Deadlock and Livelock

Deadlock (When?)

- Deadlock can occur in an interconnection network, when a group of packets cannot make progress, because they are waiting on each other to release resource (buffers, channels)
- If a sequence of waiting agents form a cycle the network is deadlocked

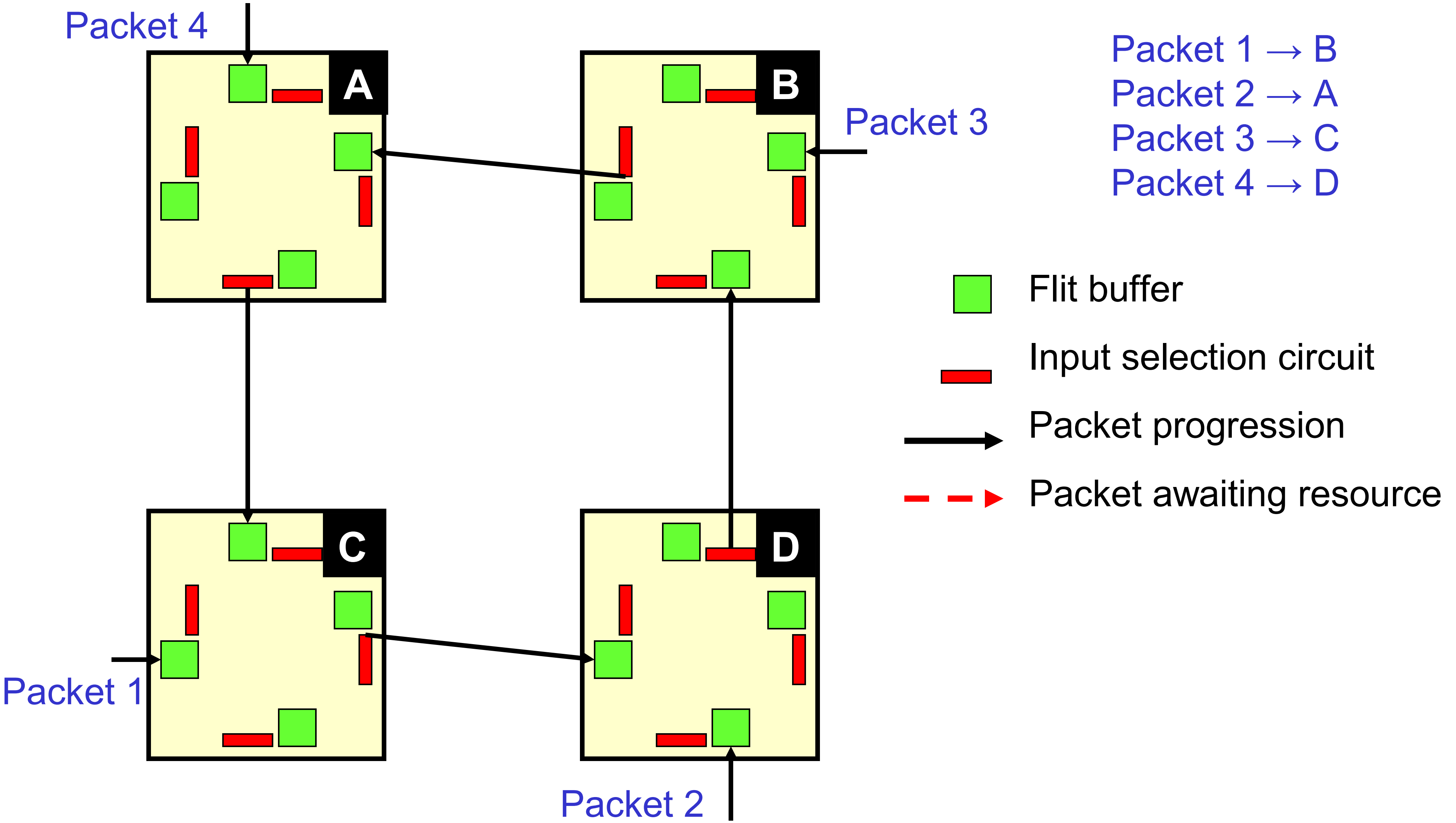
Deadlock (Why?)

- Deadlock can occur if packets are allowed to hold some resources while requesting others

