EC 553
Communication Networks

Mohamed Khedr

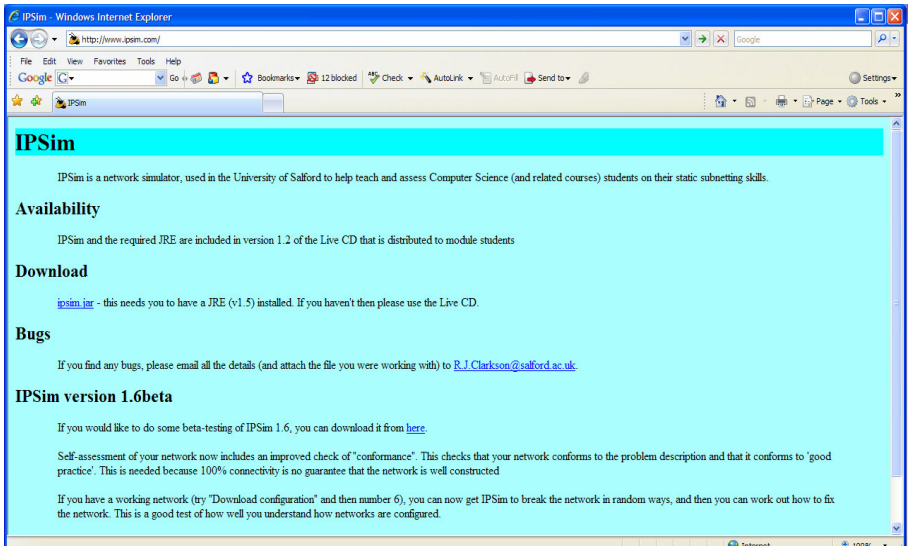
http://webmail.aast.edu/~khedr

Syllabus

- Tentatively

Week 1	Overview
Week 2	Packet Switching
Week 3	IP addressing and subnetting
Week 4	Introduction to Routing concept
Week 5	Routing algorithms
Week 6	Routing protocols
Week 7	Multiple Access I
Week 8	Multiple access II
Week 9	LAN networks
Week 10	Token ring networks
Week 11	VOIP
Week 12	WLAN
Week 13	TCP
Week 14	Congestion control
Week 15	QOS

