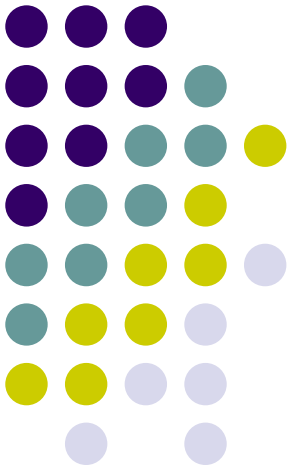
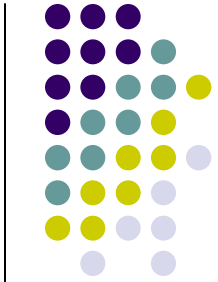
Network-on-Chip
Flow Control

Ingo Sander
ingo@imit.kth.se



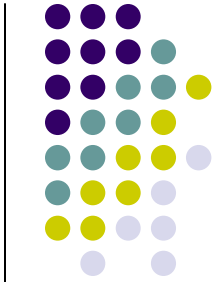
Dally: Ch 12, 13

Flow Control



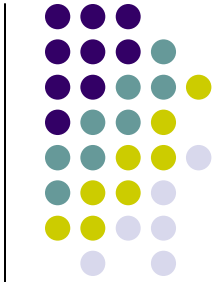
- Flow Control determines how the resources of a network, such as channel bandwidth and buffer capacity are allocated to packets traversing a network
- Goal is to use resources as efficient as possible to allow a high throughput
- An efficient flow control is a prerequisite to achieve a good network performance

Flow Control



- Flow Control can be viewed as a problem of
  - Resource allocation
  - Contention resolution
- Resources in form of channels, buffers and state must be allocated to each packet
- If two packets compete for the same channel flow control can only assign the channel to one packet, but must also deal with the other packet

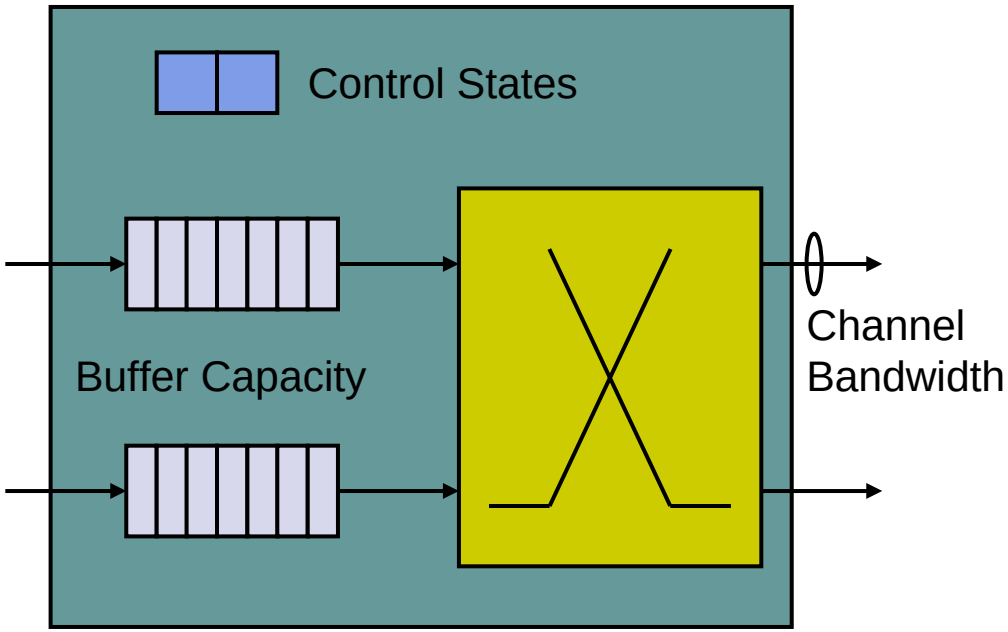
Flow Control



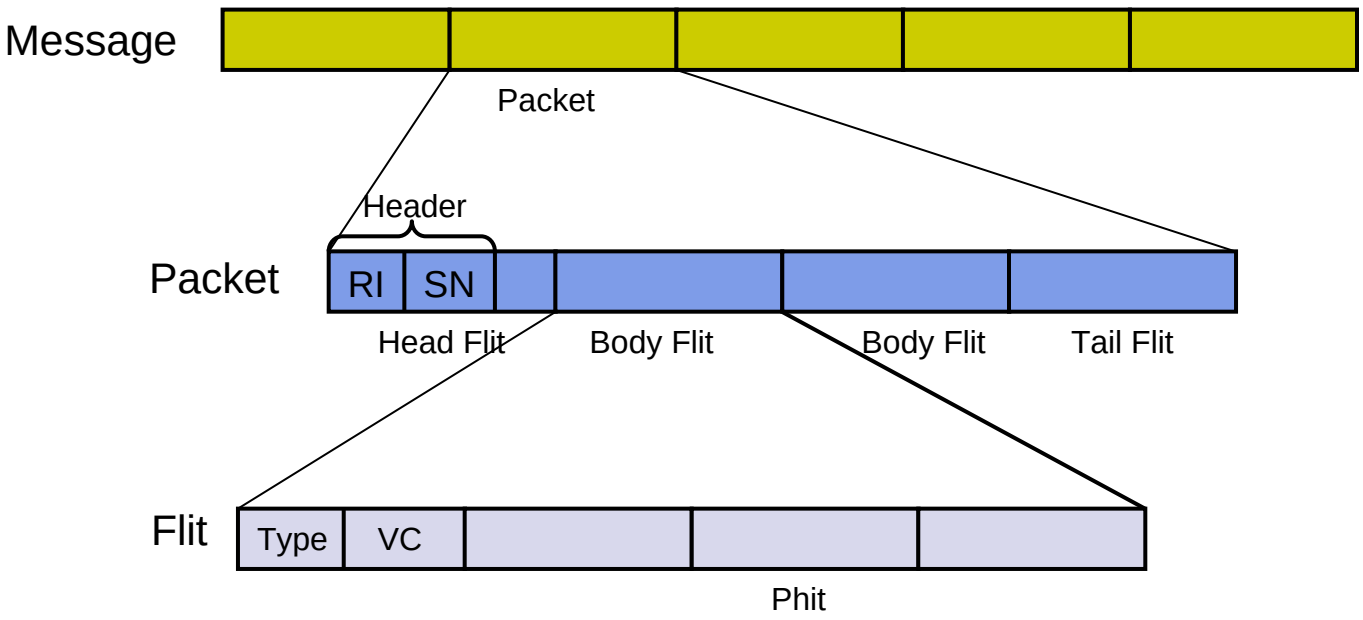
- Flow Control can be divided into
  - Bufferless flow control
    - Packets are either *dropped* or *misrouted*
  - Buffered flow control
    - Packets that cannot be routed via the desired channel are stored in buffers

Resources in a Network Node

- Control State
  - Tracks the resources allocated to the packet in the node and the state of the packet
- Buffer
  - Packet is stored in a buffer before it is send to next node
- Bandwidth
  - To travel to the next node bandwidth has to be allocated for the packet

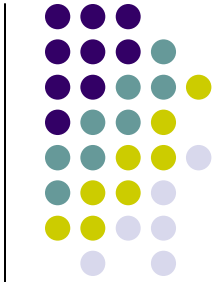


Units of Resource Allocation



Messages, Packets, Flits and Phits are handled in different layers of the network protocol

Units of Resource Allocation

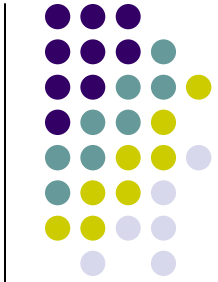


- A *message* is a contiguous group of bits that is delivered from source terminal to destination terminal. A message consists of packets.
- A *packet* is the basic unit for routing and sequencing. The control state is assigned to a packet. Packets maybe divided into flits.

Units of Resource Allocation

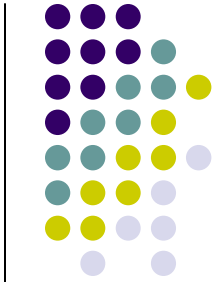
- A *flit* (flow control digit) is the basic unit of bandwidth and storage allocation. Flits do not have any routing or sequence information and have to follow the route for the whole packet.
  - Head flit allocates channel state for a packet and tail flit deallocates it
- A *phit* (physical transfer digits) is the unit that is transfered across a channel in a single clock cycle

Packets or Flits?



- Contradictory requirements on packets
  - Packets should be very large in order to reduce overhead of routing and sequencing
  - Packets should be very small to allow efficient and fine-grained resource allocation and minimize blocking latency
- Flits try to eliminate this conflict
  - Packets can be large (low overhead)
  - Flits can be small (efficient resource allocation)

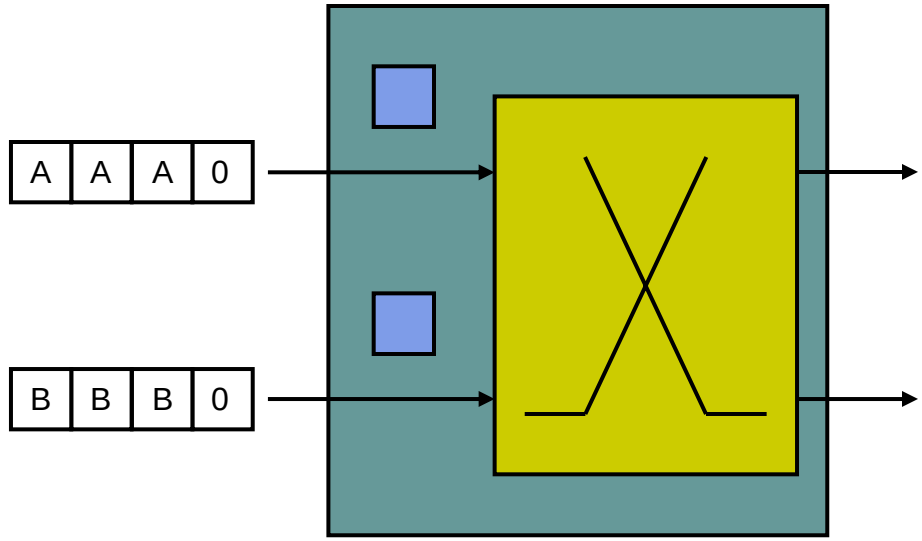
Size: Phit, Flit, Packet



- There are no fixed rules for the size of phits, flits and packets
- Typical values
  - Phits: 1 bit to 64 bits
  - Flits: 16 bits to 512 bits
  - Packets: 128 bits to 1024 bits

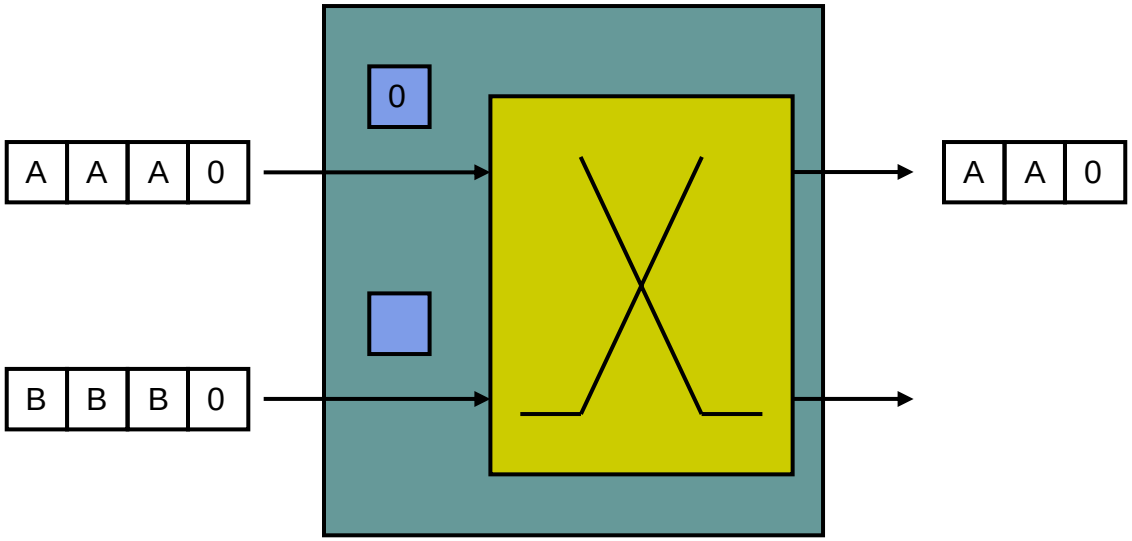
Bufferless Flow Control

- No buffers means less implementation cost
- If more than one packet shall be routed to the same output, one has to be
  - Misrouted or
  - Dropped
- In this example two packets A and B (consisting of several flits) arrive at a network node



