EE482: Advanced Computer Organization Lecture #7
Interconnection Networks Architecture and Design
Stanford University Wednesday, 2 May 2001

Flow Control

Lecture #7: Wednesday, 25 April 2001
Lecturer: Kelly Shaw
Scribe: Ben Serebrin and Dimitris Papadopoulos

1 Class administrative details

1.1 Handouts

Scribing for lecture #3 and solutions to Homework 1 were distributed.

Please remember to show your reasoning for homework assignments.

1.2 Research Paper

The research paper is due on May 16. Sources of papers include

- ACM Digital Library: www.acm.org/dl (If you view from the Stanford domain, you have free access to everything.)
- Cora: cora.whizbang.com
- Citeseer: www.citeseer.com
- IEEE online: www.comsoc.org/pubs/diglibrary.html

2 Flow Control

2.1 Review

A network has a basic topology π and a best possible throughput Θ.

A routing algorithm can reduce the throughput from the ideal Θ.

Flow control is analogous to parking lots and signaling. Flow control can also reduce the throughput from the ideal Θ.

Flow Control provides

- Contention resolution through arbitration
- Resource allocation (channels & buffers)

Flow Control tries to use resources efficiently.

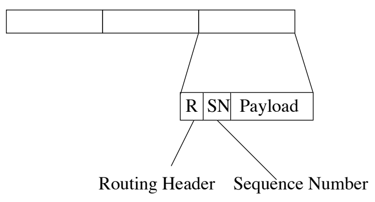

Figure 1: A message with 3 packets

2.2 What will we route?

Message: a message is a logical unit of communication. A packet is the smallest thing with a header and a payload. We break a message into *packets*, which are the basic unit of routing. Packets contain *sequence numbers* for reassembly.

Why break a message into packets? It allows buffering to be done on packet-size units rather than the larger message-size unit.

Note that each packet in a message can have different routing from the other packets. This takes advantage of path diversity to provide load balance.

Buffers are located in the input channel for our model.

Flit: packets may be broken down to flits (or they may not be). The flit is the smallest unit that must have its transfer synchronized. Flits from the same packet all take the same route.

Flits are fixed size, providing the following advantages:

- The buffer size is bounded
- Less overhead is required because flits are fixed size
- Flits can be interleaved on Virtual Channels

There are 3 types of flit:

Head flit sets up channels for the remaining flits in the packet to use

Payload flit contains only data (no routing information)

Tail flit deallocates the routing resources

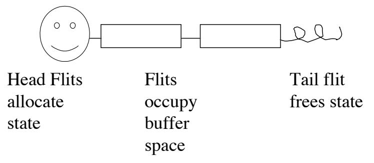

Figure 2: A happy packet

A packet may be denoted as HP*T in regular expression syntax.

Note that flits arrive in order within the packet.

A packet can be spread across multiple routers as each of its flits is at a different position along the chain of routers that the packet is using.

2.3 Optimal flit length

We need to choose the size of a flit so that it allows the best backpressure flow control.

The length of the flit is round-trip time (for the path from one router to the next and back) times the bandwidth. (seconds * $\frac{bits}{second}$ = bits) This is because this is the amount of time it takes a router to receive a packet, realize that it must halt transmission because of a flow control stoppage, and then to send the flow control message (*backpressure*) to the previous router.

Length of flit = $T_{roundtrip} * BW_{fwdirection}$

Note that $T_{roundtrip}$ does not include the serialization latency–it only takes into account the header transmission time.

2.4 Two types of Flow Control

Bufferless Without buffers, the routers cannot send backpressure, since they cannot keep the packet around while the network is stalled.

Dropping Packets use routing resources at the beginning of their paths, but are dropped if they collide along the path. This wastes the resources used in the first part of the path.

Misrouting If the desired channel is not free, then send the packet in some other direction. This increases latency–the packet is not guaranteed to ever arrive

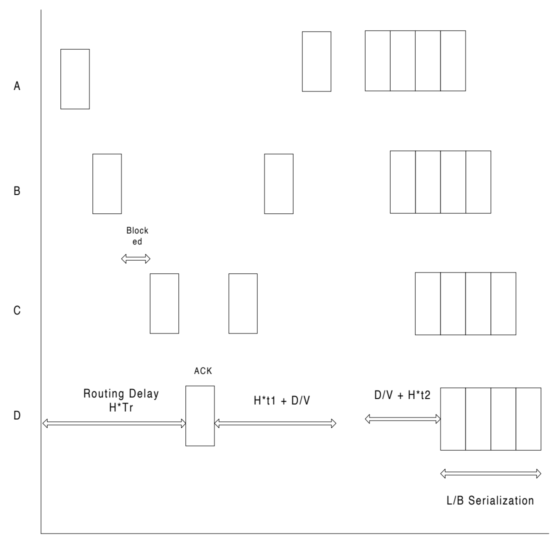

Figure 3: Circuit Switching

at its destination.

