CS405 **Computer System Architecture**

Hwang, Chapter **7**

Multiprocessors and Multicomputers

7.4 Message Passing Mechanisms

Message Passing in Multicomputers

- Multicomputers have no shared memory, and each “computer” consists of a single processor, cache, private memory, and I/O devices.

- Some “network” must be provided to allow the multiple computers to communicate.

- The communication between computers in a multicomputer is called “message passing.

Message Formats

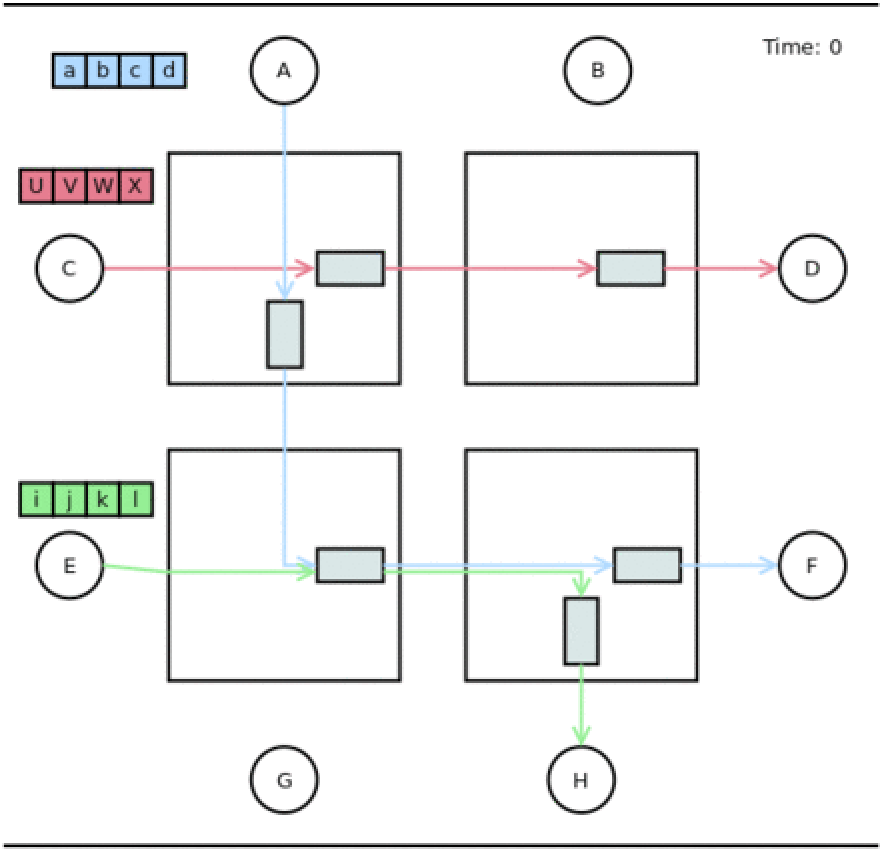
- Messages may be fixed or variable length.
- Messages are comprised of one or more packets.
- Packets are the basic units containing a destination address (e.g. processor number) for routing purposes.
- Different packets may arrive at the destination asynchronously, so they are sequence numbered to allow reassembly.
- Flits (flow control digits) are used in wormhole routing

Store and Forward Routing

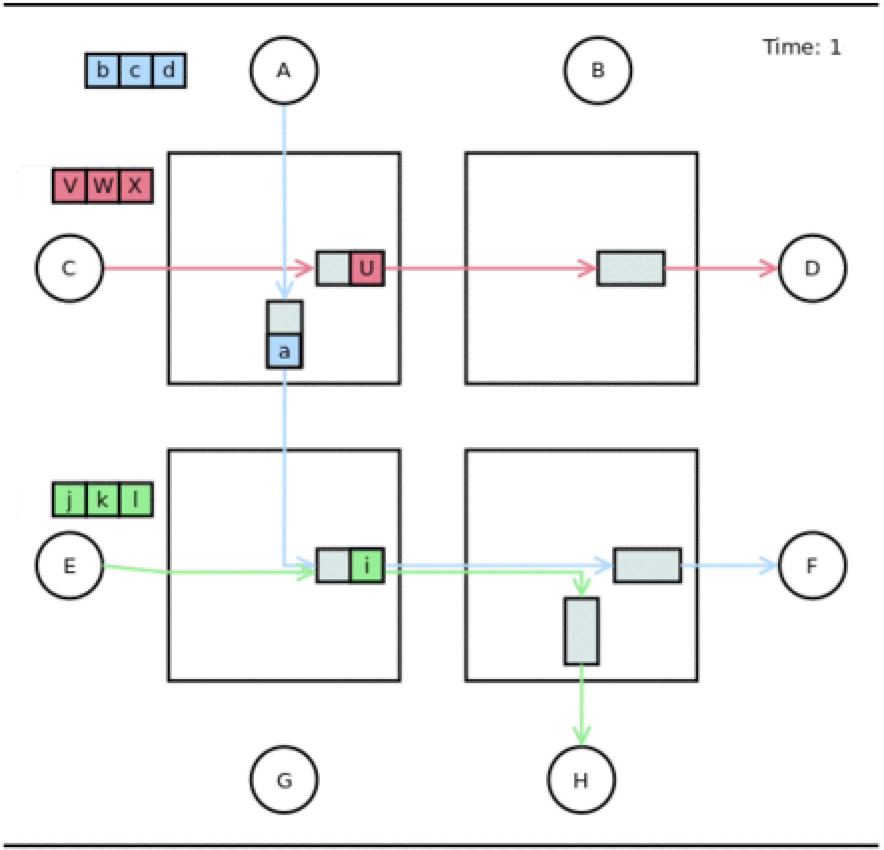
- Packets are the basic unit in the store and forward scheme.
- An intermediate node must receive a complete packet before it can be forwarded to the next node or the final destination, only if the output channel is free and the next node has available buffer space for the packet.
- The latency in store and format networks is directly related to the number of intermediate nodes through which the packet must pass.

Flits and Wormhole Routing

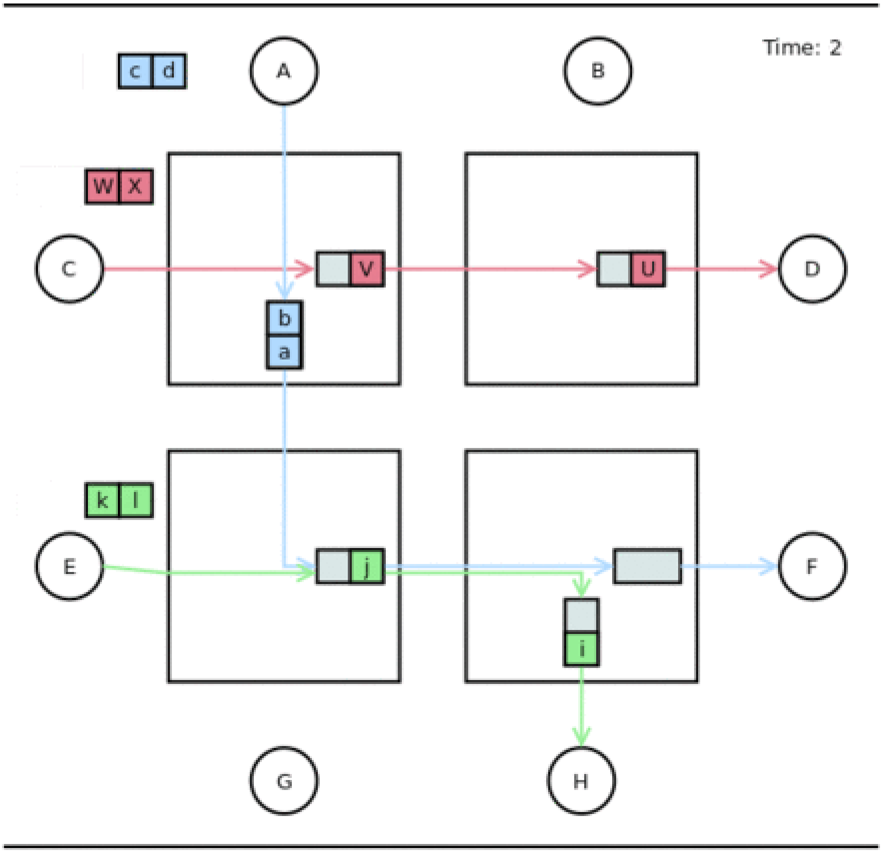
- Wormhole routing divides a packet into smaller fixed-sized pieces called **flits** (**fl**ow control dig**its**).
- The first flit in the packet must contain (at least) the destination address. Thus the size of a flit must be at least $\log_2 N$ in an N-processor multicomputer.
- Each flit is transmitted as a separate entity, but all flits belonging to a single packet must be transmitted in sequence, one immediately after the other, in a pipeline through intermediate routers.

