Flow Control Continued

Lecture #6: Thursday, 15 April 1999
Lecturer: William J. Dally
Scribe: Steve Chou, Eileen Peters

1 Review and Questions from Last Lecture

Flow Control: how to allocate resources.

Topology: the physical layout of the network

Routing: the selection of which channels to follow from the source to destination. It happens once.

Flow control happens for every flit: The idea is that we need wire bandwidth and a place to put the flits.

As can be seen in figure 1, one can visualize flow control in three steps:

1. The payload of the packet is stored into the store buffer

2. The head of packet is stored into the state buffer.

3. Bandwidth is allocated (channel)

2 Units of Routing

In this section, we define our main routing units: message, packet, and flit.

2.1 Message

A message is the logical unit of communication. It may be composed of multiple packets. (See Figure 2.) Messages are logically complete and usually perform a logical function. One example of a "message" is a server request when an application is communicating with a server.

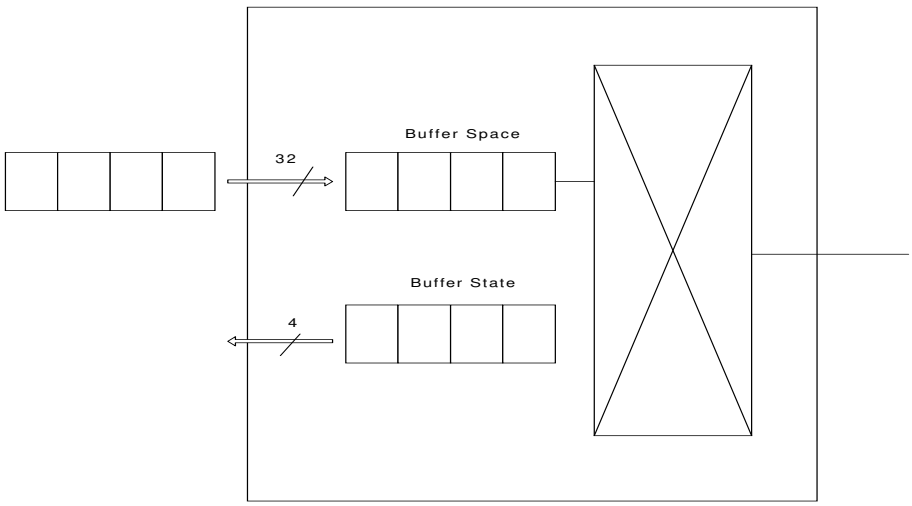

Figure 1: A buffer entering a buffer space and buffer state.

2.2 Packet

The packet is the basic unit of routing. It is the smallest thing that carries information whereby it can find where it is going (the header), as well as a payload containing the message being sent. A message may be broken into packets for several reasons.

1. A message may be too large for the buffer to store it. Using packets allows you to bound the size of the buffers.

2. A message may exceed the channel size.

3. Breaking a message into packets allows for more routing flexibility. We can send packets from the same source to the same destination but over multiple paths. This way, even within one message, breaking it into packets allows for path diversity.

2.3 Flit: Flow Control Digit

Packets are then further broken down into flits or flow control digits. The basic definition of a flit is that it has no overhead – it is all payload.

(See Figure 3). Breaking a packet into flits allows the following advantages:

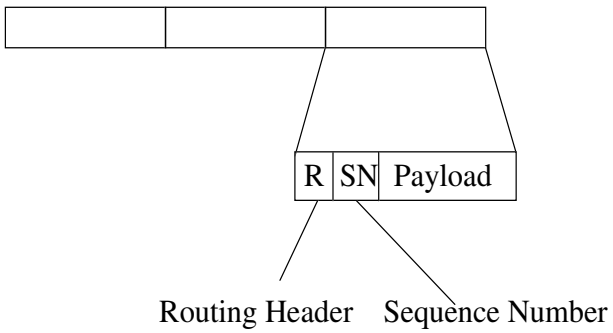

Figure 2: Components of a message.