Example of bufferless Flow Control: *circuit switching*

Circuit switching sets up a dedicated route by sending only a header flit to the target, which responds with an ACK flit. After that, the source can send data freely to the target.

The latency of the message is
Latency $= T_s + 3T_n + T_c$
Latency = (serialization latency) + (time for hops) + (contention latency)
See Figure 3.

If there is contention, the source keeps sending routing probe flits until the channel is obtained. Since we do not send payloads until we have obtained a dedicated channel, we do not need buffers.

Disadvantages: while we set up the channel, we are using up resources that we have allocated along the path. If nothing is being sent while the route still exists, then the route resources are wasted. The long setup time is bad for short messages since it represents a substantial portion of the entire transmission time.

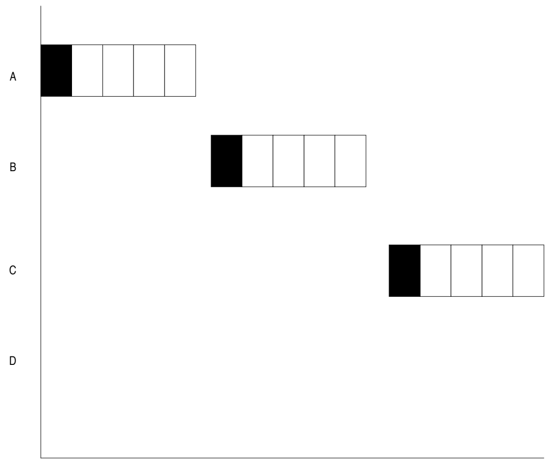

Figure 4: Store and Forward Flow Control

Advantages: This method is more efficient for longer packets. It is also easy to set up and control. After the channel is set up, we are guaranteed the bandwidth and resources.

This method of flow control was developed when wires were cheap and storage was not cheap (this is not the case today).

The average time spent blocking is $\frac{L}{2b}$: on average, we arrive after $\frac{1}{2}$ of the previous packet was sent.

The latency due to contention is (number of hops) * (probability of collision) * (delay due to collision).

2.5 Buffered Flow Control

For *packet-switched* flow control, the packets are fixed length and individually routed.

Store and Forward Each router waits until it completely receives the packet before beginning to transmit it to the next router. This is good for checksumming or error correction, but bad because each router must have enough buffering to store the entire packet.

Latency $= H * (T_s + T_r + \frac{d}{v})$, where $T_s = \frac{L}{b}$

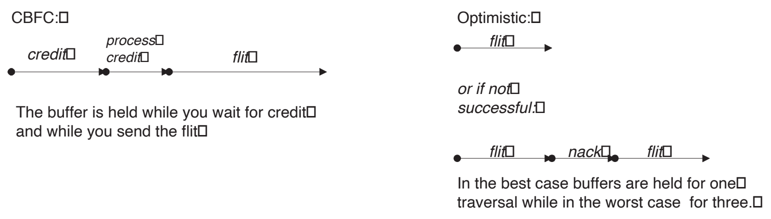
CBFC:
credit
process credit
flit
The buffer is held while you wait for credit and while you send the flit
Optimistic:
flit
or if not successful:
flit
nack
flit
In the best case buffers are held for one traversal while in the worst case for three.

Figure 5: Credit-based Flow Control

The flow control must check to see if there is sufficient buffer space available on the next router before transmitting to it.

See Figure 4.

Credit Based Flow Control Each channel at each router has a counter that decrements each time a buffer is used and increments when a buffer is freed. The router keeps the credit counters for the buffers on the channels at the *next* router; it decrements when sending and increments the counter when it receives a credit from the next router. A router will transmit to its neighbor only if it has sufficient credits along that channel. See Figure 5.

Optimistic Flow Control There is a pool of buffers within each router and a router will always send its packet, hoping that it can be received. If the target receiver has sufficient buffer space, it sends an ACK. If not, it sends a NACK and the sender must retry. When a sender receives an ACK, it may remove the packet from its own buffer.

Credit based flow control is better when bandwidth is most important, since packets are transmitted only once, when they are assured of finding buffer space at their target.

Optimistic flow control is better when latency is most important, since as soon as a packet can be buffered, it will be, without overhead for credit transmission.

Virtual Cut-Through Flow Control is a variant of Store and Forward. As soon as a router decides where to send the packet and has a channel open, it starts sending (without waiting to buffer the entire payload).

