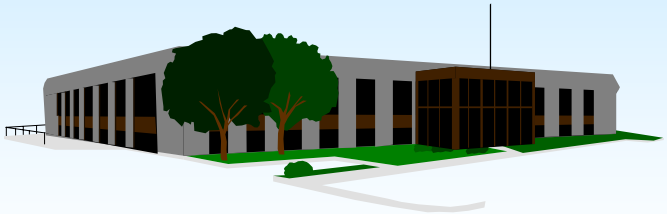
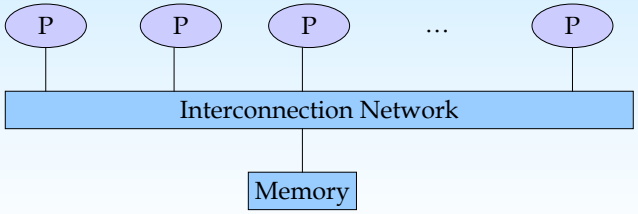
Interconnection Networks and Topologies



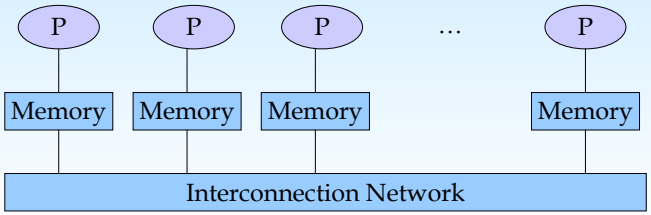
The Network is the Computer ™ – *Sun Microsystems*

Shared-memory Machine



Dancehall

Distributed-memory Machine

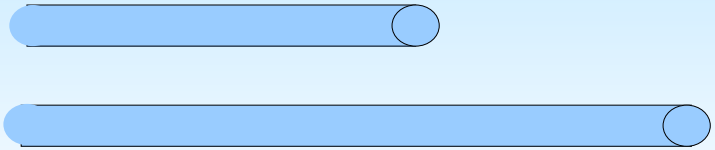


Boudoir

Static vs. Dynamic Networks

- Static networks consist of point-to-point communication links among nodes.
  - Typically used in distributed-memory machines
- Dynamic networks are built using switches and communication links; communication links are connected to one another dynamically by the switching elements to establish paths among the nodes
  - Typically used by shared-memory machines

Two Main Problems in Networks

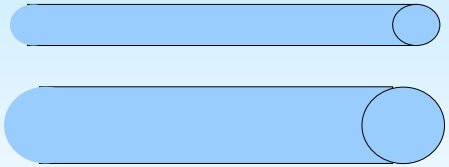


Latency: Time to travel the pipe

Short Pipe – Short Latency

Long Pipe – Long Latency

Two Main Problems in Networks



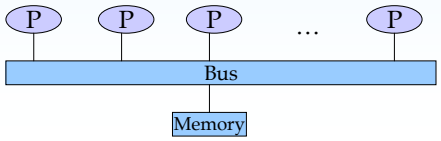
Bandwidth: Stuff that can be accommodated in the pipe