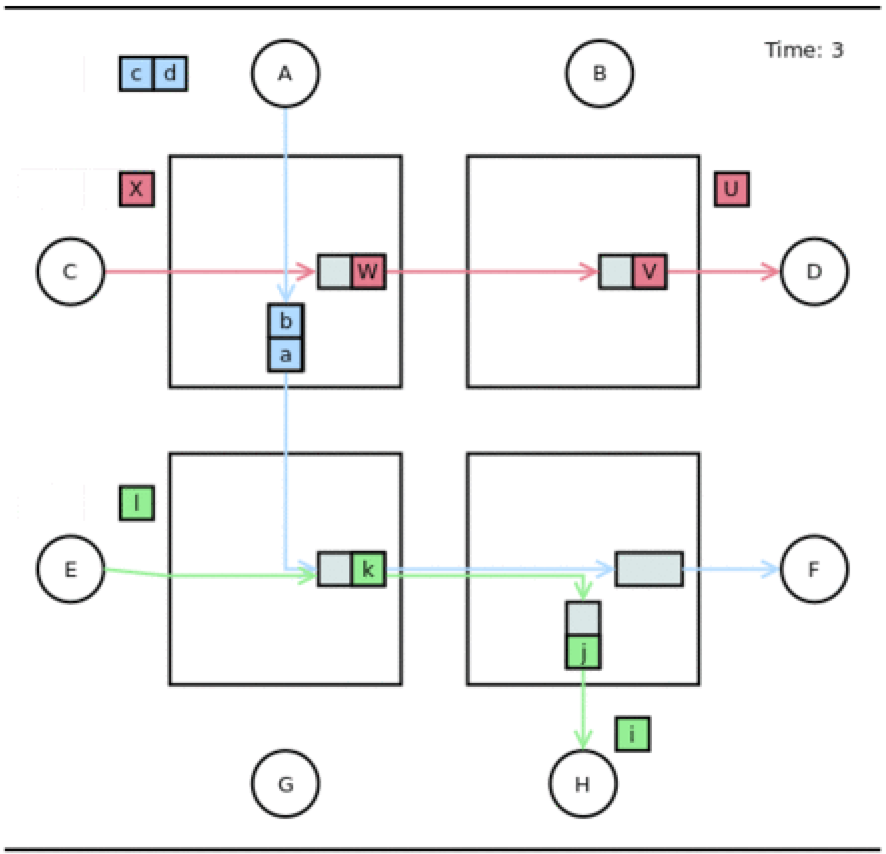


a b c d
A
B
Time: 0
U V W X
C
D
i j k l
E
F
G
H

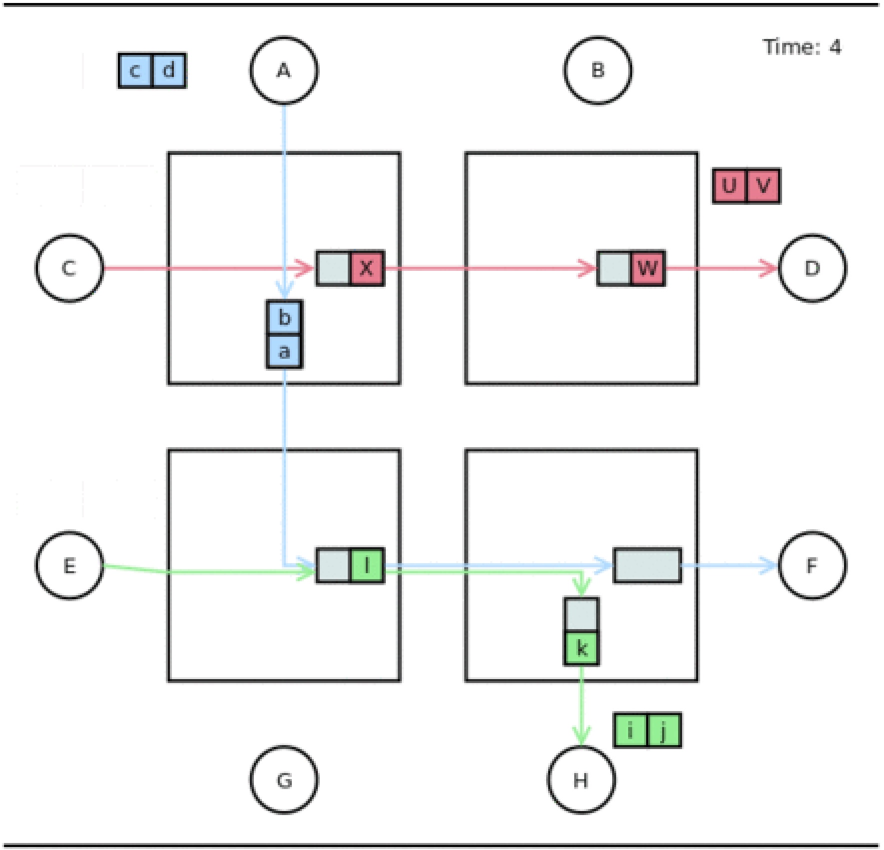


Time: 1
b c d
A
B
V W X
C
U
D
a
j k l
E
i
F
G
H

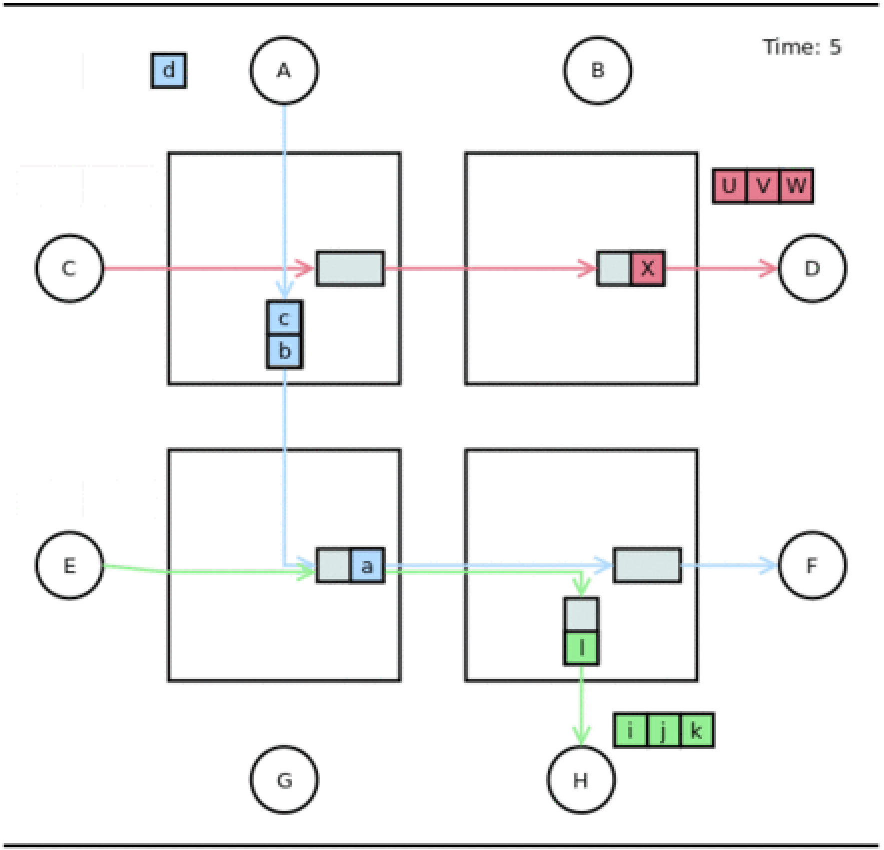

Time: 2
c d
A
B
W X
C
V
U
D
b
a
k l
E
j
F
i
G
H



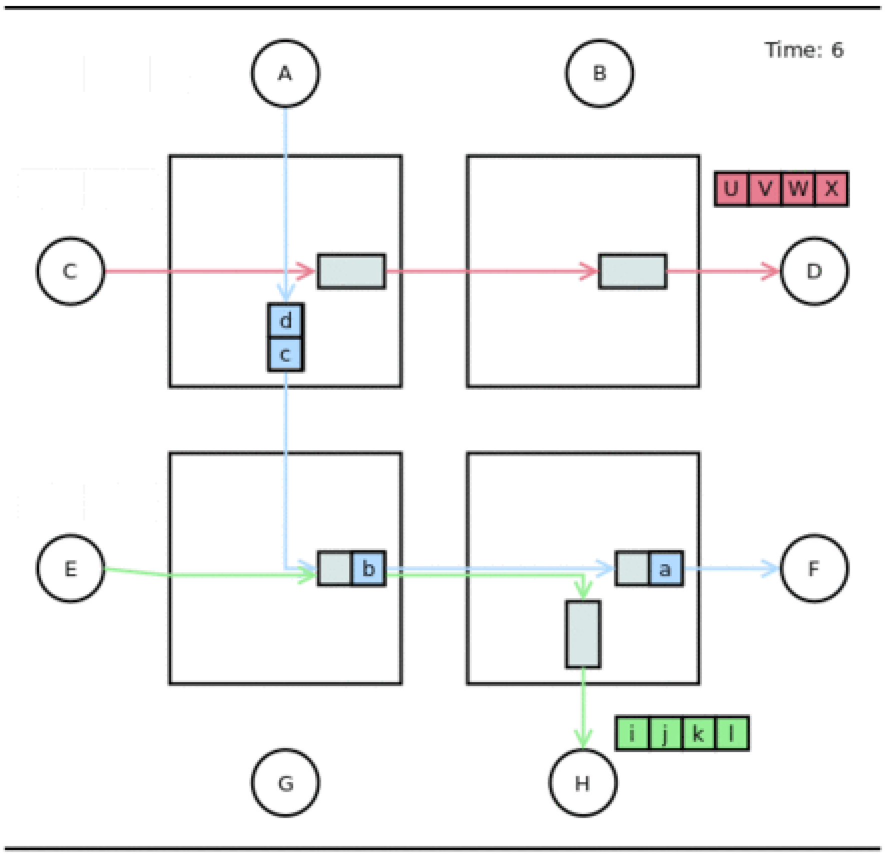
Time: 3
A
B
C
D
E
F
G
H
c
d
X
U
W
V
b
a
l
k
j
i



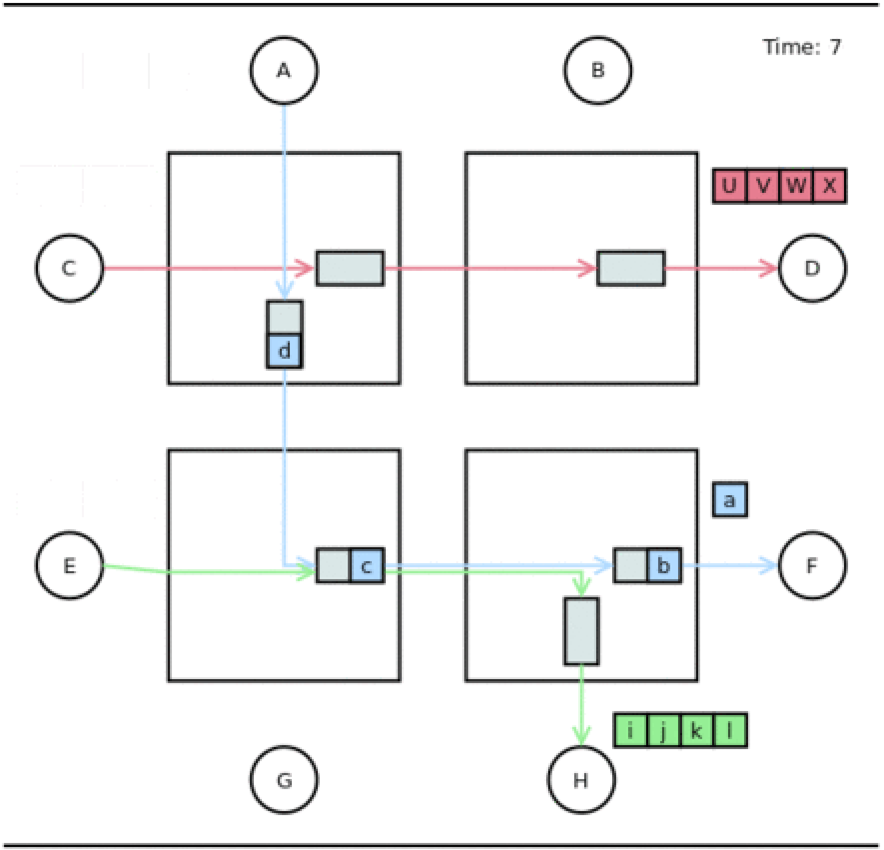
Time: 4
A
B
C
D
E
F
G
H
c
d
b
a
X
W
U
V
l
k
i
j



