Routing in Switched Networks

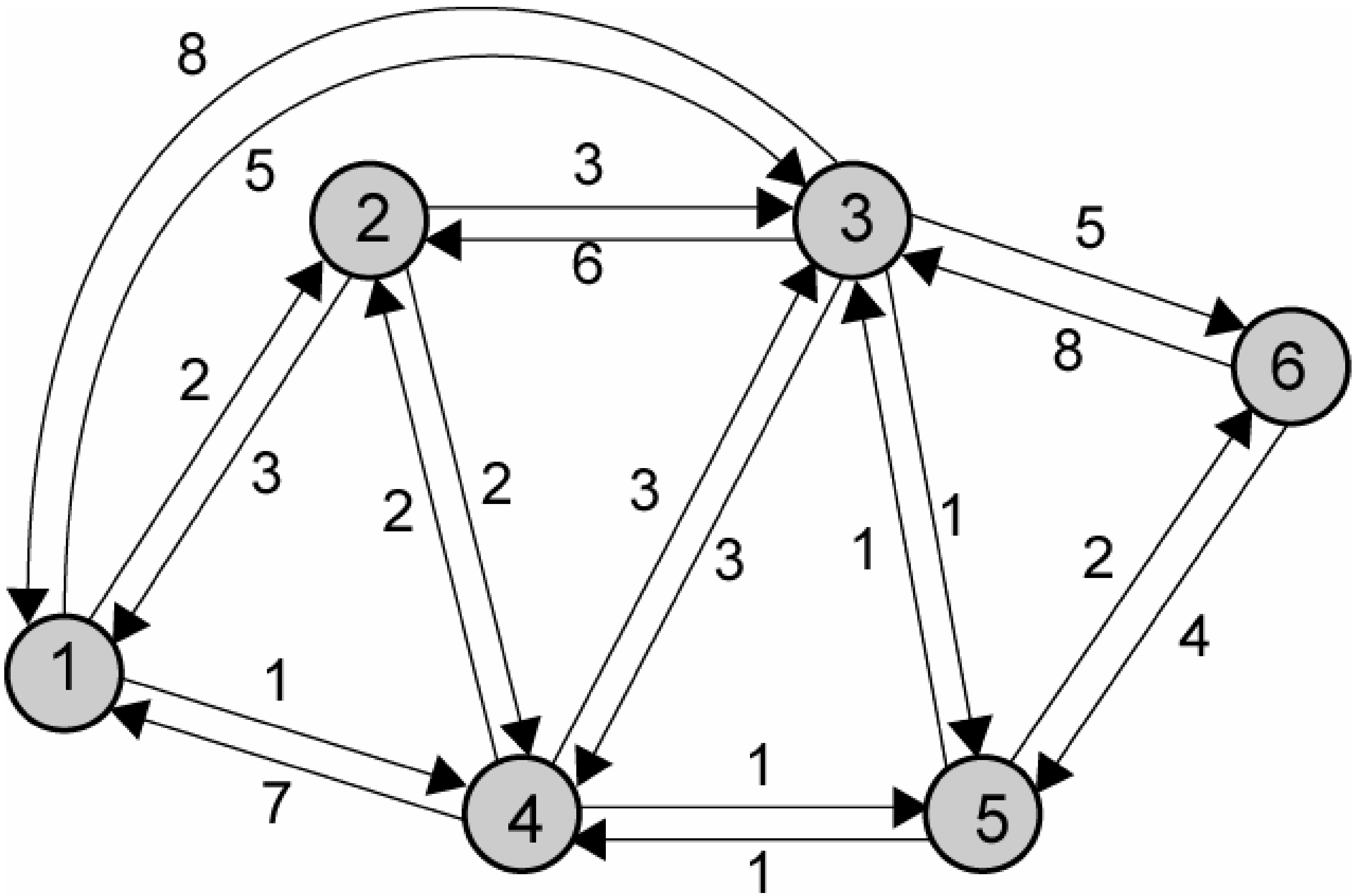
Raj Jain
Washington University
Saint Louis, MO 63131
Jain@cse.wustl.edu