Thin Pipe – Small bandwidth

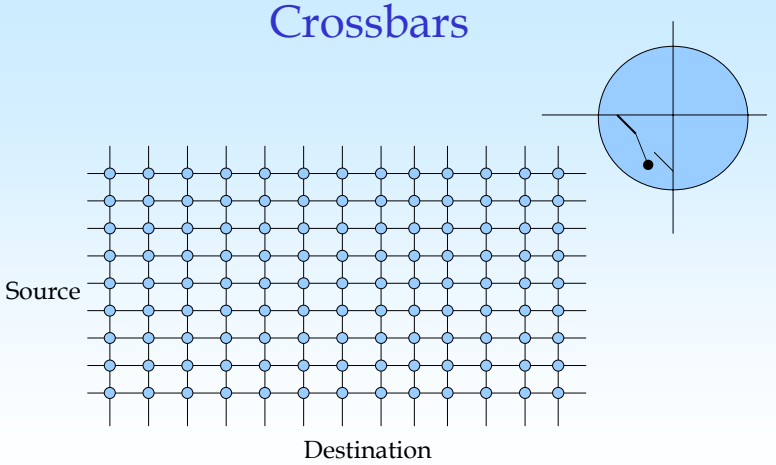
Thick Pipe – Large bandwidth

Bus-Based Networks

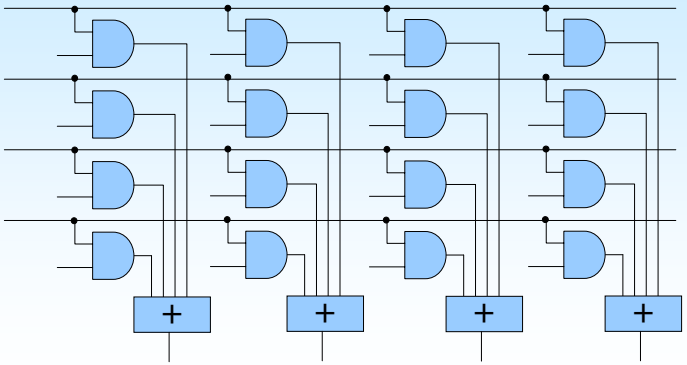
- There is a common data path called the bus that connects the nodes
- As the number of nodes increases, each node spends an increasing amount of time waiting for bus slot
- Performance saturates with small number of nodes



Crossbars



Crossbars

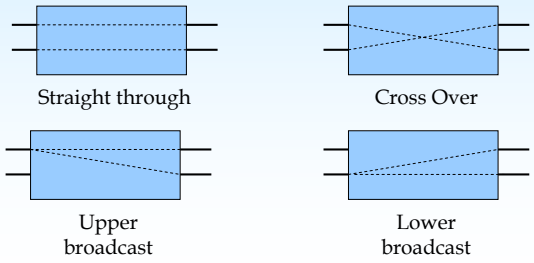


Crossbars

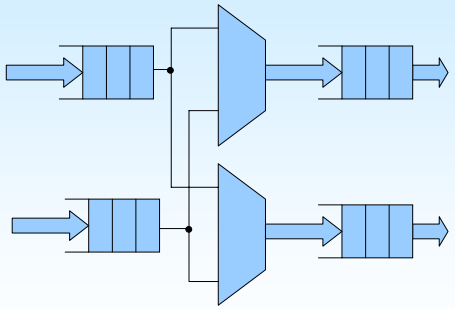
- To interconnect p nodes, we need p^2 switching elements.
- Though scalable in terms of performance, not scalable in terms of cost.

Multistage Interconnection Networks

- A broad class of dynamic switching networks can be constructed from 2x2 switches which have the following possible settings:



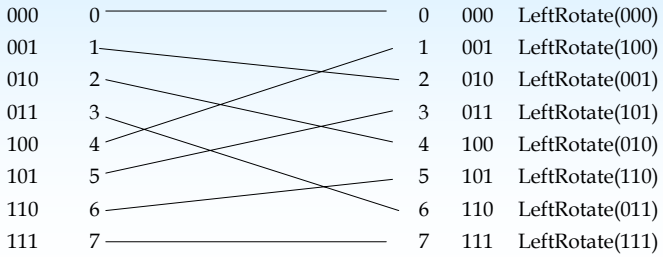
Organization of a Simple Switch



What are the multiplexer control settings for straight-through, cross over, upper broadcast, and lower broadcast?