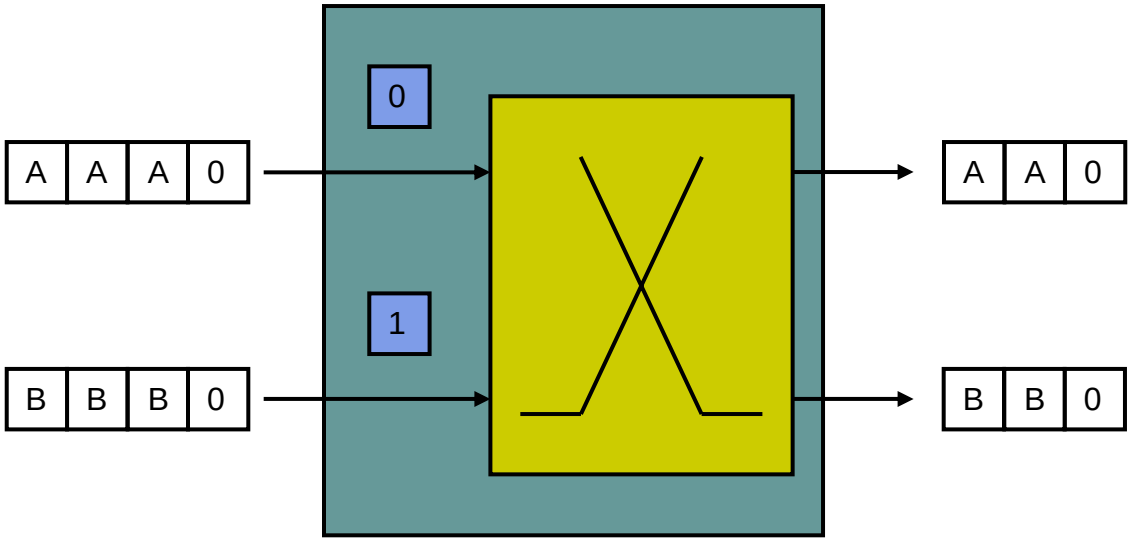
Bufferless Flow Control

- Packet B is dropped and must be resent
- There must be a protocol that informs the sending node that the packet has been dropped
  - e.g. Resend after no acknowledge has been received within a given time



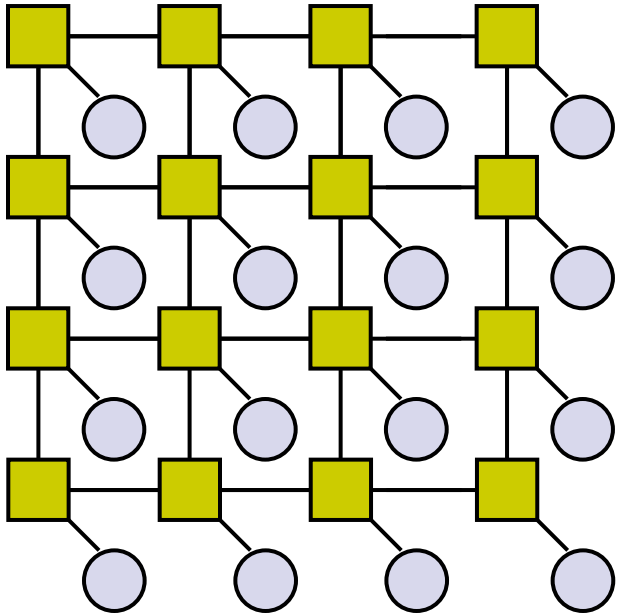
Bufferless Flow Control

- Packet B is misrouted
- No further action is required here, but
  - at the receiving node packets have to be sorted into original order

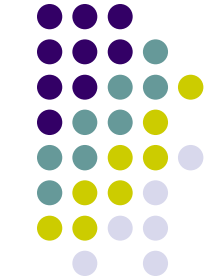


Nostrum
Bufferless Flow Control

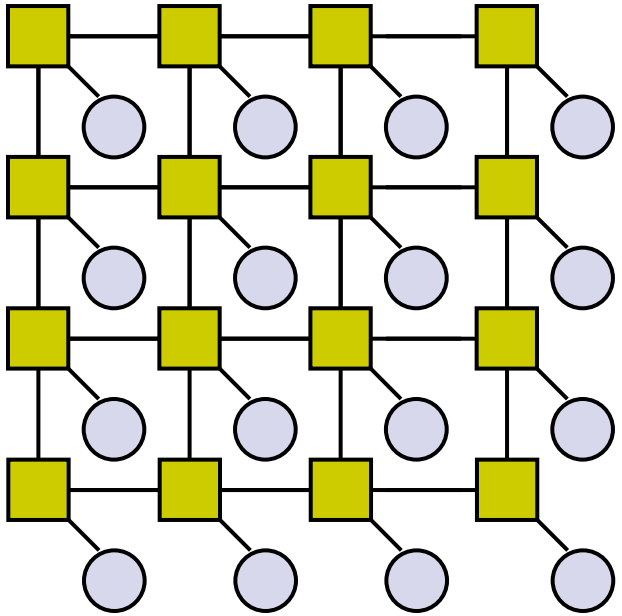
- The Nostrum NoC developed at KTH uses bufferless flow control
- The topology is a mesh and packets are routed by ”hot-potato-routing”
  - Packets have to go somewhere, they are not dropped



Nostrum
Bufferless Flow Control

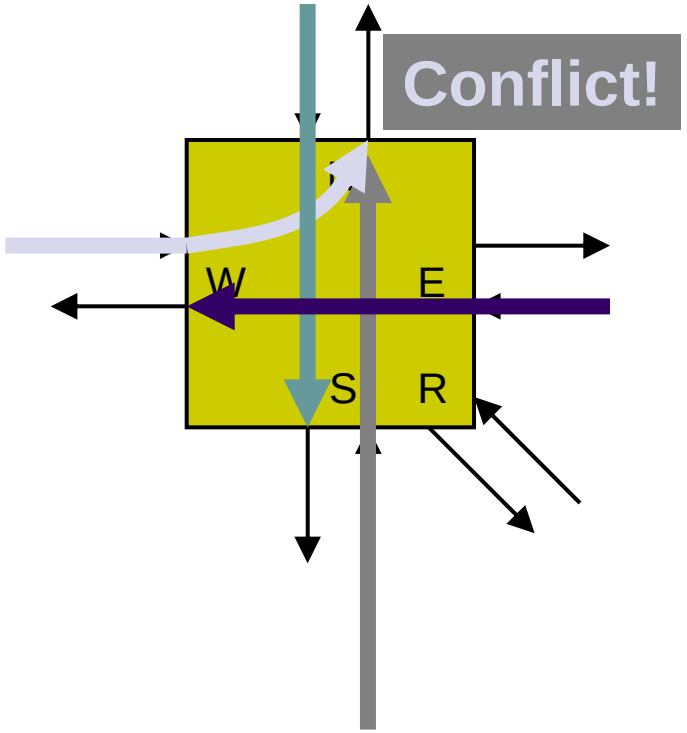


- Motivation
  - Small and fast switches!
  - Low power dissipation due to network switches
  - The network shall be operated at a load so that only few packets need to be misrouted
  - Packets are reordered in network interface



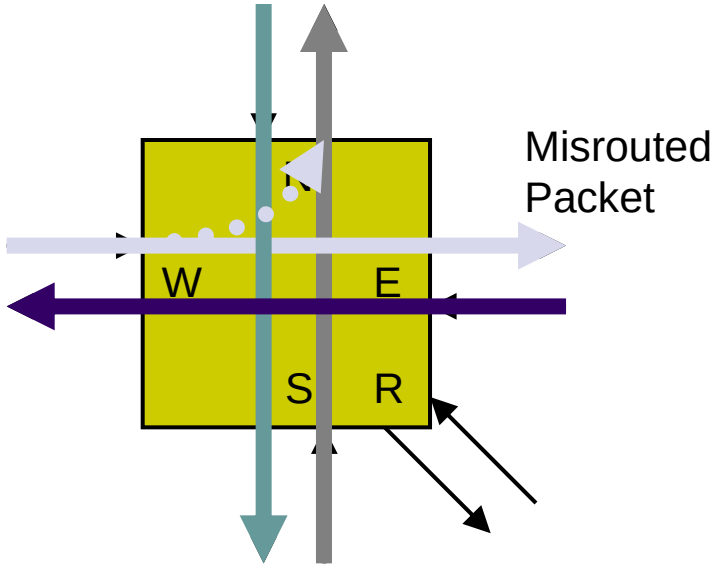
Hot-Potato Routing in Nostrum

- If packets want to take the same channel one of the packet has to be deflected (misrouted)
- Which packet is to be misrouted is decided by a routing algorithm that can be arbitrarily complex



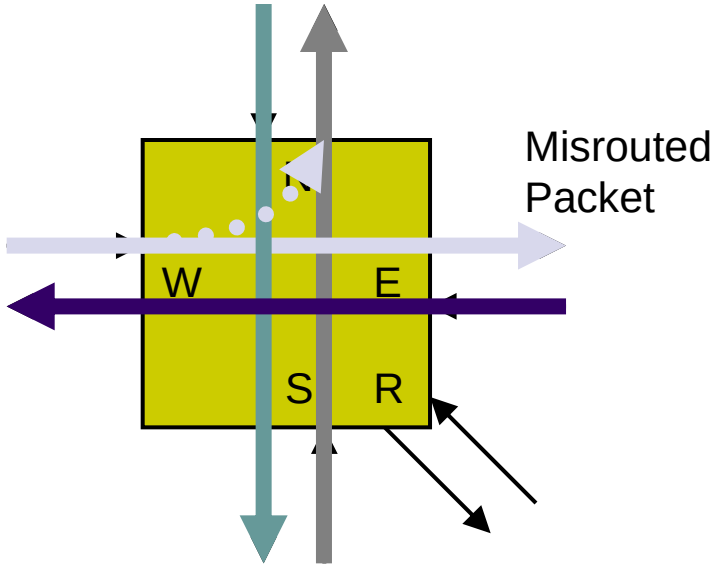
Hot-Potato Routing in Nostrum

- One possible solution of many!
- The “west-packet” is deflected and must possibly take a longer way through the network!



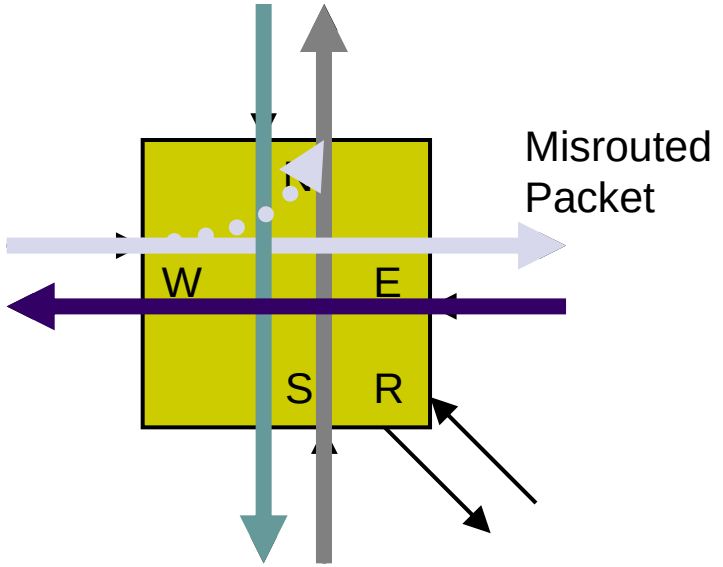
Hot-Potato Routing in Nostrum

- How shall the routing algorithm work?
- What is important?
  - Average latency should be low
  - It must be guaranteed that a packet is delivered in finite time
  - Worst case latency shall be low
  - How to guarantee quality of service?

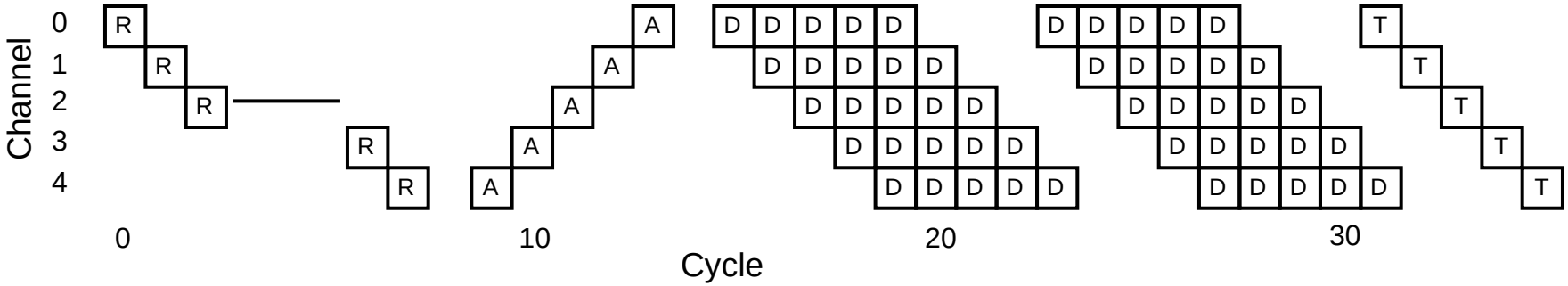


Hot-Potato Routing in Nostrum

- Is it better to prioritize
  - Old packets?
  - Young packets?
  - Packets with a short distance to its destination?
  - Packets with a long destination to its destination?
  - Shall there be another priority between packets?