These slides are available on-line at:

http://www.cse.wustl.edu/~jain/cse473-05/

- Routing algorithms
  - Dijkstra's Algorithm
  - Bellman-Ford Algorithm
- ARPAnet routing

Routing



Routing Techniques Elements

- **Performance criterion**: *Hops*, Distance, *Speed*, Delay, Cost
- **Decision time**: *Packet*, session
- **Decision place**: *Distributed*, centralized, Source
- **Network information source**: None, local, *adjacent nodes*, nodes along route, all nodes
- **Routing strategy**: Fixed, *adaptive*, random, flooding
- **Adaptive routing update time**: Continuous, *periodic*, *topology change*, major load change

Random Routing

- Node selects one outgoing path for retransmission of incoming packet
- Selection can be random or round robin
- No network info needed
- Route is typically not least cost nor minimum hop

Fixed Routing Tables

From Node / To Node

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	—

Node 1

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

Node 2

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

Node 3

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

Node 4

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

Node 5

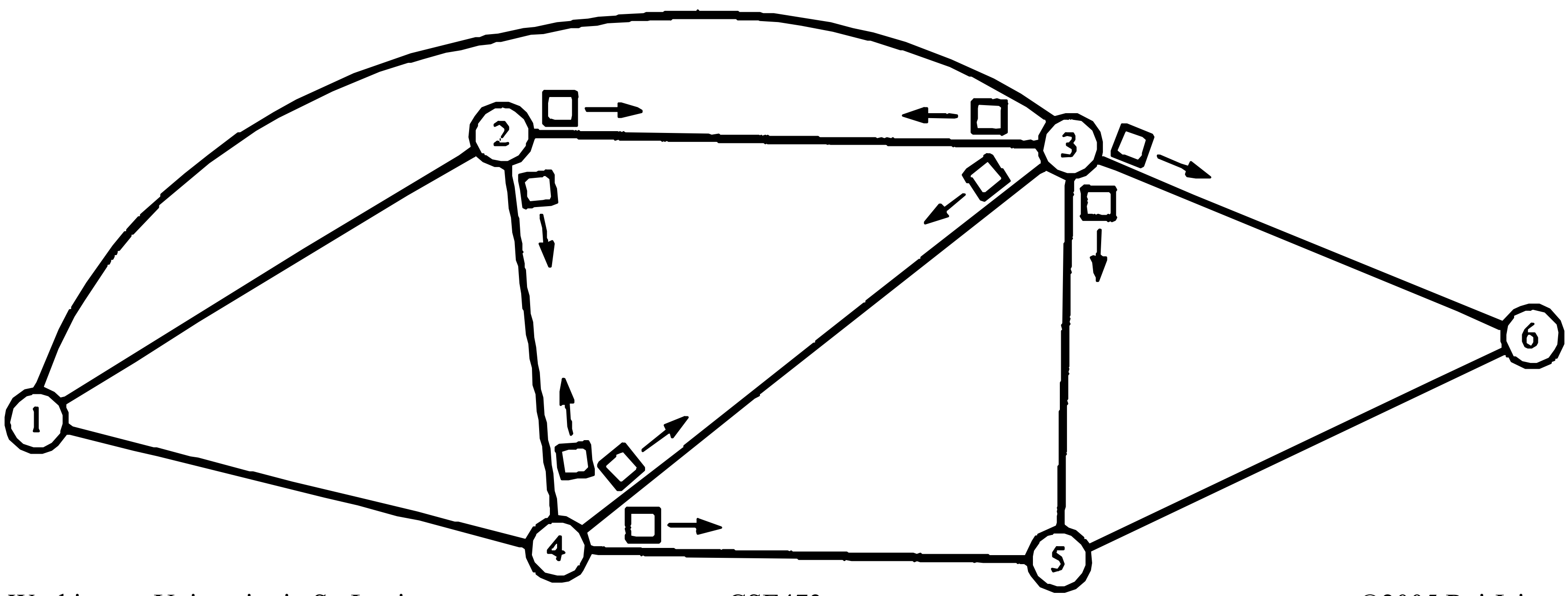
Destination	Next Node
1	4
2	4
3	3
4	4
6	6