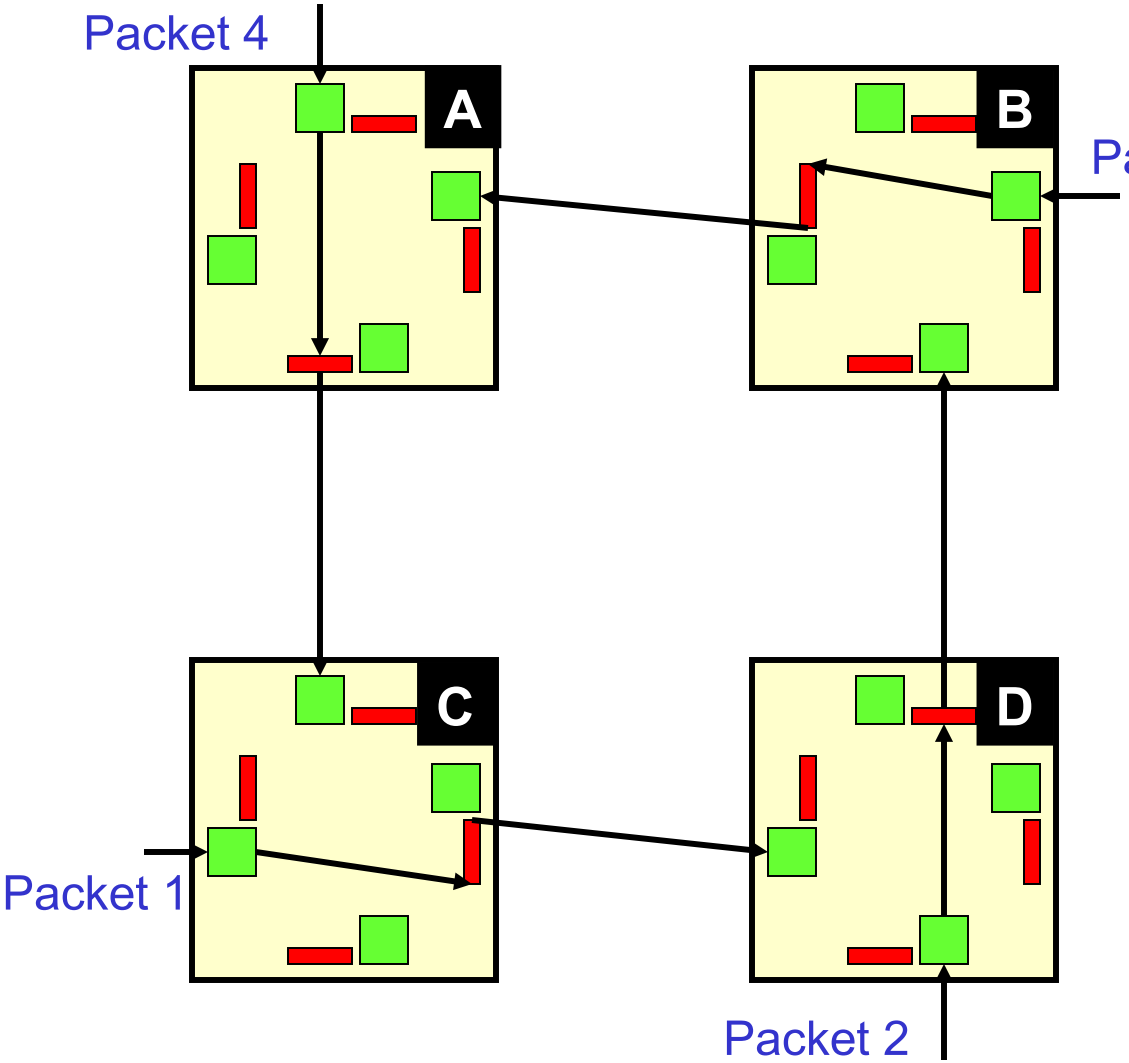
Switching Technique	Resource
Store and Forward	Buffer
Virtual Cut-Through	Buffer
Wormhole	Channel

- Because blocked packets holding channels (and their corrisponding flit buffers) remain in the network
  - ➔ Wormhole routing is particularly susceptible to deadlock

