Network Performance Analysis

Lecture #10: Thursday, 6 May 1999
Lecturer: Prof. Dally
Scribe: Derek Taylor(dat@leland.stanford.edu)
Paul Lassa(lassa@relgyro.stanford.edu)
Reviewer: Li-Shiuan Peh(lspeh@cs.stanford.edu)

This lecture covers the topic of *performance analysis* for each of the associated flow control mechanisms:

1. Dropping Flow Control
2. Blocking Flow Control
3. Queueing Flow Control
4. Virtual Channels

1 Announcements

1. The research paper deadline was extended to 18 May.
2. The checkpoint#1 deadline of the project is 11 May. This deadline covers the main part of the actual design work for the project.
3. There was a question on the project about the inconsistency of the signalling frequency (1 Gb/s) as compared to the on-chip clock rate (400MHz). Prof. Dally explained that we can simply assume a rate converter inside the chip that receives the data at the signalling frequency and provides the data on a bus to the chip internals at a rate of 5 bits every 2 clock cycles.

2 Performance Analysis Overview

What is Performance Analysis? It is little more than drawing latency and throughput curves. Figure 1 shows an example Latency curve. It is a plot of latency (dependent variable) versus offered traffic (independent variable) through the network. Note that the throughput of the network can also be determined from this graph at the point where the offered traffic requires an infinite latency. Figure 2 shows a throughput curve. This

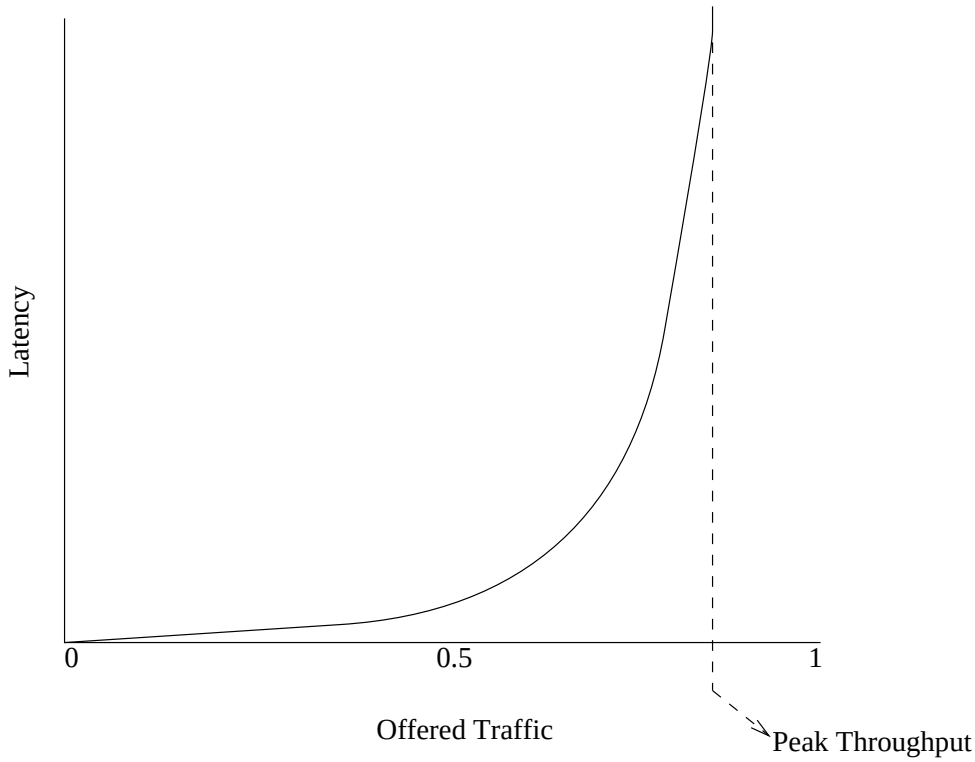

Figure 1: An example latency curve

curve is a plot of the throughput of the network as a function of the number of virtual channels.

Why do we draw these curves? The advantage of these tools is that we can get an intuitive feel for the network's performance without doing any simulation. The analysis helps us understand what kind of tradeoffs we will face in making many of the engineering decisions while we design the network. The tools in this lecture provide the ability to do paper and pencil analysis that help develop this intuitive understanding.

3 Dropping Flow Control Model

The first time we did this analysis (see Lecture #1 notes) we made the simplifying assumption that we always lose (get dropped) whenever we face contention from another packet. In general this is not the case, since we will win half of those times. In this analysis, we don't make that simplifying assumption. It turns out not to be any harder to do it correctly, so we probably should have done it correctly the first time.