Node 6

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

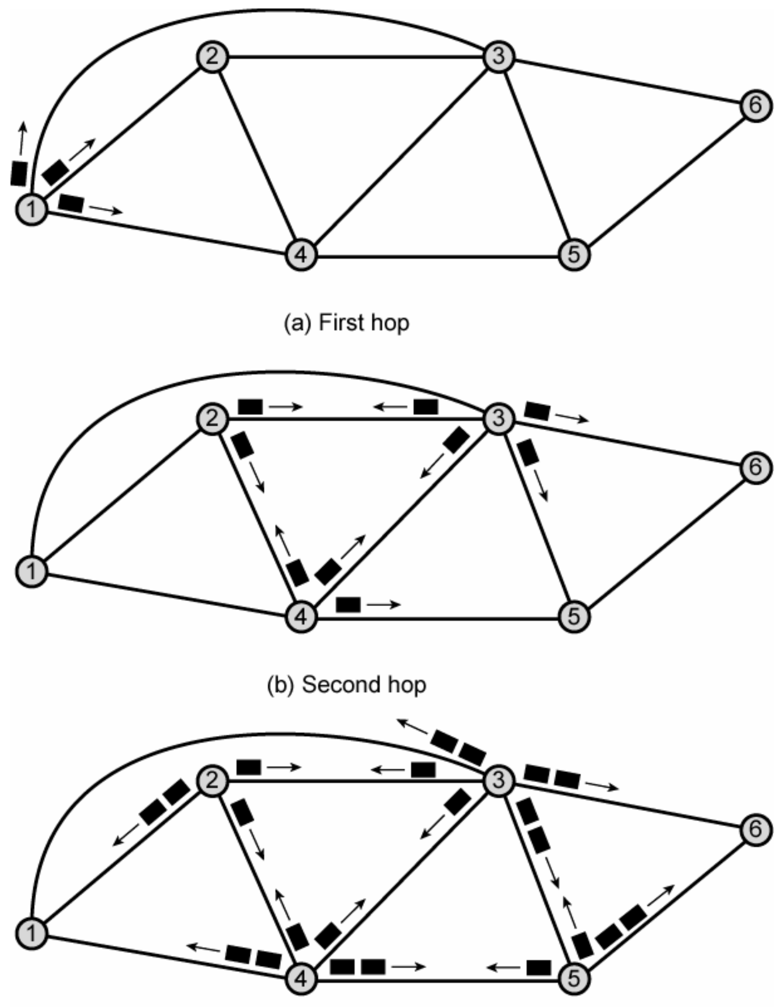
Flooding

- Packet sent by node to every neighbor
- Incoming packets retransmitted on every link except incoming link
- Each packet is uniquely numbered so duplicates can be discarded



