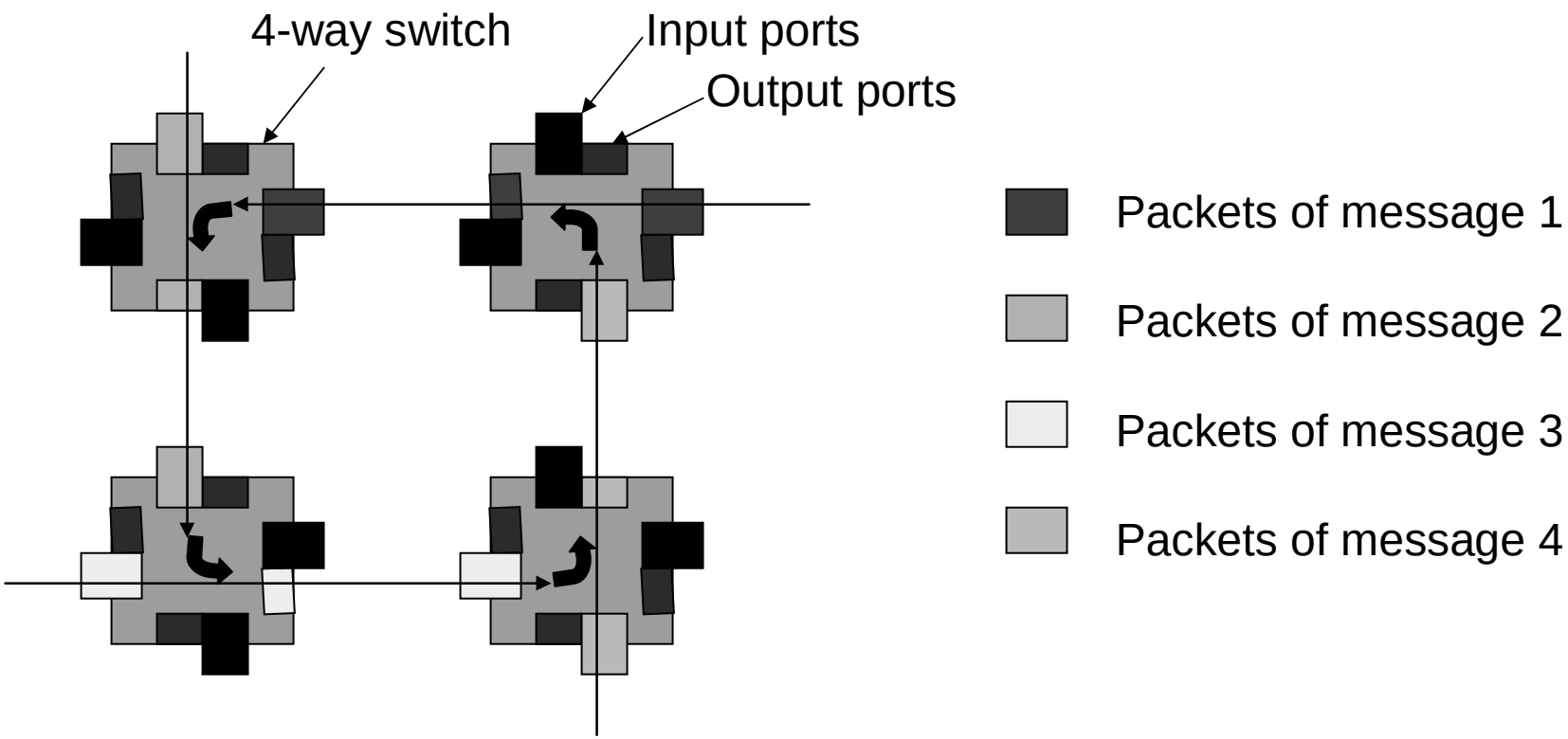
Lecture 15: Interconnection Routing

- Topics: deadlock, flow control

Deadlock

- Deadlock happens when there is a cycle of resource dependencies – a process holds on to a resource (A) and attempts to acquire another resource (B) – A is not relinquished until B is acquired

Deadlock Example

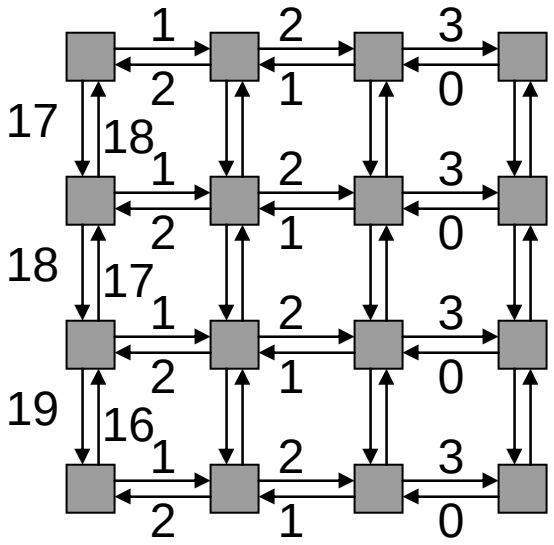


Each message is attempting to make a left turn – it must acquire an output port, while still holding on to a series of input and output ports