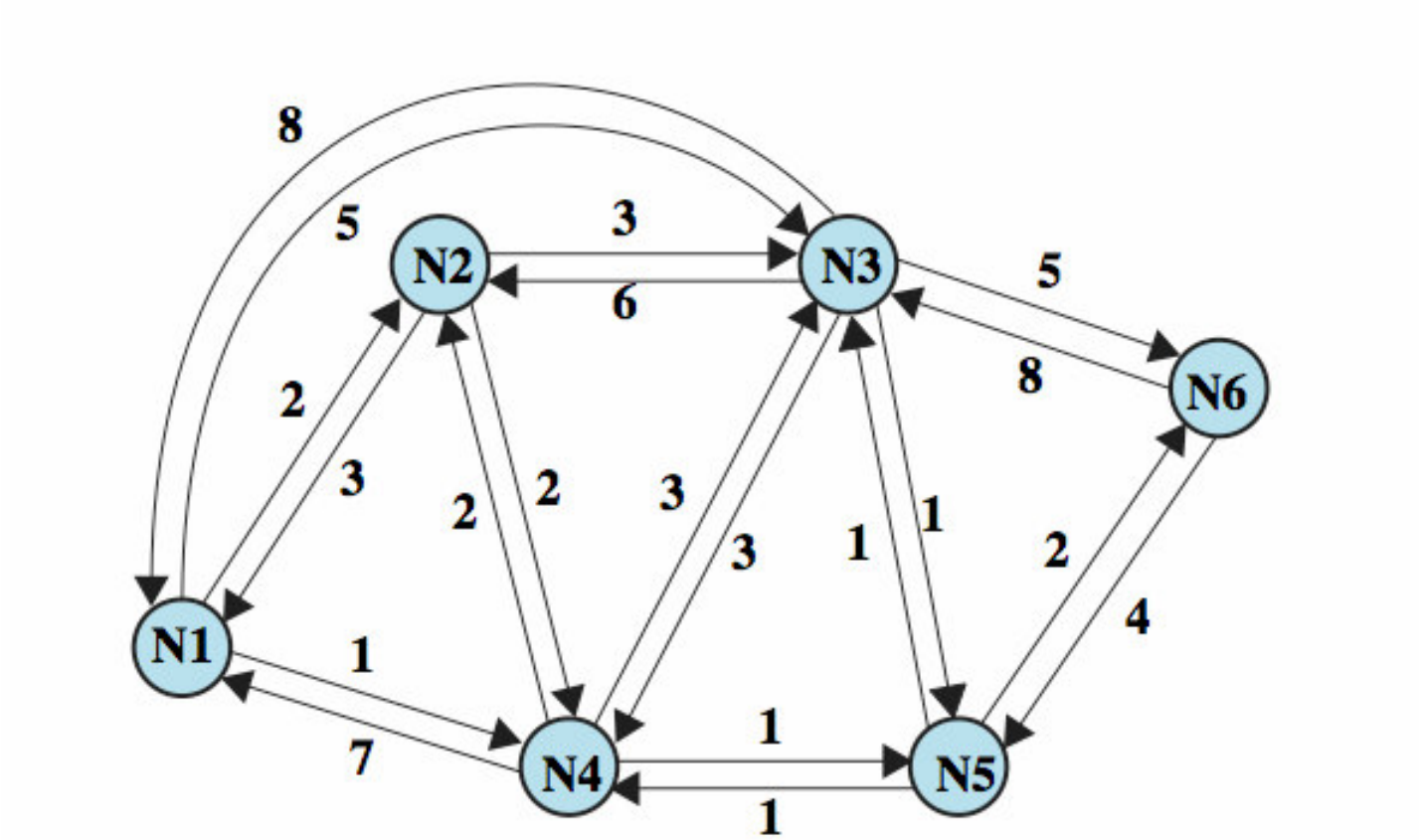
- Please download the following
  - www.ipsim.com
  - http://www.opnet.com/university_program/itguru_academic_edition/



Routing in Packet Switched Network

- key design issue for (packet) switched networks
- select route across network between end nodes
- characteristics required:
  - correctness
  - simplicity
  - robustness
  - stability
  - fairness
  - optimality
  - Efficiency
- Performance criteria
  - used for selection of route
  - simplest is "minimum hop"
  - can be generalized as "least cost