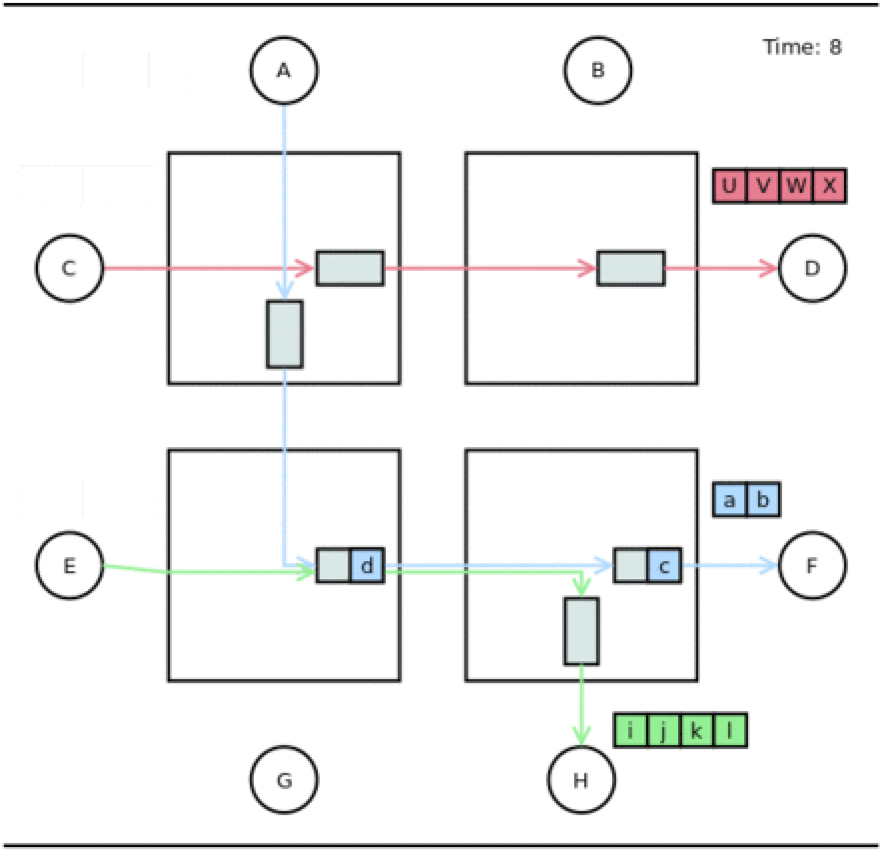
Time: 5
d
A
B
U
V
W
C
X
D
c
b
E
a
F
l
i
j
k
G
H



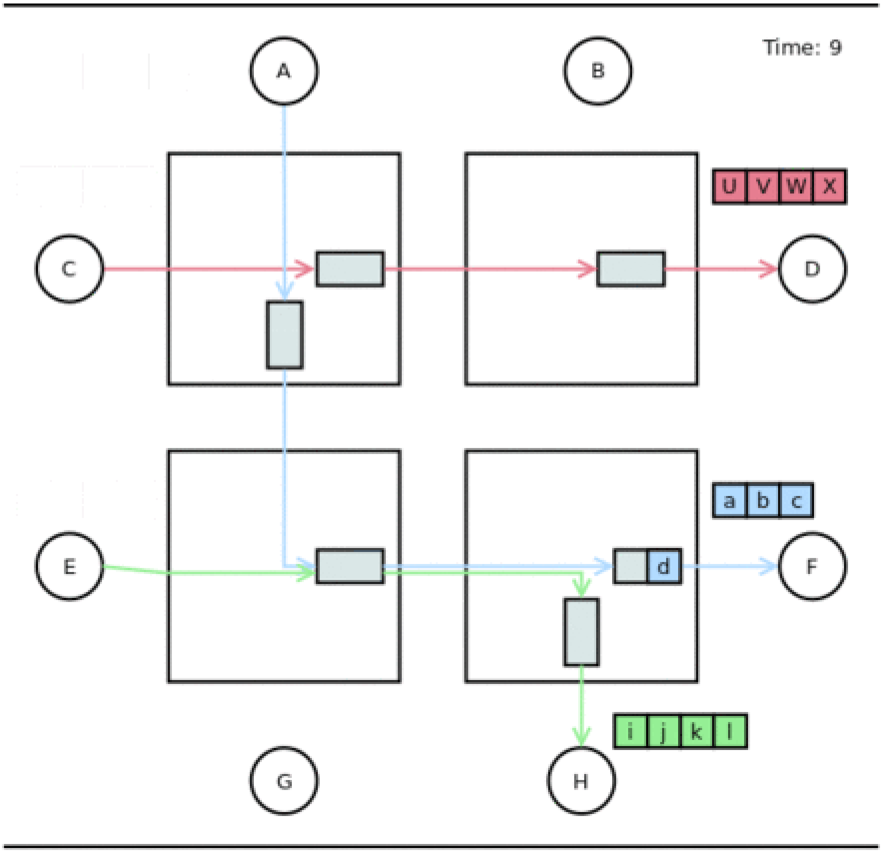
Time: 6
A
B
U
V
W
X
C
D
d
c
E
b
a
F
i
j
k
l
G
H

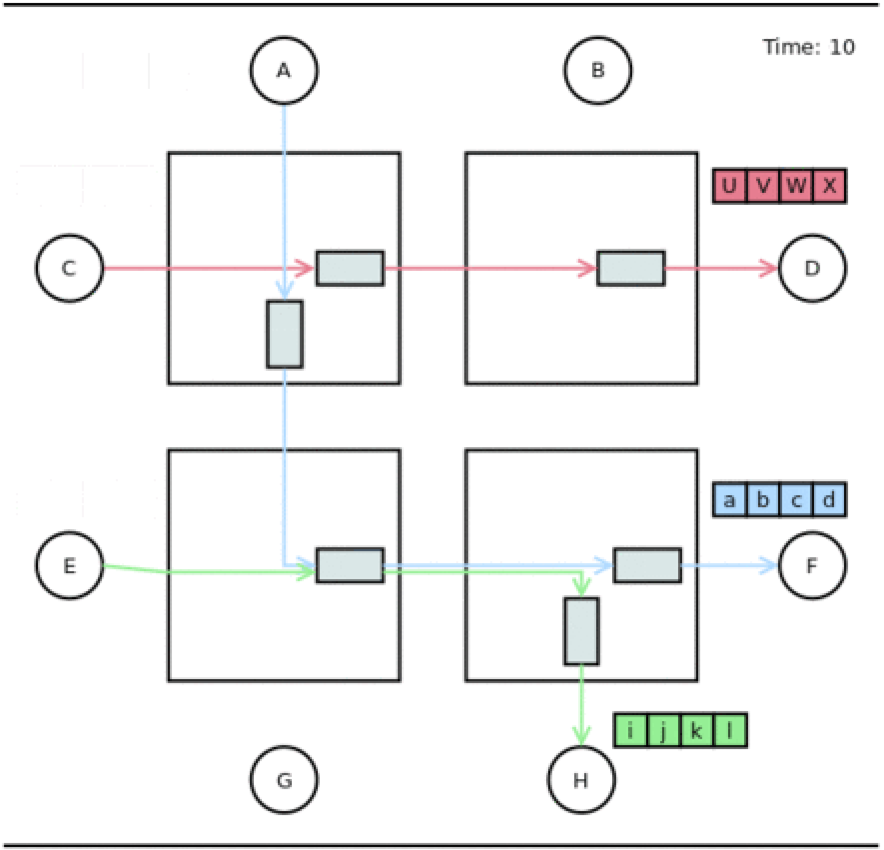


Time: 7
A
B
U
V
W
X
C
D
d
E
F
a
c
b
i
j
k
l
G
H



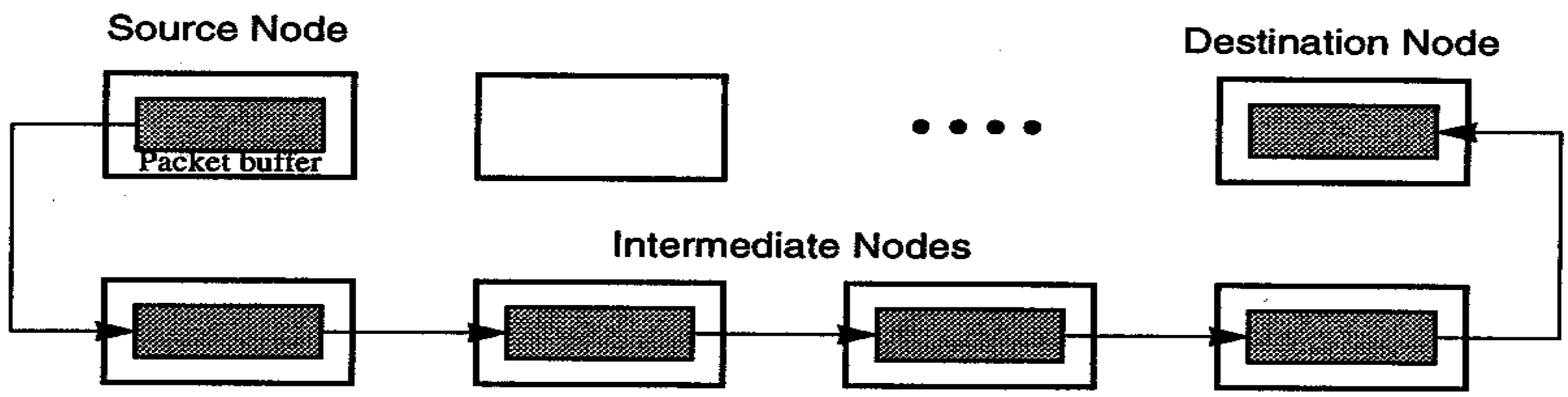
Time: 8
A
B
U
V
W
X
C
D
a
b
E
d
c
F
i
j
k
l
G
H