Perfect Shuffle

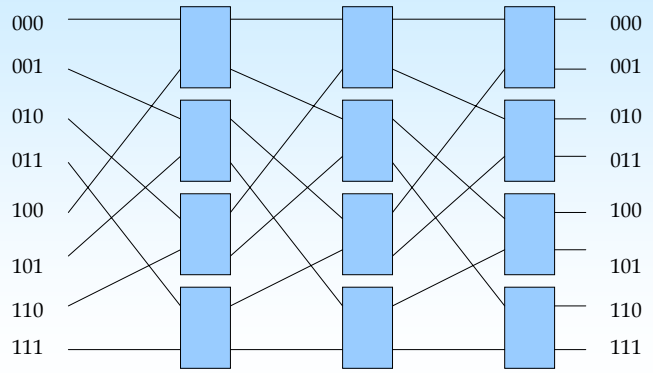
- A link exists between j and i, if
  - j = 2i (when 0 < i < p/2)
  - j = 2i+1-p (when p/2 < i < p-1)



Omega Network

- A *pxp* omega network consists of log*p* stages, with *p*/2 log *p* switches.
- The switching elements use either straight through or cross over settings.
- At each stage, a perfect shuffle interconnection pattern feeds into a set of *p*/2 switching elements

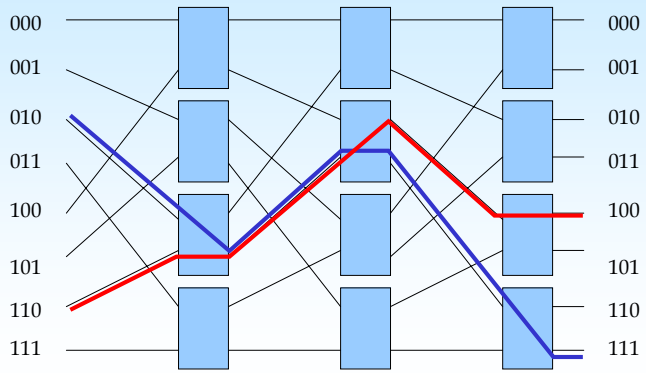
Omega Network



Routing in an Omega Network

- Let *s* and *t* be the binary representations of the source and destination of a message.
- The message traverses the link to the first switching element
- If the most significant bits of *s* and *t* are the same, then the message is routed in straight-through mode by the switch; otherwise in cross-over mode.
- The next stage switch considers the next most significant bits.

Omega Network

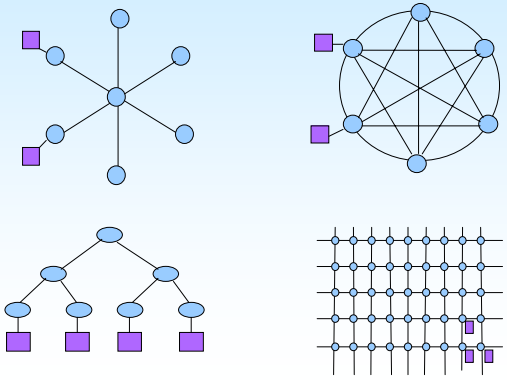


Example: (110 to 100) and (010 to 111)

Omega Network

- This network can support many simultaneous groups of point-to-point message requests, but not all possible combinations.
  - Some messages may be blocked by the settings required for other messages.
  - E.g., consider (110 to 100) and (010 to 111)
- If a single switch fails, some connections are no longer possible.

Topologies



Topological Measures

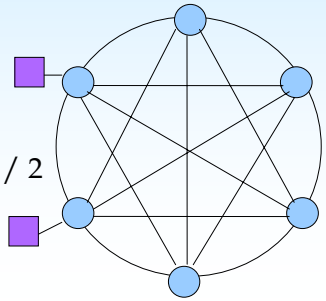
- Diameter
  - The greatest distance between any two nodes
- Node degree/valance
  - The number of edges joining it to other nodes

Topological Measures

- Bisection width
  - The minimum number of communication links that have to be removed to partition the network into two equal halves (assuming the worst case partition)
- Bisection bandwidth
  - The minimum volume of communication allowed between any two halves of the network with an equal number of nodes (assuming the worst case partition)
  - Bisection width × Link bandwidth

