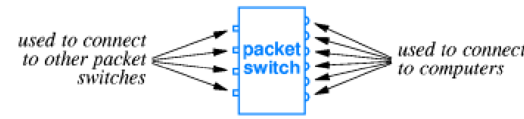
Routing

To there and back again

COMP476
Networked Computer Systems

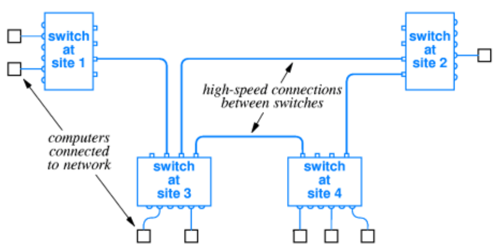
WAN – Wide Area Networks

- WANs are made of store and forward switches.



A packet switch with two types of I/O connectors: one type is used to connect to other packet switches, and the other is used to connect to computers.

Packet Switches



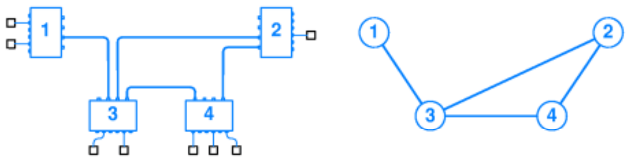
A small WAN formed by interconnecting packet switches. Connections between packet switches usually operate at a higher speed than connections to individual computers.

Packet Switch Operation

```
for all input lines {
   do forever {
        read a packet;
        if (valid) {
                output_line = table[destination_addr]
                if (queue[output_line] not full)
                        put packet on queue[output_line]
        }
   }
}
```

Routing - Network Model

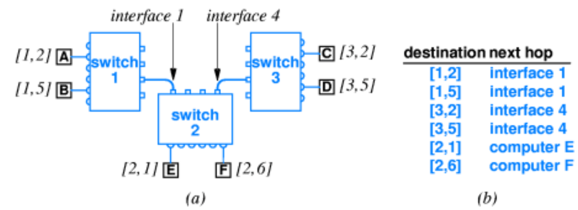
- The graph below models a network. Each node corresponds to a packet switch in the network.



Each node corresponds to a packet switch, and each edge represents a connection between the corresponding packet switches.

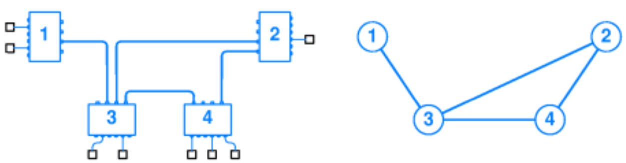
Next-Hop Forwarding

- A given switch has information about the next place (hop) to send a packet so the packet will eventually reach its destination.



The table is the next-hop forwarding information found in switch 2. Each switch has different next-hop information.

Next Hop Routing Tables

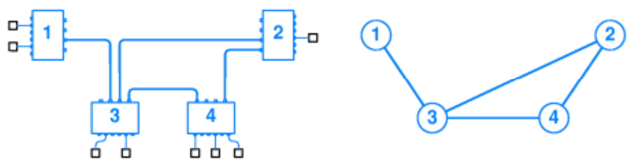


node 1			node 2			node 3			node 4	
dest	next		dest	next		dest	next		dest	next
1	-		1	3		1	1		1	3
2	3		2	-		2	2		2	2
3	3		3	3		3	-		3	3
4	3		4	4		4	4		4	-

Routing Tables with Alternatives

- Routing tables can also contain a second choice route
- In the event the first route is unavailable or congested, packets can be sent on the alternate interface.

Next Hop Routing Tables with Alternatives

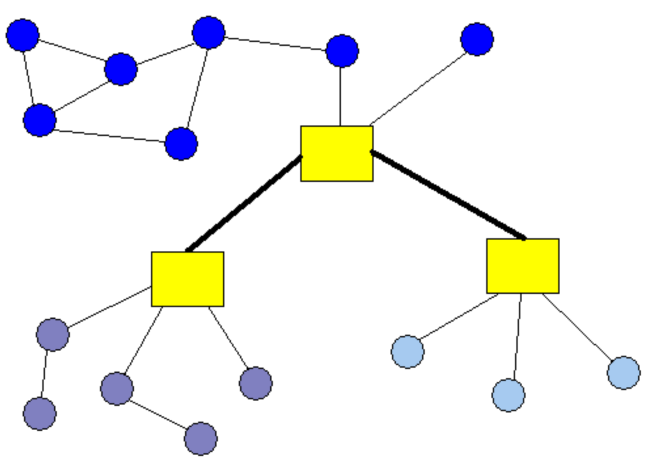


node 1			node 2			node 3			node 4		
dest	next	alt	dest	next	alt	dest	next	alt	dest	next	alt
1	-	-	1	3	4	1	1	-	1	3	2
2	3	-	2	-	-	2	2	4	2	2	3
3	3	-	3	3	4	3	-	-	3	3	2
4	3	-	4	4	3	4	4	2	4	-	-

