm

Dijkstra's Algorithm

- Finds shortest paths from given source node *s* to all other nodes by developing paths in order of increasing path length
  - Algorithm runs in stages
    - Each time adding a node with next shortest path
  - Algorithm terminates when all nodes processed by algorithm (in set *T*)
- Define:
  - *N* = set of nodes in the network
  - *s* = source node
  - *T* = set of nodes so far incorporated by the algorithm
  - $w(i, j)$ = cost from node *i* to node *j*
  - $L(n)$ = cost of least-cost path from node *s* to node *n*
  - $P(i, j)$ = path from node *i* to node *j*

Dijkstra's Algorithm

- Informal explanation of Dijkstra's algorithm applied to finding least-cost routes
  - Step 1 [Initialization]
    - $T = \{s\}$, set of nodes so far incorporated
    - $L(n) = w(s, n)$ for $n \neq s$; initial path costs to neighboring nodes are simply link costs; cost is ∞ if link does not exist. P(s,n) = s – n.
  - Step 2 [Get Next Node]
    - Find neighboring node *x* not in *T* with least-cost path from *s*
      - If two or more nodes have the same least-cost, then randomly choose one of them
    - Incorporate node *x* into *T*
  - Step 3 [Update Least-Cost Paths]
    - $L(n) = \min[L(n), L(x) + w(x, n)]$ for all $n \notin T$
    - If latter term is minimum, path from *s* to *n* is path from *s* to *x* concatenated with edge from *x* to *n*. That is: P(s,n) = P(s,x) + P(x,n)
  - Steps 2 and 3 are repeated until $T = N$

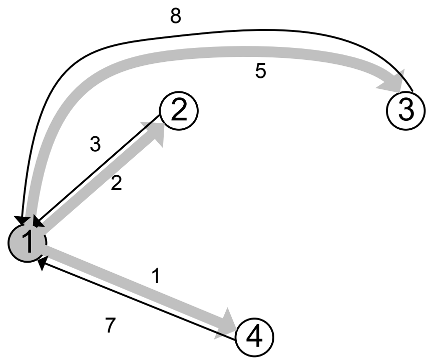
Dijkstra's Algorithm Example

- Step 1 [Initialisation]
  - T = {N1}
  - Initial least costs and paths:
    - L(N2) = 2; P(N1,N2) = N1-N2
    - L(N3) = 5; P(N1,N3) = N1-N3
    - L(N4) = 1; P(N1,N4) = N1-N4
    - L(N5) = ∞; P(N1,N5) = -
    - L(N6) = ∞; P(N1,N6) = -
- Step 2 [Get Next Node]
  - The neighbour of N1 which is not in T with least cost path is N4
    - x = N4 since L(N4) = 1
    - T = {N1, N4}
  - So now we also consider the links from/to N4
    - w(N4, N1) = 7
    - w(N4, N2) = 2
    - w(N4, N3) = 3
    - w(N4, N5) = 1
    - …
- Step 3 [Update Least-Cost Paths]
  - Determine the new L():
    - L(N2) = min[2, L(N4)+w(N4,N2)]
      = min[2, 1 + 2]
      = 2 (no change)
    - L(N3) = min[5, 1 + w(N4, N3)]
      = min[5, 1 + 3]
      = 4 (changed)
      P(N1,N3) = P(N1,N4) + P(N4,N3)
      = N1-N4-N3
    - L(N5) = min[∞, 1 + w(N4,N5)]
      = min[∞, 1+1]
      = 2 (changed)
      P(N1,N5) = P(N1,N4) + P(N4,N5)
      = N1-N4-N5
    - L(N6) = min[∞, 1 + w(N4,N6)]
      = ∞ (no change)
- Steps 2 and 3 are repeated for Iterations 3, 4, 5 and 6
  - Results shown in next slides

Dijkstra’s Algorithm Example

After Initialisation (Step 1):