Example Packet Switched Network



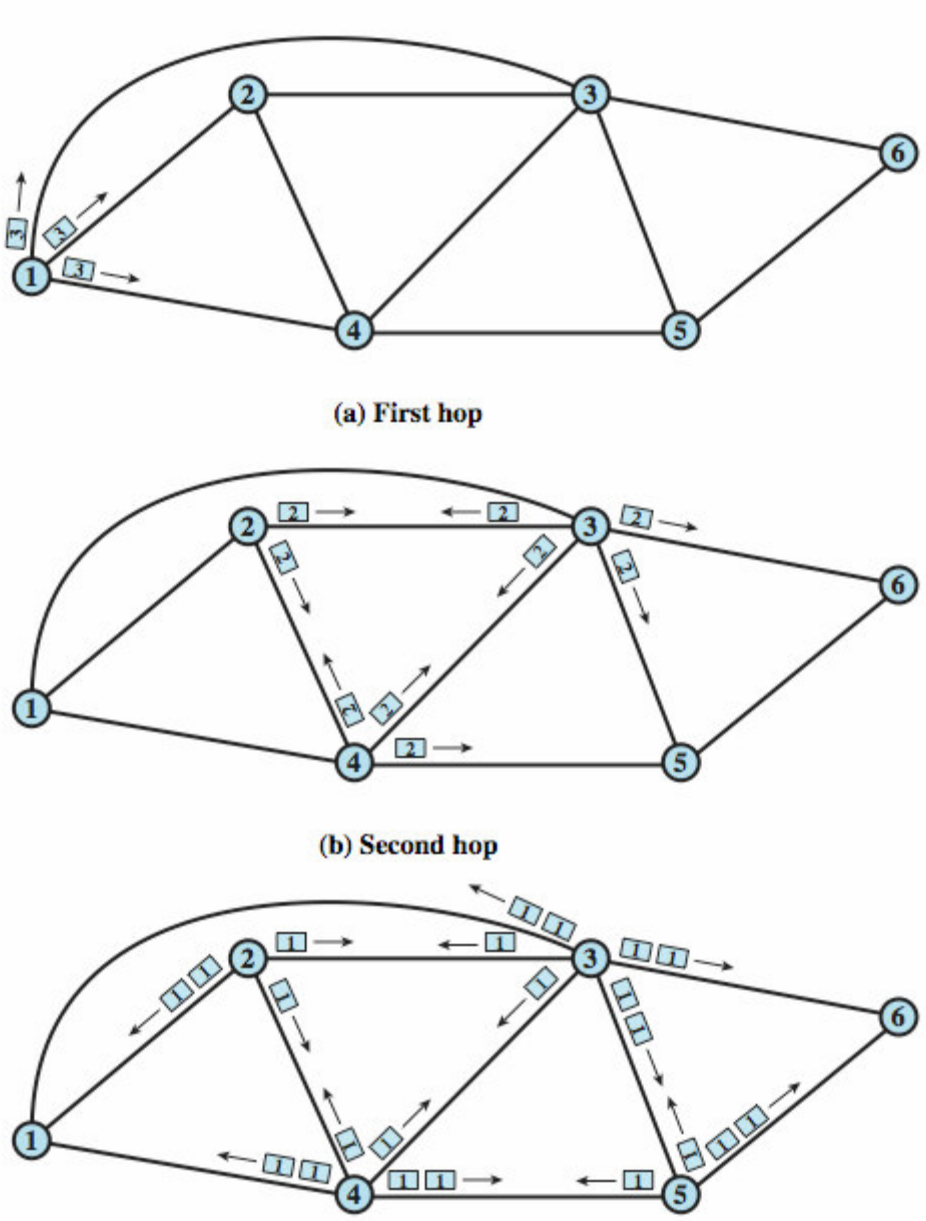
Network Information Source and Update Timing

- routing decisions usually based on knowledge of network (not always)
  - distributed routing
    - using local knowledge, info from adjacent nodes, info from all nodes on a potential route
  - central routing
    - collect info from all nodes
- issue of update timing
  - when is network info held by nodes updated
  - fixed - never updated
  - adaptive - regular updates

Routing Strategies - Flooding

- packet sent by node to every neighbor
- eventually multiple copies arrive at destination
- no network info required
- each packet is uniquely numbered so duplicates can be discarded
- need some way to limit incessant retransmission
  - nodes can remember packets already forwarded to keep network load in bounds
  - or include a hop count in packets

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 hop count route
  - can be used to set up virtual circuit
- all nodes are visited
  - useful to distribute information (eg. routing)
- disadvantage is high traffic load generated