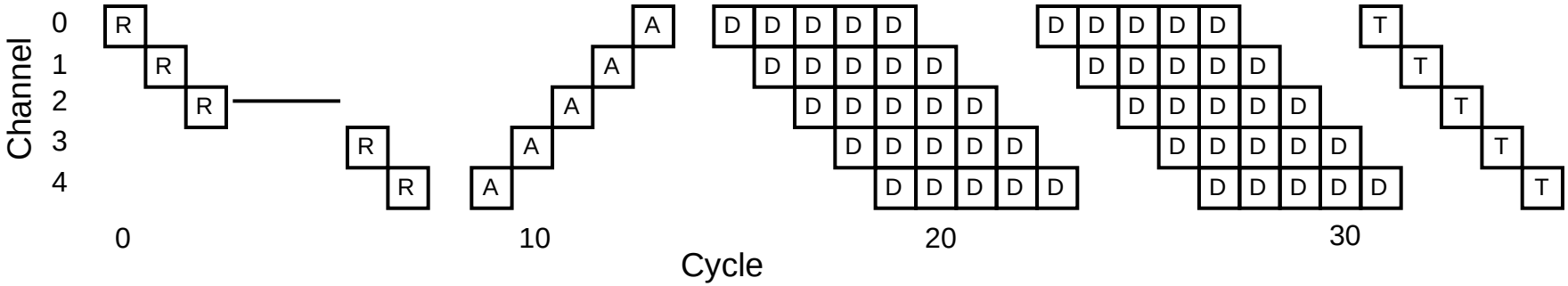


Circuit Switching



- Circuit-Switching is a bufferless flow control, where several channels are reserved to form a circuit
- A request (R) propagates from source to destination, which is answered by an acknowledgement (A)
- Then data is sent (here two five flit packets (D)) and a tail flit (T) is sent to deallocate the channels

Circuit Switching

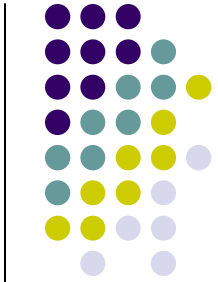


- Circuit-switching does not suffer from dropping or misrouting packets
- However there are two weaknesses:
  - High latency: $T = 3\,H\,t_r + L/b$
  - Low throughput, since channel is used to a large fraction of time for signaling and not for delivery of the payload

Buffered Flow Control

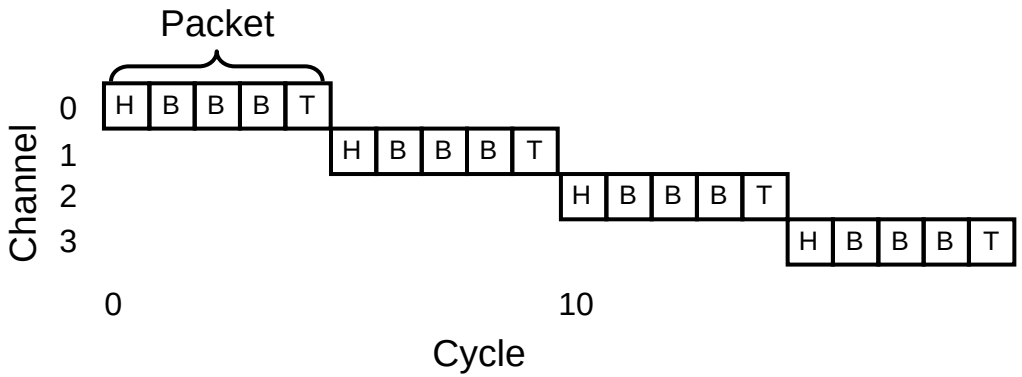
- More efficient flow control can be achieved by adding buffers
  - With sufficient buffers packets do not need to be misrouted or dropped, since packets can wait for the outgoing channel to be ready

Buffered Flow Control



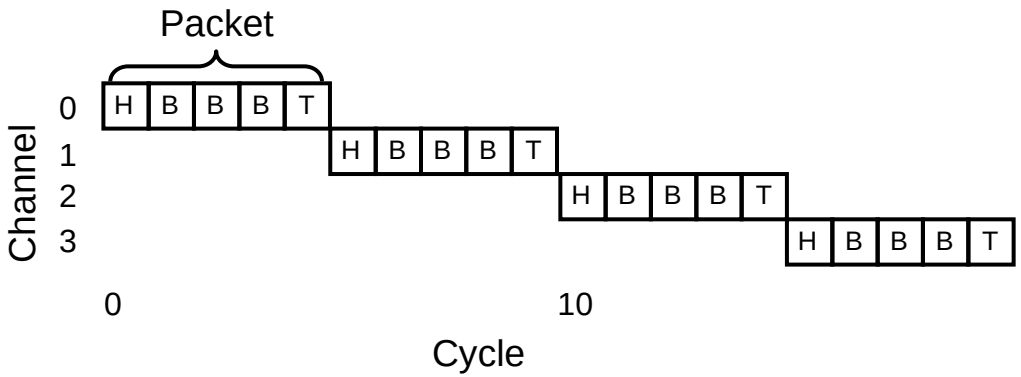
- Two main approaches
  - Packet-Buffer Flow Control
    - Store-And-Forward
    - Cut-Through
  - Flit-Buffer Flow Control
    - Wormhole Flow Control
    - Virtual Channel Flow Control

Store and Forward Flow Control



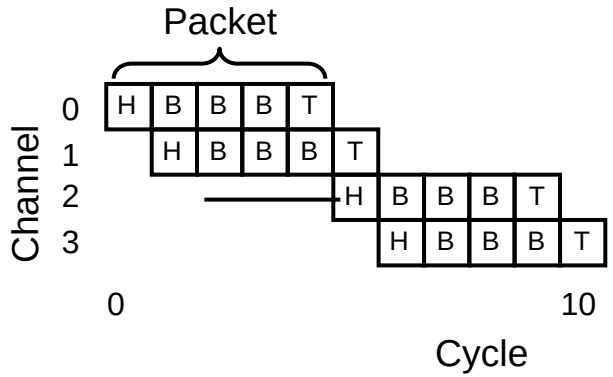
- Each node along a route waits until a packet is completely received (stored) and then the packet is forwarded to the next node
- Two resources are needed
  - Packet-sized buffer in the switch
  - Exclusive use of the outgoing channel

Store and Forward Flow Control



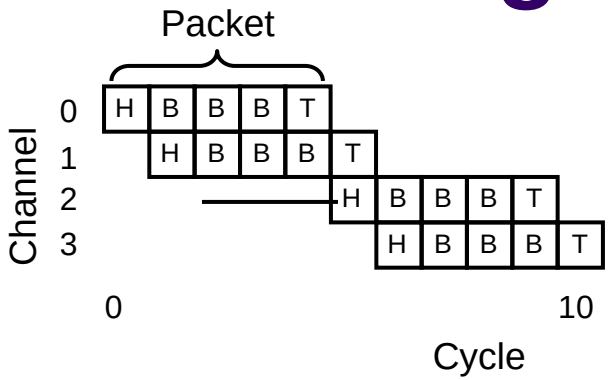
- Advantage: While waiting to acquire resources, no channels are being helt idle and only a single packet buffer on the current node is occupied
- Disadvantage: Very high latency
  - $T = H\,(t_r + L/b)$

Cut-Through Flow Control



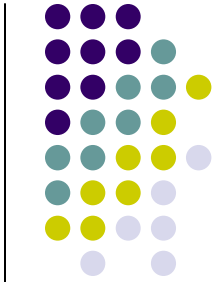
- Transmission on the next channel starts directly when the new header flit is received (otherwise it behaves like Store-Forward)
- Channel is released after tail flit

Cut-Through Flow Control



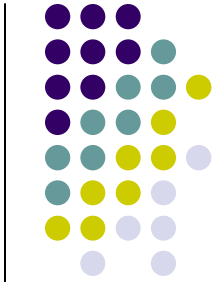
- Advantages
  - Cut-through reduces the latency
    - $T = H\,t_r + L/b$
  - Very high channel utilization
- Disadvantages (also valid for Store-and Forward)
  - No good utilization of buffers, since they are allocated in units of packets (also valid for Store-and Forward)
  - Contention latency is increased, since packets must wait until a whole packet leaves the occupied channel

Wormhole Flow Control



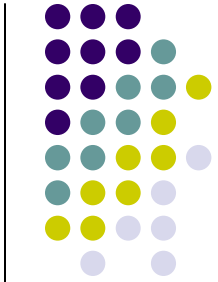
- Wormhole flow control operates like cut-through, but with channel and buffers allocated to flits rather than packets
- When the head flit arrives at a node, it must acquire three resources before it can be forwarded to the next node along a route
  - A virtual channel (channel state) for the packet
  - One flit buffer
  - Bandwidth corresponding to one flit
- Body flits use a virtual channel acquired by the head flit and have to acquire one flit buffer and bandwidth corresponding to one flit
- Tail flits behave like body flits, but release also the channel

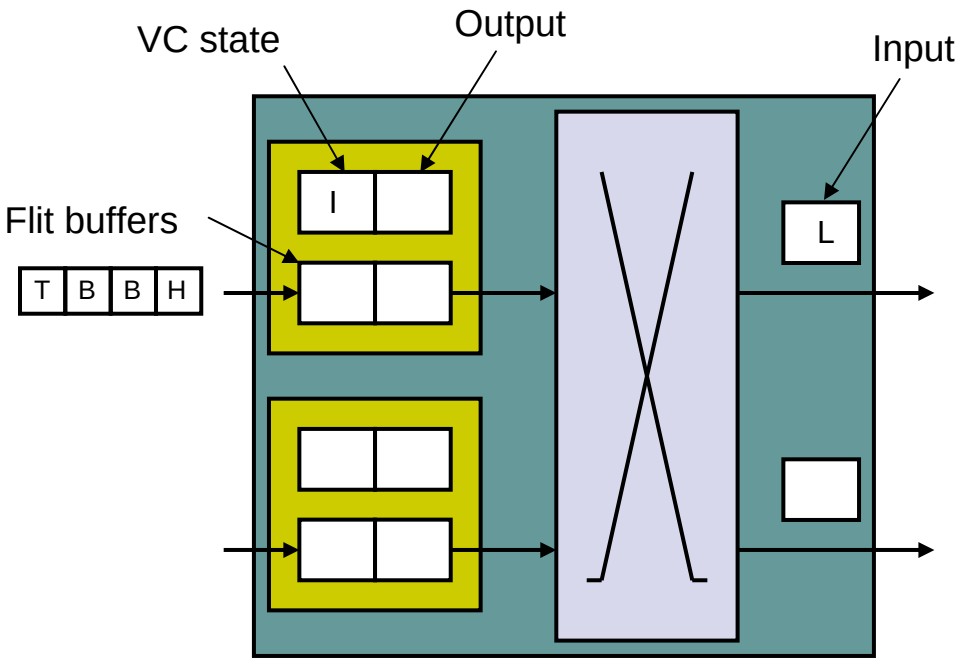
Wormhole Flow Control



- Virtual channels hold the state needed to coordinate the handling of flits of a packet over a channel
- Comparison to cut-through
  - wormhole flow control makes far more efficient use of buffer space
  - Throughput maybe less, since wormhole flow control may block a channels mid-packets