1. Breaking the packet into smaller sizes allows flits to be interleaved.

2. Breaking a packet into flits allows us to ensure a specified size for a basic unit.

3. This is less overhead because the router knows exactly how long a flit is. (c.f. each packet has a routing header and sequence number). Although we defined a flit as having no overhead, it actualy has a little information that it carries to distinguish virtual channels. Virtual channels will be described in more detail in lecture 7.

Flit components Flits do not have to carry header information. Each individual flit may have a separate function. For example, a head flit allocates the state and arbitrates for the channel. A payload flit carries information and the tail flit is responsible for closing any allocated channels. The tail flit frees the allocated buffer state, and allows the next head flit coming to occupy that state. (see Figure 3)

Flit navigation How does a flit know where it is going? You can think of it as holding onto the tail of the head flit. Also, flits of two different packets can interleave, all travelling over the same wire.

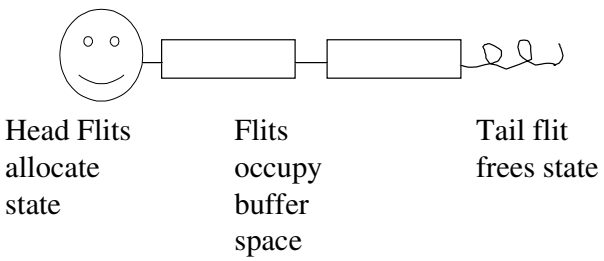

Figure 3: Components of a flit: head, body and tail.

The length of the flit should be equal to the round trip time times the bandwidth (seconds * bits/second = bits):

Length of flit = $(T_{roundtrip} * BW_{fwddirection})$

A flow control unit should be sized for the time it takes to do something, e.g. to send back pressure. If the flit size is made larger, granularity is lost.

Example: A 32 bit wide channel (W = 32) is running at one gigahertz (B = 1 GHz). If it takes four clock cycles to send a message across and back ($T_{roundtrip} = 4$), then the flit size should be (32 bits/CC) * 4 CC = 128 bits.

3 Circuit Switching

Figure 4 depicts a simple circuit switching example, which consists of 3 main steps.

1. The sender transmits the head flit.

2. When the head arrives at the destination, a message is relayed to the source that it has the channel.

3. The sender receives the head acknowledgement and sends the rest of the packet.

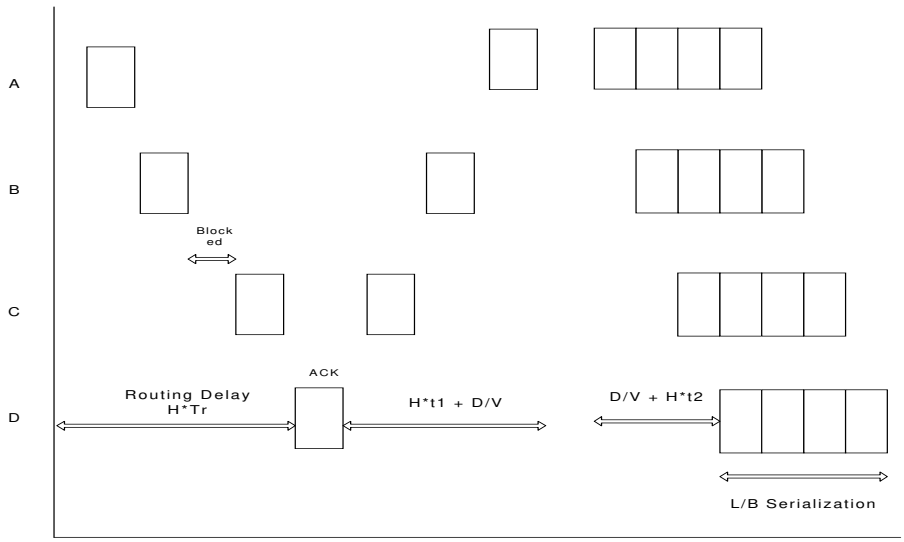

Figure 4: Circuit switching time-space diagram.