What direction should we analyze the network? It is more natural if we analyze the effect of contention starting at the destination and working back to the source. We can derive relationships between the channel loads on the input and output and iteratively

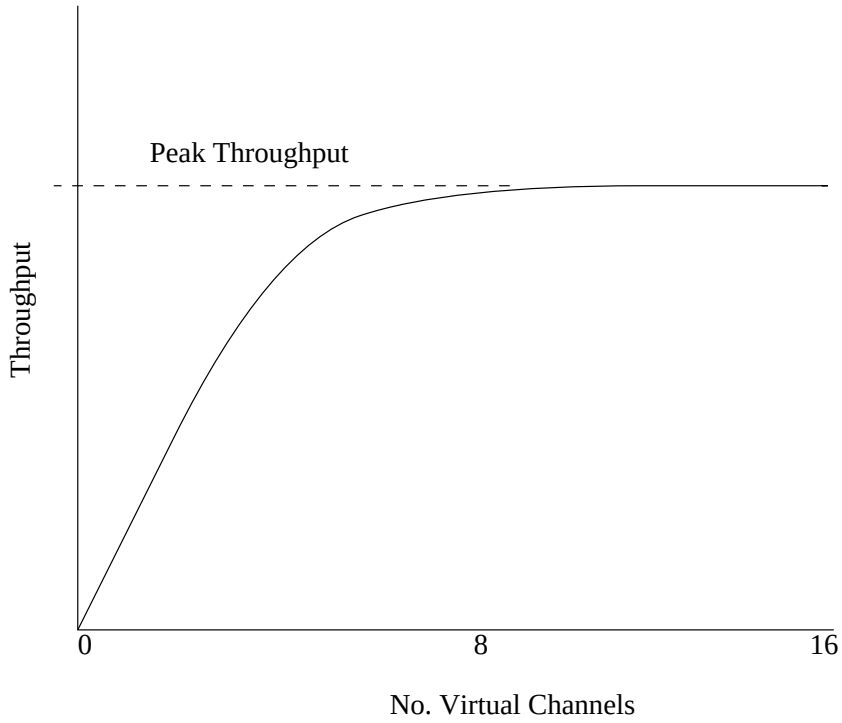

Figure 2: An example throughput curve

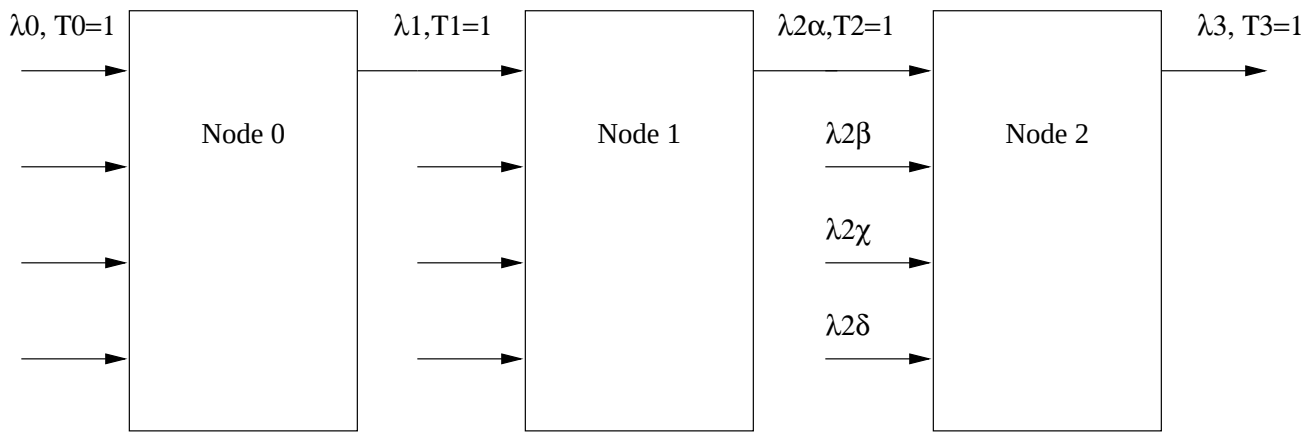

Figure 3: Arrival rate diagram through three nodes

solve the problem for the solution.

To help us analyze a network, we define λ_i to be the rate of packets over the channel (packets/time) and T_i to be the time that a packet occupies a resource. Figure 3 shows this graphically. We normalize this time to one cycle at the beginning of our analysis

4	k	4	k	8	k
0.75	(k-1)/k	0.75	(k-1)/k	0.875	(k-1)/k
lambda_i	I	lambda_i	I	lambda_i	I
3	0.307	8	0.143	3	0.276
2	0.398831	7	0.160179	2	0.363876
1	0.569045	6	0.18205	1	0.533849
0	0.992724	5	0.210837	0	1.001815
		4	0.250438		
		3	0.308356		
		2	0.401122		
		1	0.573721		
		0	1.007042		

Table 1: Spreadsheet analysis using Dropping Flow Control

($T_n = 1$). For each term, we can analyze what happens to the network by writing an expression for λ_i in terms of λ_{i+1} and an expression for T_i in terms of T_{i+1}.

