Chapter 8: Packet Switching

Raj Jain

Raj Jain is now at
Washington University in Saint Louis
Jain@cse.wustl.edu
http://www.cse.wustl.edu/~jain/

- Circuit, datagram, virtual circuit switching
- Routing algorithms
- ARPAnet routing

Datagram vs Virtual Circuit

Connectionless vs connection-oriented

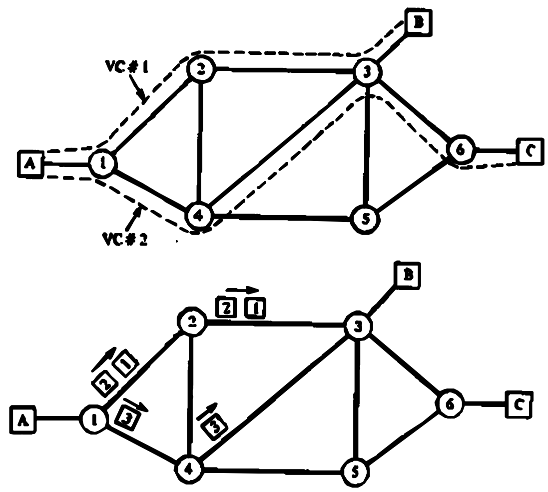


Fig 8.4

Circuit vs Datagram vs VC

Circuit Switching	Datagram	Virtual Circuit
Dedicated transmission path	No dedicated path	No dedicated path , Shared path
Continuous transmission of data	Bursty	Bursty
No buffering required	Buffers required	Buffers required
Path fixed at connection setup	Different packets may take different paths	Path fixed at connection setup
Call setup delay	Queueing delays	Call setup + queueing delays
Overload blocks new calls	Overload increases queueing delays	Overload may block new calls. May increase queueing delays
Source and destination have the same speed	Source and destination may have different speed	Source and destination may have different speed
Bandwidth is reserved. Unused bandwidth is wasted	Bandwidth is dynamically shared among users	Bandwidth is reserved as well as dynamically shared
No overhead bits after call setup	Overhead bits in each packet	Less overhead bits in each packet
Switches keep state	Switches donot keep state	Switches keep state
No or negligible loss	Loss possible	Loss possible

On link failure — Connection continues — VC broken

Table 8.1

External vs Internal VC

- External: End-to-end (Transport layer)
- Internal: On the path (Network layer)

End-to-end (External)		On the path (Internal): Datagram	On the path (Internal): Virtual Ckt
	No connection (Datagram)	UDP/IP	
	Connection (VC)	TCP/IP	SNA TYMNET

Routing

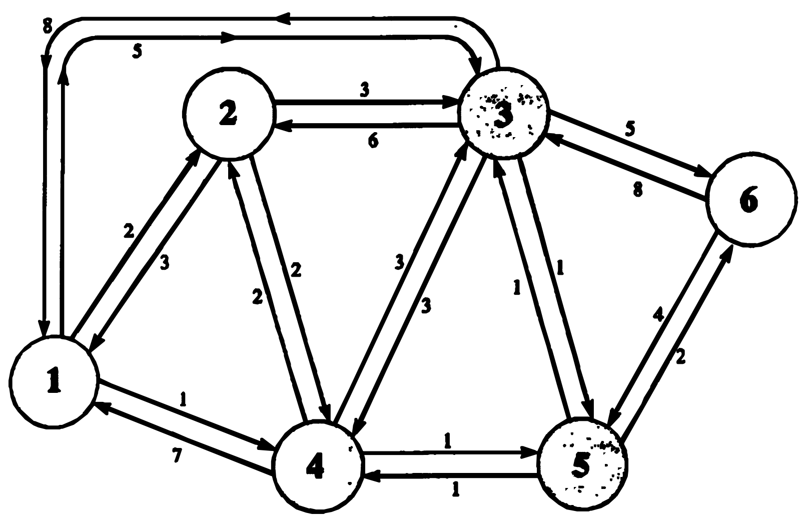


Fig 8.8

Rooting or Routing

- *Rooting* is what fans do at football games, what pics do for truffles under oak trees in the Vaucluse, and what nursery workers intent on propagation do to cuttings from plants.
- *Routing* is how one creates a beveled edge on a table top or sends a corps of infanctrymen into full scale, disorganized retreat

Ref: Piscitello and Chapin, p413

Routeing or Routing

- Routeing: British
- Routing: American
- Since Oxford English Dictionary is much heavier than any other dictionary of American English, British English generally prevalis in the documents produced by ISO and CCITT; wherefore, most of the international standards for routing standards use the routeing spelling.