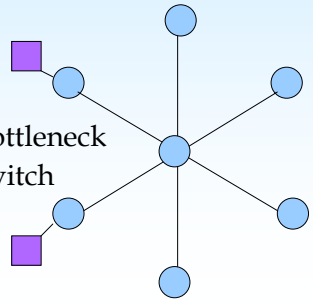
Fully Connected Network

- The most powerful interconnection network topology
  - Each node is directly connected to *all* other nodes
- Advantages
  - Low diameter (always 1)
- Disadvantage
  - Does not scale well
  - Large number of links: $p(p\text{-}1) / 2$



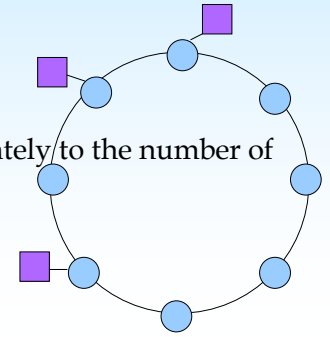
Star-Connected Network

- All communications go through the central switch
- Advantages
  - Low diameter (2)
  - Only p links
- Disadvantage
  - Central switch becomes the bottleneck
  - Large degree (p) for central switch



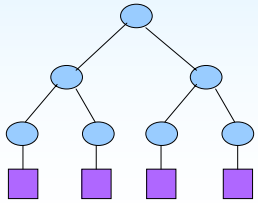
Rings

- A simple ring is just a linear array with the end nodes linked.
- Advantage
  - Low degree (always 2)
- Disadvantage
  - Diameter grows proportionately to the number of nodes



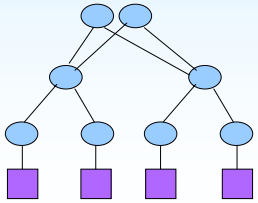
Trees

- Message routing from *a* to *b* involves the root of the smallest sub-tree containing *a* and *b*
- Linear array & star-connected networks are special cases of trees



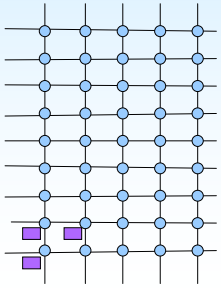
Fat Trees

- Bottleneck at higher levels of the tree
  - Increase links at these levels: *fat* tree



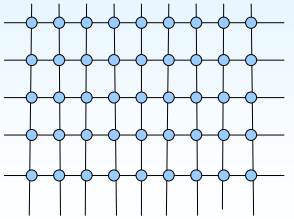
Mesh and Torus

- In a mesh, nodes are laid out along the axis of a *d*-dimensional grid
  - A *Torus* is a mesh whose edges wrap around
- These topologies are popular
  - They are simple to construct
  - Their diameter grows as sqrt(*p*)
  - Can be expanded easily
  - Matrix calculations fit this topology



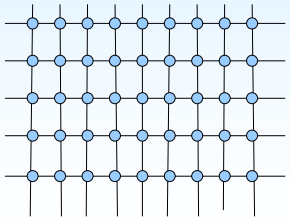
XY-Routing

- A message is sent along the X dimension until it reaches the column of the destination processor, and then along the Y dimension until it reaches its destination.



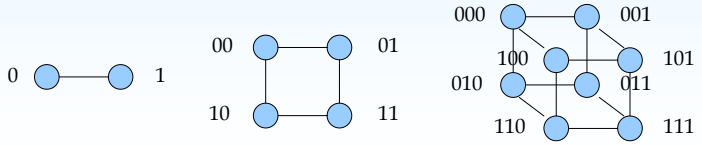
XY-Routing

- A message transferred from $P_{(sx,sy)}$ through to $P_{(dx,dy)}$ passes through $P_{(sx+1,sy)}$, $P_{(sx+2,sy)}$, …, $P_{(dx,sy)}$, and then through $P_{(dx,sy+1)}$, $P_{(dx,sy+2)}$, … to reach $P_{(dx,dy)}$.