Deadlock-Free Proofs

- Number edges and show that all routes will traverse edges in increasing (or decreasing) order – therefore, it will be impossible to have cyclic dependencies

- Example: k-ary 2-d array with dimension routing: first route along x-dimension, then along y

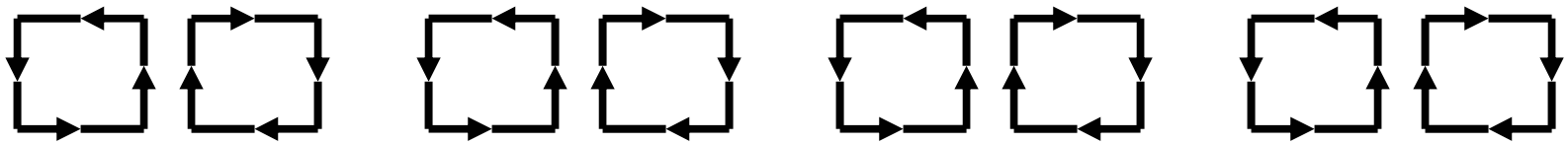


Breaking Deadlock I

- The earlier proof does not apply to tori because of wraparound edges
- Partition resources across multiple virtual channels
- If a wraparound edge must be used in a torus, travel on virtual channel 1, else travel on virtual channel 0

Breaking Deadlock II

- Consider the eight possible turns in a 2-d array (note that turns lead to cycles)
- By preventing just two turns, cycles can be eliminated
- Dimension-order routing disallows four turns
- Helps avoid deadlock even in adaptive routing



West-First | North-Last | Negative-First | Can allow deadlocks

Packets/Flits

- A message is broken into multiple packets (each packet has header information that allows the receiver to re-construct the original message)
- A packet may itself be broken into flits – flits do not contain additional headers
- Two packets can follow different paths to the destination Flits are always ordered and follow the same path
- Such an architecture allows the use of a large packet size (low header overhead) and yet allows fine-grained resource allocation on a per-flit basis

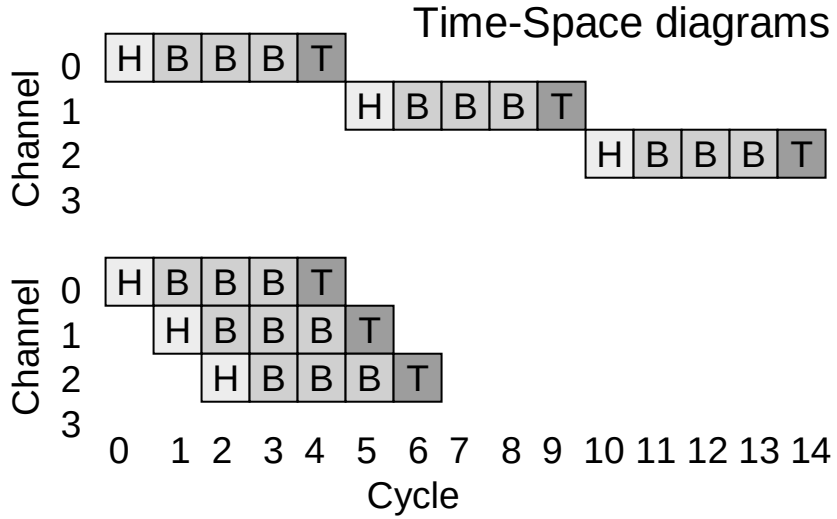
Flow Control

- The routing of a message requires allocation of various resources: the channel (or link), buffers, control state
- Bufferless: flits are dropped if there is contention for a link, NACKs are sent back, and the original sender has to re-transmit the packet
- Circuit switching: a request is first sent to reserve the channels, the request may be held at an intermediate router until the channel is available (hence, not truly bufferless), ACKs are sent back, and subsequent packets/flits are routed with little effort (good for bulk transfers)

Buffered Flow Control

- A buffer between two channels decouples the resource allocation for each channel – buffer storage is not as precious a resource as the channel (perhaps, not so true for on-chip networks)

- Packet-buffer flow control: channels and buffers are allocated per packet
  - Store-and-forward
  - Cut-through



Flit-Buffer Flow Control (Wormhole)

- Wormhole Flow Control: just like cut-through, but with buffers allocated per flit (not channel)