Time: 9
A
B
U V W X
C
D
a b c
E
d
F
i j k l
G
H

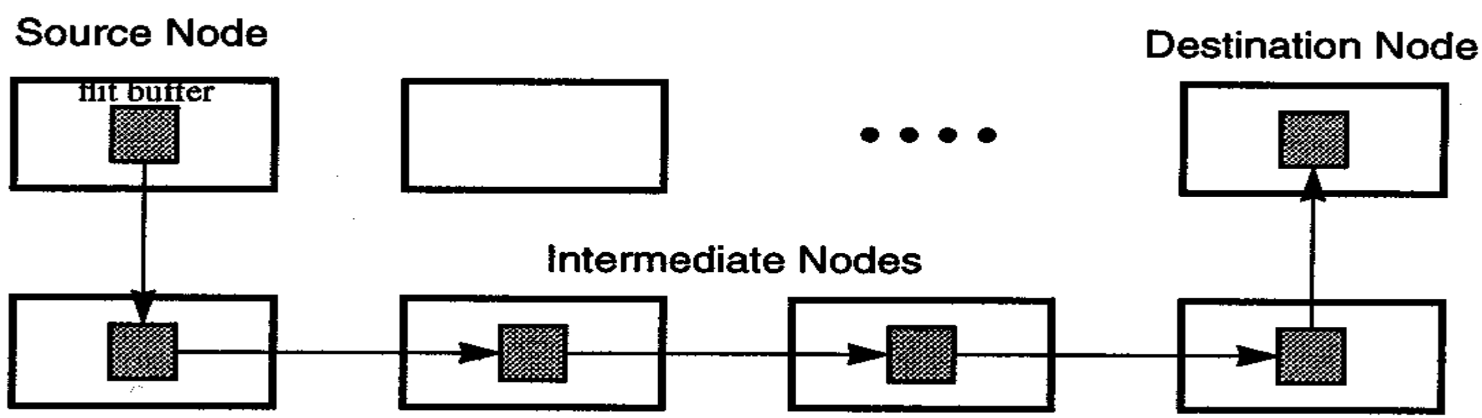


Time: 10
A
B
C
D
E
F
G
H
U
V
W
X
a
b
c
d
i
j
k
l

Store and Forward vs. Wormhole



(a) Store-and-forward routing using packet buffers in successive nodes

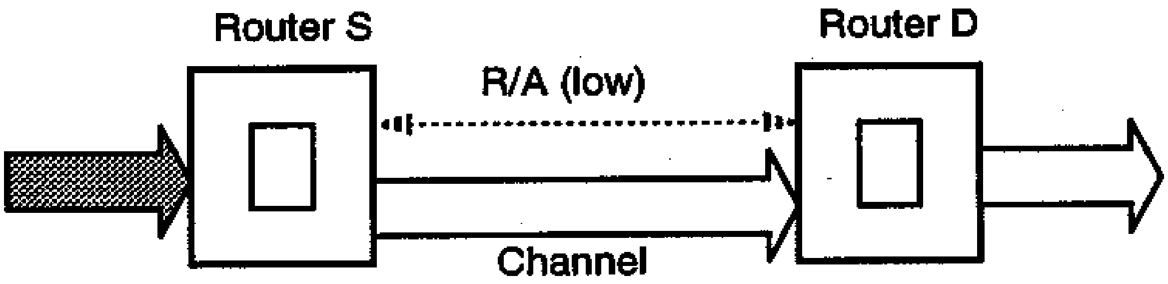


(b) Wormhole routing using flit buffers in successive routers

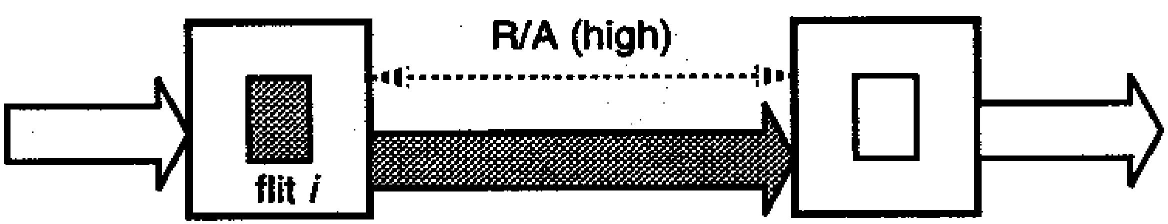
Asynchronous Pipelining

- Each intermediate node in a wormhole network, and the source and destination, each have a buffer capable of storing a flit.
- Adjacent nodes communicate requests and acknowledgements using a one-bit ready/request (R/A) line.
  - When a receiver is ready, it pulls the R/A line low.
  - When the sender is ready, it raises the R/A line high and transmits the next flit; the line is left high.
  - After the receiver deals with the flit (perhaps sending it on to another node), it lowers the R/A line to indicate it is ready to accept another flit.
  - The cycle repeats for transmission of other flits.

Wormhole Node Handshaking

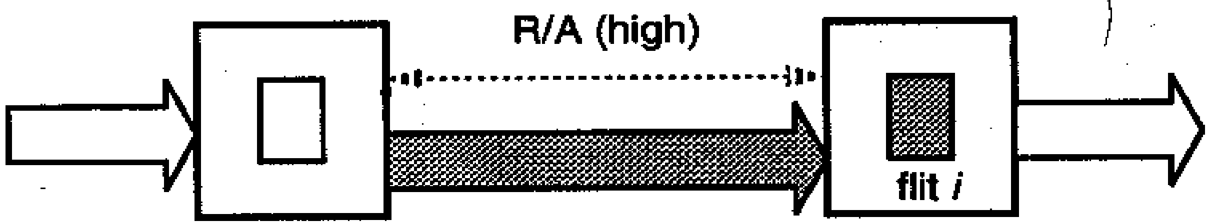


(a) D is ready to receive a flit

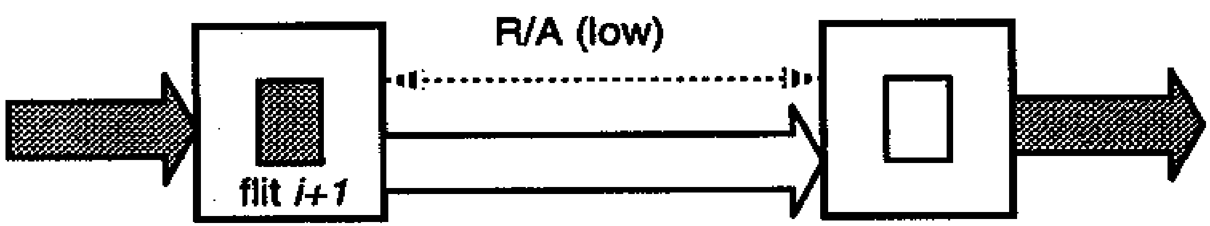


(b) S is ready to send flit i

Wormhole Node Handshaking



(c) Flit i is received by D



(d) Flit i is removed from D's buffer and flit $i+1$ arrives at S's buffer

Asynchronous Pipeline Speeds

- An asynchronous pipeline can be very efficient, and use a clock speed higher than that used in a synchronous pipeline.
- The pipeline can be stalled if buffers or successive channels in the path are not available during certain cycles.
- A packet could be "buffered, blocked, dragged, detoured" – and just knocked around, in general – if the pipeline stalls.

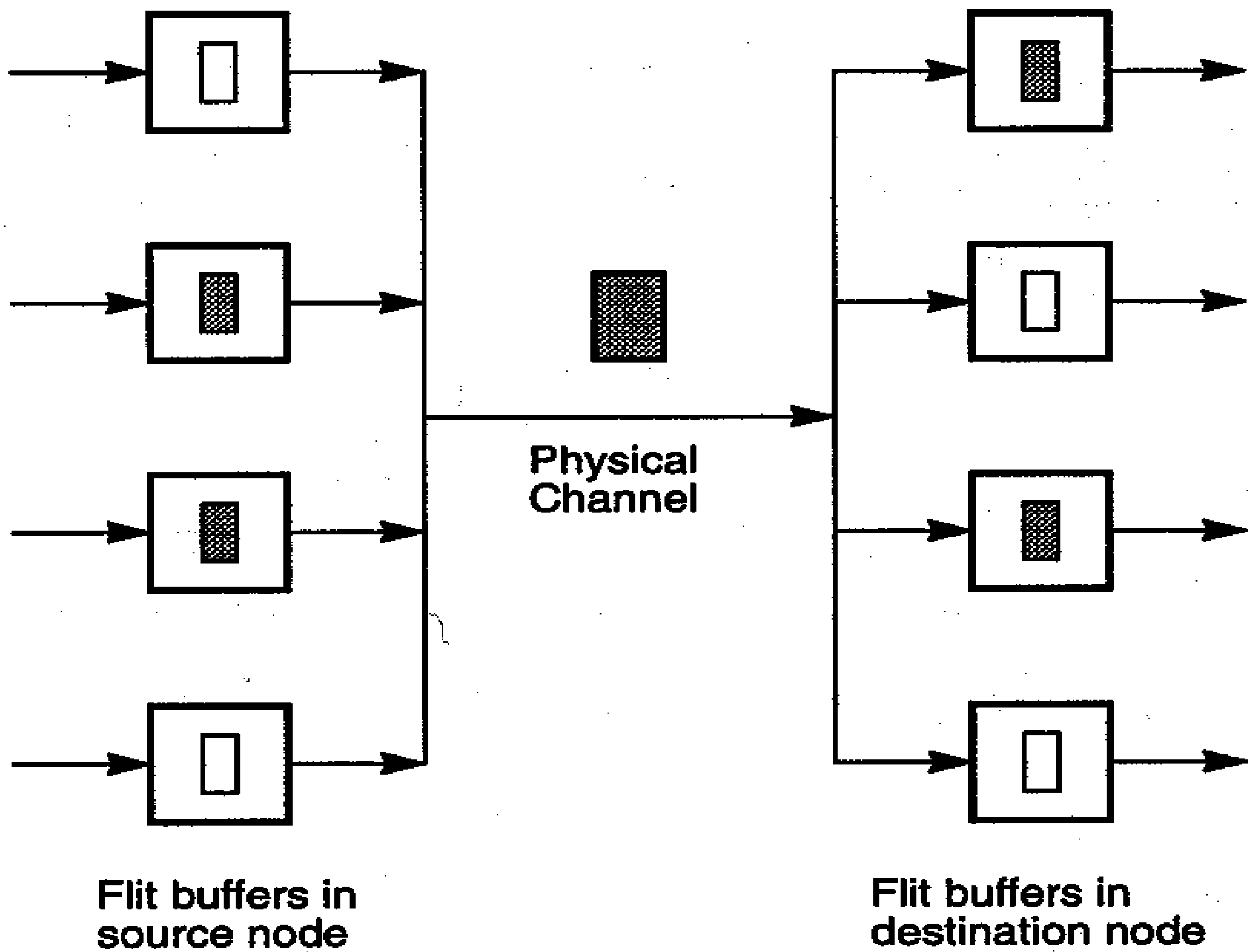
Latency

- Assume
  - D = # of intermediate nodes (routers) between the source and destination
  - L = packet length (in bits)
  - F = flit length (in bits)
  - W = the channel bandwidth (in bits/sec)
- Ignoring network startup time, propagation and resource delays:
  - store and forward latency is $L/W \times (D+1)$, and
  - wormhole latency is $L/W + F/W \times D$.
- F is usually much smaller than L, and thus D has no significant effect on latency in wormhole systems.