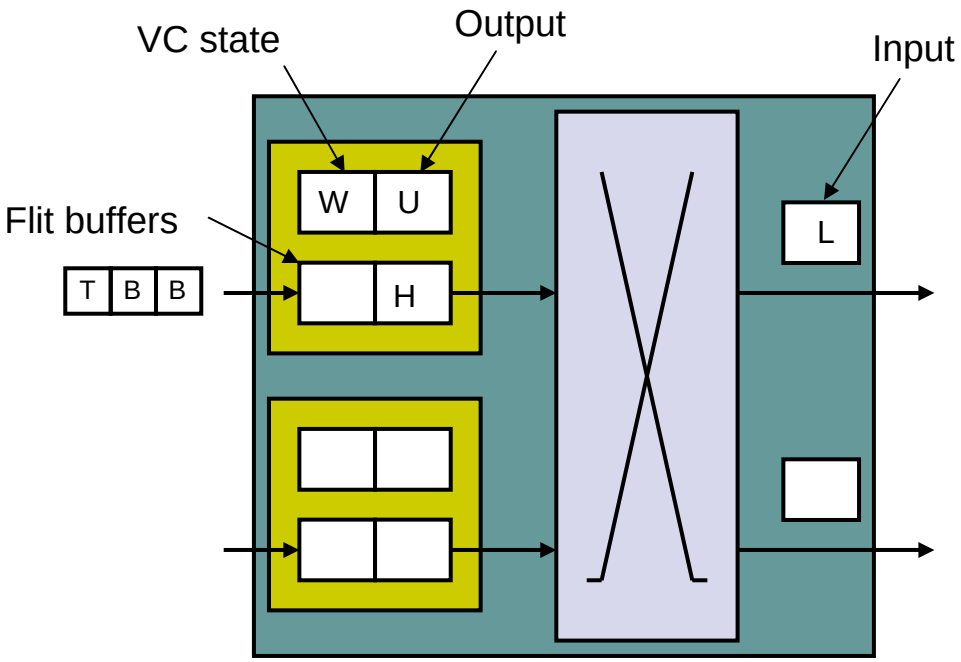
Example for Wormhole Flow Control





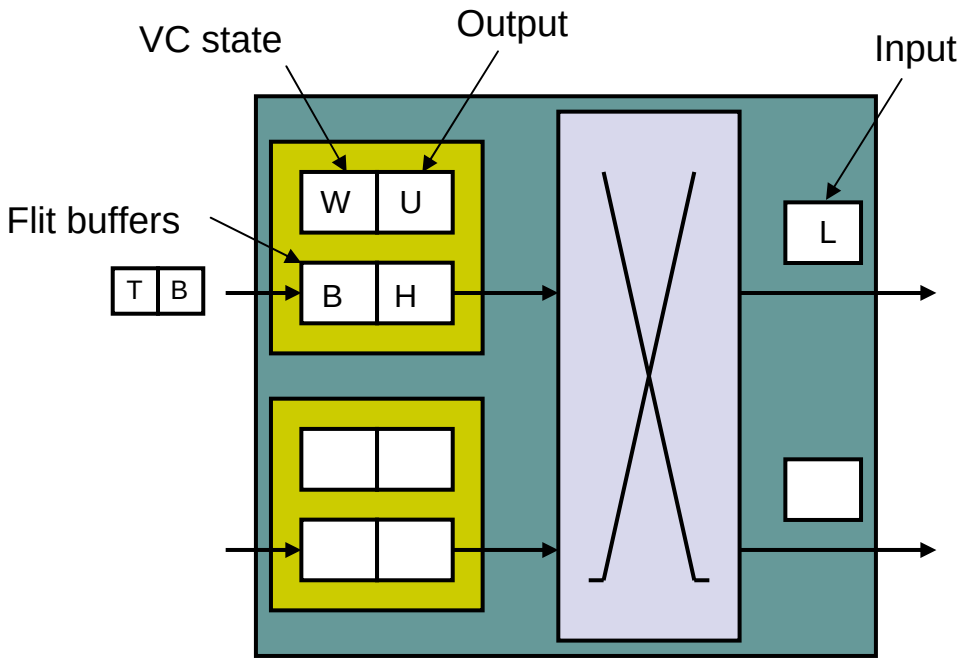
- Input virtual channel is in idle state (I)
- Upper output channel is occupied, allocated to lower channel (L)

Example for Wormhole Flow Control



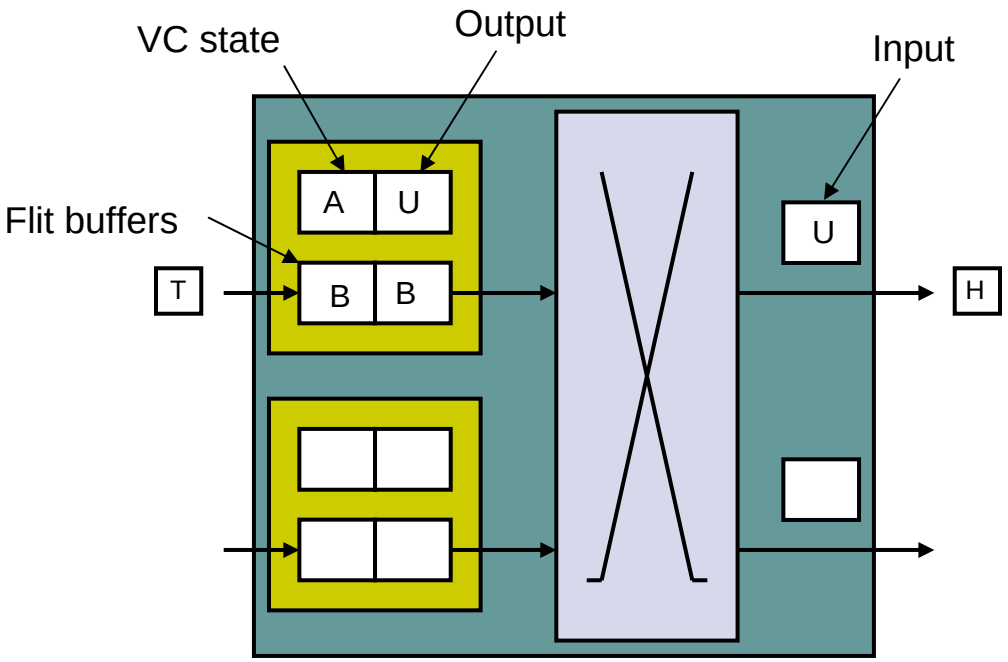
- Input channel enters the waiting state (W)
- Head flit is buffered

Example for Wormhole Flow Control



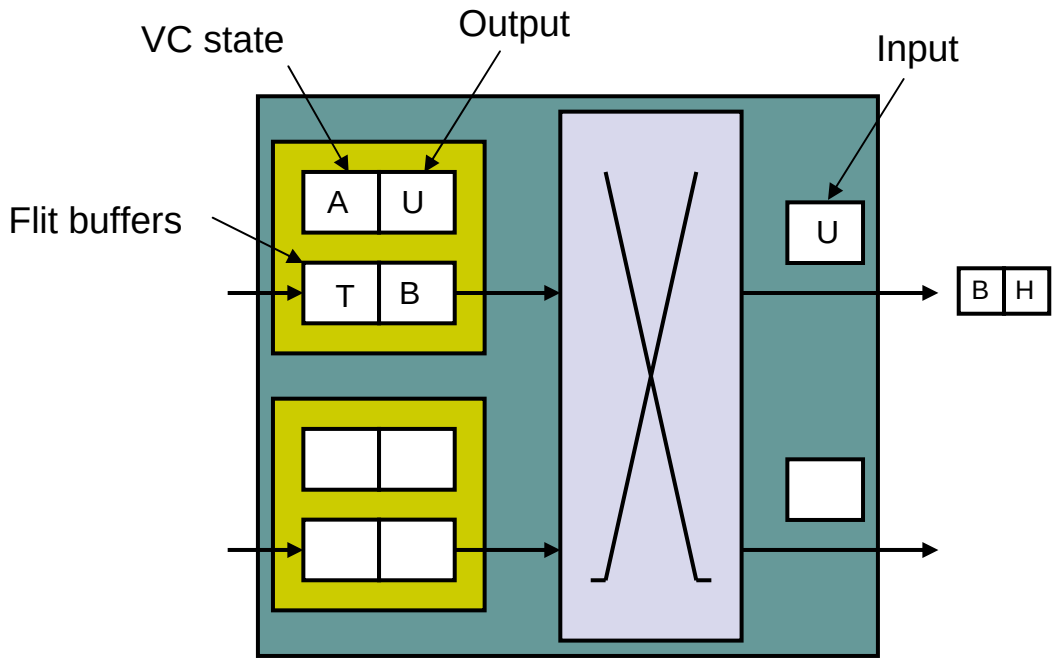
- Body flit is also buffered
- No more flits can be buffered, thus congestion arises if more flits want to enter the switch

Example for Wormhole Flow Control



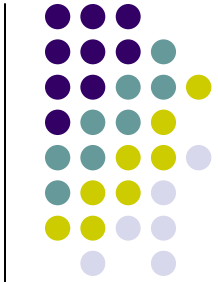
- Virtual channel enters active state (A)
- Head flit is output on upper channel
- Second body flit is accepted

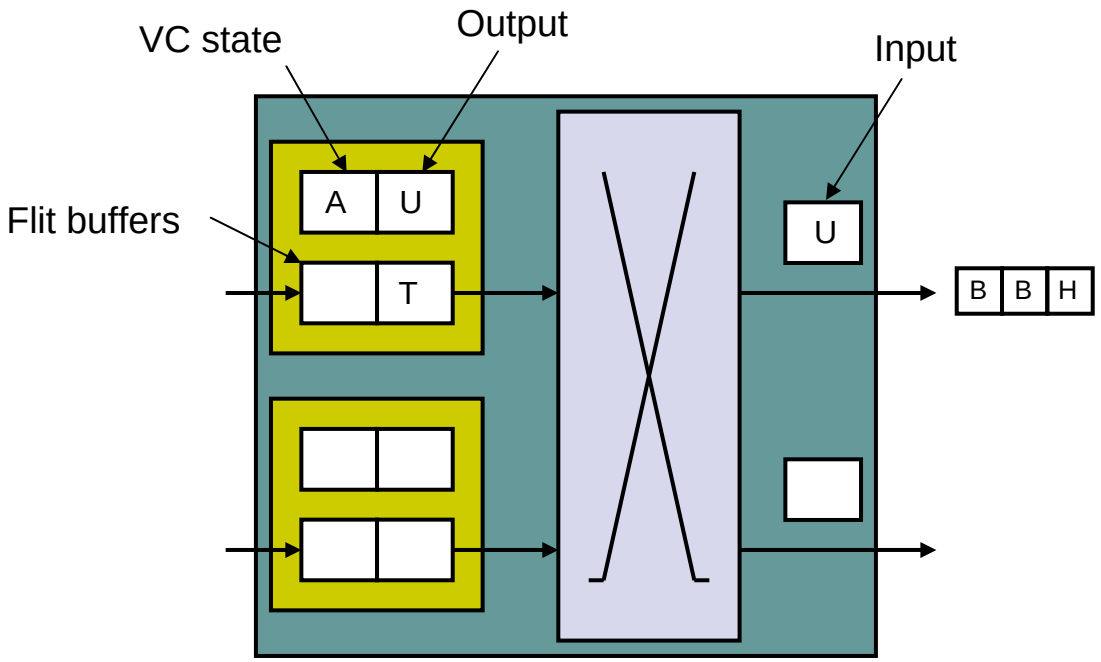
Example for Wormhole Flow Control



- First body flit is output
- Tail flit is accepted

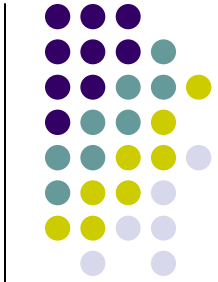
Example for Wormhole Flow Control

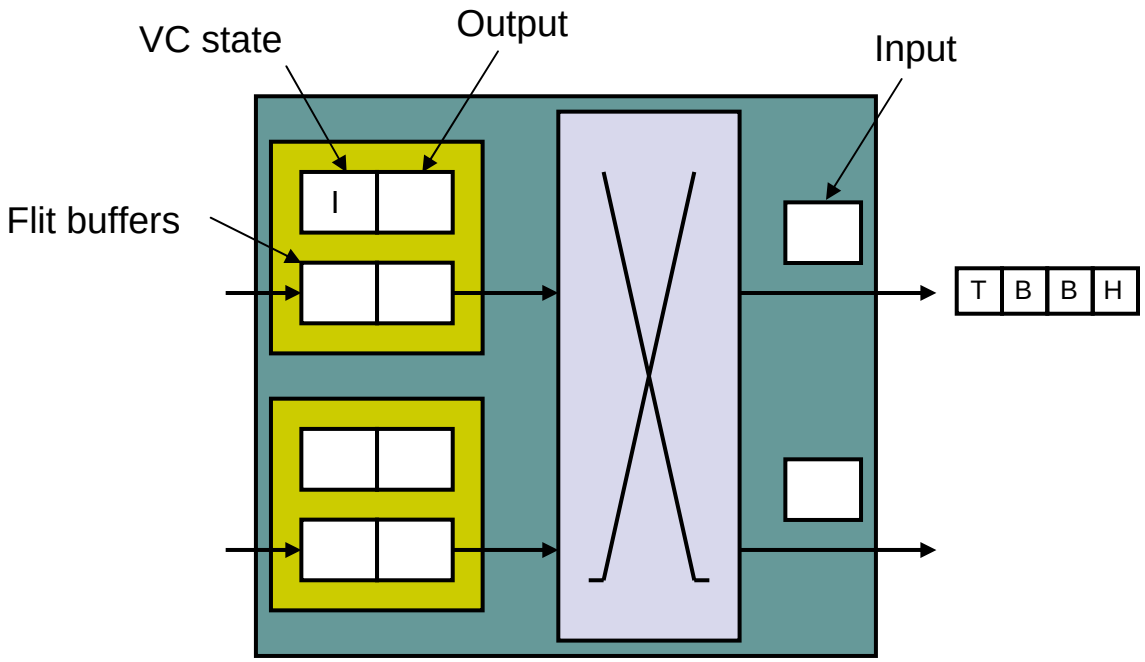




- Second body flit is output

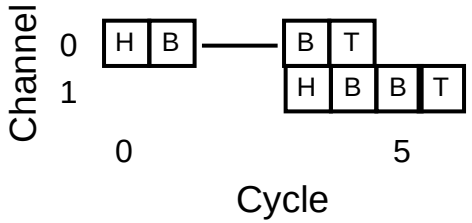
Example for Wormhole Flow Control





- Tail flit is output
- Virtual channels is deallocated and returns to idle state