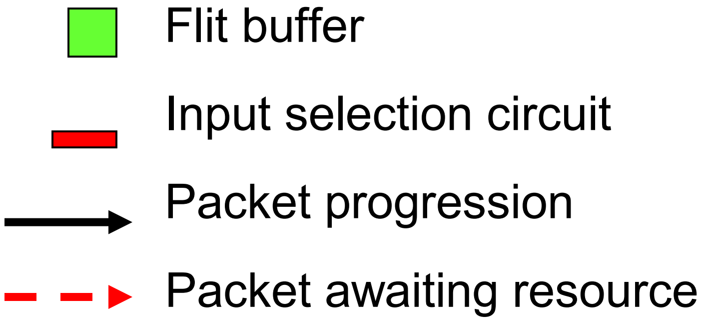
Wormhole Routing Deadlock Example



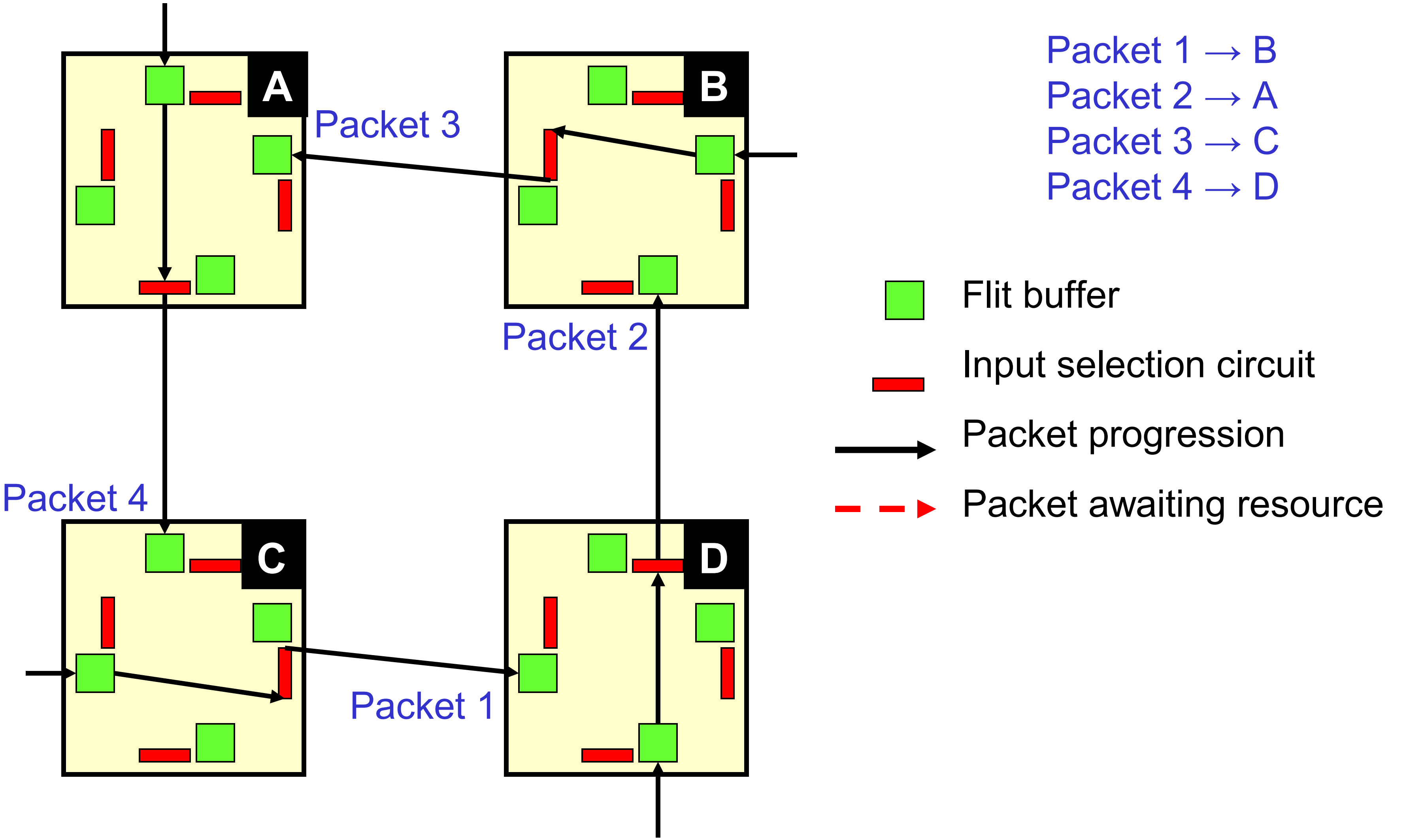
Wormhole Routing Deadlock Example



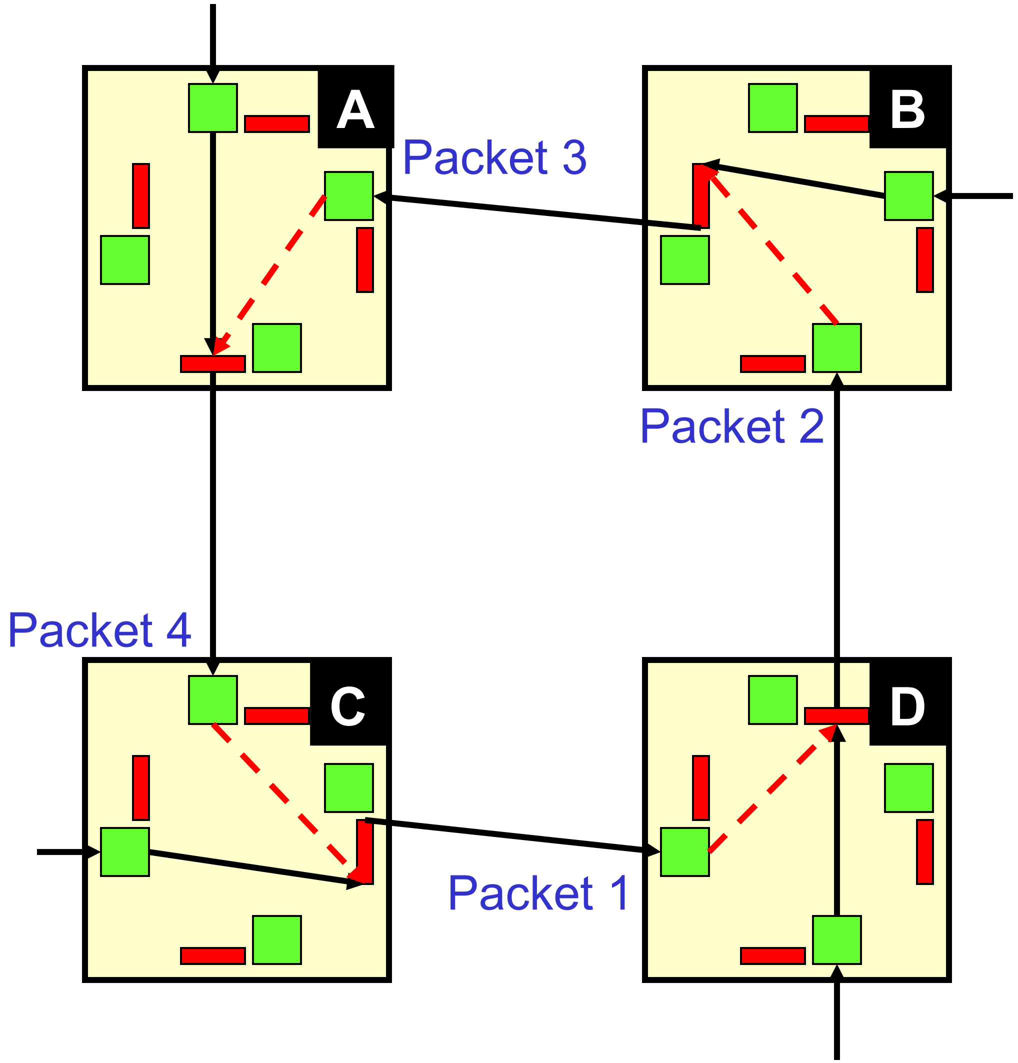
Packet 1 → B
Packet 2 → A
Packet 3 → C
Packet 4 → D