For dropping flow control, we see immediately that $T_i = T_{i+1} = 1$, since messages never hold on to a channel for longer than the time required to transmit them. However, we see that $\lambda_i > \lambda_{i+1}$ since some of the channels are dropped at each stage through the network, determined by the amount of contention faced at each node.

If we assume a uniform bit rate at all the input ports on a node, we can write the probability that the node is busy as

$$P_{busy_other} = \frac{k-1}{k}\lambda_{i+1} = P_{drop}$$

Then we can determine the probability that the packet gets through the node as

$$\lambda_i(1 - P_{drop}) = \lambda_{i+1}$$

$$\lambda_i = \frac{\lambda_{i+1}}{1 - (\frac{k-1}{k}\lambda_{i+1})}$$

One way to analyze this expression is by entering the formula for each stage into a spreadsheet and iteratively solving for the output rate at which the input rate is one. This gives the saturation throughput.

One student also suggested that for this formula, a closed form could be found for λ_{i+1} in terms of λ_i. However, it turns out to be just as easy to solve the problem by using a spreadsheet and watching the results converge.

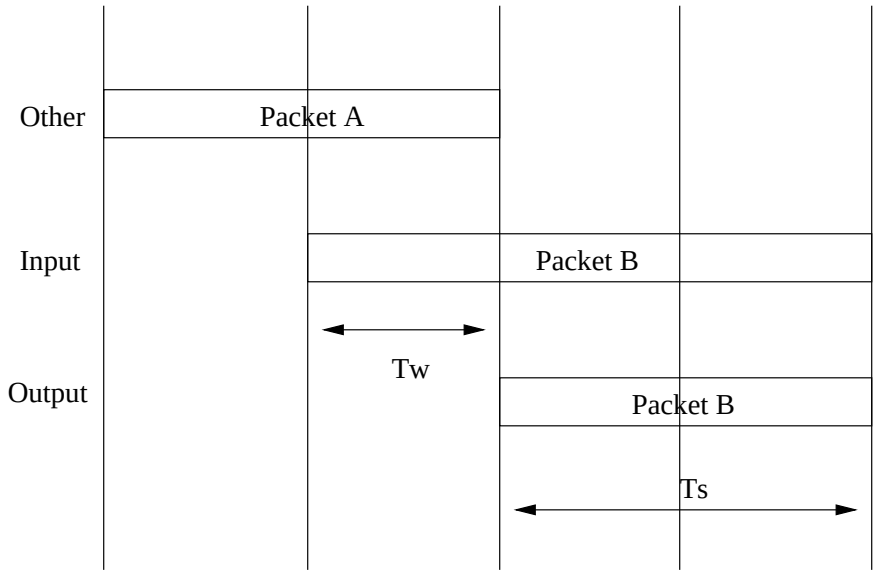

Figure 4: Time diagram demonstrating the average collision in blocking flow control

What happens if we add more stages to the network? From the spreadsheet analysis shown in Table 1, we see that the network saturates at 30.7% capacity for 4 stages. If we increase the number of stages, we also see that the network saturates at 14.3% capacity for 8 stages. As the number of stages increases, the throughput decreases, since there are more opportunities to collide with other packets. Also, as the radix increases, the throughput through each node decreases. Intuitively, there are more potential packets to collide with. Overall, however, increasing the radix is desirable because it results in fewer stages. Usually, the fewer number of stages has greater impact on increasing the throughput than does the increased probability of a collision at a single node.

4 Blocking Flow Control

For this case, we will assume a flow control mechanism that provides for instantaneous halting of a communication channel. That is, when messages block, they queue up within the input channel as if the channel itself had state. We assume no buffering for this flow control methodology.

Figure 4 illustrates the time diagram for the case of collisions in this flow control scheme.

Since the network never drops any packets, we have

$$\lambda_i = \lambda_{i+1} = offered_rate$$

However, in this case, we have increasing delay at each stage due to the increasing probability of collisions (i.e. $T_i > T_{i+1}$).

We can write an expression for the probability of collisions as follows

$$P_{collision} = \frac{k-1}{k}\lambda_{i+1}T_{i+1}$$

Also, to first order, the expected waiting time given a collision is given by

$$E(T_w|collision) = T_{i+1}/2$$

This expression is accurate only to first order since a packet may collide with more than one packet entering the node at the same time. However, in practice, no more complicated model is required since networks can seldom hold a sufficient number of packets to make the second order term dominant.