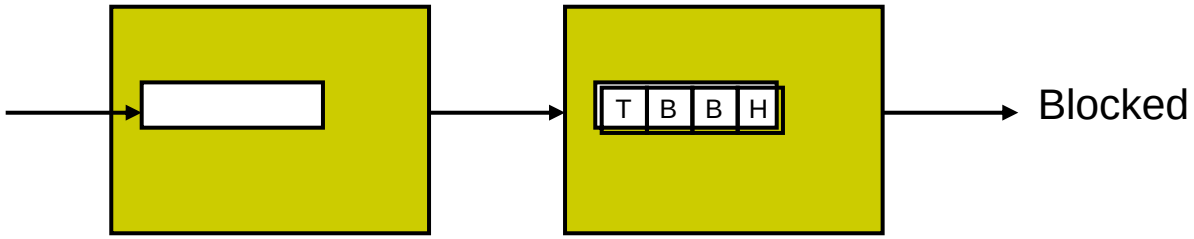
Wormhole Flow Control



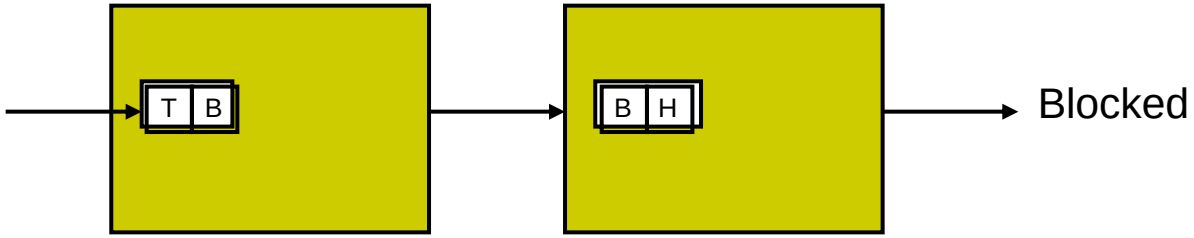
- The main advantage of wormhole to cut-through is that buffers in the routers do not need to be able to hold full packets, but only need to store a number of flits
- This allows to use smaller and faster routers

Blocking
Cut-Through and Wormhole

Cut-Through (Buffer-Size 1 Packet)

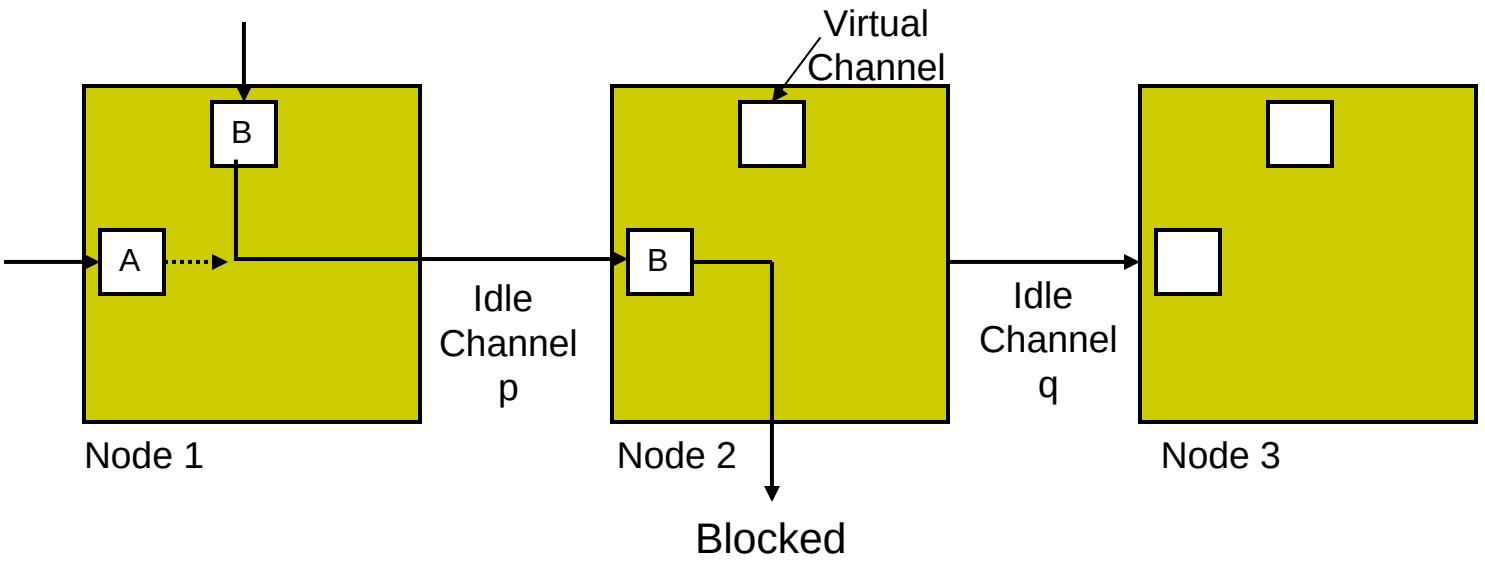


Wormhole (Buffer-Size 2 Flits)



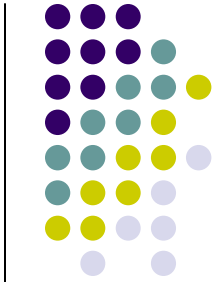
- If a packet is blocked, the flits of the wormhole packet are stored in different routers

Wormhole Flow Control



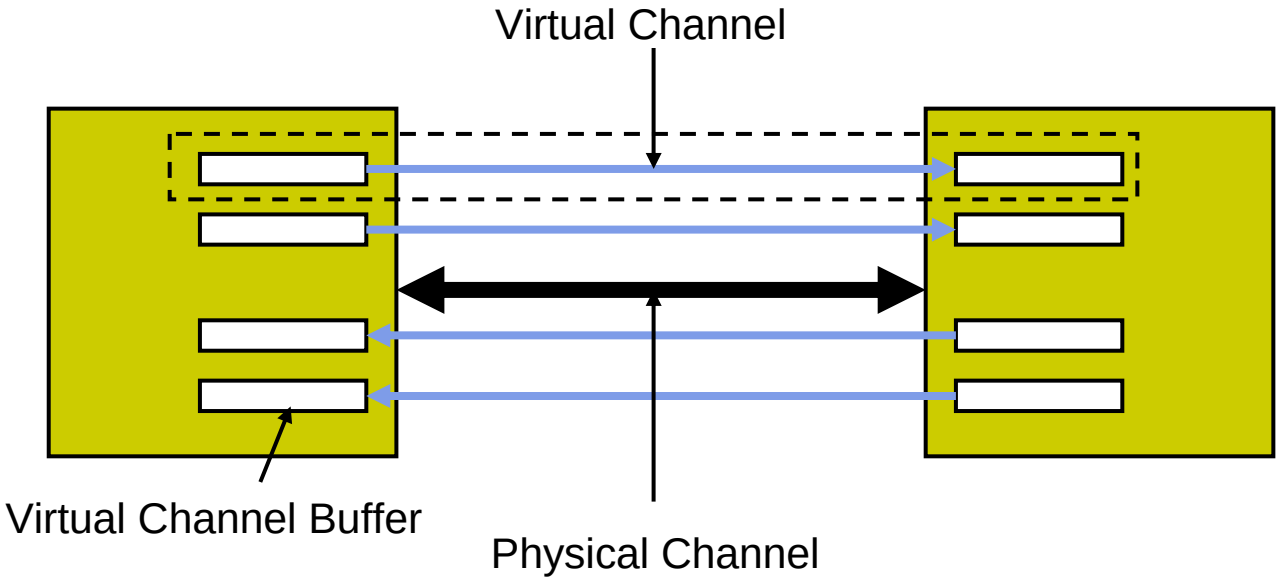
- There is only one virtual channel for each physical channel
- Packet A is blocked and cannot acquire channel p
- Though channels p and q are idle packet A cannot use these channels since B owns channel p

Virtual Channel-Flow Control



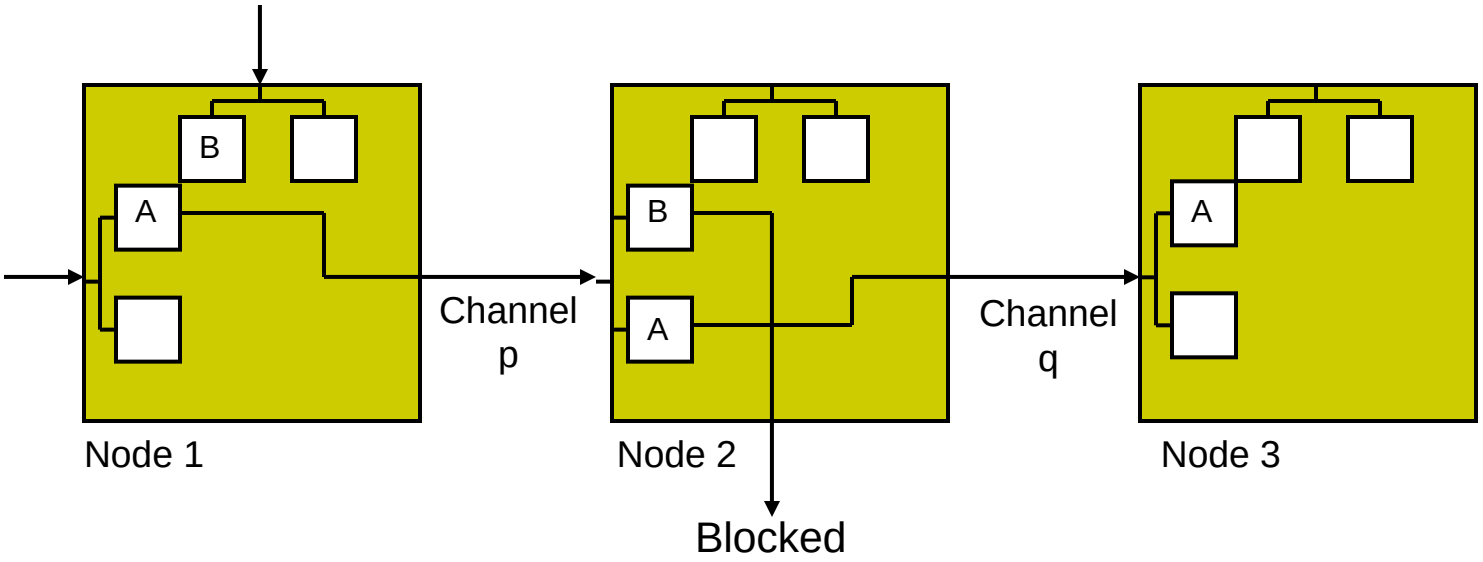
- In virtual channel flow-control several channels are associated with a single physical channel
- This allows to use the bandwidth that otherwise is left idle when a packet blocks the channel
- Unlike wormhole flow control subsequent flits are not guaranteed bandwidth, since they have to compete for bandwidth with other flits

Concept of Virtual Channels



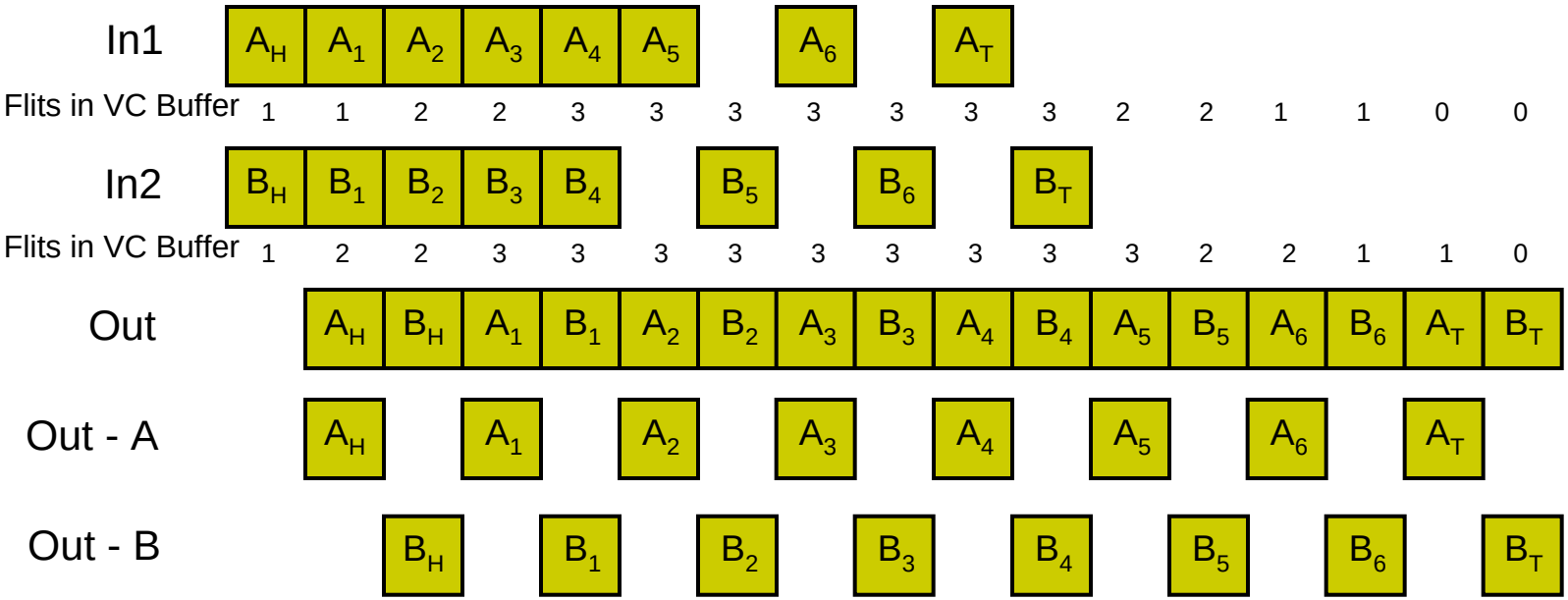
- A physical channel is shared by several virtual channels
- Naturally the speed of each virtual channel connection is reduced