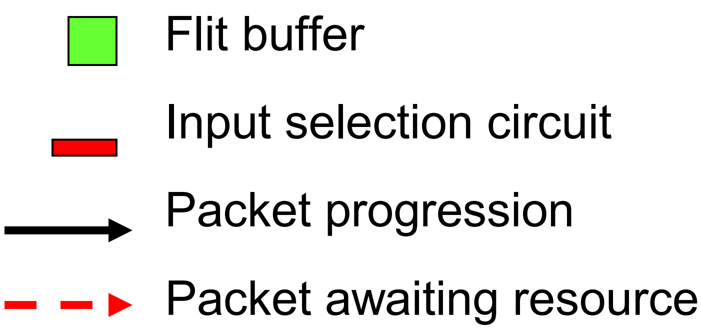
Wormhole Routing Deadlock Example



Wormhole Routing Deadlock Example



Packet 1 → B
Packet 2 → A
Packet 3 → C
Packet 4 → D



How to Solve Deadlock Problems

- Allow the preemption of packets involved in a potential deadlock situation
- Preemption packets can be
  - Rerouted
    - Adaptive nonminimal routing techniques
  - Discarded
    - Packets recovered at the source and restransmitted
- Not used in most direct networks architectures
  - Requirements of low-latency and reliability

How to Solve Deadlock Problems

- More commonly, deadlock is avoided by the routing algorithm
  - ➔ By ordering network resources and requiring that packets use these resources in strictly monotonic order
    - ✓ Circular wait is avoided

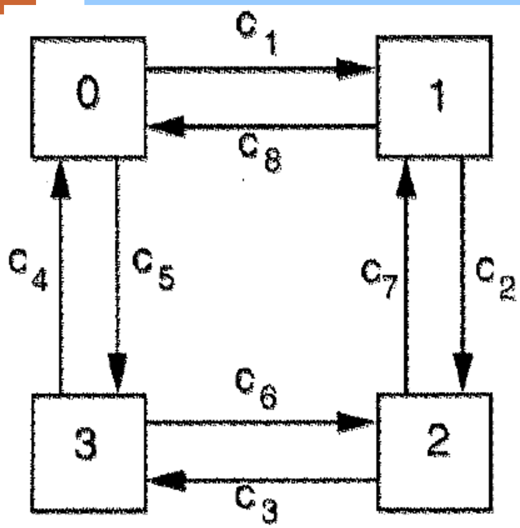
Channel Dependecy Graph

- In wormhole routing networks channels are the critical resources
- There is a *Direct Dependency* from l_i to l_j if l_j can be used immediately after l_i by messages destined to some node *n*
- The *Channel Dependency Graph* for a network and a routing algorithm is a direct graph $D=G(L,D)$
  - ➔ L consists of all the unidirectional channels in the network
  - ➔ D includes the pairs (l_i, l_j) if there is a direct dependency from l_i to l_j

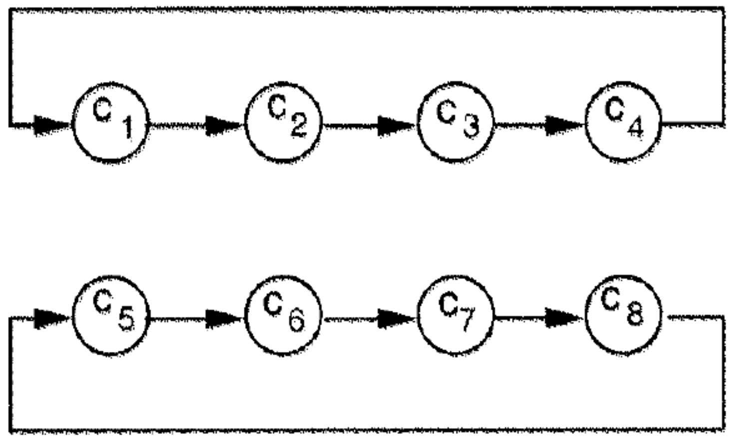
Duato's Theorem

- A routing function R is *deadlock-free* if there are no cycles in its channel dependency graph

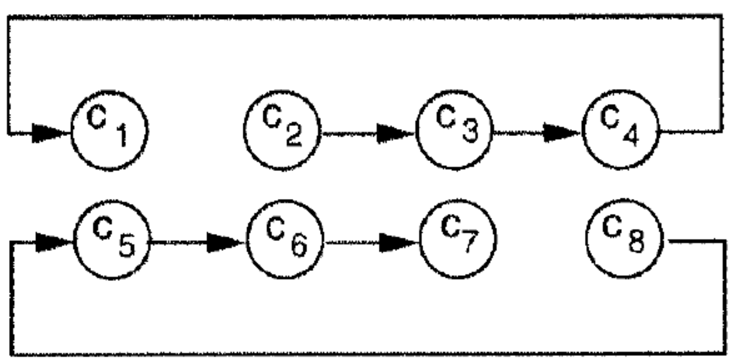
Channel Dependency Graph Method



(a)



(b)



(c)