Virtual Channels

- The channels between nodes in a wormhole-routed multicomputer are shared by many possible source and destination pairs.
- A “virtual channel” is a pair of flit buffers (in nodes) connected by a shared physical channel.
- The physical channel is “time shared” by all the virtual channels.
- Other resources (including the R/A line) must be replicated for each of the virtual channels.

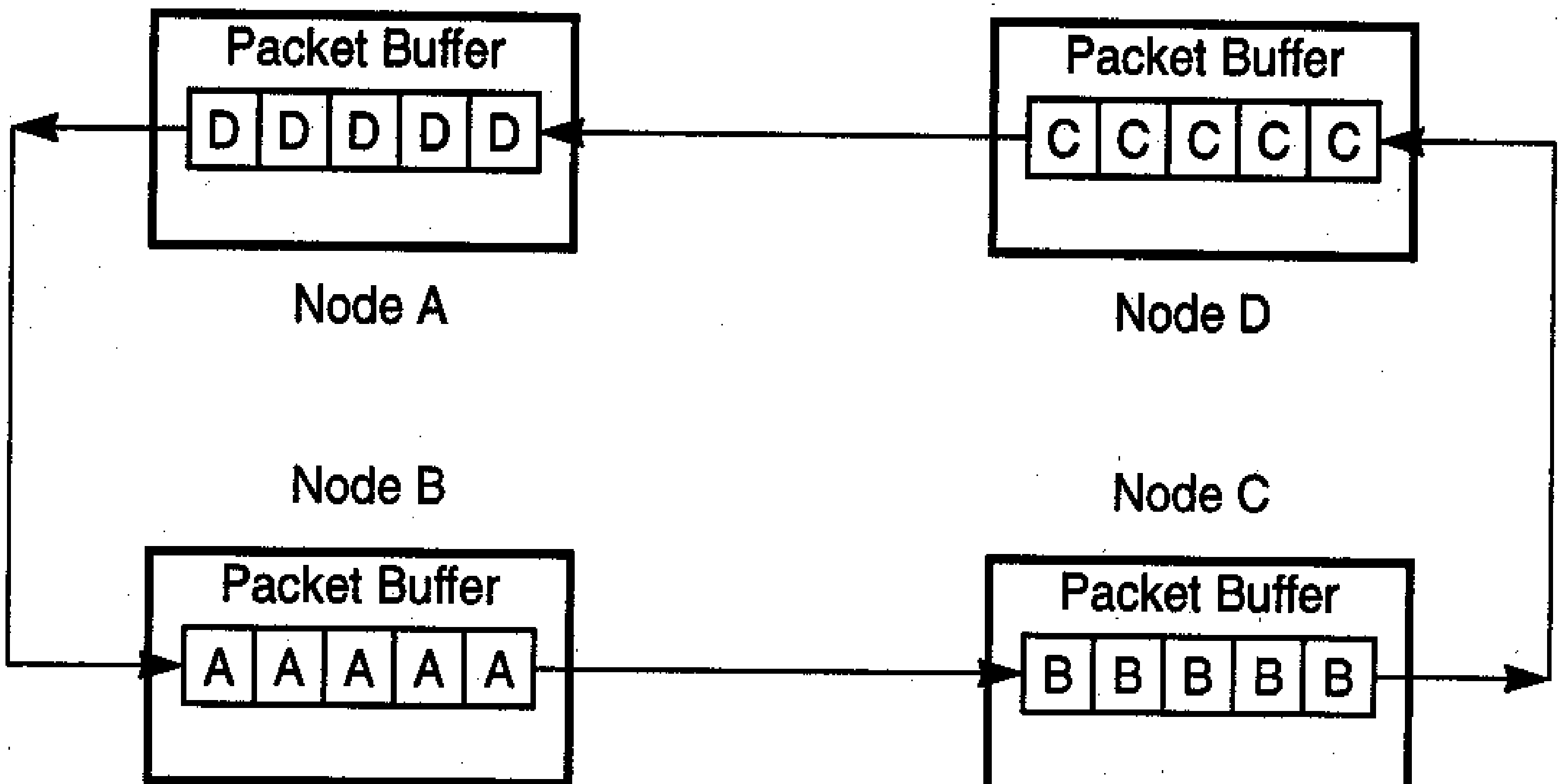
Virtual Channel Example



Deadlock

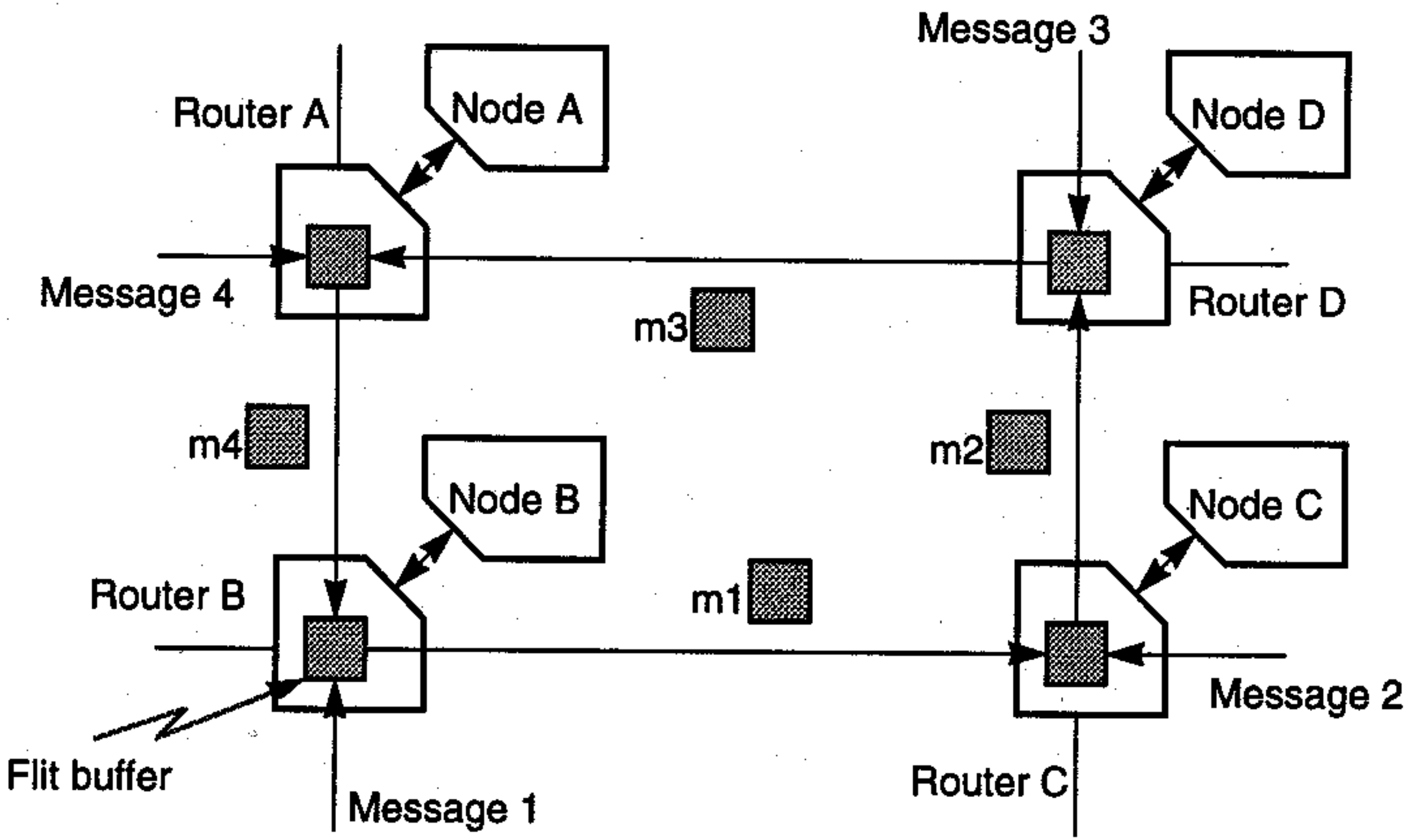
- Deadlock can occur if it is impossible for any messages to move (without discarding one).
  - Buffer deadlock occurs when all buffers are full in a store and forward network. This leads to a circular wait condition, each node waiting for space to receive the next message.
  - Channel deadlock is similar, but will result if all channels around a circular path in a wormhole-based network are busy (recall that each “node” has a single buffer used for both input and output).

Buffer Deadlock in a Store and Forward Network



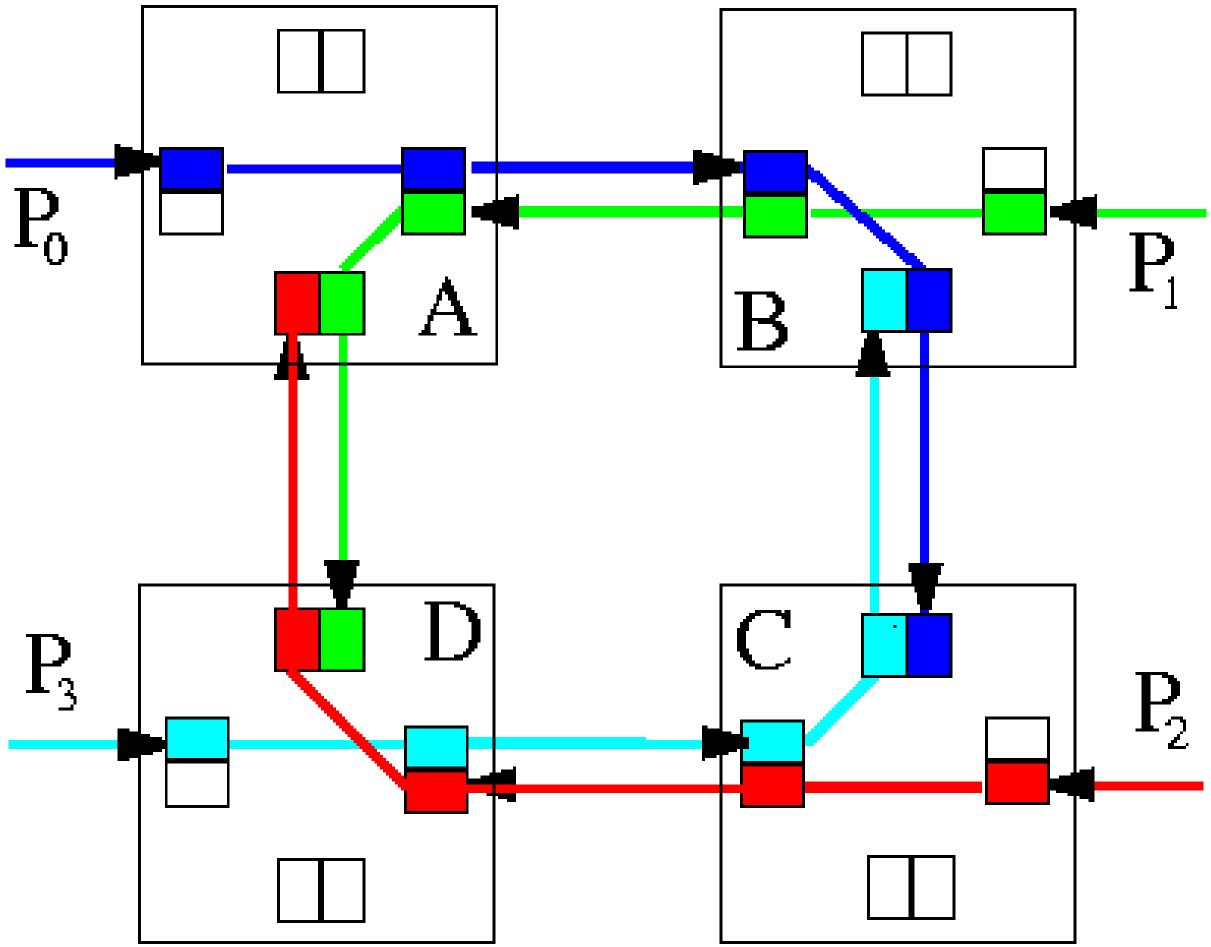
(a) Buffer deadlock among four nodes with store-and-forward routing