Virtual Channel Flow Control



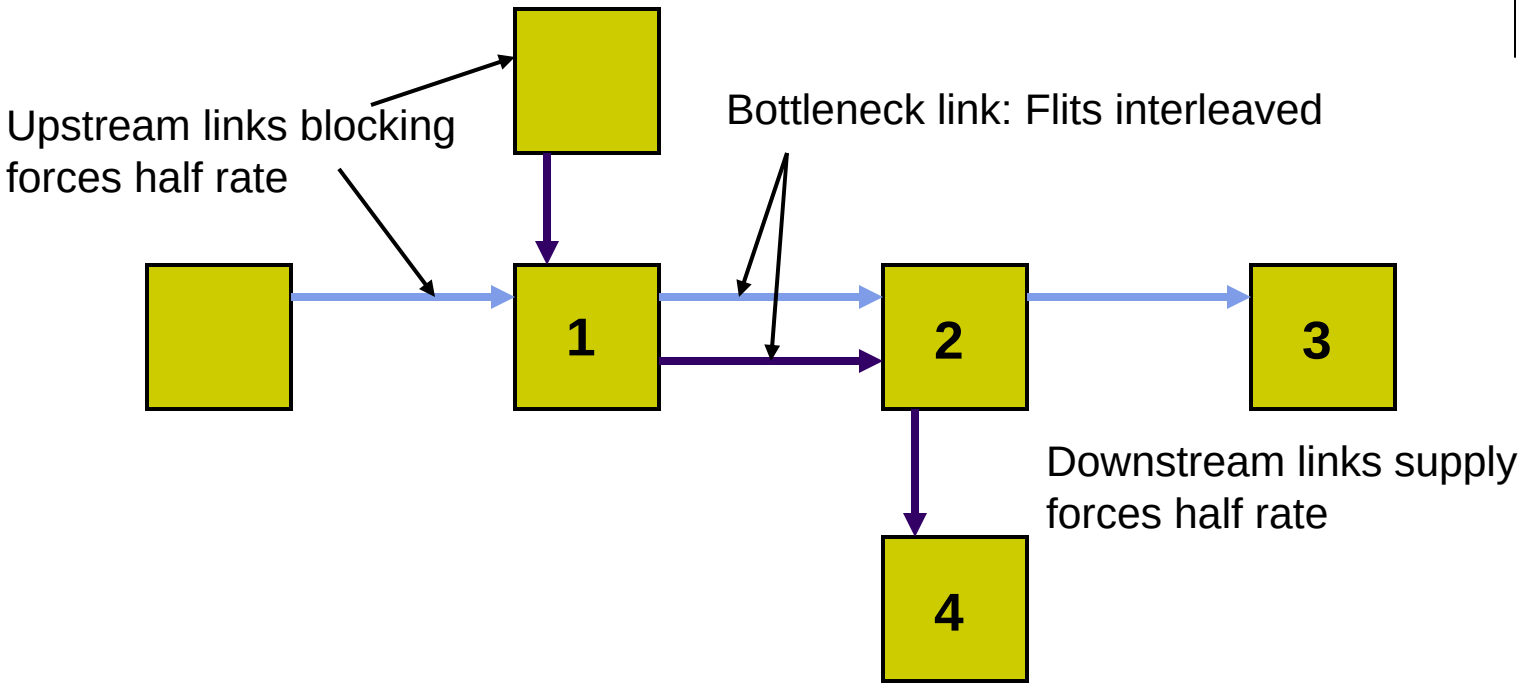
- There are several virtual channels for each physical channel
- Packet A can use a second virtual channel and thus proceed over channel p and q

Virtual Channel Flow Control Fair Bandwidth Arbitration



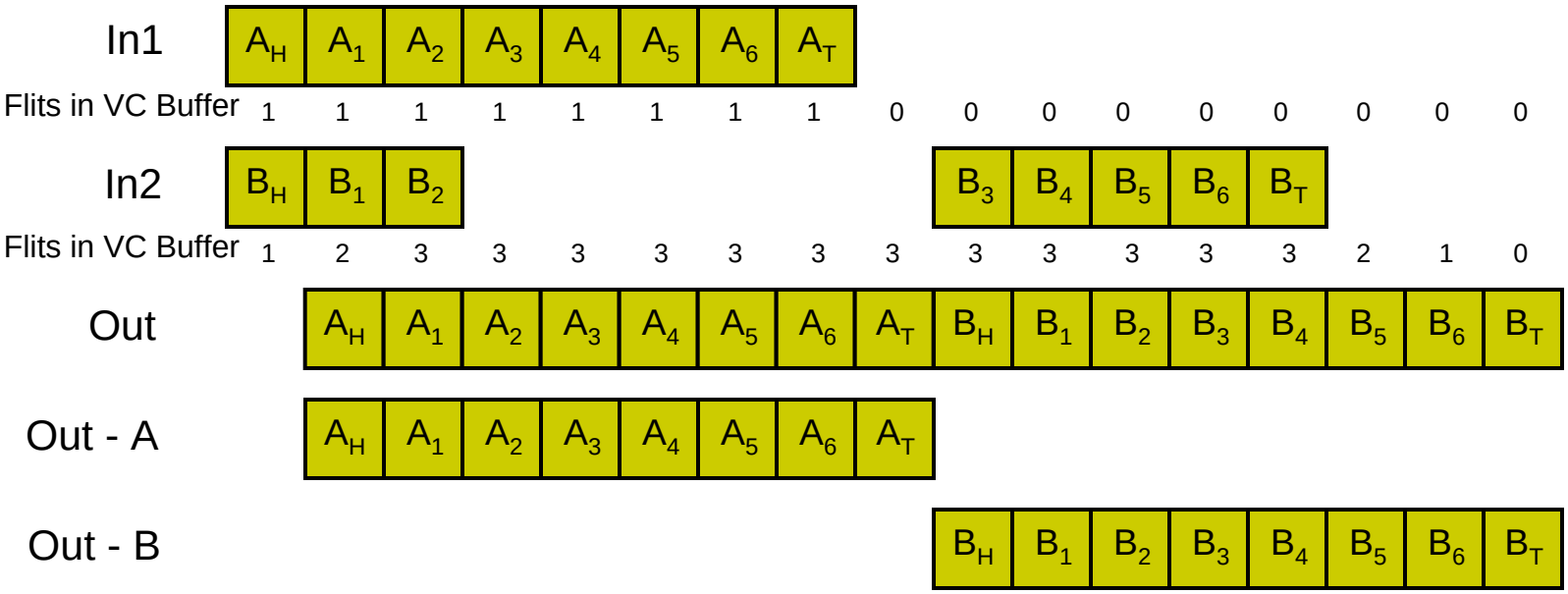
- The virtual channels interleave their flits
- This results in a high average latency

Virtual Channel Flow Control



- A bottleneck like that between node 1 and 2 can cause lower throughput rate upstreams and downstreams

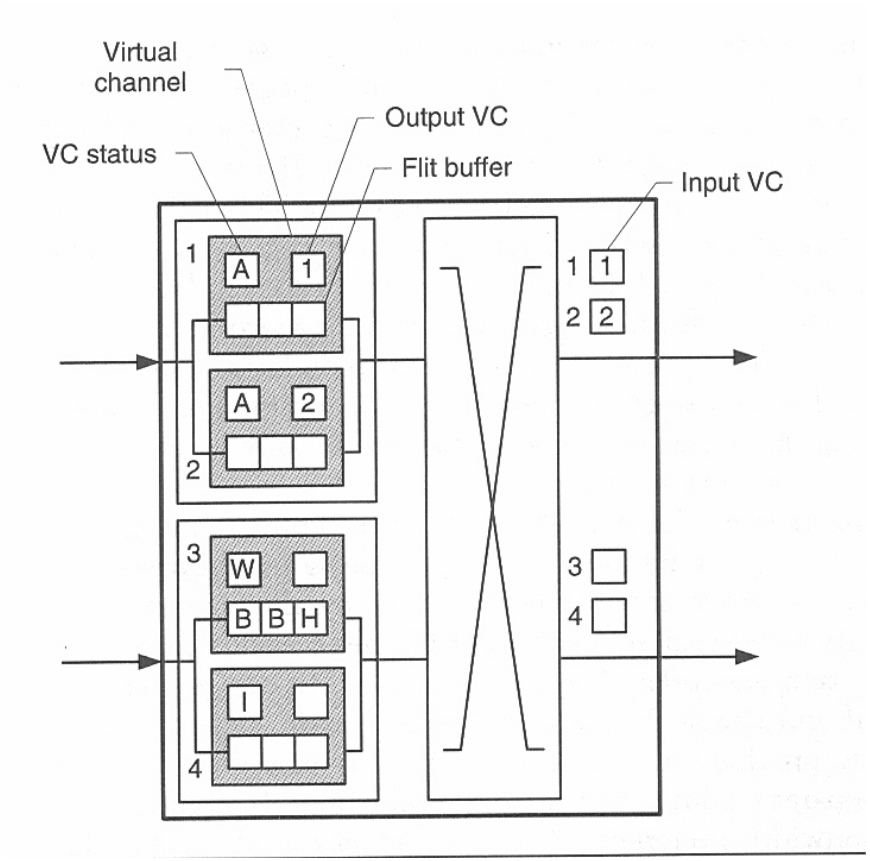
Virtual Channel Flow Control Winner-Take-All Arbitration



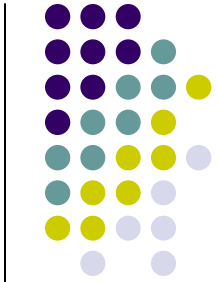
- A winner-take all arbitration reduces the average latency with no throughput penalty

Virtual Channel Router

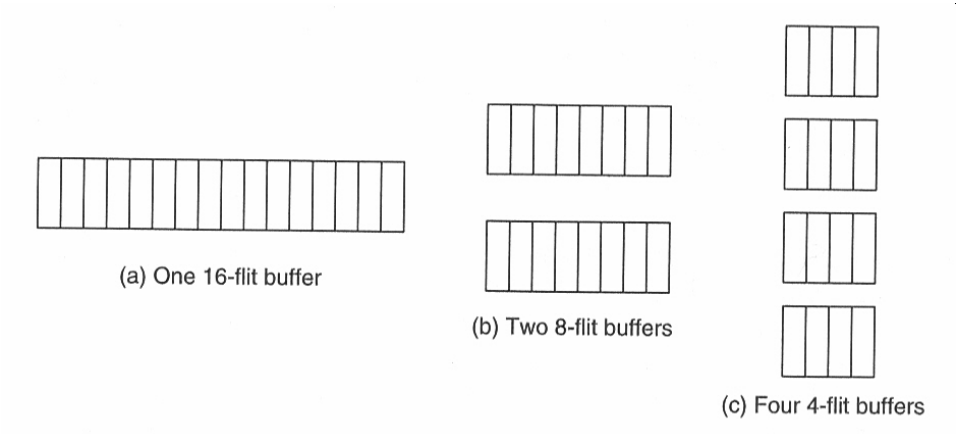
- Router has
  - 2 input channels
  - 2 output channels
  - 2 virtual channels
  - 4 flit buffers



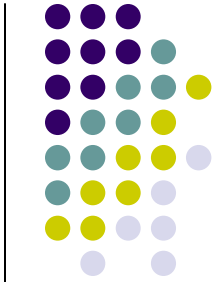
Virtual Channel Flow Control Buffer Storage



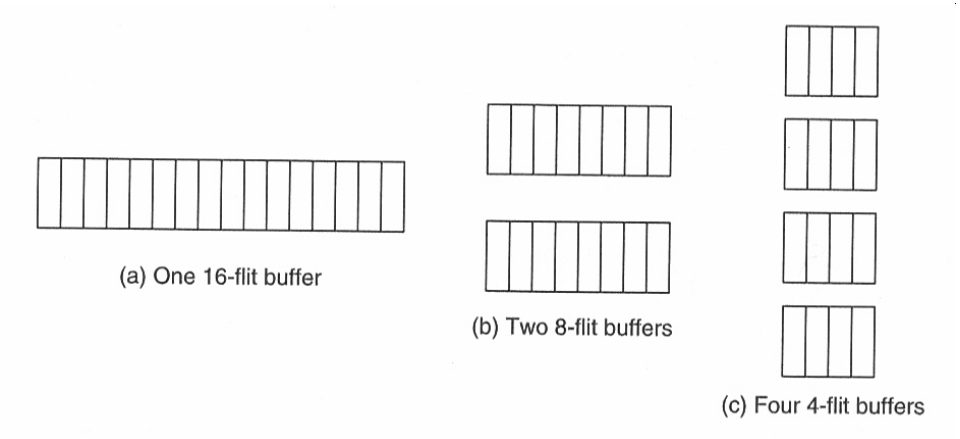
- Buffer storage is organized in two dimensions
  - Number of virtual channels
  - Number of flits that can be buffered per channel