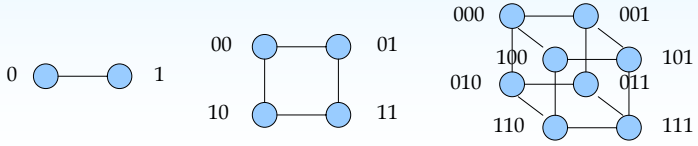
Hypercube Network

- Advantages
  - Perfect routing is easy to implement
  - Diameter grows logarithmically with the number of nodes
  - Hypercubes have lots of links



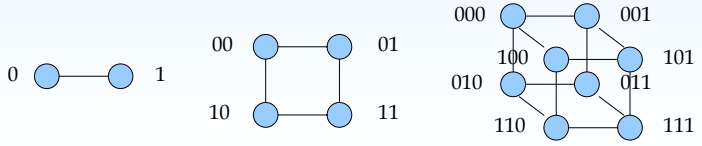
Hypercube Network

- In a *d*-dimensional hypercube, each node is directly connected to *d* other nodes.
- A *d*-dimensional hypercube can be partitioned into two *(d-1)*-dimensional hypercubes easily



Hypercube Network

- Node ids in a *d*-dimensional hypercube contain *d* bits (in their binary representation)
- Two nodes are connected by a link if and only if their binary ids differ at exactly one bit position
- The number of bits at which two ids differ is the Hamming distance

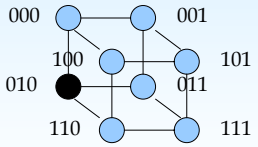


E-cube Routing

- Like XY-routing, E-cube routing is based on the source and destination processors' labels P_s and P_d.
- The minimum distance between P_s and P_d is given by the number of 1s in $P_s \oplus P_d$.
- Processor P_i computes and sends the message along dimension k where k is the position of the least significant non-zero bit in $P_i \oplus P_d$. Here i starts with s (i.e., the source processor).

E-cube Routing

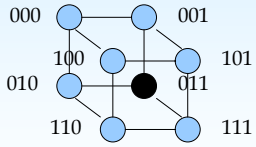
- Let P_s = 010 and P_d = 111.



$P_i \oplus P_d = 101$

E-cube Routing

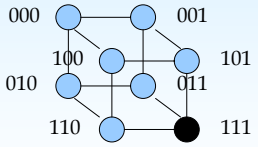
- Let P_s = 010 and P_d = 111.



$P_i \oplus P_d = 100$

E-cube Routing

- Let P_s = 010 and P_d = 111.



$P_i \oplus P_d = 000$

Routing

- The process of deciding how best to deliver a message.
  - Which path to take?
  - What resources are needed?
  - How to avoid deadlock?

Which path to take?

- Traverse a link that reduces the distance to the destination.
- Some routing schemes allow a message to take a path which increases the distance to the destination if the links required for an optimal path are in high demand.

What resources are needed

- Buffer space to store the message, on each node. This depends on the message forwarding strategy.
- Link bandwidth.
  - Can a link be committed to transfer a message packet in this cycle?

Avoiding deadlock

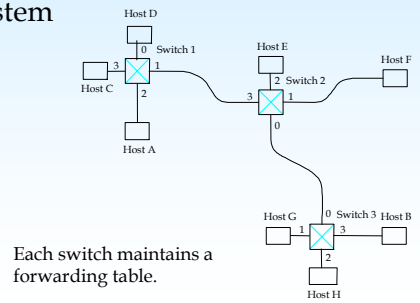
- Allocate shared resources in a fixed global order.
- Back off and retry later. This can lead to *livelock*.
  - Livelock is the recurrent occurrence of resource allocation sequences that lead to back-off in all the participants. It is the busy-waiting analog of deadlock.