Channel Deadlock with Wormhole Routing



(b) Channel deadlock among four nodes with wormhole routing; shaded boxes are flit buffers

Deadlock Avoidance - New Virtual Circuits (timeshared)



Flow Control

- If multiple packets/flits demand the same resources at a given node, then there must be some policy indicating how the conflict is to be resolved.
- These policies then determine what mechanisms can be used to deal with congestion and deadlock.

Packet Collision Resolution

- Consider the case of two flits both wanting to use the same channel or the same receive buffer at the same time.
- How is the "collision" resolved? Who gets the resource? What happens to the other flit?

Virtual Cut-Through Routing

- Solution: temporarily store one of the packets in a different buffer.
- Positive:
  - No messages lost
  - Should perform as well as wormhole with no conflicts
- Negative:
  - Potentially large buffer required (with potentially large delays).
  - Not suitable for routers.
  - Cycles must be avoided

Blocking

- Solution: prevent one of the messages from advancing while the other uses the buffer/channel.
- Positive:
  - Messages are not lost.
- Negative
  - Node sending blocked packet is idled.

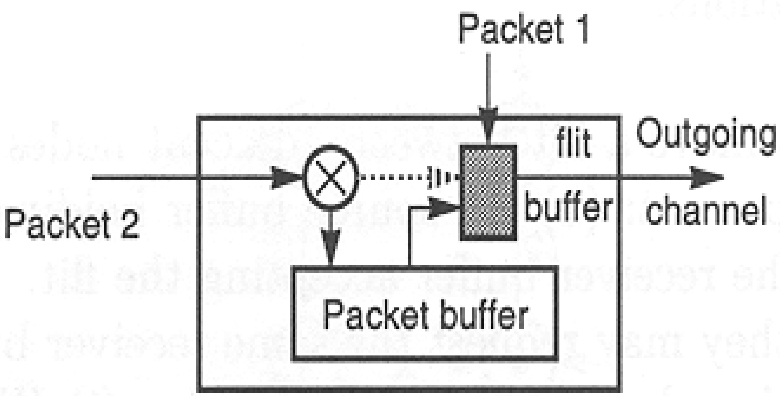
Discarding

- Solution: drop one of the messages in contention for the buffer/channel.
- Positive:
  - Simple to implement
- Negative:
  - Loses messages, resulting in a severe waste of resources.

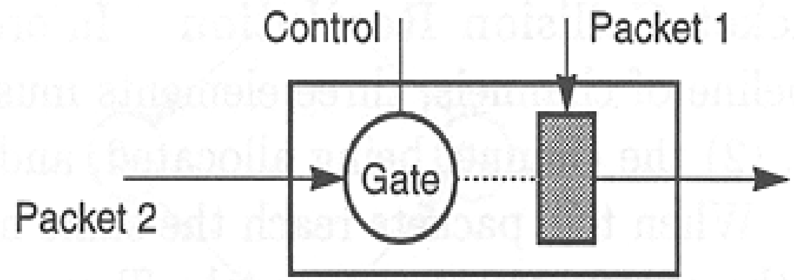
Detour

- Solution: send the conflicting message somewhere (anywhere) else.
- Positive:
  - Simple to implement
- Negative:
  - May waste more channel resource than necessary
  - May cause other resources to be idled
  - May cause livelock (e.g. four dining philosophers, with two seated across from each other conspiring to starve the other two).

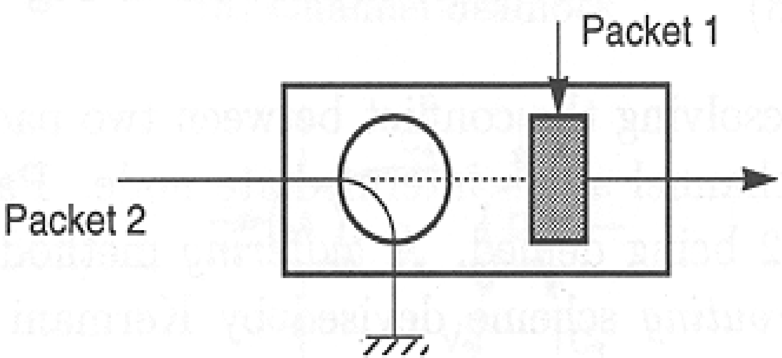
Collision Resolution Techniques



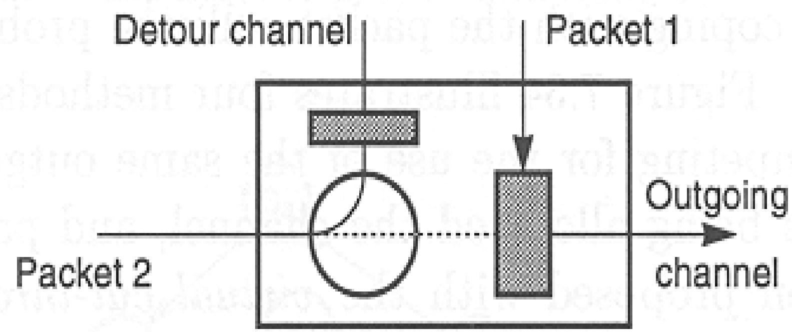
(a) Buffering in virtual cut-through routing



(b) Blocking flow control



(c) Discard and retransmission



(d) Detour after being blocked

Routing

- Deterministic routing: the path from source to destination is determined uniquely from the source and destination addresses.
- Adaptive routing: the path may depend on network conditions.

Deterministic Routing : Dimension Ordering

- Dimension ordering algorithms are based on the selection of a sequence of channels following a specified order.
- For example, routing in a two-dimensional mesh is called X-Y routing, because the X-dimension routing path is decided before choosing the Y-dimension path.
- In hypercubes, the example algorithm is called E-cube routing, and again specifies the sequence of channels to be used.

E-cube Routing on a Hypercube

- Assume the system has N = 2^n nodes; the dimensions of the hypercube are numbered 1, 2, …, n.
- Each node has a binary address with n bits (numbered n-1 to 0). The i^{th} bit in a node address corresponds to the i^{th} dimension.
- Source address = s, destination address = d.
- Algorithm:
  - Compute direction bit $r_i = s_{i-1}$ xor d_{i-1} for all dimensions. Now set $i = 1$ and $v = s$.
  - Route from the current node v to the next node v xor 2^{i-1} if $r_i = 1$; skip this step if $r_i = 0$.
  - Move to dimension $i + 1$ (i.e. $i \leftarrow i + 1$). If $i <= n$, go to the previous step.

E-cube Routing Example

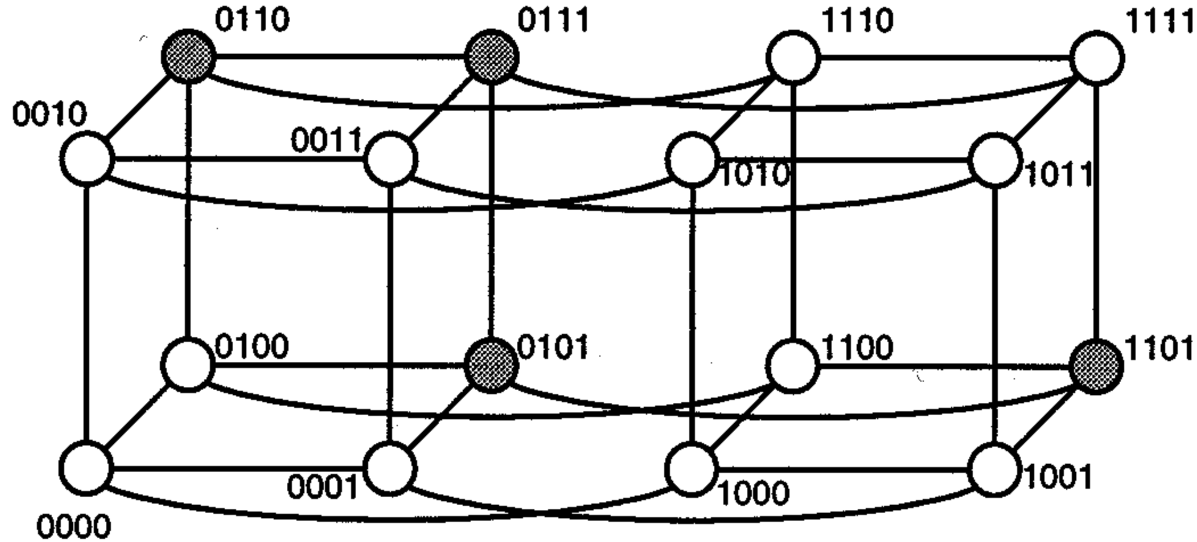
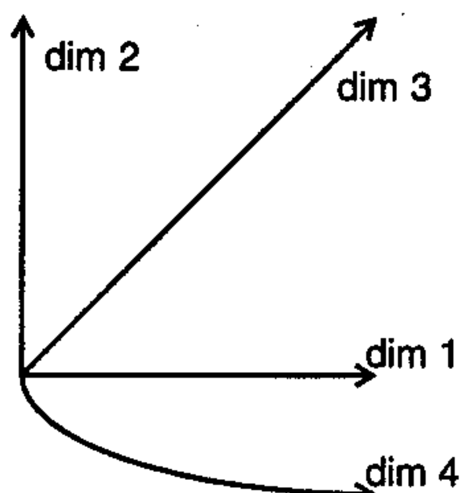


Figure 7.… 16 nodes.