Ref: Piscitello and Chapin, p413

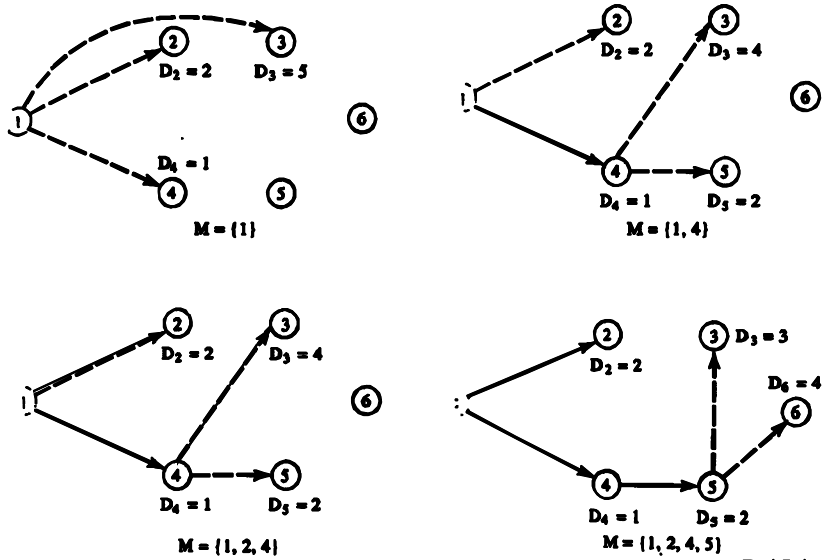
Routing Techniques Elements

- **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

Dijkstra's Algorithm

- Goal: Find the least cost paths from a given node to all other nodes in the network
- Notation:
  d_{ij} = Link cost from i to j if i and j are connected
  D_n = Total path cost from s to n
  M = Set of nodes so far for which the least cost path is known
- Method:
  - Initialize: M={s}, $D_n = d_{sn}$
  - Find node w $\notin$ M, whose Dn is minimum
  - Update D_n

Example



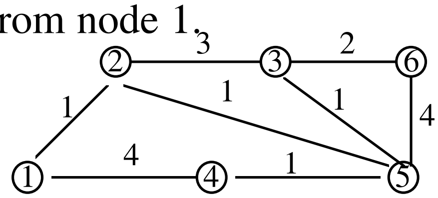
Example (Cont)

	M	D2	Path	D3	Path	D4	Path	D5	Path	D6	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

Table 8.4a

Dijkstra's routing algorithm

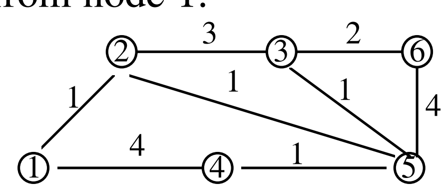
- Apply to the following network and compute paths from node 1.



	M	D2 Path	D3 Path	D4 Path	D5 Path	D6 Path
1						
2						
3						
4						
5						
6						

Dijkstra's routing algorithm

- Apply to the following network and compute paths from node 1.



	M	D2	Path	D3	Path	D4	Path	D5	Path	D6	Path
1	{1}	1	1-2	∞	-	4	1-4	∞	-	∞	-
2	{1,2}	1	1-2	4	1-2-3	4	1-4				
3	{1,2,3}	1	1-2	4	1-2-3	4	1-4	2	1-2-5	6	1-2-3-6
4	{1,2,3,5}	1	1-2	4	1-2-3	3	1-2-5-1	2	1-2-5	6	1-2-3-6
5	{1,2,3,4,5}	1	1-2	4	1-2-3	3	1-2-5-1	2	1-2-5	6	1-2-3-6
6	{1,2,3,4,5,6}	1	1-2	4	1-2-3	3	1-2-5-1	2	1-2-5	6	1-2-3-6

Bellman-Ford Algorithm

- Notation:
  h = Number of hops being considered
  $D^{(h)}_n$ = Cost of h-hop path from s to n
- Method: Find all nodes 1 hop away
  Find all nodes 2 hops away
  Find all nodes 3 hops away
  - Initialize: $D^{(h)}_n = \infty$ for all $n \neq s$; $D^{(h)}_n = 0$ for all h
  - Find jth node for which h+1 hops cost is minimum
    $D^{(h+1)}_n = \min_j [D^{(h)}_j + D_{jn}]$