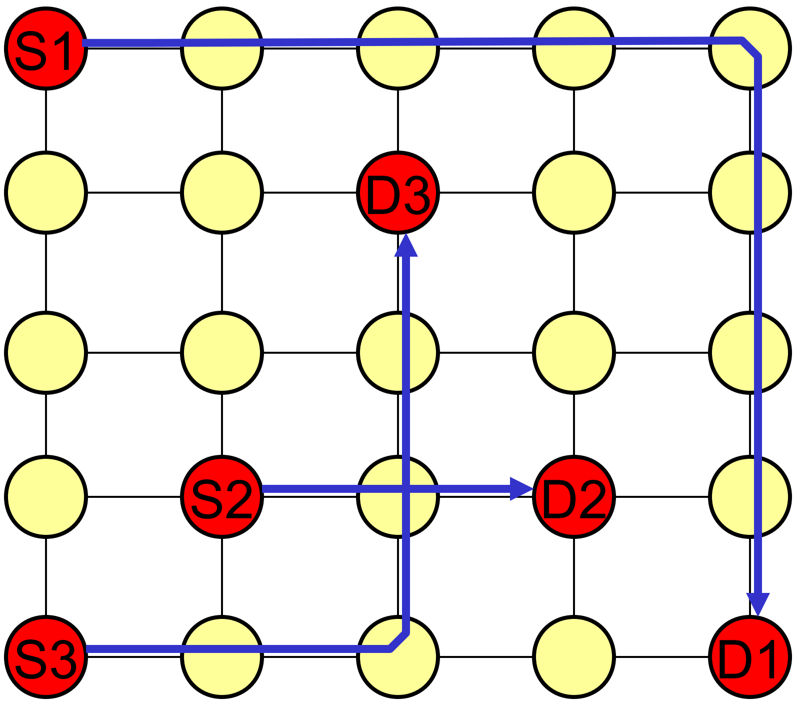
- The routing is still minimal
  - ➔ However, to send a packet from 0 to 2, the packet **must** be forwarded through node 3

Deterministic Routing

- An approach to designing a deadlock-free routing algorithm for a wormhole-routed network is to ensure that cycles are avoided in the channel dependency graph
  - Assign a unique number to each channel
  - Allocate channels to packets in strictly ascending (or descending) order
- If the behavior of the algorithm is independent of current network conditions
  - Deterministic routing

Dimension-ordered Routing

- Each packet is routed in one dimension at a time
- Arriving at the proper coordinate in each dimension before proceeding to the next dimension
- Enforcing a strictly monotonic order on the dimension traversed



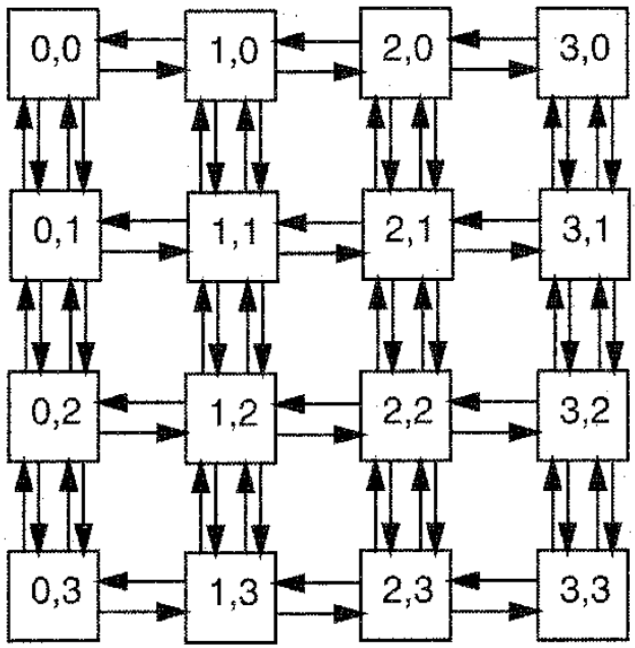
Adaptive Routing

- The main problem of a deterministic routing is that it cannot respond to dynamic network conditions
  - Congestion
  - Faults
- Adaptive routing must address deadlock issue

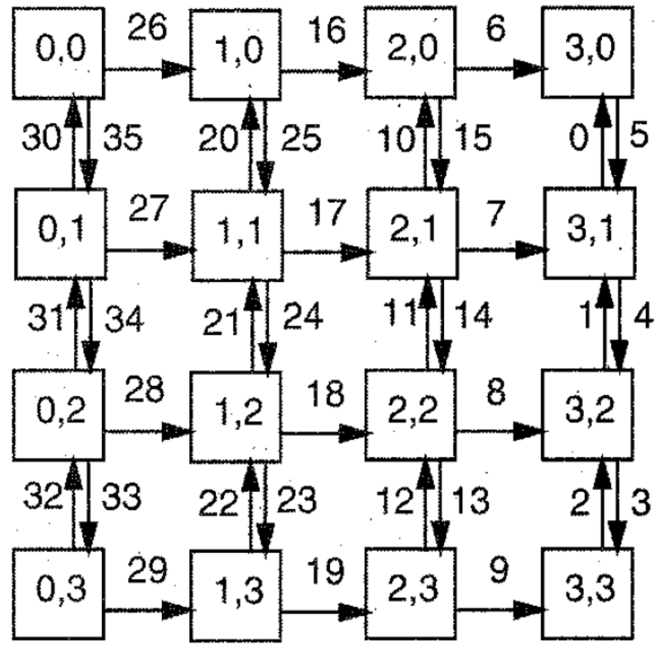
Minimal Adaptive Routing

- Partitioning of the channels into disjoint subsets
  - Each subset constituite a corresponding subnetwork
  - Packets are routed through different subnetworks
    - Depending on the location of destination nodes