Source: s=0110

Destination: d=1101

Route:
0110→0111→0101→1101

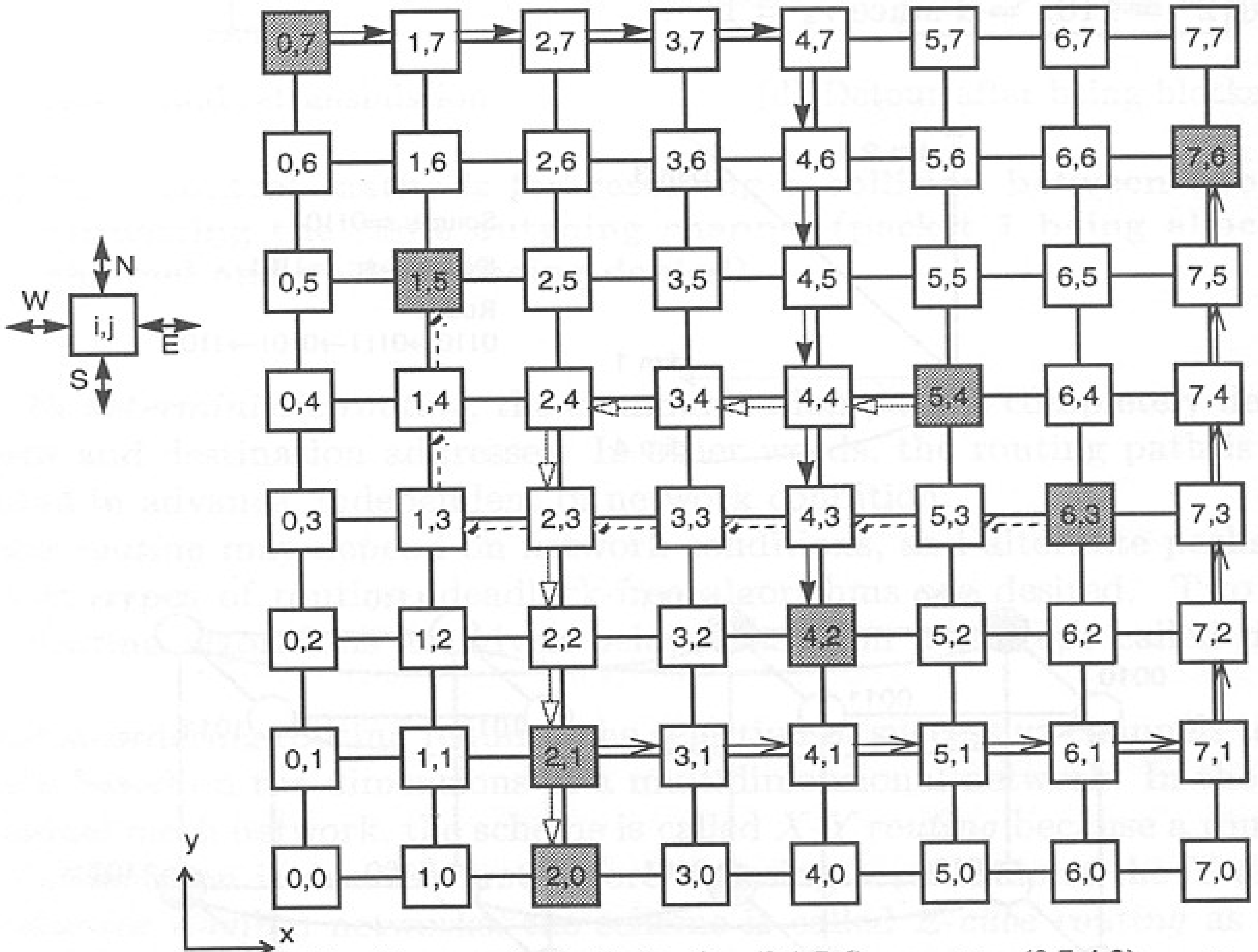
E-Cube Routing Example (Detail)

- Source Address s = 0110, n = 4 (dimension of cube)
- Destination Address d = 1101
- "Direction Bits" r = 0110 xor 1101 = 1011
- Route from 0110 to 011$\underline{1}$ because r = 101$\underline{1}$
- Route from 0111 to 01$\underline{0}$1 because r = 10$\underline{1}$1
- Skip dimension 3 because r = 1$\underline{0}$11
- Route from 0101 to $\underline{1}$101 because r = $\underline{1}$011

X-Y Routing on a 2-D Mesh

- X-Y routing is similar, in concept, to E-cube routing in that the route from the source to the destination is determined completely from their addresses.
- In X-Y routing, the message travels "horizontally" (in the X-dimension) from the source node to the "column" containing the destination, where the message travels vertically.
- There are four possible direction pairs, east-north, east-south, west-north, and west-south.

X-Y Routing Example



Four (Source; destination) pairs: (2,1;7,6) (0,7;4,2) (5,4; 2,0) (6,3; 1,5)

Dimension Ordering Characteristics

- In general, X-Y routing can be expanded to an n-dimensional mesh.

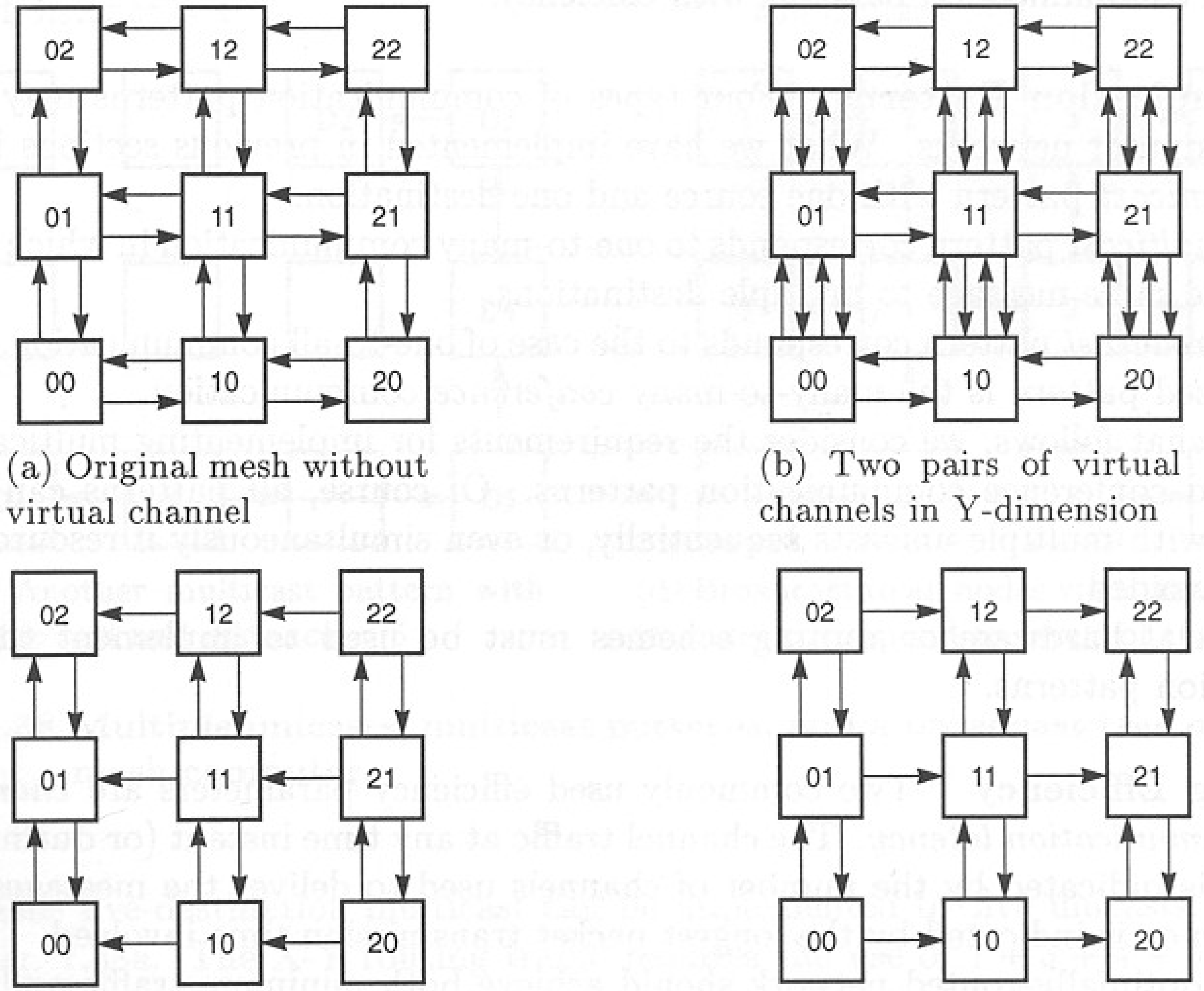
- Both X-Y routing and E-cube routing can be shown to be deadlock free.

- Both techniques can be used with store-and-forward or wormhole routing networks to produce minimal routes.

Adaptive Routing

- The main purpose of adaptive routing is to avoid deadlock.
- Adaptive routing makes use of virtual channels between nodes to make routing more economical and feasible to implement.
- Virtual channels allow the network to exhibit different characteristics at different times (that is, it "adapts").
- For example, (c) and (d) on the next slide are adaptive configurations of (a), but they prevent deadlock from occurring, since they allow only west-{north/south} routing (in c), or east-{north/south} routing (in d).

Adaptive Use of Virtual Channels to Avoid Deadlock



(a) Original mesh without virtual channel

(b) Two pairs of virtual channels in Y-dimension

(c) For a westbound message

(d) For an eastbound message

Communication Patterns

- Four possible patterns
  - Unicast – traditional one to one communication
  - Multicast – one to many communication, with one message sent to multiple destinations
  - Broadcast – one to all communication, with one message sent to every possible destination
  - Conference – many to many communication
- Note that each of these can be implemented using simple sequential transmission of messages (unicast).