Figure 19.28 Next-hop routing

Routing table for host A

Destination	Route
Host B	R1, R2, Host B

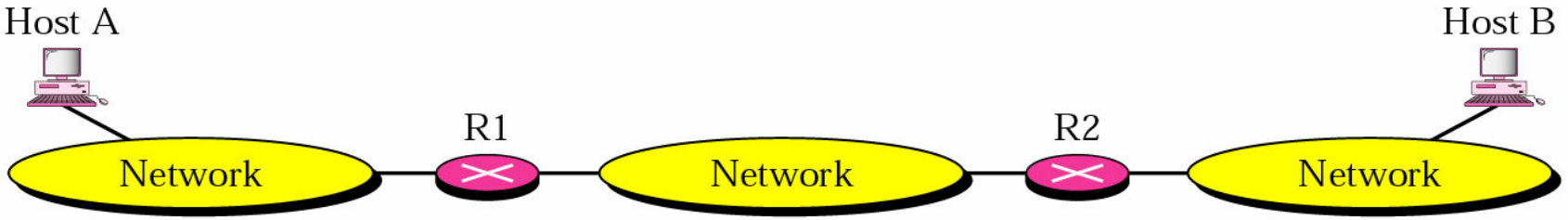
Routing table for R1

Destination	Route
Host B	R2, Host B

Routing table for R2

Destination	Route
Host B	Host B

a. Routing tables based on route



Routing table for host A

Destination	Next Hop
Host B	R1

Routing table for R1

Destination	Next Hop
Host B	R2

Routing table for R2

Destination	Next Hop
Host B	—

b. Routing tables based on next hop

Figure 19.29 Network-specific routing

Routing table for host S based on host-specific routing

Destination	Next Hop
A	R1
B	R1
C	R1
D	R1

Routing table for host S based on network-specific routing