Flooding

- Uses all possible paths
- Uses minimum hop path Used for source routing



(a) First hop

(b) Second hop

(c) Third hop

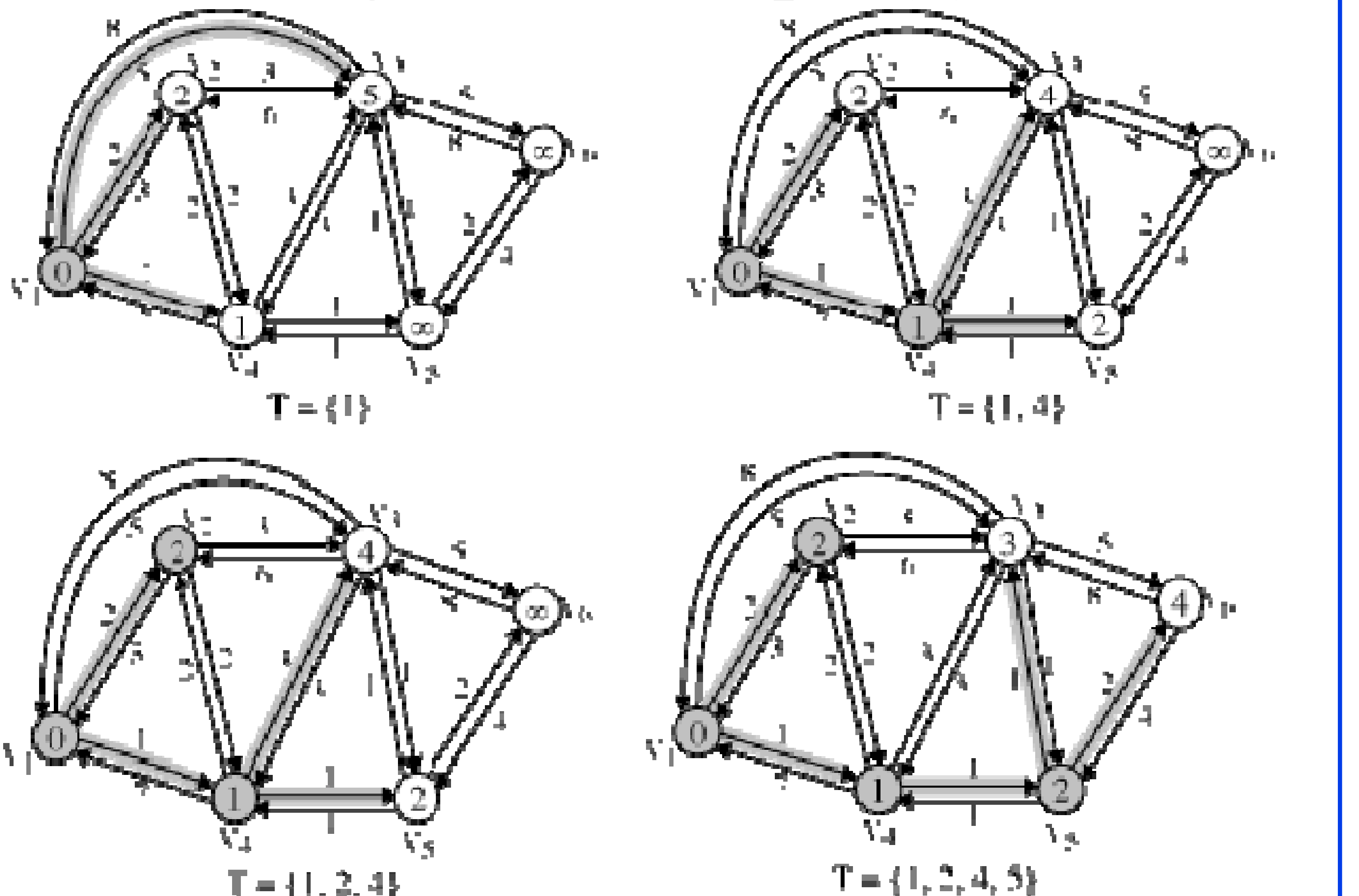
Adaptive: Distance Vector vs Link State

- **Distance Vector**: Each router sends a vector of distances to its neighbors. The vector contains distances to all nodes in the network.
  Older method. Count to infinity problem.
- **Link State**: Each router sends a vector of distances to all nodes. The vector contains only distances to neighbors. Newer method. Used currently in internet.