Example

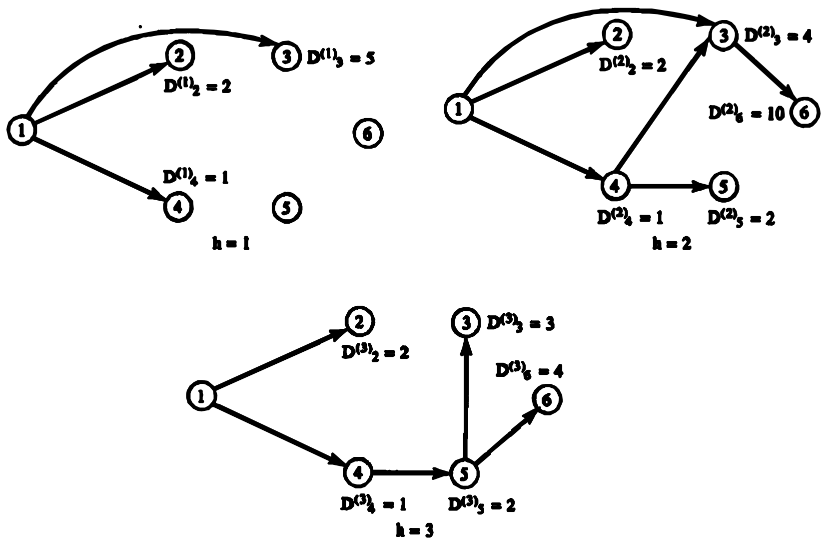


Fig 8.9b

Example (Cont)

h	$D_2^{(h)}$	Path	$D_3^{(h)}$	Path	$D_4^{(h)}$	Path	$D_5^{(h)}$	Path	$D_6^{(h)}$	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	1-5-4-3	1	1-4	2	1-4-5	4	1-4-5-6
4	2	1-2	3	1-5-4-3	1	1-4	2	1-4-5	4	1-4-5-6

Table 8.4b

Flooding

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

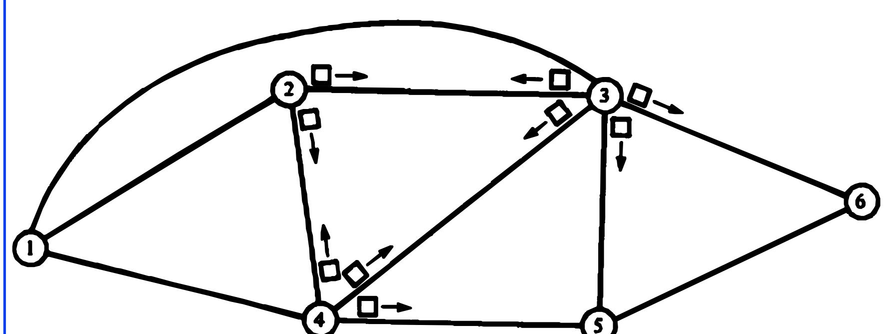


Fig 8.11b

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

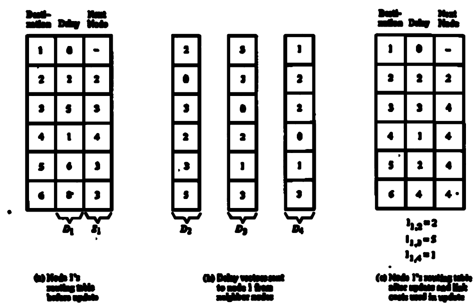


Fig 8.13

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

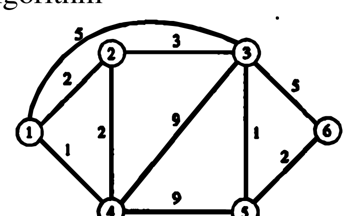


Fig 8.14

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

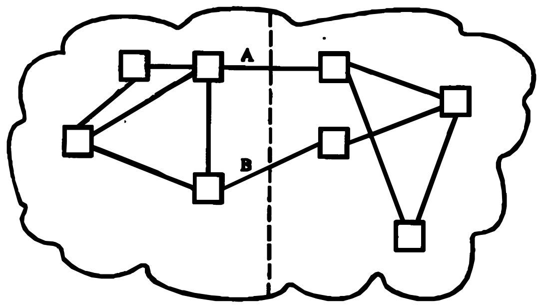


Fig 8.15

Routing Algorithm

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

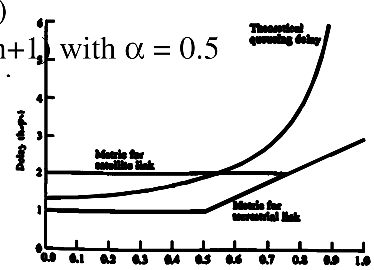
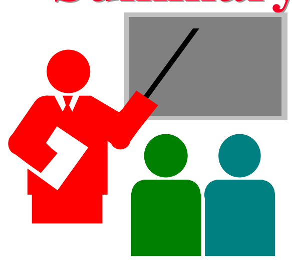


Fig 8.16

Summary



- Connection-oriented and Connectionless
- Routing: Least-cost, Flooding, Adaptive
- Dijkstra's and Bellman-Ford algorithms
- ARPAnet

Homework

- Excercise 8.4 (in b assume a unidirectional single loop), 8.10, 8.15
- Due: Next class