Destination	Next Hop
N2	R1

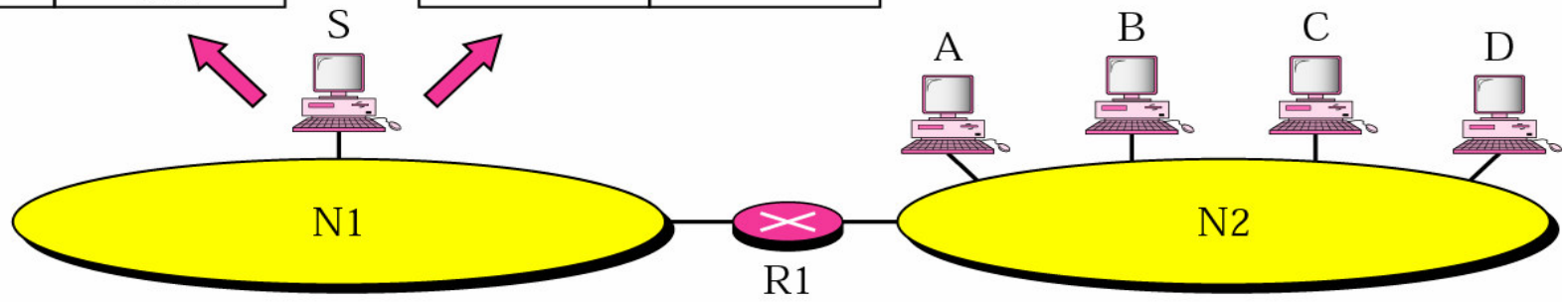


Figure 19.30 Host-specific routing

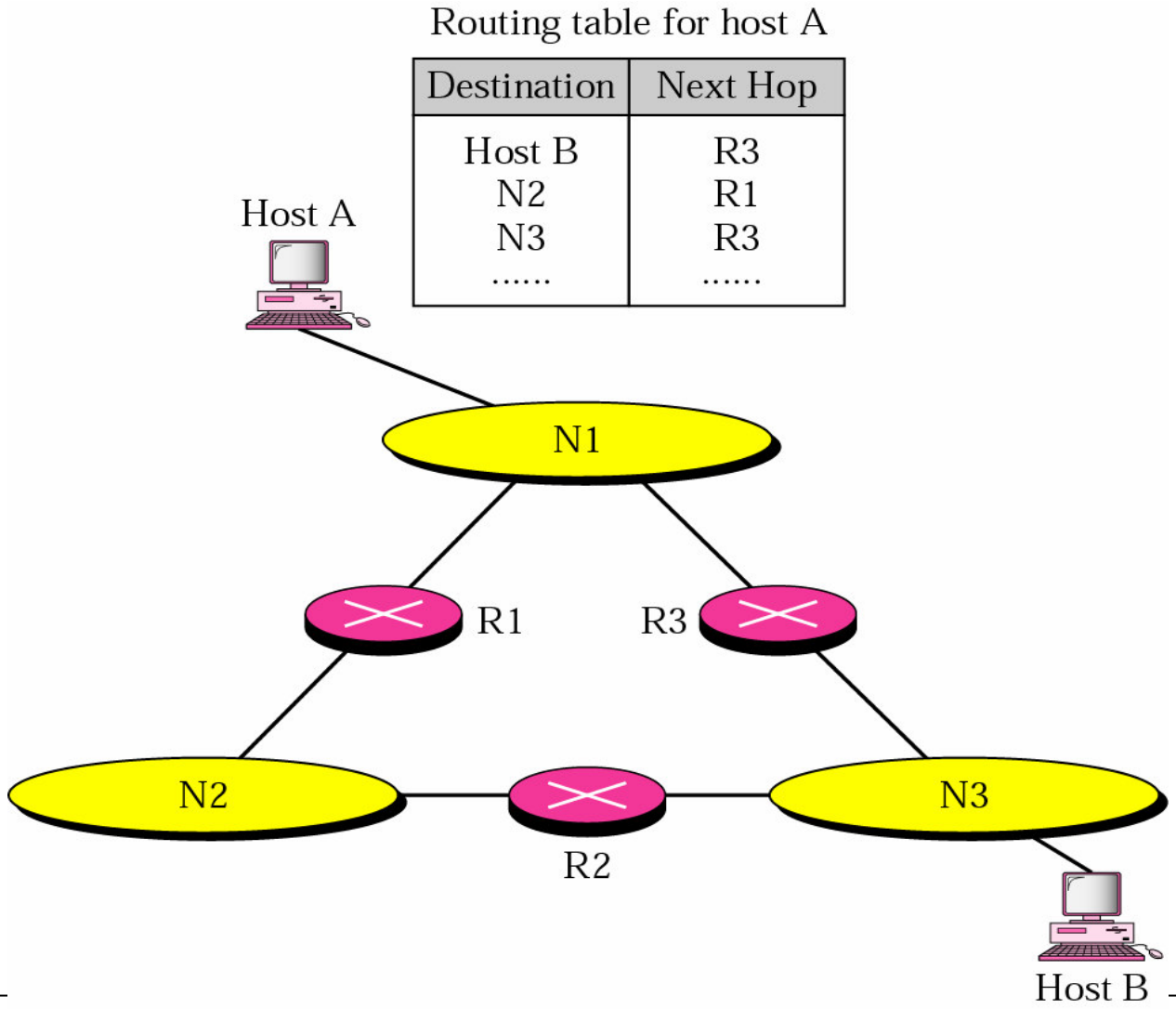


Figure 19.31 Default routing

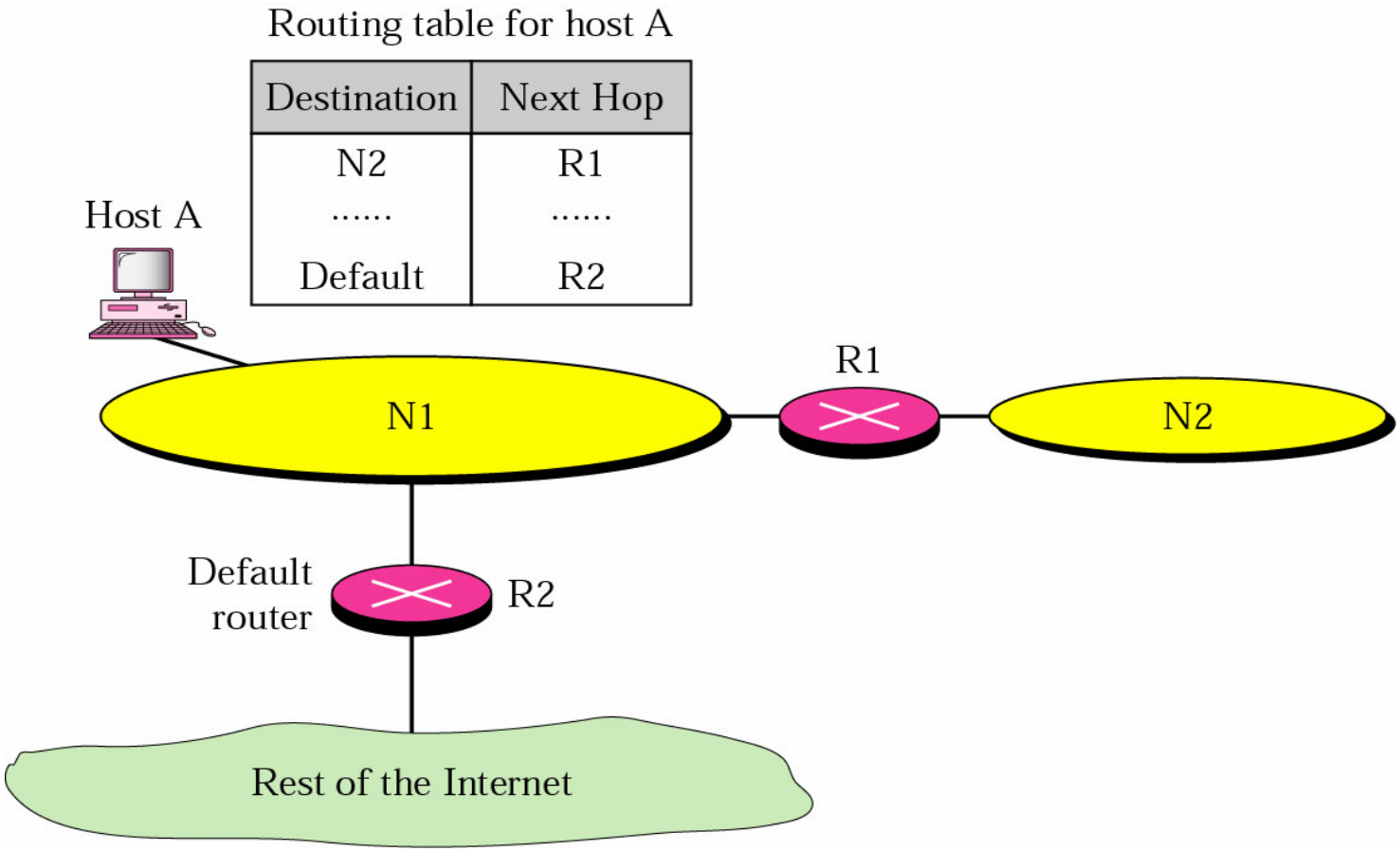


Figure 19.32 Classful addressing routing table

	Mask	Destination address	Next-hop address	Interface
	/8	14.0.0.0	118.45.23.8	m1
Host-specific →	/32	192.16.7.1	202.45.9.3	m0
	/24	193.14.5.0	84.78.4.12	m2
Default →	/0	/0	145.11.10.6	m0

Example 10

Using the table in Figure 19.32, the router receives a packet for destination 192.16.7.1. For each row, the mask is applied to the destination address until a match with the destination address is found. In this example, the router sends the packet through interface m0 (host specific).

Example 11

Using the table in Figure 19.32, the router receives a packet for destination 193.14.5.22. For each row, the mask is applied to the destination address until a match with the next-hop address is found. In this example, the router sends the packet through interface m2 (network specific).

Example 12

Using the table in Figure 19.32, the router receives a packet for destination 200.34.12.34. For each row, the mask is applied to the destination address, but no match is found. In this example, the router sends the packet through the default interface m0.