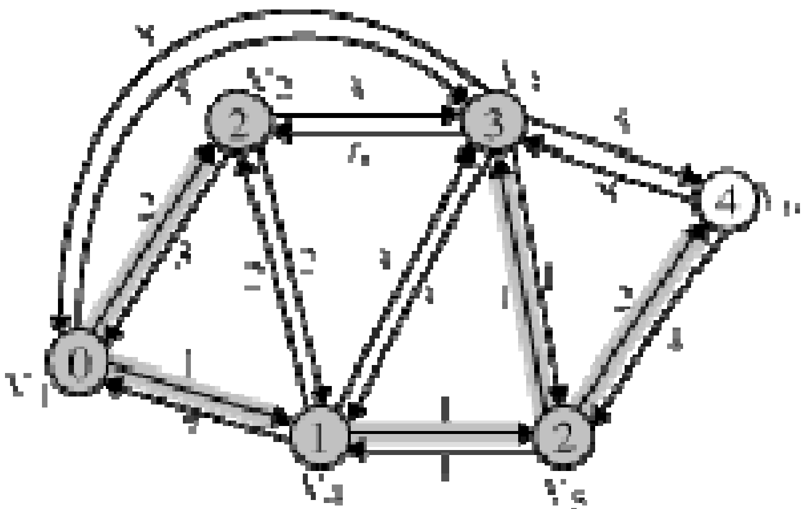
Dijkstra's Algorithm

- Goal: Find the least cost paths from a given node to all other nodes in the network
- Notation:
  w(i,j) = Link cost from i to j if i and j are connected
  L(n) = Total path cost from s to n
  T = Set of nodes so far for which the least cost path is known
- Method:
  - Initialize: $T=\{s\}$, $L(n) = w(s,n)$ for $n \neq s$
  - Find node $x \notin T$, whose L(x) is minimum
  - Update $L(n) = \min[L(n), L(x) + w(x,n)]$ for all $n \notin T$

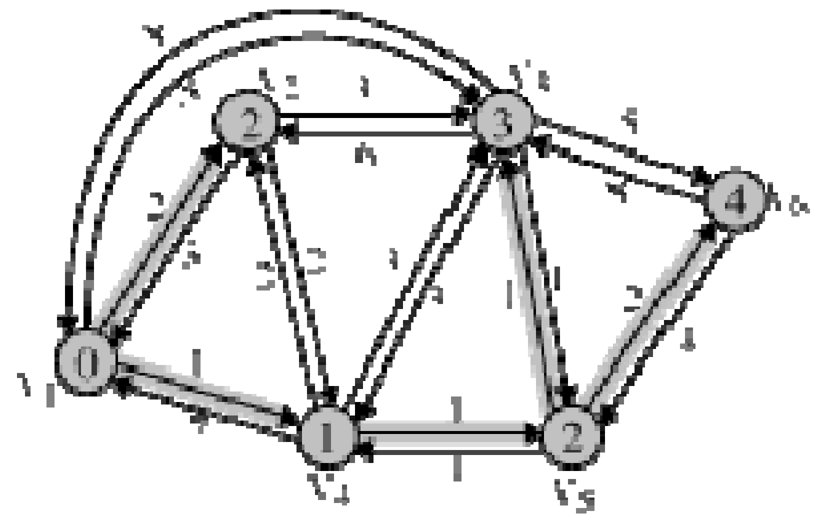
Dijkstra Example (1)



Dijkstra Example (2)