Hierarchical Addressing

- Hierarchical addresses can be used to simplify routing.
- Local routing occurs without knowledge of distant nodes.
- If a packet needs to go to a distant node, it is given to a higher level node.
- The higher level nodes get the packet to the right group.
- Local routing gets the packet to the right node in the distant group.

Hierarchical Routing



Route Generation

- Global routing is done by local decisions.
- Routing table can be created:
  - By a central system and distributed to nodes
  - By sending node for each packet
  - “Learned” by each node from neighbors

Optimal Routes

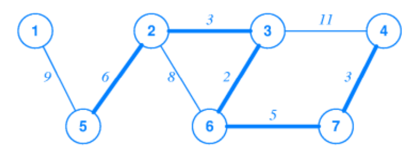
- There are several algorithms for computing the optimal path.

  •Dijkstra's algorithm
  •Floyd's algorithm
  •distributed algorithm

Dijkstra's Algorithm

(One of the more popular methods for computation of the optimal path)

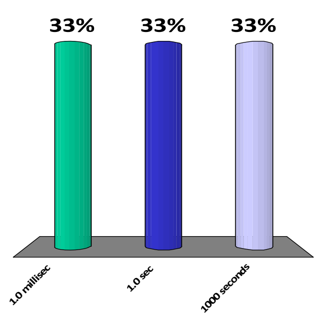
- Solves the problem of finding the shortest path from a point in a graph (the *source*) to a destination.
- Dijkstra's algorithm computes shortest paths in a graph by using weights on edges as a measure of distance. A path with the fewest number of edges may not be the path with least weight.



** A graph with weights assigned to edges. The shortest path between nodes 4 and 5 is shown darkened. The distance along the path is 19, the sum of the weights on the edges.*

Dijkstra's Algorithm is $O(n^2)$. If it takes 1.0 nsec per node and there are a million roads in the U.S., how long does it take to calculate a shortest path?

1. 1.0 millisec
2. 1.0 sec
3. 1000 seconds



Floyd's Algorithm

Start with a table giving the distance from each source to each destination. Immediate neighbors use the link distance. Other nodes have an infinite distance.

		Destination			
		1	**2**	**3**	**4**
source	**1**	0	∞	4	∞
	2	∞	0	5	1
	3	4	5	0	7
	4	∞	1	7	0

Floyd's Algorithm

For each non-infinite link, see if that is a shorter way to other destinations. If shorter, replace the current distance with the sum of the distance to an intermediate plus the distance from the intermediate to the destination.

		Destination			
		1	**2**	**3**	**4**
source	**1**	0	∞	4	∞
	2	∞	0	5	1
	3	4	5	0	7
	4	∞	1	7	0

Floyd's Algorithm

Node 1 can get to node 3 in 4 time units. Node 3 can get to node 2 in 5 time units. The time to get from node 1 to 3 to 2 is 4+5=9.

		Destination			
		1	**2**	**3**	**4**
source	**1**	0	9	4	∞
	2	∞	0	5	1
	3	4	5	0	7
	4	∞	1	7	0

Floyd's Algorithm

Repeat for each source node, for each intermediate and for each destination. Node 1 can get to node 4 by 1 → 3 (4 time units) then 3 → 4 (7 time units for a total of 11).

		Destination			
		1	**2**	**3**	**4**
source	**1**	0	9	4	11
	2	9	0	5	1
	3	4	5	0	7
	4	11	1	7	0

Floyd's Algorithm

Iterate until no more changes are made.

Node 3 → 4 by node 3 → 2 then 2 → 4.

		Destination			
		1	**2**	**3**	**4**
source	**1**	0	9	4	10
	2	9	0	5	1
	3	4	5	0	6
	4	10	1	6	0

Distributed Routing

- Each node computes the time to send a packet to its neighbors.
- Periodically, each node shares its routing table with its neighbors.
- Use the minimum(current path,
  (time to neighbor + neighbor to destination))