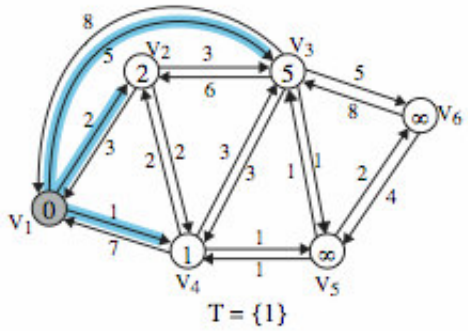
Dijkstra's Algorithm

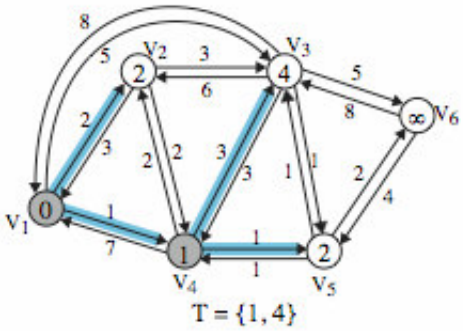
- finds shortest paths from given source node `s` to all other nodes
- by developing paths in order of increasing path length
- algorithm runs in stages (next slide)
  - each time adding node with next shortest path
- algorithm terminates when all nodes processed by algorithm (in set T)

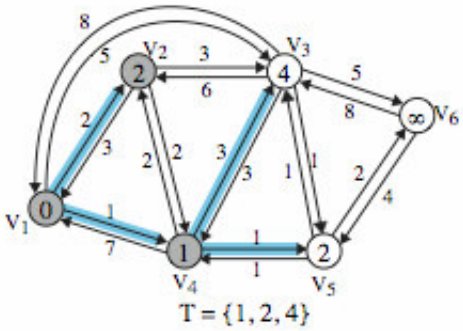
Dijkstra's Algorithm Method

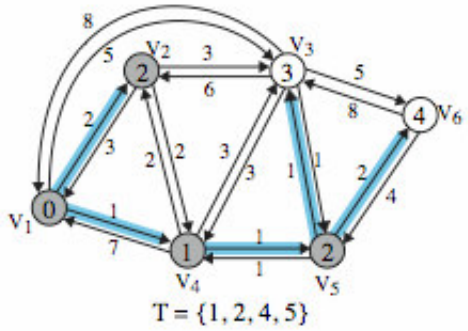
- Step 1 [Initialization]
  - T = {s} Set of nodes so far incorporated
  - L(n) = w(s, n) for n ≠ s
  - initial path costs to neighboring nodes are simply link costs
- Step 2 [Get Next Node]
  - find neighboring node not in T with least-cost path from s
  - incorporate node into T
  - also incorporate the edge that is incident on that node and a node in T that contributes to the path
- 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