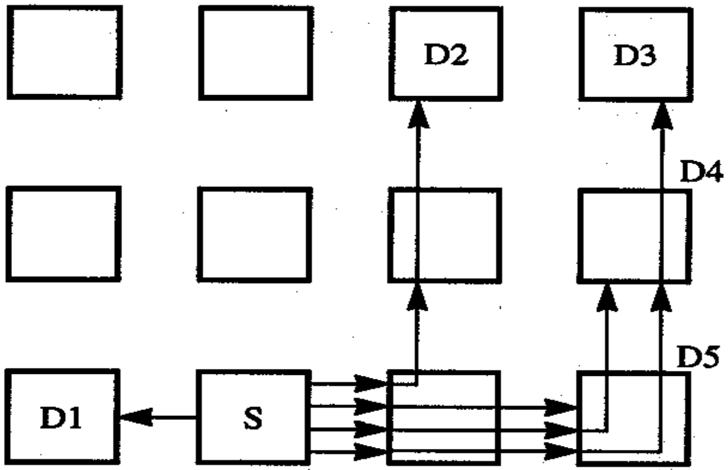
Efficiency Parameters

- Two common efficiency parameters are:
  - channel traffic – the number of channels used at any time instant to deliver messages
  - communication latency – the longest time required for any packet to reach its destination
- An optimal network would minimize both of these parameters for the communication patterns it uses.
- However, these efficiency parameters are interrelated, and achieving minimums in each may not be possible.
- Latency is more important than traffic in a store-and-forward network.
- Traffic demand is more important than latency in a wormhole-routed network.

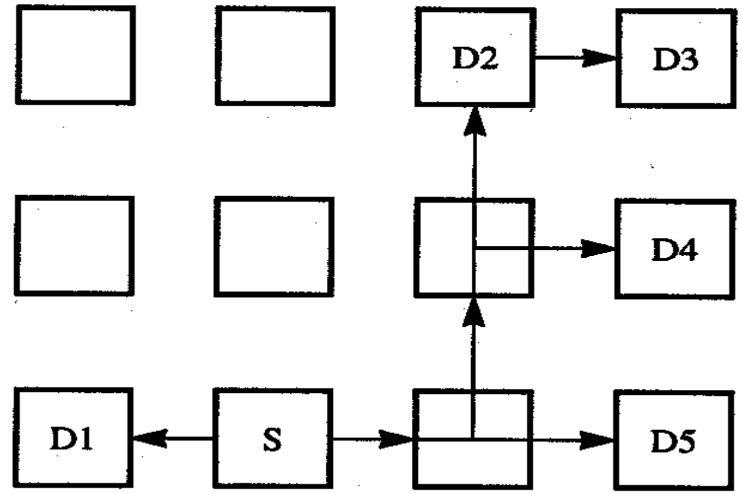
Example 5-Destination Multicast

- (a) Five unicasts, with traffic demand = 13 and latency = 4 (assuming one "hop" per unit time).
- (b) Tree multicast with branching at multiple levels, with traffic demand = 7 and latency = 4.
- (c) Tree multicast with only one branching node, with traffic demand = 6 and latency = 5.
- (d) Broadcast to all nodes with spanning tree.

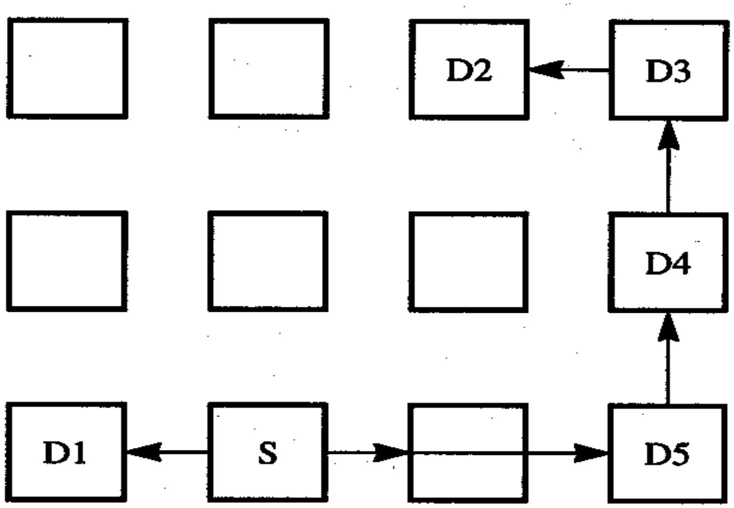
Multicast & Broadcast Patterns



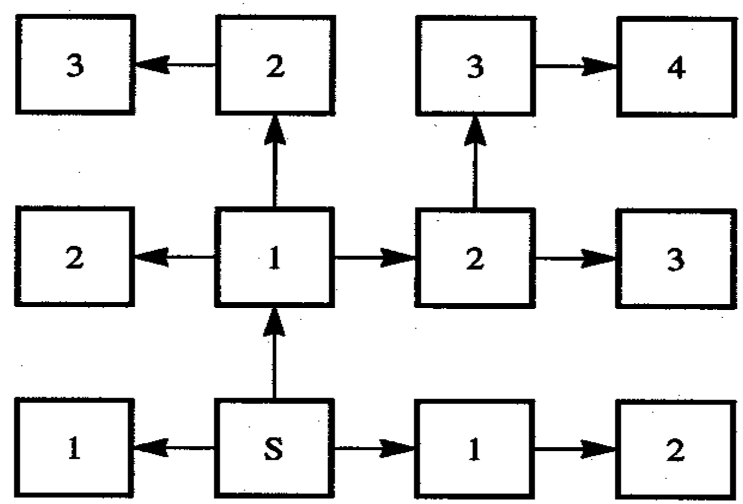
(a) Five unicasts with traffic = 13 and distance = 4



(b) A multicast pattern with traffic = 7 and distance = 4



(c) Another multicast pattern with traffic = 6 and distance = 5

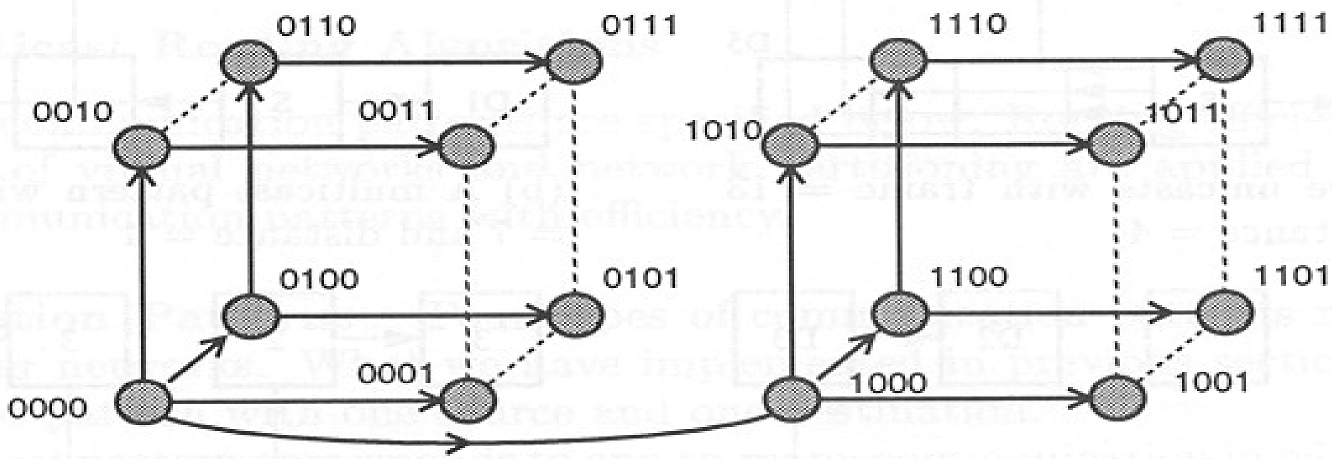


(d) Broadcast to all nodes via a tree (numbers in nodes correspond to levels of the tree)

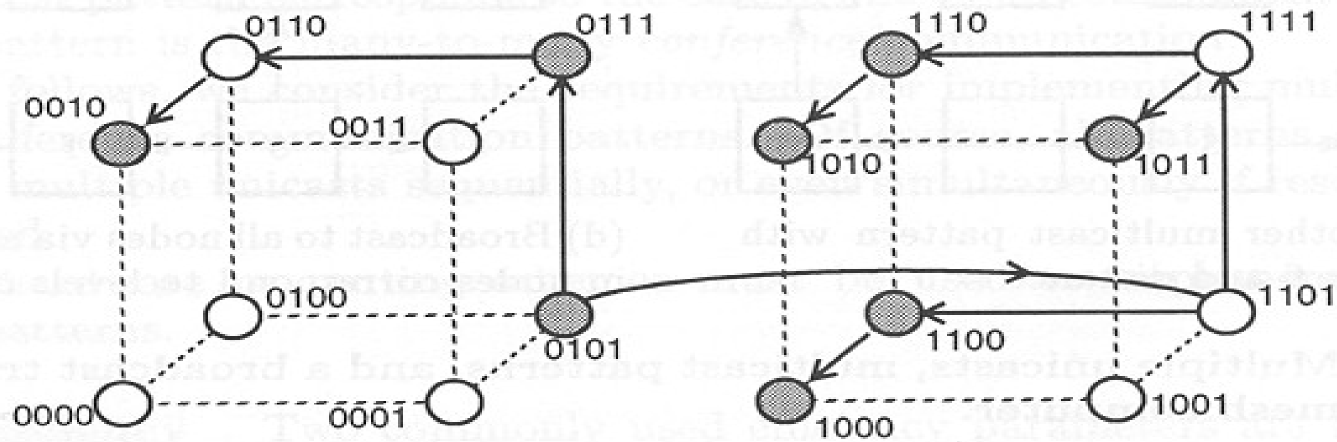
Hypercube Multicast/Broadcast

- Broadcast on a hypercube of dimension n will have a latency not exceeding n.
- A greedy algorithm for building a tree selects, at each node, the nodes in dimensions that will reach the largest number of remaining destinations (e.g. find the minimum cover set).
- In the event of a tie, any of the tied dimensions can be selected (which means the resulting tree is not necessarily unique).
- Note that all communication channels at each level of the multicast/broadcast tree must be ready at the same time, or else additional buffering might be required.

Broadcast & Multicast on Hypercube



(a) Broadcast tree for a 4-cube rooted at node 0000



(b) A multicast tree from node 0101 to seven destination nodes 1100, 0111, 1010, 1110, 1011, 1000, and 0010

Virtual Networks

- With multiple virtual channels between nodes, it is possible to dynamically reconfigure a network into one of perhaps many different "virtual networks."
- The advantages of having many such virtual networks are
  - routing needs can be used to tailor networks that yield results with simple and efficient routing algorithms
  - deadlock can be completely eliminated (e.g. by not allowing cycles to exist in the virtual network)
- Of course, adding channels to the network will increase the cost

Network Partitioning

- Another benefit of having virtual channels between nodes is the ability to dynamically partition a network into multiple subnetworks for multicast communication.
- Each subnet can carry a different multicast message at the same time.