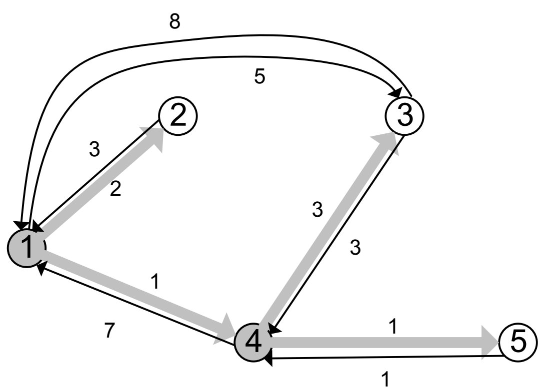
Iter.	T	L(N2)	Path	L(N3)	Path	L(N4)	Path	L(N5)	Path	L(N6)	Path
1	{1}	2	1-2	5	1-3	1	1-4	∞	-	∞	-



Dijkstra’s Algorithm Example

After iteration 2 (Steps 2 and 3):

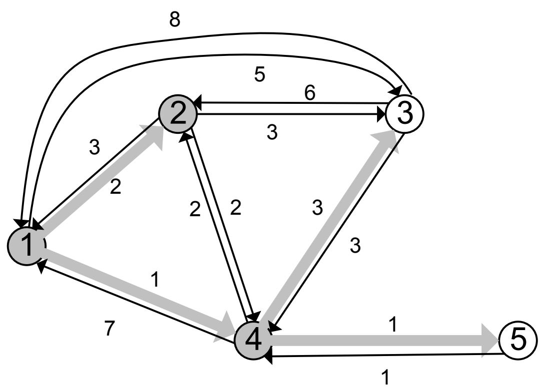
Iter.	T	L(N2)	Path	L(N3)	Path	L(N4)	Path	L(N5)	Path	L(N6)	Path
1	{1}	2	1-2	5	1-3	**1**	1-4	∞	-	∞	-
2	{1,4}	2	1-2	4	1-4-3	1	1-4	2	1-4-5	∞	-



Dijkstra's Algorithm Example

After iteration 3 (Steps 2 and 3 repeated):

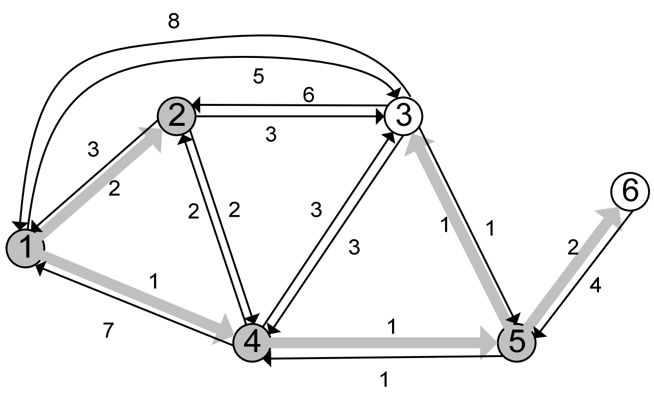
Iter.	T	L(N2)	Path	L(N3)	Path	L(N4)	Path	L(N5)	Path	L(N6)	Path
1	{1}	2	1-2	5	1-3	1	1-4	∞	-	∞	-
2	{1,4}	**2**	1-2	4	1-4-3	1	1-4	2	1-4-5	∞	-
3	{1, 2, 4}	2	1-2	4	1-4-3	1	1-4	2	1-4-5	∞	-



Dijkstra’s Algorithm Example

After iteration 4 (Steps 2 and 3 repeated): :

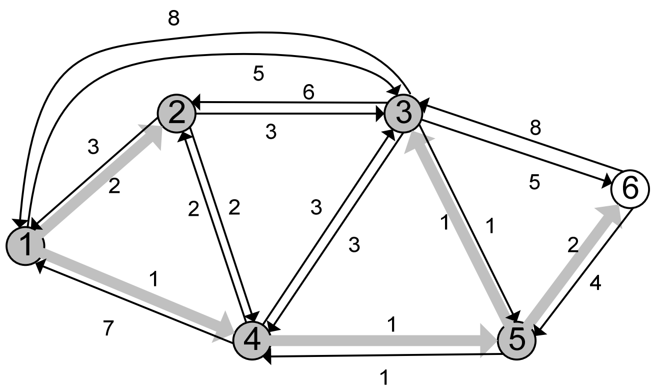
Iter.	**T**	**L(N2)**	**Path**	**L(N3)**	**Path**	**L(N4)**	**Path**	**L(N5)**	**Path**	**L(N6)**	**Path**
1	{1}	2	1-2	5	1-3	1	1-4	∞	-	∞	-
2	{1,4}	2	1-2	4	1-4-3	1	1-4	2	1-4-5	∞	-
3	{1, 2, 4}	2	1-2	4	1-4-3	1	1-4	**2**	1-4-5	∞	-
4	{1, 2, 4, 5}	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6