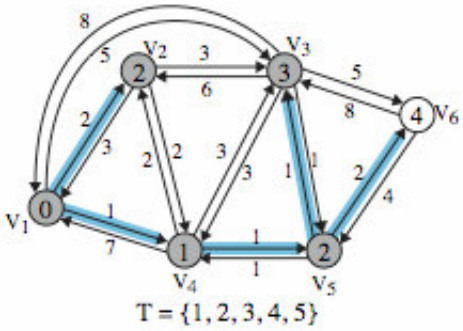
Dijkstra's Algorithm Example

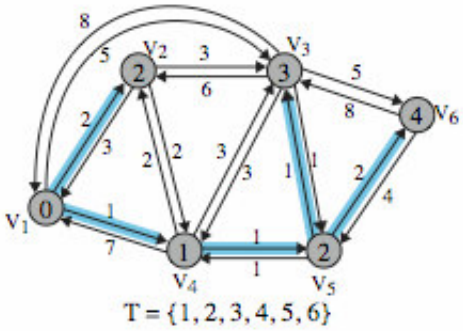












Dijkstra's Algorithm Example

Iter	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

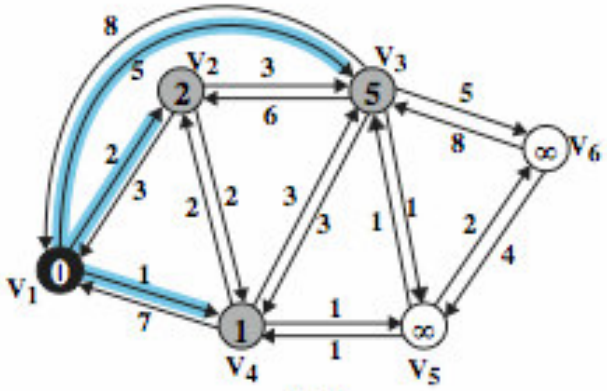
Bellman-Ford Algorithm

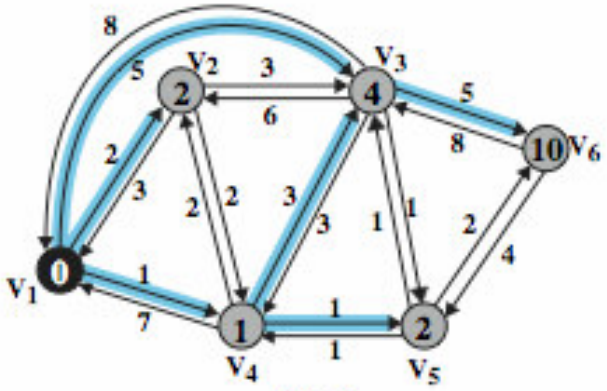
- find shortest paths from given node subject to constraint that paths contain at most one link
- find the shortest paths with a constraint of paths of at most two links
- and so on

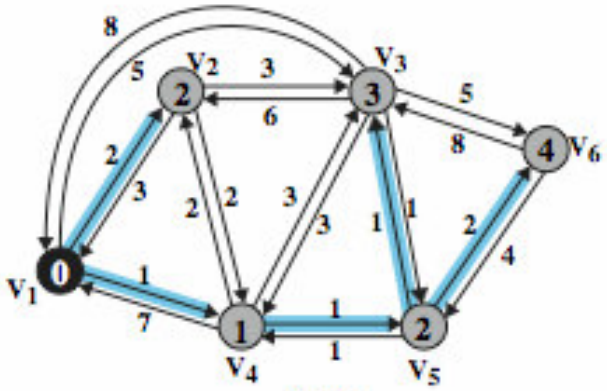
Bellman-Ford Algorithm

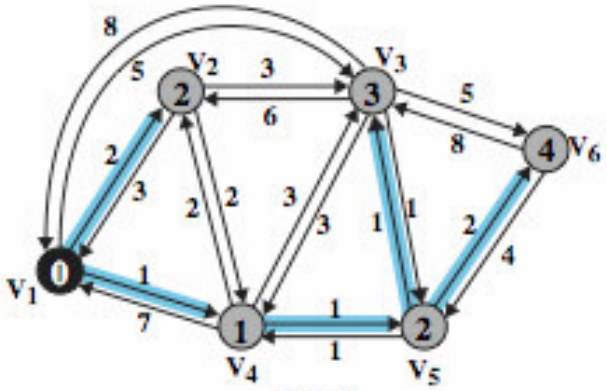
- step 1 [Initialization]
  - $L_0(n) = \infty$, for all $n \neq s$
  - $L_h(s) = 0$, for all h
- step 2 [Update]
  - for each successive $h \geq 0$
    - for each $n \neq s$, compute: $L_{h+1}(n) = \min_j [L_h(j) + w(j,n)]$
  - connect n with predecessor node j that gives min
  - eliminate any connection of n with different predecessor node formed during an earlier iteration
  - path from s to n terminates with link from j to n

Example of Bellman-Ford Algorithm









Results of Bellman-Ford Example

h	$L_h(2)$	Path	$L_h(3)$	Path	$L_h(4)$	Path	$L_h(5)$	Path	$L_h(6)$	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

Comparison

- results from two algorithms agree
- Bellman-Ford
  - calculation for node n needs link cost to neighbouring nodes plus total cost to each neighbour from s
  - each node can maintain set of costs and paths for every other node
  - can exchange information with direct neighbours
  - can update costs and paths based on information from neighbors and knowledge of link costs
- Dijkstra
  - each node needs complete topology
  - must know link costs of all links in network
  - must exchange information with all other nodes

Evaluation

- dependent on
  - processing time of algorithms
  - amount of information required from other nodes
- implementation specific
- both converge under static topology and costs
- both converge to same solution
- if link costs change, algs attempt to catch up
- if link costs depend on traffic, which depends on routes chosen, may have feedback instability