Details In circuit switching, the idea is that we send the head first down the network and keep sending clones of the head until it reaches the destination. Once the head reaches the destination, it uses a backward channel to tell the router at the source node to begin sending the payload. In Dally's example illustrated in figure 4, he points out that it takes three traversals of the network to establish a route before any useful work is accomplished. Using the assumption that each hop takes 4 cycles, it takes 36 cycles to establish the circuit.

Latency= $T_s + 3T_h + T_c$

$=$ serialization + 3*T_{hop} + contention.

$T_s =$ L/b

If a head flit is blocked, the part of the channel already allocated is held for a while, but no work is being done yet. When this occurs, certain resources will be held for an indefinite period. Since the head flit is constantly transmitted, a route is eventually established through brute force.

For example: The channel from A to B will be held if the head is blocked at B waiting for C. In circuit switching, the routing resources are held for a certain minimum

period in which there is no work done. Only once the route has been fully established do packets traverse the network. Circuit switching was developed when wires were cheap and storage was expensive. However, nowadays, wires are expensive, so this is not an optimal method. Circuit switching can be viewed through the polite analogy: the sender asks "may I send?", the destination replies "yes you may", and then finally they sender transmits the message.

Circuit switching duty factor depends on the packet length. For example, here in Figure 4, only 4 of the 56 channel times are used efficiently. The efficiency of circuit switching increases as packet length increases.

Average time spent blocking is L/2b (where L is the length of acquistion of the channel). This is the probability of colliding.

Latency due to contention = num. hops * prob of collision * delay due to collision.

Advantages

+ simple to implement

+ The resources are guaranteed for the duration of the packet. Thus we have a guaranteed bandwidth during a certain time. This can be useful, e.g. with Multimedia applications.

Disadvantages

- wasteful use of resources

- high latency of establishing the path because sender is being polite.

4 Store and Forward Switching

Figure 5 depicts a store and forward packet switching example, which consists of 2 main steps.

1. A packet enters a node and the entire packet is buffered. It remains in this buffer until the entire packet has been received.

2. When the whole packet has been received, it is forwarded to the next node. This is useful for such things as calculating check sums from check bits in the tail for error detection and correction.

If a channel is not available, The whole packet stays in the buffer until the channel is free.

Latency = H * (T_s+T_r) + d/v (T_s = L/b)

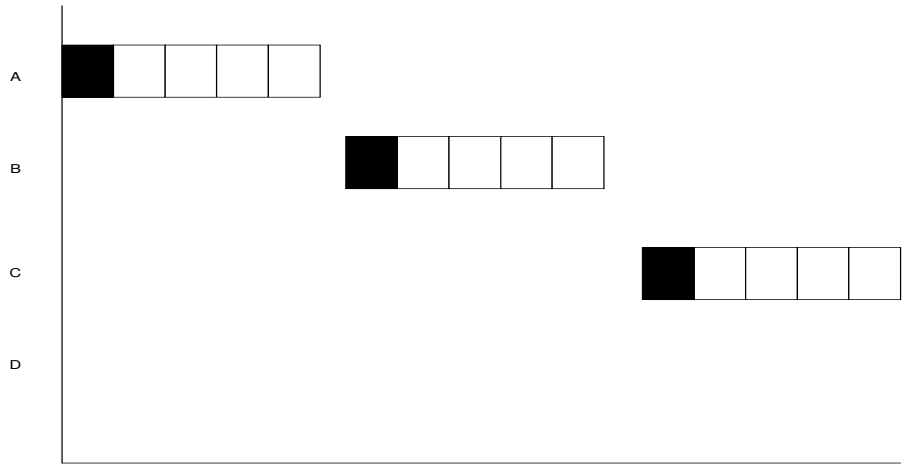

Figure 5: .