Dijkstra's Algorithm Example

After iteration 5 (Steps 2 and 3 repeated): :

Iter.	T	L(N2)	Path	L(N3)	Path	L(N4)	Path	L(N5)	Path	L(N6)	Path
1	{1}	2	1-2	5	1-3	1	1-4	∞	-	∞	-
2	{1,4}	2	1-2	4	1-4-3	1	1-4	2	1-4-5	∞	-
3	{1, 2, 4}	2	1-2	4	1-4-3	1	1-4	2	1-4-5	∞	-
4	{1, 2, 4, 5}	2	1-2	**3**	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6
5	{1, 2, 3, 4, 5}	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6

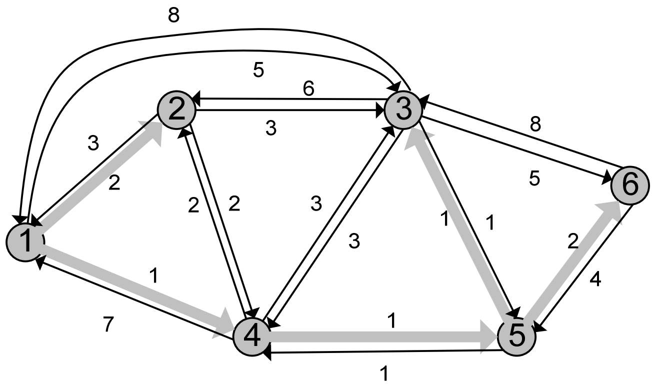


Dijkstra's Algorithm Example

After iteration 6 (Steps 2 and 3 repeated): :

Iter.	T	L(N2)	Path	L(N3)	Path	L(N4)	Path	L(N5)	Path	L(N6)	Path
1	{1}	2	1-2	5	1-3	1	1-4	∞	-	∞	-
2	{1,4}	2	1-2	4	1-4-3	1	1-4	2	1-4-5	∞	-
3	{1, 2, 4}	2	1-2	4	1-4-3	1	1-4	2	1-4-5	∞	-
4	{1, 2, 4, 5}	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6
5	{1, 2, 3, 4, 5}	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6
6	{1, 2, 3, 4, 5, 6}	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6

T now contains all nodes (that is T = N), hence the algorithm stops. We now have the least cost paths from N1 to all other nodes.



Link State Routing: Step 4

- Each node builds its routing table based on the shortest path tree
  - Remember, no need to store the entire path, only the next node
  - Example: routing table for Node 1

Dest	Next	Cost
N2	N2	2
N3	N4	3
N4	N4	1
N5	N4	2

(The cost is not necessary, but often included)

  - Nodes N2, N3, N4, N5 and N6 would also create their routing tables (from the results of applying Dijkstra's algorithm)

- As new LSPs are received (due to topology changes), Steps 3 and 4 are repeated so that the "best" routes are always used

Summary – Concepts

- Communication networks are formed by connecting devices across multiple links
- Switching is the method of delivering data between source and destination across multiple links
  - Stations or end-user devices act as sources and destinations of data
  - Switches connect the links and forward data between source and destination
- Circuit and Packet Switching techniques determine how to deliver data across one or more paths between source and destination
- Routing determines what path to take between source and destination
  - The aim is to find the "best" path
- There are different routing metrics, strategies, algorithms and protocols available

Summary – Practice

- Circuit switching was developed for traditional telephone networks and is still used today in those (and other) networks
- Packet switching was developed to be more efficient for delivering computer generated data over networks
- Packet switching is the concept used in the Internet and in almost all new Wide Area Networks (WANs) today

- Adaptive routing strategies are needed for almost all WANs
  - In smaller networks, fixed or flooding strategies are viable
- Dijkstra and Bellman Ford are two of the most common algorithms for determining the shortest path between source and destination
  - They, and some variants, are implemented by routing protocols used in the Internet today
  - The trade-offs between the different routing protocols often depend on the size of networks, the amount of traffic and the rate at which the network changes