Minimal Adaptive Routing



Double Y-channel 2D mesh



+X subnetwork and labeling

- If the destination node is to the right (left) of the source ➔ The packet will be routed through the +X (-X) subnetwork

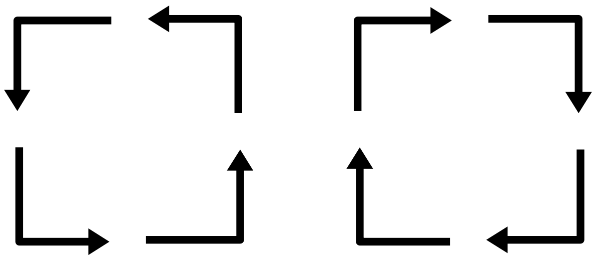
Minimal Adaptive Routing

- Has been demonstrated that a *k*-ary *n*-cube can be partitioned into 2^{n-1} subnetwork, and k^n channels per level
  - ➔The number of additional channels increases rapidly with *n*
- While this approach does provide minimal fully adaptive routing, the cost associated with the additional channels makes it impratical when *n* is large

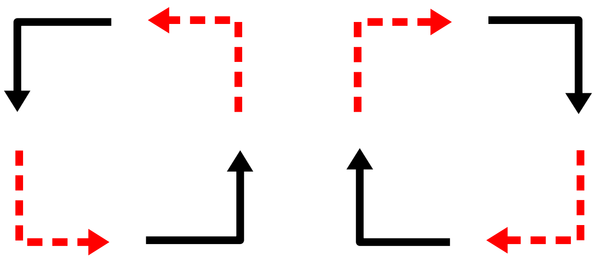
The Turn Model

- The *Turn Model* provide a systematic approach to the development of maximally adaptive routing algorithms
  1. Classify channels according to the direction in which they route packets
  2. Identify the turns that occur between one direction and another
  3. Identify the simple cycles these turns can forms
  4. Prohibit one turn in each cycle