- A head flit must acquire three resources at the next switch before being forwarded:
  - channel control state (virtual channel, one per input port)
  - one flit buffer
  - one flit of channel bandwidth

  The other flits adopt the same virtual channel as the head and only compete for the buffer and physical channel

- Consumes much less buffer space than cut-through routing – does not improve channel utilization as another packet cannot cut in (only one VC per input port)

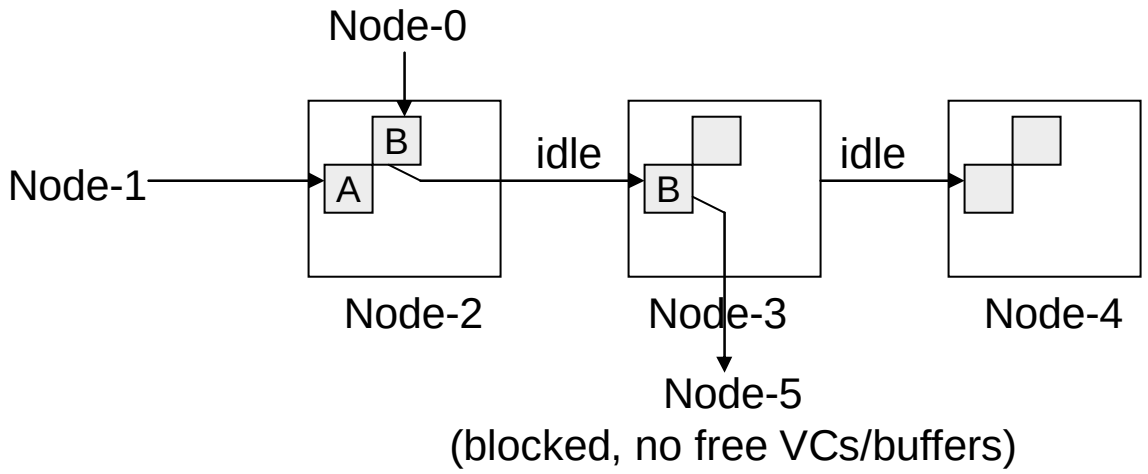
Virtual Channel Flow Control

- Each switch has multiple virtual channels per phys. channel
- Each virtual channel keeps track of the output channel assigned to the head, and pointers to buffered packets
- A head flit must allocate the same three resources in the next switch before being forwarded
- By having multiple virtual channels per physical channel, two different packets are allowed to utilize the channel and not waste the resource when one packet is idle

Example

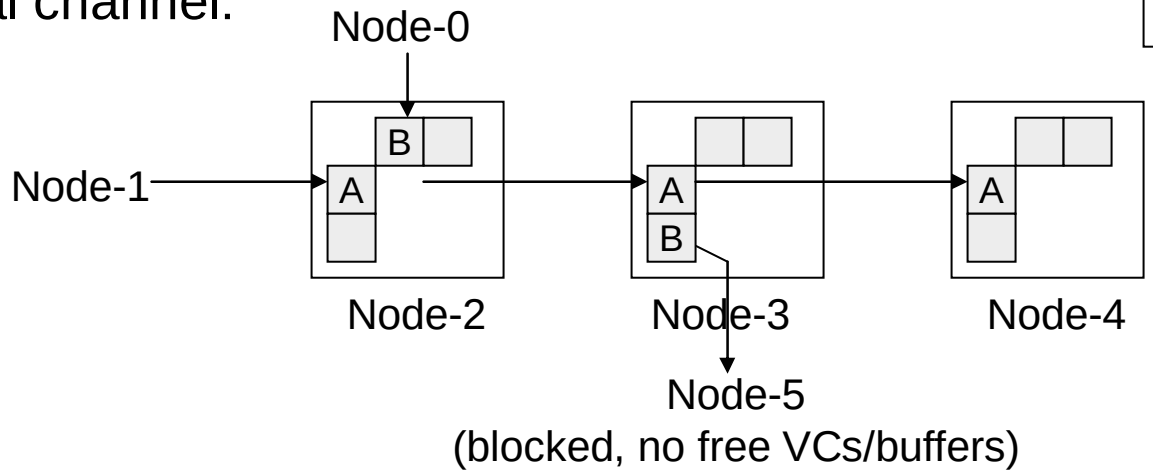
- Wormhole:

A is going from Node-1 to Node-4; B is going from Node-0 to Node-5



Traffic Analogy:
B is trying to make a left turn; A is trying to go straight; there is no left-only lane with wormhole, but there is one with VC

- Virtual channel:



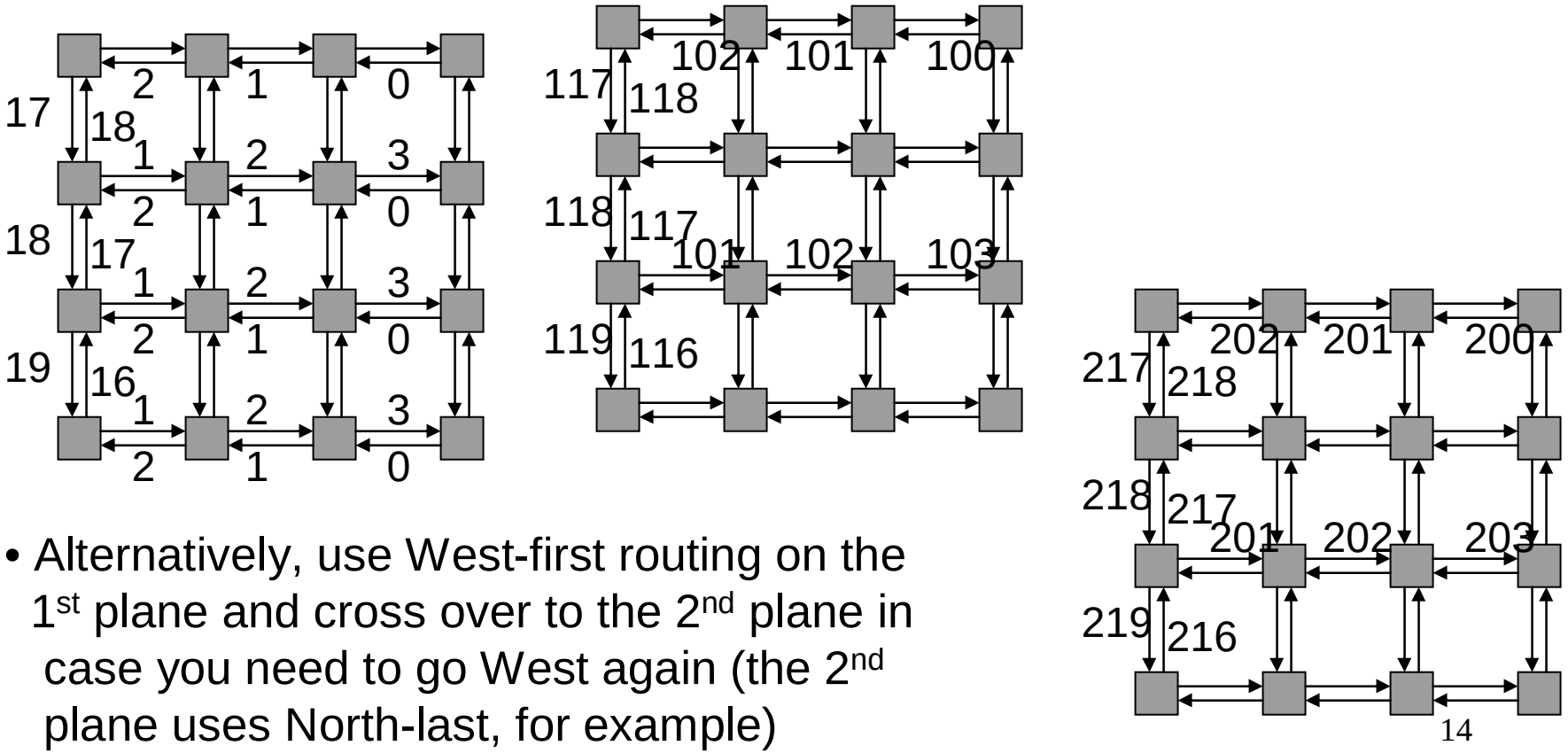
Buffer Management

- Credit-based: keep track of the number of free buffers in the downstream node; the downstream node sends back signals to increment the count when a buffer is freed; need enough buffers to hide the round-trip latency

- On/Off: the upstream node sends back a signal when its buffers are close to being full – reduces upstream signaling and counters, but can waste buffer space

Deadlock Avoidance with VCs

- VCs provide another way to number the links such that a route always uses ascending link numbers



- Alternatively, use West-first routing on the 1st plane and cross over to the 2nd plane in case you need to go West again (the 2nd plane uses North-last, for example)

Projects

- Deadlines:
  - 3/2: decide on teams, talk to me about initial idea
  - 3/9: rd related work, talk to me about abstract/simulator
  - 3/16: understand simulator aspects, write 1-page pitch (articulate idea, expected benefits, compare against related work, list of experiments/simulator changes)
  - Start experiments on simulator

- Evaluation:
  - Submit written reports (April end)
  - Peer reviews of reports (mid May)
  - My evaluation: 20%, Peer evaluation: 20%, Quality of reviews: 10%

Title

- Bullet