Flow Control

Recall that a topology is defined by π, (the set of all nodes and channels), and ideal throughput Θ (theta).

Both choice of routing algorithms and flow control can reduce throughput from ideal max.

Flow control provides [1] contention resolution through arbitration and [2] resource allocation (channels and buffers).

What is to be routed?

Message is the logical unit of communication

=> Messages are subdivided into packets containing metadata such as header and sequence number. With the sequence number, messages can later be reassembled if packets are routed differently (for load balancing via path diversity)

=> Flits (flow control unit) are the smallest unit that can be routed individually. Flits from the same packet all take the same route and must arrive in order (no sequence number for reassembly). Flits are categorized into the following types:

[1] Head flit: sets up channels for the remaining flits in the packet to use

[2] Payload flit: contains only data (no routing information)

[3] Tail flit: deallocates routing resources

Keep in mind that a packet can be split across multiple routers while the individual flits are transiting.

Optimal flit length => length = T-roundtrip * BW where BW is the bandwidth in bits/s, and T-roundtrip is the roundtrip time between source and destination routers

2 types of flow control

Bufferless: without buffers, the routers cannot send back pressure since they can't hold the packet while the network is stalled

Dropping flow control => packets use routing resources at the beginning of their paths, and they are dropped if there is a collision. This wastes routing resources.

Misrouting => if the desired channel isn't free, then the packet is routing to some other direction. This method has no guarantees on packets arriving to their destination.

Circuit switching => attempts to set up a dedicated route by sending only the header flit until the destination sends an ACK. Then with the channel locked, it sends the rest of the packet.
This method is characterized by a latency of L = T-s (serialization latency) + 3*T-n (hop latency) + T-c (contention latency)

Disadvantages: channel is idle while the path is being locked. The long setup time is bad for short messages.
Advantages: useful for longer packets as the flow on the locked channel is stable.

Buffered flow control:

Store and forward => each router waits until it completely receives the packet before beginning to transmit to the next router.
Pros: good for data integrity operations, e.g. checksums/error correction
Cons: bad for storage as each router must have enough buffer space for an entire packet
See notes for latency equation.

Credit based flow control => each channel has a counter that tracks availability in the next router's receive buffer. Next router periodically issues tokens that credit more space back to the sender.
Pros: better when bandwidth is scarce as transmission is only attempted when there is space guaranteed.

Optimistic flow control => router optimistically sends packets/flits to next router hoping it will be received. If the target has space, it sends an ACK, else NACK. When the sender receives the ACK, it flushes the packet out.
Pros: better when latency is most important since no overhead is required for credit transmission

Virtual cut-through flow control => similar to store-and-forward except that if the next hop router is available, the router begins transmitting the packet as soon as possible (a virtual cut through to the next router). Buffer size must still be the same as store-and-forward in the event the next hop channel is blocked, backpressure is applied, and the packet must be stored until there is no more contention.

Wormhole flow control => same as virtual-cut-through flow control, but the unit handled is the flit.

From here on we will expound on the topics covered above from another source….

Flow Control

When a message that needs to be routed from a source node S to a destination node D, the flow control method chosen must reserve resources such as channel bandwidth and/or buffers to accommodate transit

Load balancing may be accommodated for by packetizing messages and using alternate paths from packet to packet

Packets also need to reserve control states from the node in order to be appropriately sent

Packets can be subdivided into flits. Recall packets have sequence number and may be reordered at destination.

Flits do not have sequence numbers and thus be sent sequentially, but they do have virtual channel IDs so the destination router knows how to reconstruct the packet.

Head flits request flow control resources to be allocated. Tail flits deallocate resources.

Types of flow control

Circuit switching => (bufferless) [1] head is sent to destination, [2] ACK is sent to source, [3] payload and tail flits are sent to destination, and tail flit deallocates resources from source node. Latency of establishing a link is characterized by a min of 2*H*T-router (consistent with above description: this term just represents a 2*T-n component where the remaining 1*T-n comes from sending the beginning of the payload)

Store and forward => (buffered) see above for description. Expounding on performance: this method suffers from lack of pipelining. Latency is given by H * L/b, where L is the length of the packet and b is the bandwidth. A good flow control method should have H and L/b summed rather than multiplied. Hence for long routes, this method incurs lots of delay.

Virtual cut-through => (buffered) attempts to resolve latency issue of store and forward by allocating and transmitting to next hop as soon as the header flit is received. Disadvantage is that a CRC for error correction would not reach until the final destination which may lead an error to propagate. And still, the channel is held for the entire length of the packet.

Wormhole cut-through => (buffered) advantage is that granularity has shrunk to the size of the flit. Looking at the latency from store-and-forward, H*L/b, this realizes a lower latency as L is now the length of a flit. Still the channel is held for the entire length of the packet and the problem may be compounded as flits may span multiples nodes.

Enter virtual channels…

Virtual channels are independent logical channels that are mux'ed on a single physical channel. Originally developed to prevent deadlock in networks which use wormhole routing.

(this is distinct from Head of Line (HOL) blocking which serves to resolve input blocking at the destination node by creating queues for each source output. Virtual channels resolve blocking entirely.)

Virtual channels split up buffers as well as channel bandwidth so two packets can transit the same path at the same time

Architecture of a node employing Virtual Channels

Figure 8 shows an example block diagram employing VCs.

Virtual channel buffers are placed at the input to the node.

Buffer state for the next node is placed at the node's output ports.
Buffer state contents:
- count of the next node's available buffers for a particular VC
- bit to indicate if that VC has been allocated

Crossbar connects inputs and outputs.