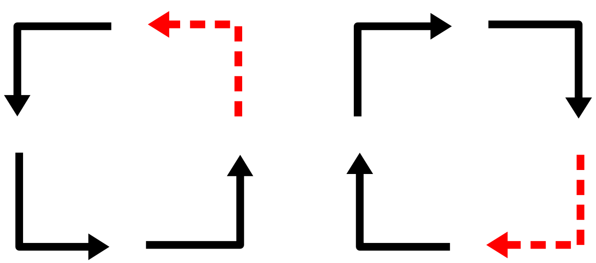
The Turn Model



Abstract cycles in a 2D mesh

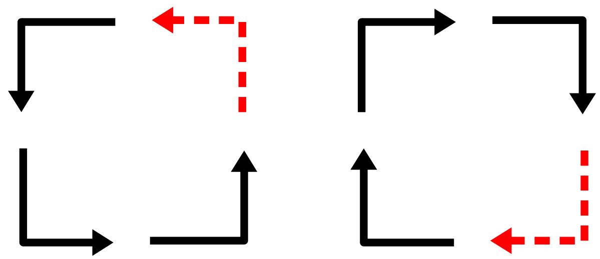


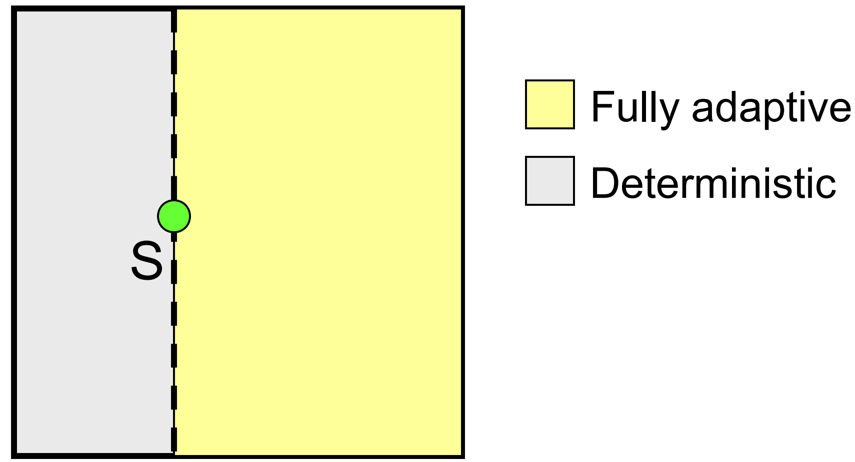
Four turns allowed in XY routing



Six turns allowed in *west-first* routing

West-First Routing

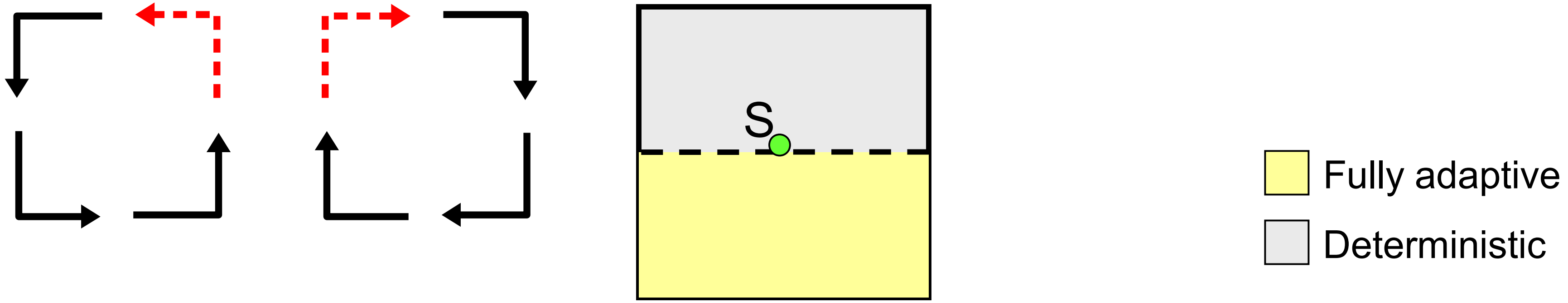




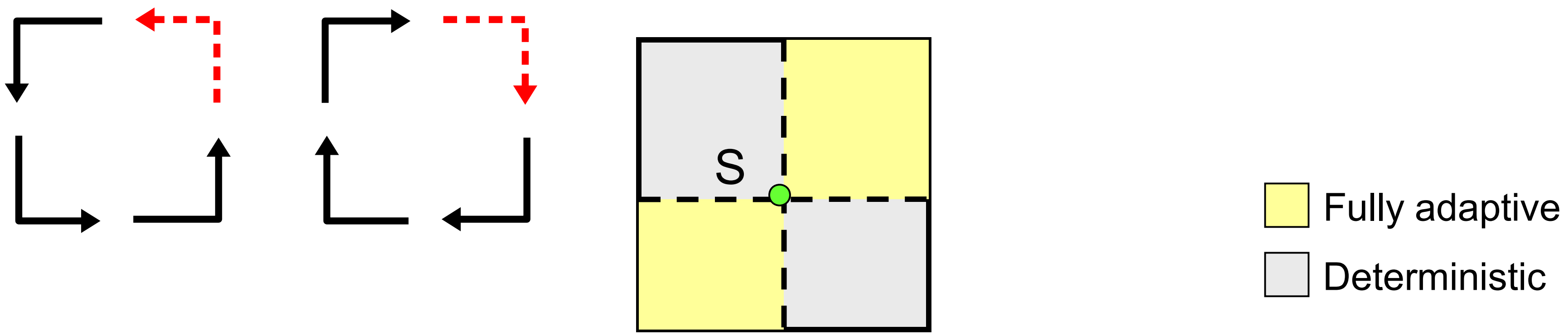
- First route a packet west, if necessary, and then adaptively south, east, and north
  - The algorithm is fully adaptive if the destination is on the right-hand side (east) of the source
    - Otherwise it is deterministic

Routing Algorithms based on Turn Model

North-last



Negative-first



Routing in Reconfigurable Networks

- Routing algorithms have been developed for each type of network topology
- If the topology change ➔ Routing algorithm has to change
- The router must be flexible and programmable to allow for the implementation of different deadlock-free algorithms
- Two techniques
  - ➔ Source routing
  - ➔ Table-lookup routing

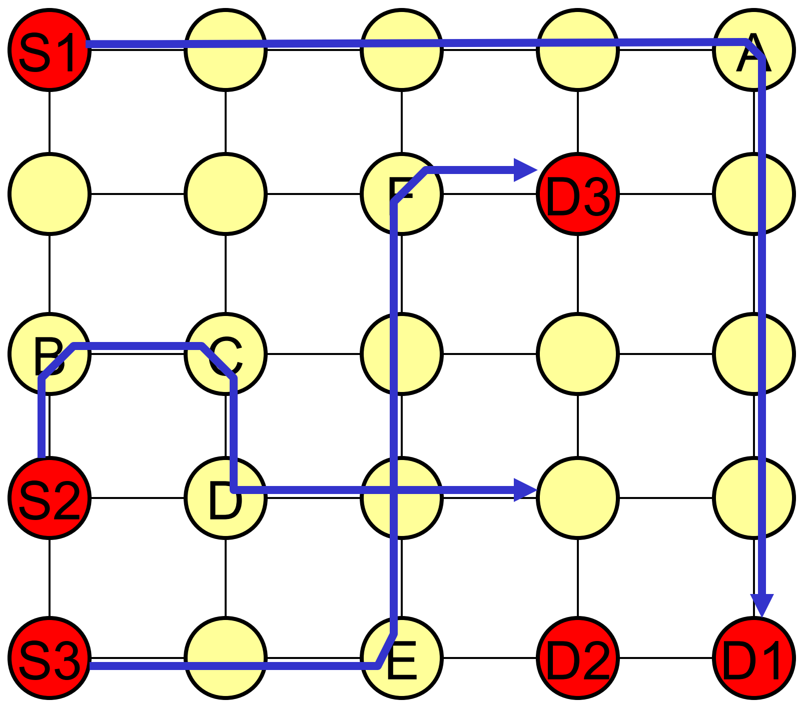
Source Routing

- The source node specifies the routing path
- The packet must carry complete routing information in the packet header
  - ➔ It is important to minimize packet length
- Street-sign routing
  - ➔ $[<Node_1,Turn_1>,<Node_2,Turn_2>,\ldots]$
  - ➔ By default packets arriving from the input channel +X (+Y) will be forwarded to -X (-Y)
  - ➔ If current node is $Node_i$ then turn $Turn_i$ is taken and $<Node_2,Turn_2>$ is removed from the header