T = {1, 2, 3, 4, 5}



T = {1, 2, 3, 4, 5, 6}

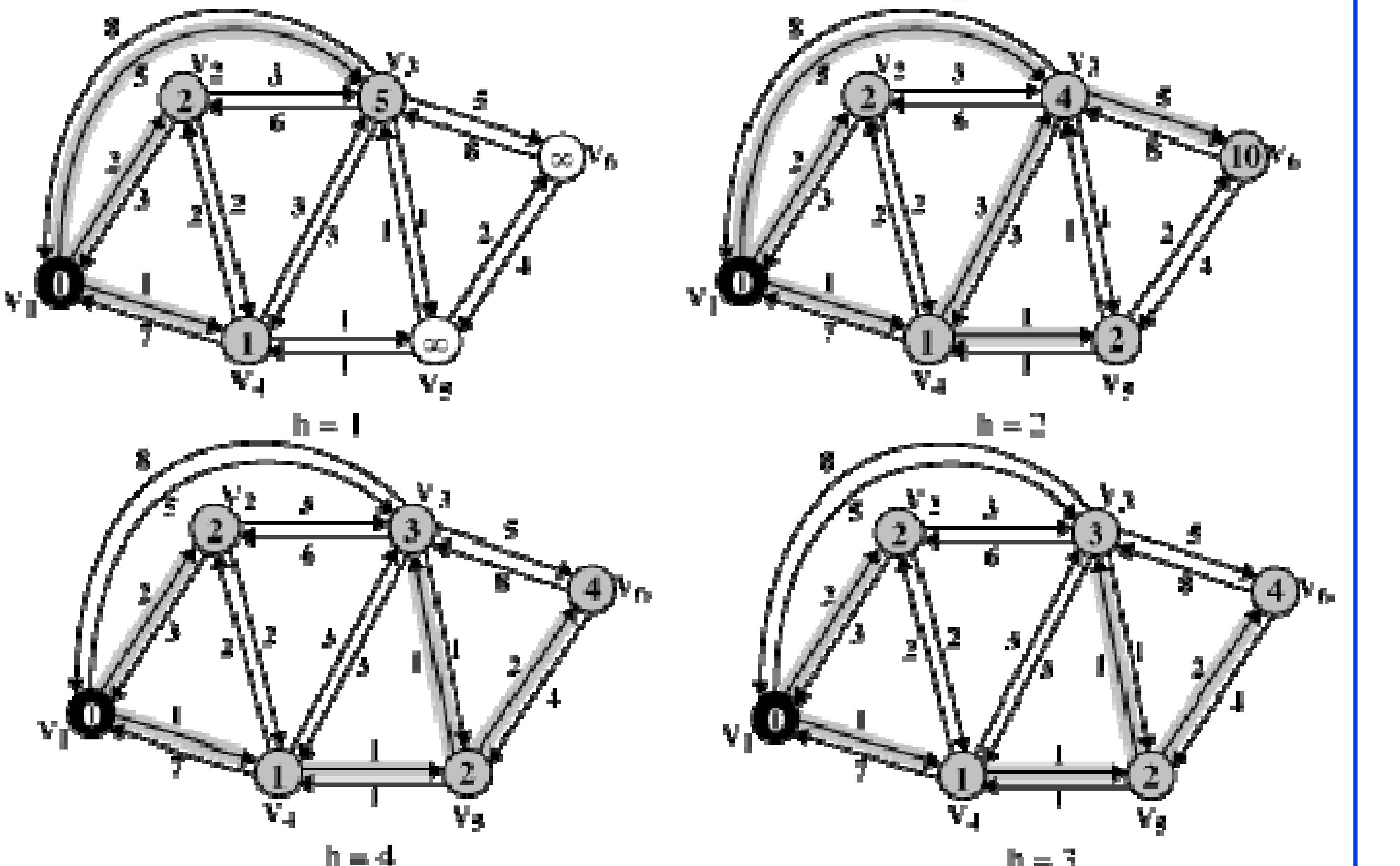
Dijkstra Example (3)

	T	L(2)	Path	L(3)	Path	L(4)	Path	L(5)	Path	L(6)	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

Bellman-Ford Algorithm

- Notation:
  s = Source node
  w(i,j) = link cost from i to j
  h = Number of hops being considered
  $L_h(n)$ = Cost of h-hop path from s to n with $\leq$ h hops
- Method: Find all nodes 1 hop away
  Find all nodes 2 hops away
  Find all nodes 3 hops away
  - Initialize: $L_0(n) = \infty$ for all $n \neq s$; $L_h(s) = 0$ for all h
  - Find jth node for which h+1 hops cost is minimum
    $L_{h+1}(n) = \min_j [L_h(j) + w(j,n)]$

Bellman-Ford Example



Bellman-Ford Example (Cont)

h	$D(h_2)$	Path	$D(h_3)$	Path	$D(h_4)$	Path	$D(h_5)$	Path	$D(h_6)$	Path
0	∞	-	∞	-	∞	-	∞	-	∞	-
1	2	1-2	5	1-3	1	1-4	∞	-	∞	-
2	2	1-2	4	1-4-3	1	1-4	2	1-4-5	10	1-3-6
3	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6
4	2	1-2	3	1-4-5-3	1	1-4	2	1-4-5	4	1-4-5-6