Virtual Channel Flow Control Buffer Storage

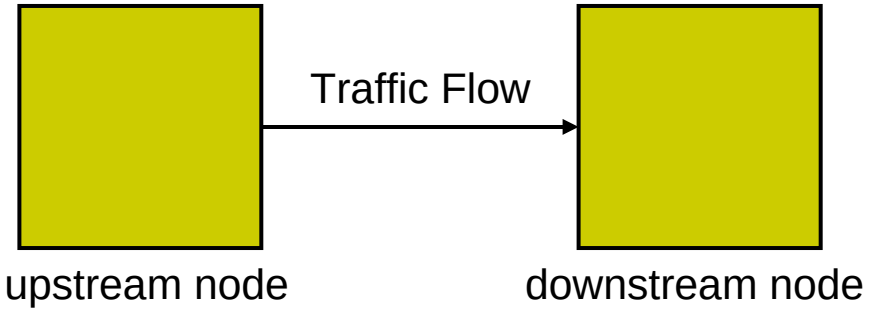


- Virtual channel buffer shall at least be as deep as needed to cover round-trip credit latency
- In general it is usually better to add more virtual channels than to increase the buffer size

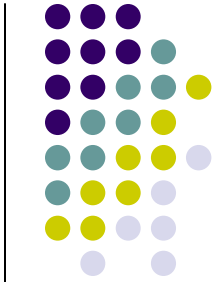


Buffer Management

- In buffered flow control nodes there is a need for communication between nodes in order to inform about the availability of buffers
- *Backpressure* informs upstream nodes that they must stop sending to a downstream node when the buffers of that downstream node are full

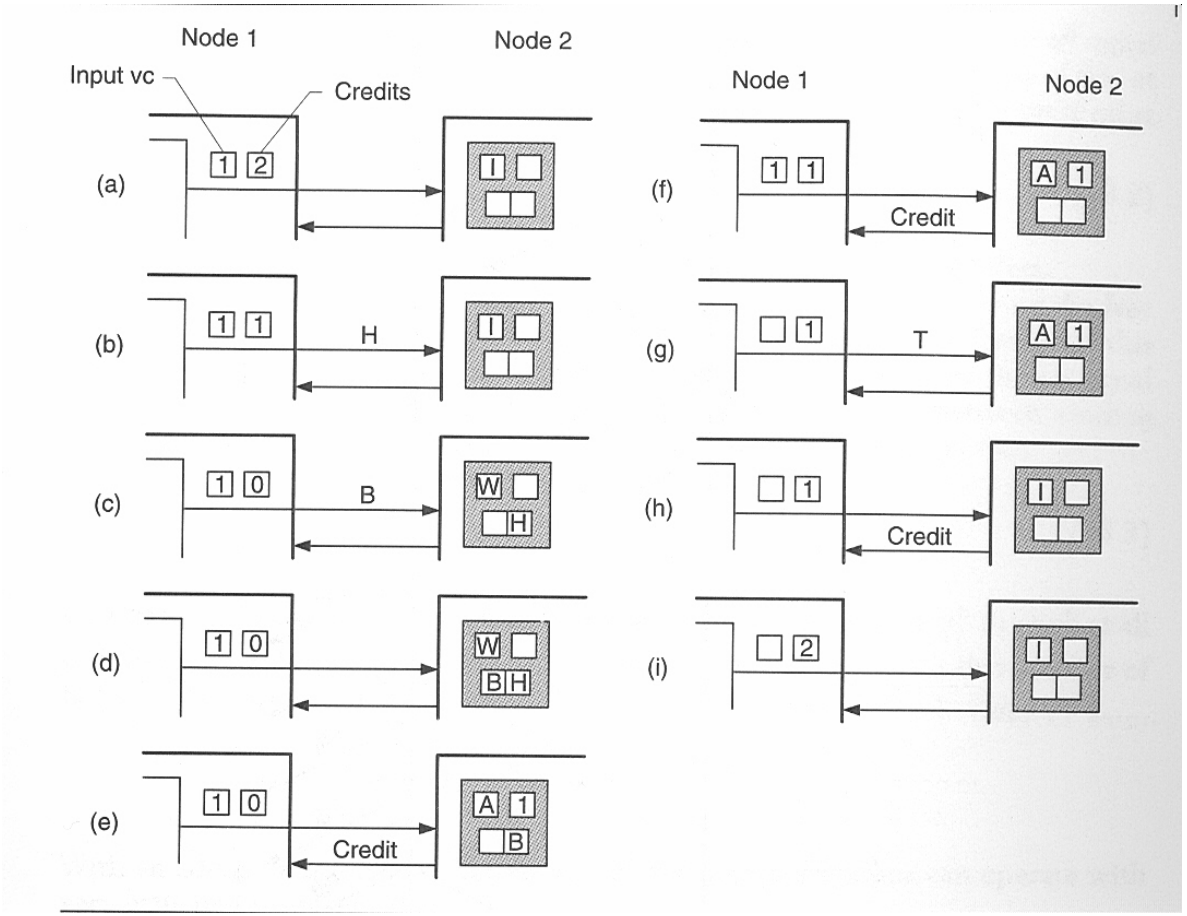


Credit-Based Flow Control



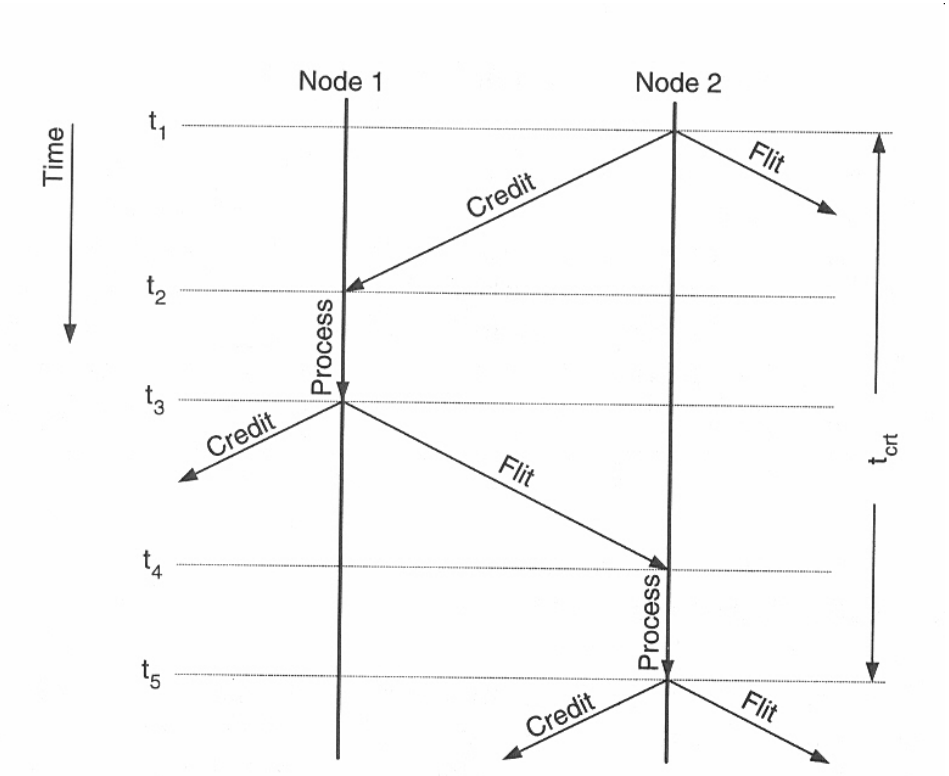
- The upstream router keeps a count of the number of free flit buffers in each virtual channel downstream
- Each time the upstream router forwards a flit, it decrements the counter
- If a counter reaches zero, the downstream buffer is full and the upstream node cannot send a new flit
- If the downstream node forwards a flit, it frees the associated buffer and sends a *credit* to the upstream buffer, which increments its counter

Credit-Based Flow Control



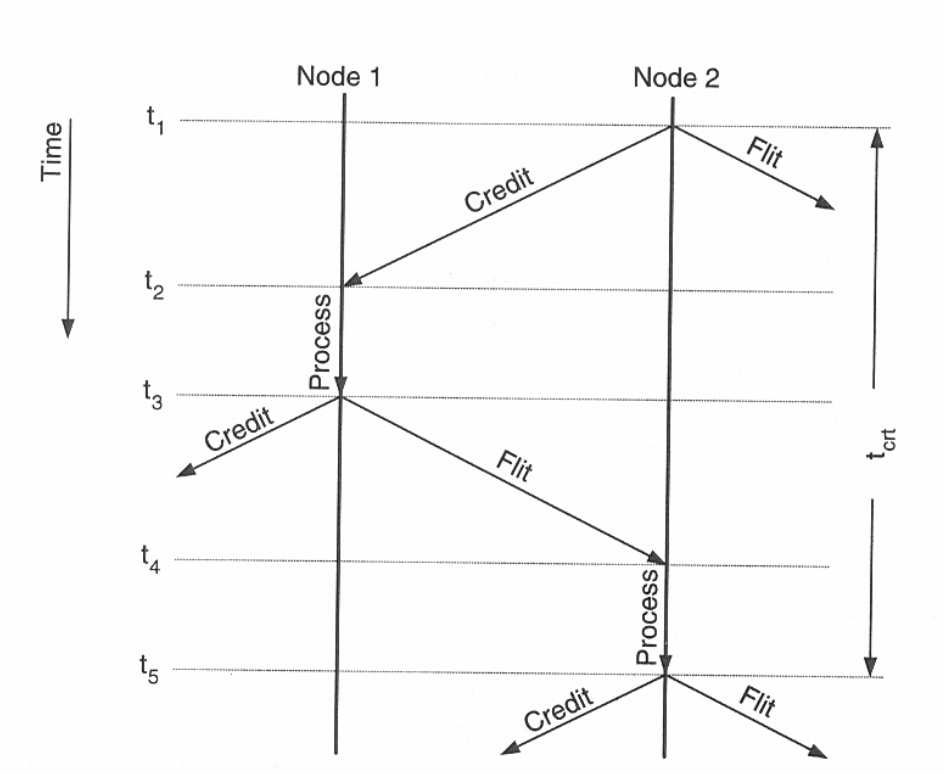
Credit-Based Flow Control

- The minimum time between the credit being sent at time t_1 and a credit send for the *same* buffer at time t_1 is the *credit round-trip delay* t_{crt}
- The credit round-trip delay including wire delay is a critical parameter for any router because it determines the maximum throughput that can be supported by the flow control mechanism



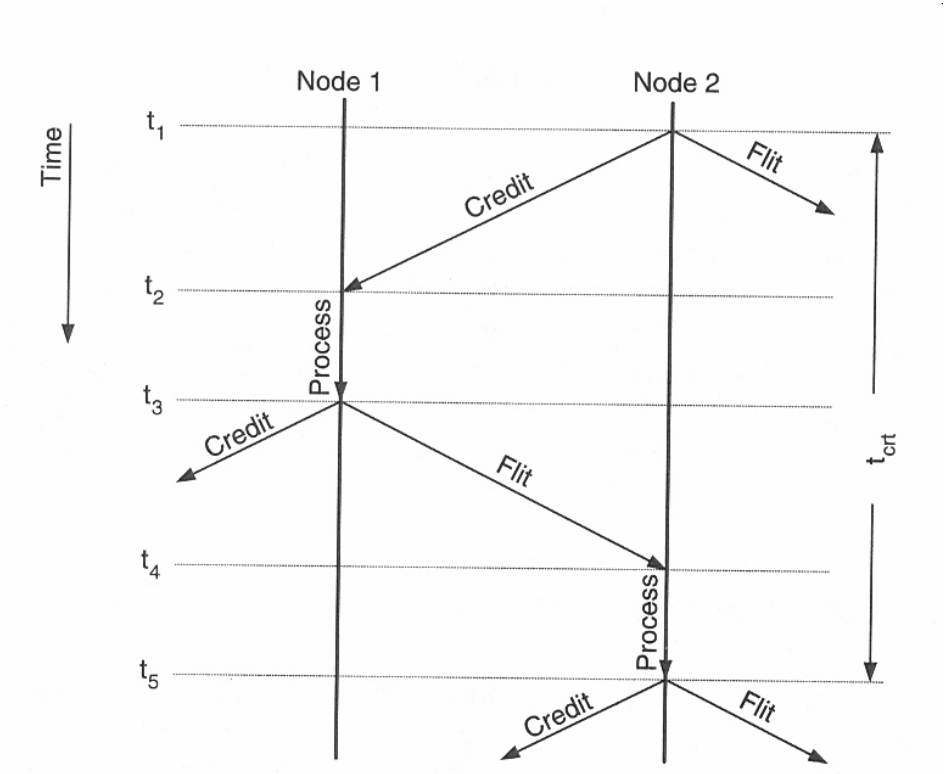
Credit-Based Flow Control

- If there would only be a single flit buffer, a flit wait for a new credit and the maximum throughput is limited to one flit for each t_{crt}
- The bit rate would be then L_f / t_{crt} where L_f is the length of a flit in bits



Credit-Based Flow Control

- If there would be *F* flit buffers on the virtual channel, *F* flits could be sent before waiting for the credit, which gives a throughput of *F* flits for each t_{crt} and a bit rate of FL_f / t_{crt}