How do you know if there is space for a packet at the next node?

There are two methods, credit-based flow control (figure 6) and optimistic flow control (figure 7). To send a packet you need two things: channel and buffer space.

Credit based flow control- Each channel has a counter that is decremented when a flit is sent. Once the counter reaches zero, it indicates that all the buffers at the destination have been occupied. The counter increments when the source receives a credit from the downstream node. Credit based flow control only works if there's a dedicated buffer per channel. (Otherwise multiple senders could spend the last credit at once.)

Optimistic flow control- Optimistic flow control is very similar to credit based flow control except it allows a buffer pool. The sources transmit their packets and hope that there is an open buffer space available downstream.

As shown in figure 7, the sender assumes there is a buffer, sends a packet, and receives an ack or nack. If it is a nack, it retries. It doesn't give up it's own buffer until it receives an ack.

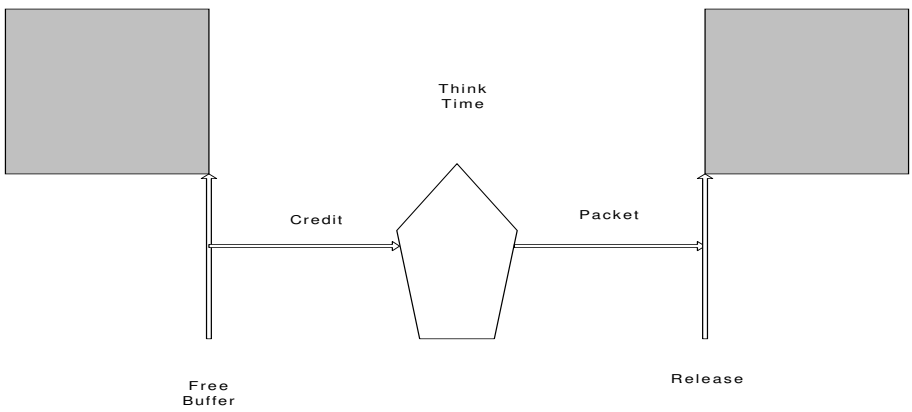

Figure 6: .

Comparison of CBFC and Optimistic With CBFC, as shown in Figure 6, two buffers (source and dest) are held for 2 traversals – one traversal for request, then some think time, and then the packet time before the source buffer is free.

Comparatively, Figure 7 shows Optimistic Flow Control, where in the best case, two buffers are held for just 1 traversal. However, in the worst case, they are held for 3 traversals, so Optimistic is, on average, similar to CBFC.

In circuit switching, certain channels were blocked once communication routes had been established. The head blocks until the full channel is available.

In store and forward, there are elements of real flow control – we have blocking and buffering. Our previous implementation of store and forward flow control involved storing the entire packet before forwarding it on to the next node. A more efficient implementation would be to forward the packet to the next node once the head is received, but before the entire packet is received. This is called virtual cut through routing.

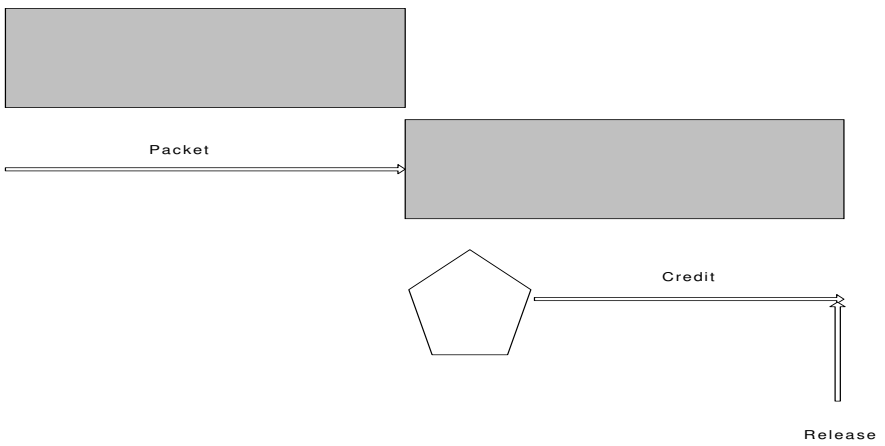

Figure 7: .