ARPAnet Routing (1969-78)

- Features: Cost=Queue length,
- Each node sends a vector of costs (to all nodes) to neighbors. Distance vector
- Each node computes new cost vectors based on the new info using Bellman-Ford algorithm

ARPAnet Routing Algorithm

Desti-nation	Delay	Next node
1	0	Ñ
2	2	2
3	5	3
4	1	4
5	6	3
6	8	3
	D^1	S^1

(a) Node 1ås routing table before update

D^2	D^3	D^4
2	3	1
0	3	2
3	0	2
2	2	0
3	1	1
5	3	3

(b) Delay vectors sent t neighbor nodes

Desti-nation	Delay	Next node
1	0	Ñ
2	2	2
3	3	4
4	1	4
5	2	4
6	4	4

$1^{1,2}$ = 2

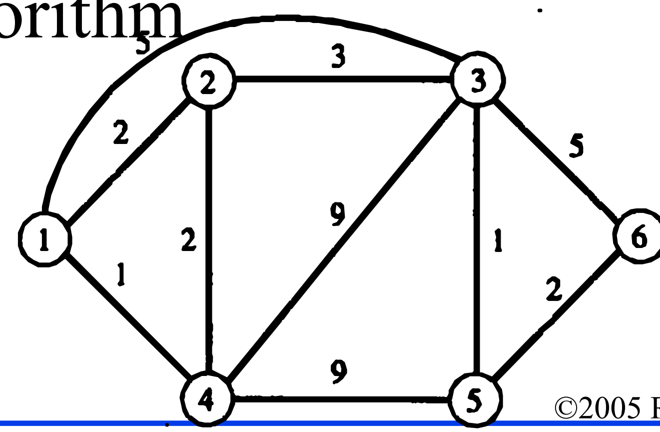
$1^{1,3}$ = 5

$1^{1,4}$ = 1

(c) Node 1ås routing table after update and link c

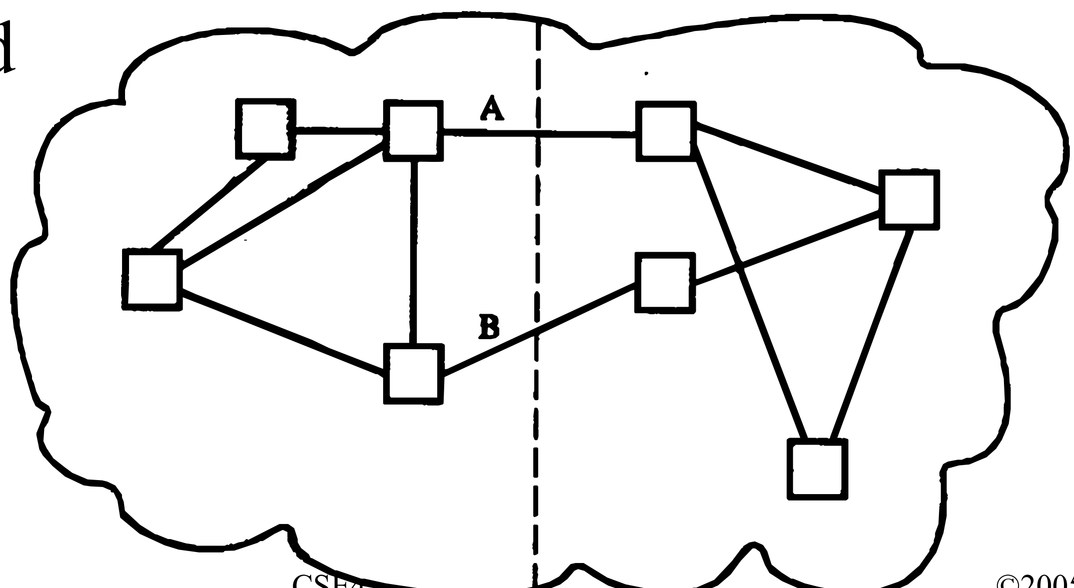
ARPAnet Routing (1979-86)