Source Routing

[<A,-Y>]

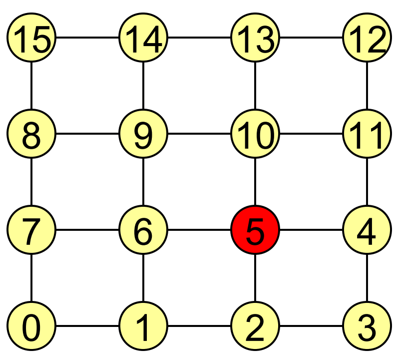
[<B,+X>;
<C,-Y>;
<D;+X]



[<E,+Y>;<F,+X>]

Table-lookup Routing

- Amenable to reconfigurable topologies
- An obvious implementation
  - ➔ Given a destination node address the corresponding entry in the table indicates which outgoing channel should be used to forward the packet
  - ➔ Not practical
    - ✓ Large table is inefficient in the use of chip area

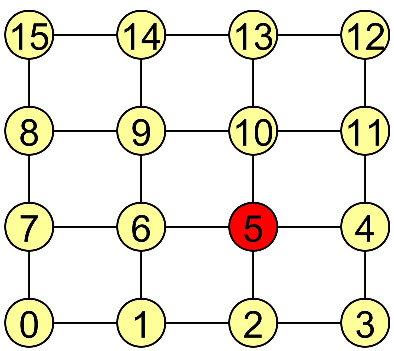


Routing table for node 5

Dst	Out
0	West
1	West
2	South
3	East
4	East
5	-
6	West
7	West
8	West
9	West
10	North
11	East
12	East
13	North
14	West
15	West

Table-lookup Routing

- One way to reduce the table size is to define a *range of addresses* to be associated with each routing channel



Routing table for node 5

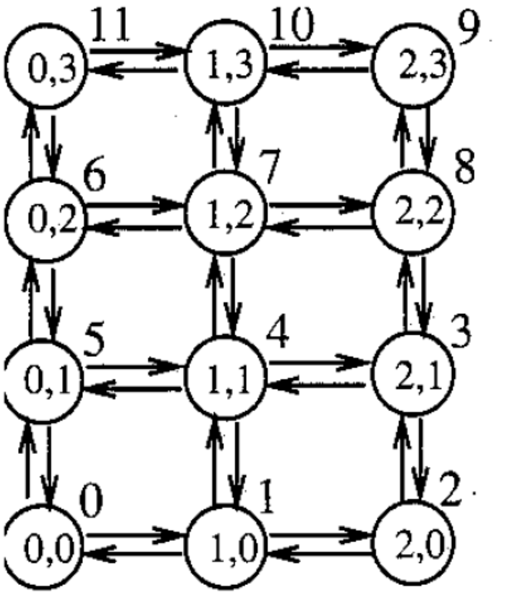
Dst	Outs
>=10	North
[6,10[	West
]2,4]	East
<=2	South

Table-lookup Routing

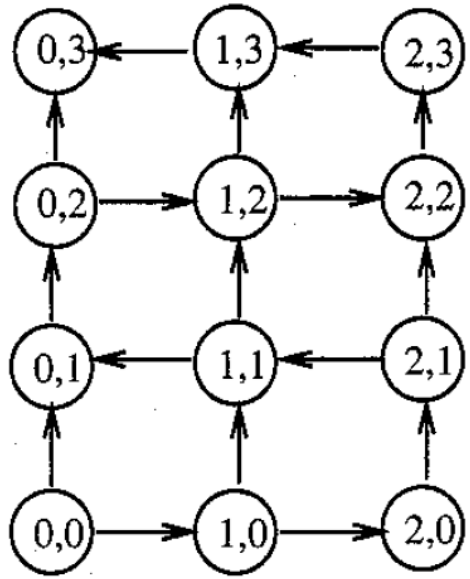
- How to assign appropriate labels to nodes so that minimal deadlock-free routing results?
  - ➔For 2D mesh topology with N nodes, use a particular Hamiltonian path of the network graph
- For a *m*×*n* mesh

$$l(x,y) = \begin{cases} y*n+x & \text{if } y \text{ is even} \\ y*n+n-x-1 & \text{if } y \text{ is odd} \end{cases}$$

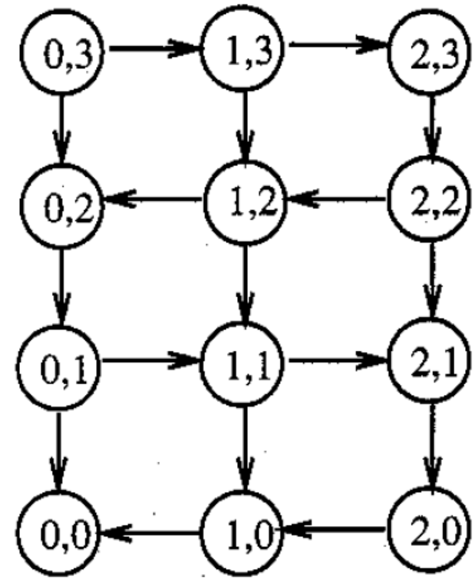
Table-lookup Routing



Physical network



High-channel network



Low-channel network

- This labeling divides the netwok into two subnetworks
  - ➔ Both subnetworks are acyclic ➔ Table-lookup routing algorithm is deadlock free