5 Virtual Cut-Through and Wormhole Flow Control

Latency decreased In virtual cut-through flow control, latency is decreased because as soon as the head is received, a node immediately begins sending the packet received thus far to the next node.

Latency Equation The following latency equation from store and forward D/v + H*(T_s + T_r) + T_c becomes D/v + H*T_r + T_s + T_c. The effect is to remove the Ts from being multiplied by H – only pay serialization once.

The problem with Virtual Cut-Through is that the buffer size must still be packet-sized.

Wormhole flow control allows for smaller buffers compared to virtual cut-through switching. In wormhole switching, packets are also pipelined. However the packet is broken up into flits.

A Virtual Channel (VC) is a logical channel on the same physical channel (PC). Virtual channels are allocated to packets, while Physical channels get allocated to flits.

How to size the buffer for wormhole flow control

We need twice the storage – what is there and what you need to be able to hold while telling the sender to stop. That is, we allocate only enough buffer storage to hold what you normally hold and to hold what arrives while applying back pressure. a.k.a. 2 latches per clock. A good buffer size is b*T_{rt} * 2. The x2 factor in this equation gives room for calculations.

Flit length should be .5 the size of round-trip delay. The .5 factor gives more granularity.

6 Deadlock

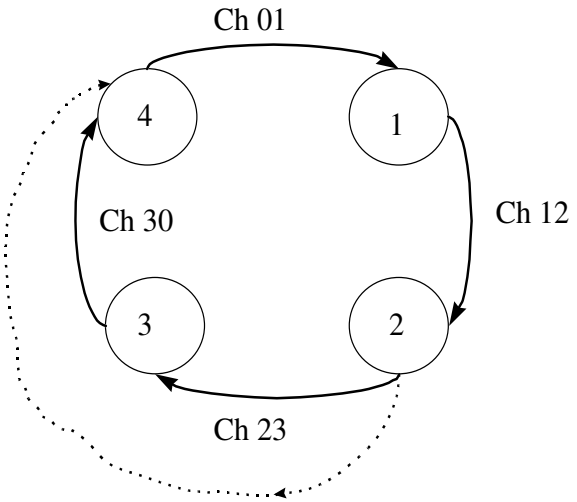

Figure 8: .

In the above example (Figure 9), the network is deadlocked because each packet is waiting for the other packet to get out of the way. As a result neither packet can reach its destination. Now, instead we can route on virtual channels to break cycles that cause deadlock.

With deterministic routing, the routing functino defines the subgraph.

R:C*N goes to C.

Notation explanation: Routing function takes a channel and a node gives a channel.

R(chan, node) goes to chan, chan

Tree saturation- single point of contention (e.g. a cycle) will grow to the rest of the tree/graph. The entire network then becomes blocked. Virtual channels allow us to make a network deadlock free.

Cache Coherent Machines Cycles can result from things other than just individual packets cycling as in the case of cache coherency protocols. For example, a node A may send a request to node B. Node B then forwards this request to a third node, C. Then C sends the answer to node A. Such cycles can result in deadlock situations.

7 Quality of Service (QoS)

In the last few minutes of lecture, Prof. Dally introduced QoS. QoS is just what it sounds like – the quality of service provided to a customer. There can be different standards in latency and reliability. For example, a high-priority customer's packets may never be dropped, but for a best-effort customer, some packets could be dropped, and their service may be slower.

Virtual channels give good performance, and allow for different priorities of QoS (e.g. Best Effort vs. High Priority messsages).