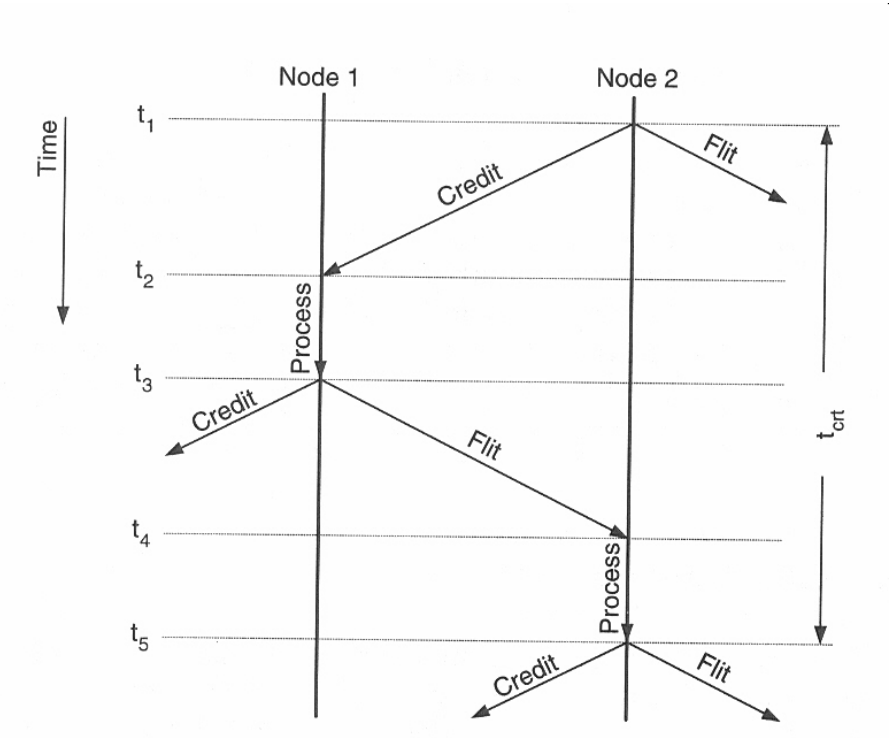


Credit-Based Flow Control

- In order not to limit the throughput by low level flow control the flit buffer should be at least

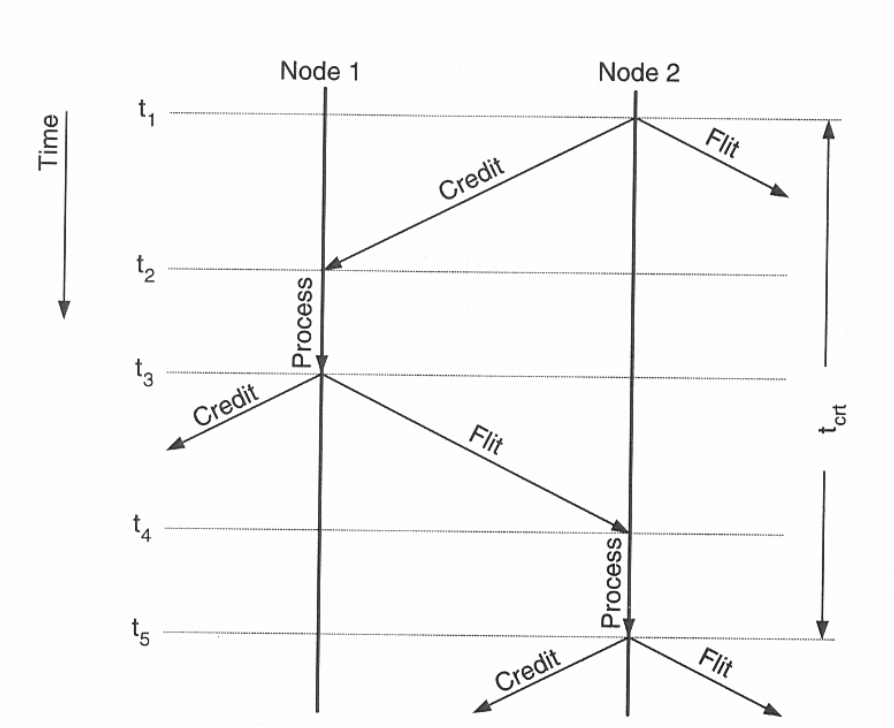
$$F \geq \frac{t_{crt} b}{L_f}$$

where *b* is the bandwidth of a channel



Credit-Based Flow Control

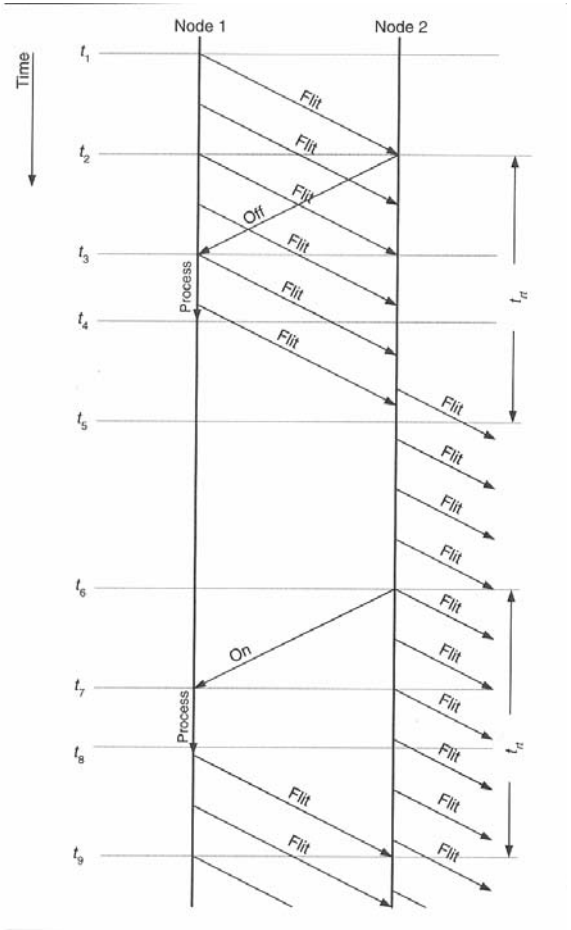
- For each flit sent downstream a corresponding credit is set upstream
- Thus there is a large amount of upstream signaling, which especially for small flits can represent a large overhead!



$$F \geq \frac{t_{crt} b}{L_f}$$

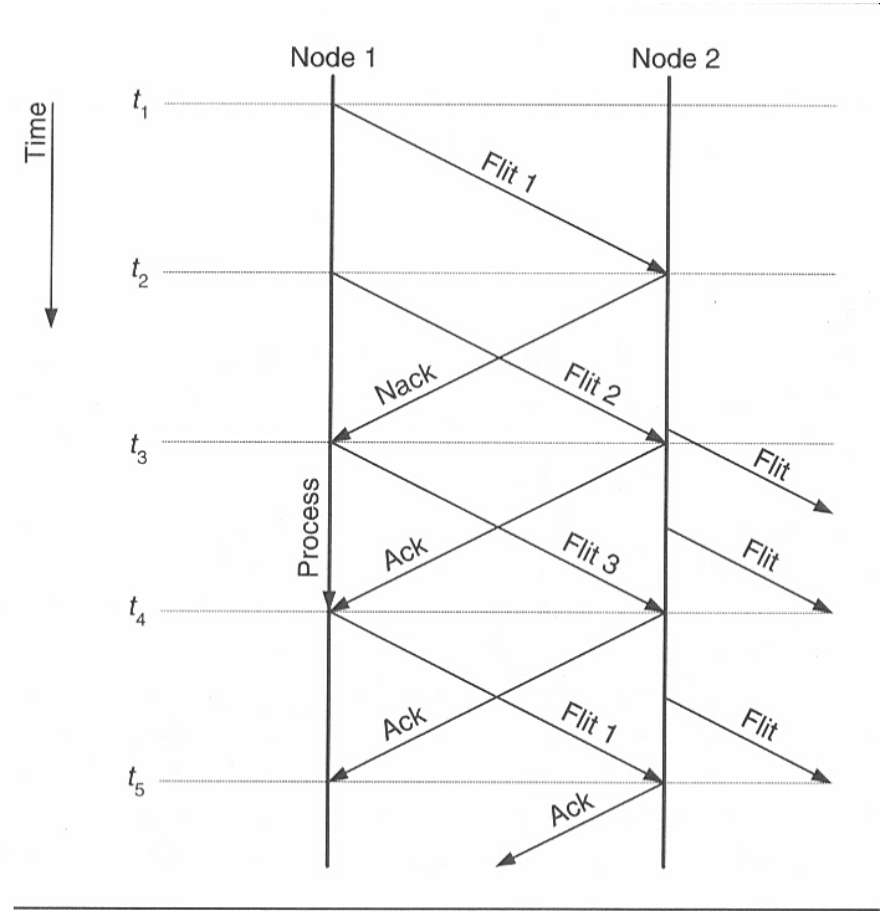
On/Off Flow Control

- On/off Flow control tries to reduce the amount of upstream signaling
- An *off* signal is sent to the upstream node, if the number of free buffers falls below the threshold F_{off}
- An *on* signal is sent to the upstream node, if the number of free buffers rises above the threshold F_{on}
- With carefully dimensioned buffers on/off flow control can achieve a very low overhead in form of upstream signaling

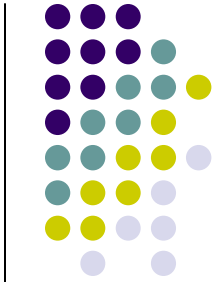


Ack/Nack Flow Control

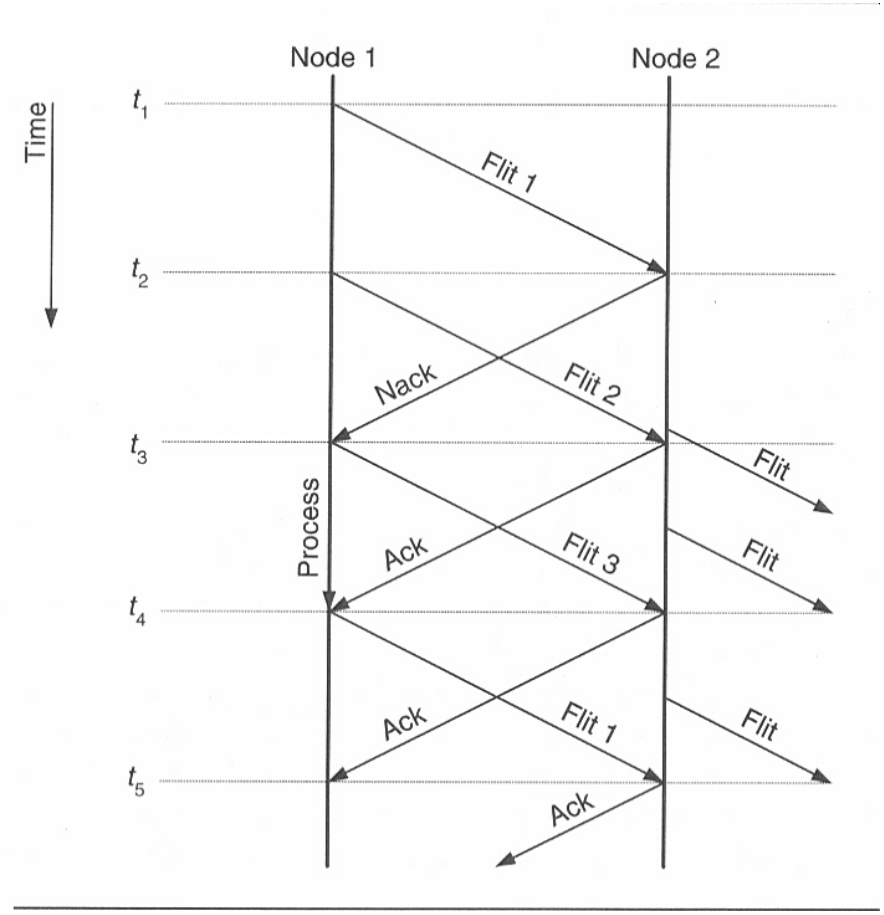
- In ack/nack flow control the upstream node sends packets without knowing, if there are free buffers in the downstream node



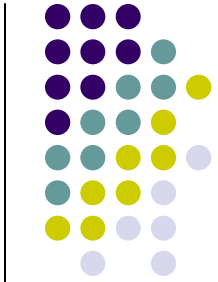
Ack/Nack Flow Control



- If there is no buffer available
  - the downstream node sends *nack* and drops the flit
  - the flit must be resend
  - flits must be reordered at the downstream node
- If there is a buffer available
  - the downstream node sends *ack* and stores the flit in a buffer

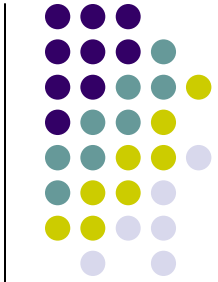


Buffer Management



- Because of its buffer and bandwidth inefficiency *ack/nack* is rarely used
- *Credit-based* flow control is used in systems with a small number of buffers
- *On/off* flow control is used in systems that have a large number of flit buffers

Summary



- An efficient flow control is very important for the performance of the network
- In bufferless flow control some packets might be dropped or misrouted
- In buffered flow control packets or flits can be stored in buffers
- Flit-based virtual channels lead to a better utilization of the network
- Buffered flow control need some overhead in form of upstream signaling