General Routing Methods

- Datagram or connectionless
  - Make sure each packet has enough info to enable switching decision
- Virtual circuit or connection-oriented
  - Circuit is constructed before packets are sent

Datagram Switching

- No connection setup phase
- Each packet forwarded independently
- Also known as *connectionless* model
- Analogy: postal system



Datagram Switching

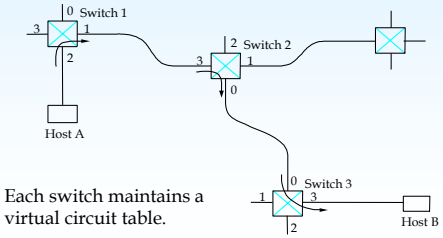
- There is no round trip time delay waiting for connection setup; a host can send data as soon as it is ready.
- Source host has no way of knowing if the network is capable of delivering a packet or if the destination host is even up.

Datagram Switching

- Since packets are treated independently , it is possible to route around link and node failures.
- Since every packet must carry the full address of the destination, the overhead per packet is higher than for the connection-oriented model.

Virtual Circuit Switching

- Explicit connection setup (and tear-down) phase
- Subsequence packets follow same circuit
- Also known as *connection-oriented* model
- Analogy: phone call



Virtual Circuit Switching

- Typically wait full round trip time for connection setup before sending first data packet.
- While the connection request contains the full address for destination, each data packet contains only a small identifier, making the per-packet header overhead small.

Virtual Circuit Switching

- If a switch or a link in a connection fails, the connection is broken and a new one needs to be established.
- Connection setup provides an opportunity to reserve resources.

Communication costs

- Communication latency is the time to communicate a message from one processor to another. It is the sum of
  - the time to prepare the message for transmission, and
  - the time taken by the message to traverse the network to its destination
  - the time to receive and unpack the message at destination

Communication costs

- *Time of flight*: Time first bit of the message takes to arrive at the receiver
- *Transmission time*: Time spent by the message in the network (excludes time of flight)
- *Transport latency*: time of flight + transmission time

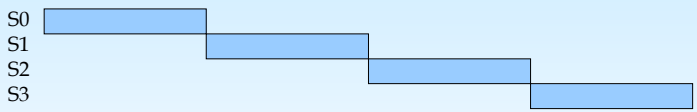
Communication costs

- Startup time (t_s) is the time required to prepare the message for transmission at the source processor
- Per-hop time (t_h) is the time taken by the header of the message to travel between two directly-connected processors.
- Per-packet transfer time (t_p) is the time a packet takes to traverse a link.

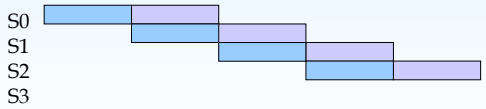
Store-and-Forward Routing

- Each processor on the path of message traversal forward the message to the next processor after it has received and stored the *entire* message.
- A message of size m (in number of packets) traversing l links will therefore incur a transmission latency of $(mt_p + t_h)l = lt_h + lmt_p$
  - Can be approximated to mt_pl since t_h is generally small compared to mt_p.

Store-and-Forward Routing



Message sent over a store & forward network



Same message broken into two parts and sent over the network

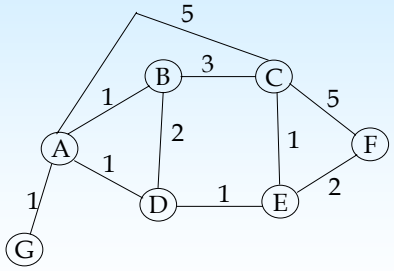
Wormhole & Cut-Through Routing

- Examine the header, and start forwarding the message without waiting for the arrival of the entire message
  - *Wormhole*: When the head is blocked, the body & tail are spread in the network.
    - Potential blocking of other messages in the network
  - *Cut-through*: When the head is blocked, the body & tail get to the switch where the head is.
- Transmission latency: $lt_h + mt_p$

Routing Table

- Each node maintains a set of triples
  - **(Destination, Cost, NextHop)**
- Exchange updates directly connected neighbors
  - periodically (on the order of several seconds)
  - whenever its table changes (called *triggered* update)
- Each update is a list of:
  - **(Destination, Cost)**
  - **(Destination, Cost, NextHop)**
- Update local table if a "better" route is received
- Refresh existing routes; delete if they time out