Latency is $H * T_r + T_s + \frac{D}{V} + T_c$

If there is contention, it is handled as in Store and Forward (incoming data is buffered upon arrival). Buffer size must be the same as in Store and Forward–we must be able to hold an entire packet to allow for backpressure to work.

Wormhole Flow Control is the same as Virtual Cut-Through Flow Control, except we use flits instead of packets. This gives us the same advantages as before, but we can use smaller buffers because we only need to keep flits stored rather than packets. Wormhole Flow Control is analogous to pipelining.

As in packet switching, the good buffer size is $b * T_{rt} * 2$, where the factor of 2 is added because each router requires a think time, and the buffer granularity is a flit-size. (A flit size is $2T_r$, and the actual time required here is slightly higher because of the think time. Here we assume that the think time is less than a flit transmission time.)

Under contention, the flit at the point of contention is held up and a back-pressure control signal is sent back along the path. Flits will be spread out, waiting at each router along the chain until the contention is cleared. Latency is the same as with Virtual Cut-Through.

See Figure 6 for both Wormhole and Virtual Cut-through.

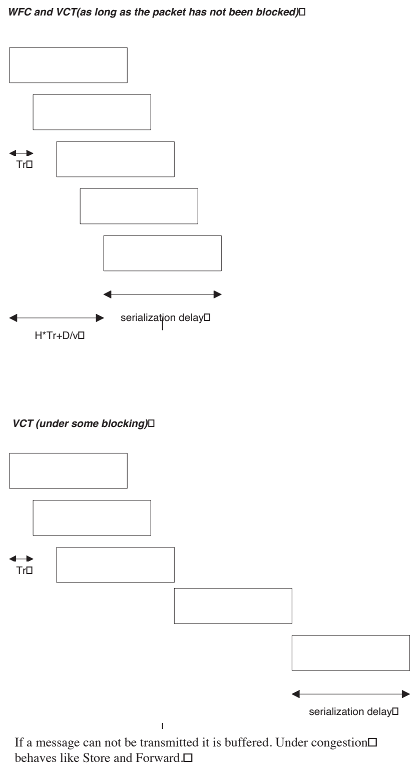
WFC and VCT(as long as the packet has not been blocked)
Tr
serialization delay
H*Tr+D/v
VCT (under some blocking)
Tr
serialization delay
If a message can not be transmitted it is buffered. Under congestion
behaves like Store and Forward.

Figure 6: Wormhole and Virtual Cut-Through Flow Control

2.6 Virtual Channels

Instead of having one FIFO buffer on each physical channel, we have multiple logical channels, each with its own buffer.

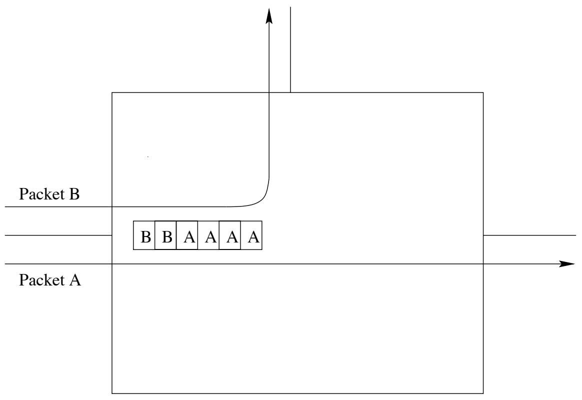

Figure 7: Without Virtual Channels

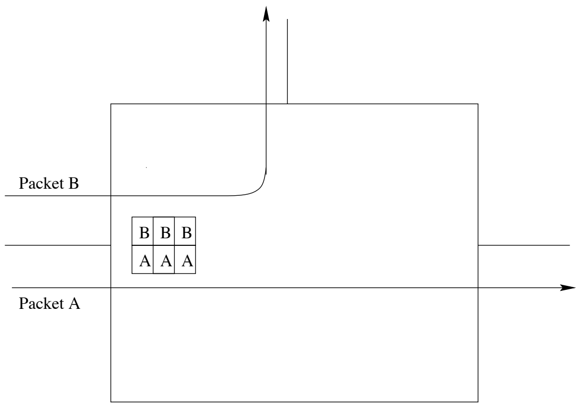

Figure 8: With Virtual Channels

This example demonstrates the advantage of using VCs. Packet A and packet B arrive on the west port as shown, with A arriving first. If A is blocked because the east port is busy, then B may not leave the router, even though its destination, the north port, is free. See Figure 7. With VCs, B goes to a separate FIFO from A and thus may leave while A is still waiting for its port to be freed. See Figure 8.

This has the advantage that if for a given input channel, one virtual channel may be blocked, but the packet behind it on the same input channel may not be blocked and thus may be sent immediately. Without virtual channels, the second packet would be blocked until the first packet was transmitted.