States: {I = Idle, R = routing, V = VC allocation, A = active, T = tail deallocation}

R => router gets a head flit and tries to find a physical output channel
V => router attempts to give a head flit a VC
A => router sends the body flits
T => router sends the tail, deallocates VC, and goes back to R or I

Referring back to the buffer state contents.
- If the count reaches 0, then the source router must hold the head flit.
- Once the next hop router gets space, it issues a credit back to the source router either via sideband signal or piggy-backing off of another flit.
- The pending flit can now be sent.
- Once the source router reaches the tail flit, the VC is deallocated

VC allocation
1. Head flit arrives at the input port and is placed in the head register corresponding to the VC
2. Router generates a direction vector based on the routing relation for the indexed by the contents of the head register.
   Direction vector corresponds to a list of output ports ordered by preference that could be used by the packet.
3. Another module reads the state of the allocate bits on all VCs of all output ports to determine availability.
4. The router must then perform bipartite matching (matching between 2 disjoint sets)
5. Once the matching is done, the incoming packet is allocated to the VC

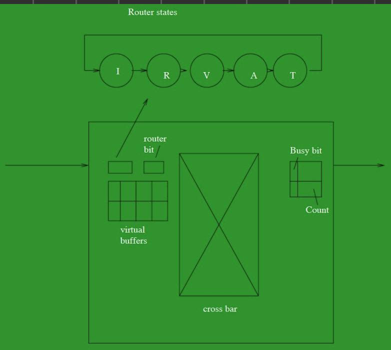

Figure 8: A canonical node/router using virtual channels

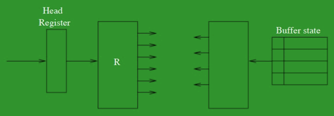

Figure 9: Details of Implementation of VCs

Switch bandwidth allocation
=> This is accomplished as another bipartite matching problem between input ports and output ports.

Output channel bandwidth allocation
=> Bandwidth is split between VCs using arbitration. Arbitration scheme determines the fairness of the split. Can be greedy, fair, priority based, and/or incorporate max queueing time.

VC buffer sizing
=> (RTT + think time at src + think time at dest) * bandwidth
=> This calculation finds the appropriate size. Smaller sizes result in local buffer emptying before upstream router data finishes transit to local node thus idling resources. Larger sizes result in block packet occupying fewer nodes and marginal improvements to throughput.

Virtual Channel Use Cases

Ideally networks should be non-interfering. This means one node getting clogged should not cause another node to get clogged. In practice what happens is one node gets clogged by being sent traffic several times what it can handle causing neighboring nodes to slowly start getting clogged. This is referred to as *tree saturation*

Using virtual channels can address tree saturation by allowing traffic to flow through to a neighboring node despite a block existing for one packet to the neighboring node.

VCs also support QOS e.g. traffic may be low priority like file transfers or high priority like system/control messages. Low priority traffic may be interrupted to accommodate short bursts of high priority traffic with respective VC's allocated for each type. Thus low latency is afforded for high priority while high throughput is still maintained for low priority.

Following charts depict the above use cases.

Not covered in these notes yet are:

=> Flit Reservation Flow Control (see "08 flow control.pdf")

=> Time Domain Multiplexing (see "08 flow control.pdf")

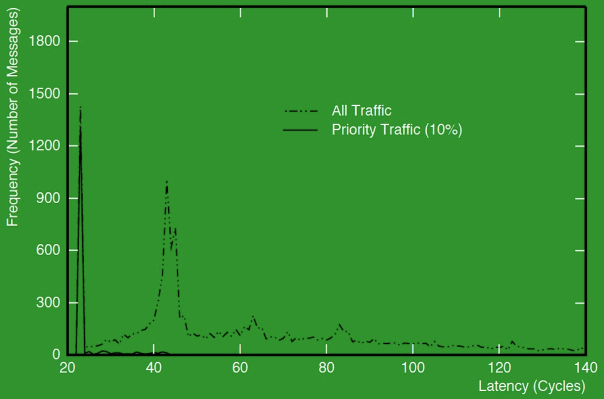

Figure 10: Latency of High Priority vs Low priority Packets

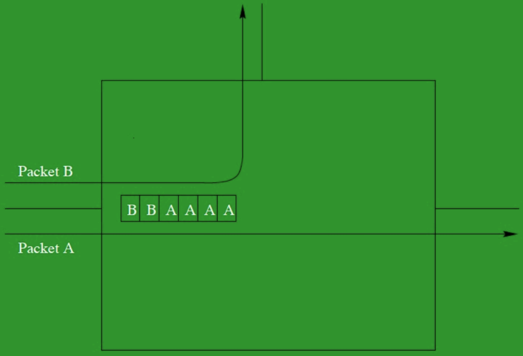

Figure 11: Throughput without VCs

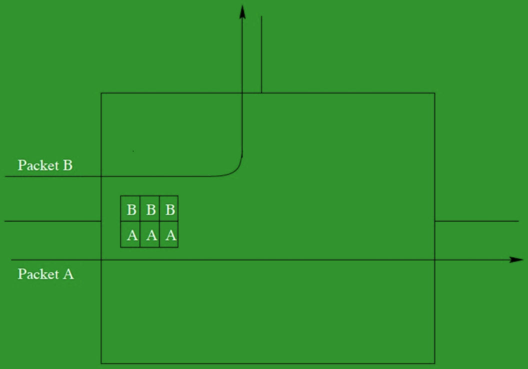

Figure 12: Throughput with VCs