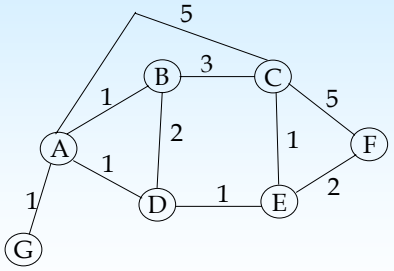
Example



A's routing table

Destination	Cost	Next hop
B	1	B
C	3	D
D	1	D
E	2	D
F	4	D
G	1	G

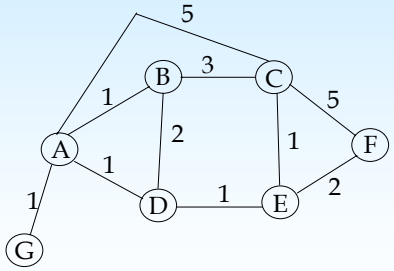
Example



B's routing table

Destination	Cost	Next hop
A	1	A
C	3	C
D	2	D
E	3	D
F	5	D
G	2	A

Example

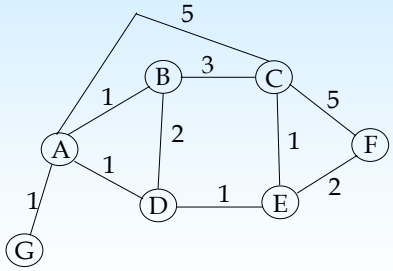


D's routing table

Destination	Cost	Next hop
A	1	A
B	2	B
C	2	E
E	*x*	E
F	3	E
G	2	A

D detects that link to E failed, sets distance to E to *x*.

Example

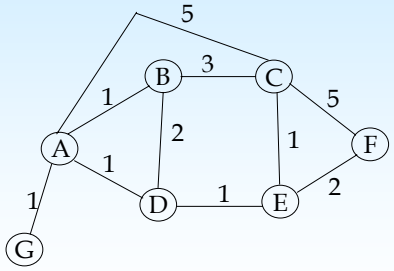


B's routing table

Destination	Cost	Next hop
A	1	A
C	3	C
D	2	D
E	*x*	D
F	*x*	D
G	2	A

B sets distances to E and F to *x* after update from D

Example



B's routing table

Destination	Cost	Next hop
A	1	A
C	3	C
D	2	D
E	4	C
F	8	C
G	2	A

B sets distance to E to 4 and F to 8 after update from C

Classifying Routing

- Minimal vs. non-minimal
  - Minimal routing always selects one of the shortest paths between the source and the destination
  - Non-minimal routing may use a longer path
- Deterministic vs. adaptive
  - Deterministic uses a unique path based on the source and destination
  - Adaptive determines the path based on the current state of the network

Dimension-ordered Routing

- This is a deterministic minimal routing
- Links to traverse are based on a numbering scheme determined by the dimension of the link
  - 2-dimensional mesh uses XY-routing
  - A hypercube uses E-cube routing

Flow & Congestion Control

- Necessary to regulate traffic on the network to avoid "jams".
  - *Flow control* describes how traffic between pairs of senders & receivers is regulated
  - *Congestion control* describes how the collective traffic on the network is regulated
- Solutions: discard packets, backpressure flow control, credit-based flow control, …

Further Reading

- Feng, T. Y. (1981). *A Survey of Interconnection Networks*. IEEE Computer, 14 (12), 12-27.
- Saad, Y. and Schultz, M. H. (1988) *Topological Properties of Hypercubes*. IEEE Transactions on Computers. Volume 37, 867-872.
- Hennessy, J. and Patterson, D. *Computer Architecture: A Quantitative Approach*, Chapter 8.
- Shay, W. *Understanding Communications and Networks*.