- Problem with earlier algorithm: Thrashing (packets went to areas of low queue length rather than the destination), Speed not considered
- Solution: Cost=Measured delay over 10 seconds
- Each node floods a vector of cost to neighbors. Link-state. Converges faster after topology changes.
- Each node computes new cost vectors based on the new info using Dijkstra's algorithm



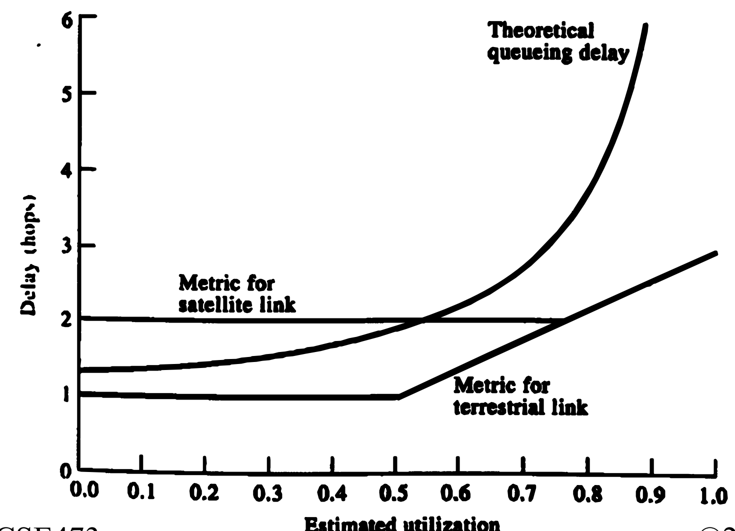
ARPAnet Routing (1987+)

- Problem with 2nd Method: Correlation between delays reported and those experienced later : High in light loads, low during heavy loads
  ⇒ Oscillations under heavy loads
  ⇒ Unused capacity at some links, over-utilization of others, More variance in delay more frequent updates More overhead

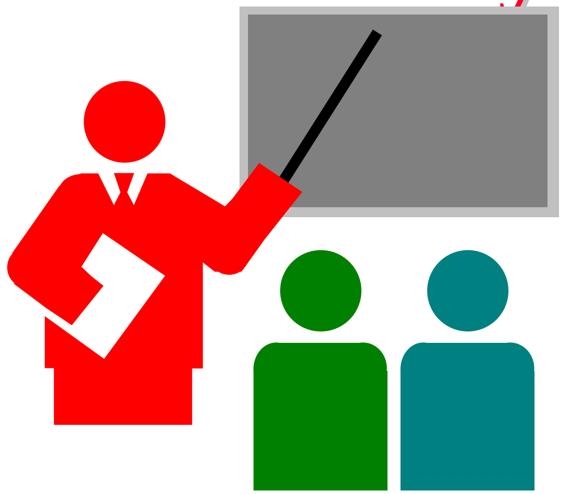


Routing Algorithm

- Delay is averaged over 10 s
- Link utilization = $\rho = 2(T_s - T)/(T_s - 2T)$
  where T=measured delay,
  T_s = service time per packet (600 bit times)
- Exponentially weighted average utilization
  $U(n+1) = \alpha\, U(n) + (1-\alpha)\rho(n+1)$
  $= 0.5\, U(n) + 0.5\, \rho(n+1)$ with $\alpha = 0.5$
- Link cost = fn(U)



Summary



- Distance Vector and Link State
- Routing: Least-cost, Flooding, Random, Fixed
- Dijkstra's and Bellman-Ford algorithms
- ARPAnet

Reading Assignment

- Read Chapter 12 of Stallings' 7th edition and try to answer all review questions.

Homework

Prepare the routing calculation table for node 1 in the following network using (a) Dijkstra's algorithm (b) Bellman Ford Algorithm