The waiting time of a packet (T_w) is given by

$$T_{w,i} = P_{collision}E(T_w|collision) = \frac{\lambda_{i+1}T_{i+1}^2}{2}\frac{k-1}{k}$$

Thus, the waiting time at each node is given by

$$T_i = T_{i+1}(1 + \frac{\lambda_{i+1}T_{i+1}}{2}\frac{k-1}{k})$$

This network will saturate when the duty factor equals one, or

$$\lambda_0 T_0 = 1$$

Once, again, the easiest way to analyze this network is on a spreadsheet where we iterate over these equations until the network saturates. We see from the spreadsheet analysis in Table 2 that the network saturates at about 52.5% for 4 stages and about 30% for 8 stages. Thus, the blocking network also worsens as the number of stages increases. The blocking protocol (though unrealistic) operates at not quite two times the performance of dropping flow control. Intuitively, we can understand this by comparing to dropping flow control. In dropping flow control, a packet uses up some number of resources and then gets dropped, completely wasting the resources used. In blocking flow control, resources get used at inefficient levels, but they are never completely wasted since at least eventually the packets will succeed in getting through the network. This wastage in dropping flow control accounts for the difference between the two schemes.

Finally, we should add that this total time due to contention adds to the Latency of the network. From our base equation for latency, we add a new term for contention, yielding the following equation for latency

$$Latency = L/b + D/v + Ht_r + T_c$$

where T_c is the time due to contention. This contention time is the sum of the waiting times at each stage of the network. That is

$$T_c = \sum_{i=1}^{n} T_{w,i}$$

0.525	lambda		0.28	lambda	
4	k		4	k	
0.75	(k-1)/k		0.75	(k-1)/k	
I	T_I	lambda*T_I	I	T_I	ambda*T_I
3	1	0.525	8	1	0.28
2	1.196875	0.6283594	7	1.105	0.3094
1	1.4789	0.7764227	6	1.233208	0.345298
0	1.909495	1.0024848	5	1.392892	0.39001
			4	1.596607	0.44705
0.49	lambda		3	1.864268	0.521995
8	k		2	2.229196	0.624175
0.875	(k-1)/k		1	2.750973	0.770273
I	T_I	lambda*T_I	0	3.545598	0.992768
3	1	0.49			
2	1.214375	0.5950438			
1	1.530515	0.7499525			
0	2.032684	0.996015			

Table 2: Spreadsheet analysis using Blocking Flow Control

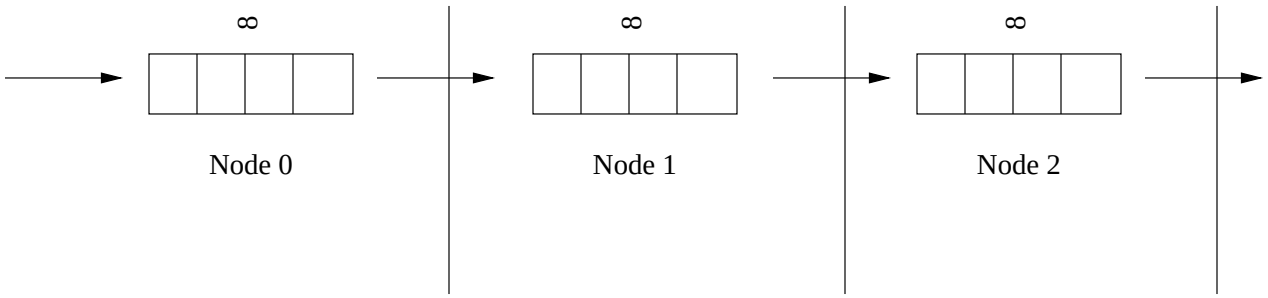

Figure 5: Infinite buffering diagram

5 Buffering Flow Control

This analysis assumes an infinite queue within the network nodes, as shown in Figure 5.

Since we assume infinite buffering and we never drop packets, we have

$$\lambda_i = \lambda_{i+1}$$

Furthermore, since buffering allows us to free input channels as soon as the packet is transmitted, we have

$$T_i = T_{i+1}$$

Furthermore, we can operate this network at $\lambda = 1$, since we always have a place to put the traffic as long as we assume an infinite buffer. For the case where $\lambda = 1$, however, we really will use that infinite buffer to capacity!

The time spent waiting in the buffers due to delay (assuming a M/D/1 queuing model) is given by

$$T_c = \frac{\lambda}{2(1-\lambda)}$$

Note that since all λ are equal, we drop the subscript.

Professor Dally next commented on how publication results sometimes are incorrect in that they show the delay with a finite number of inputs and a finite number of stages saturating at infinity. In practice, however, this cannot be the case since we can easily compute a theoritical maximum to the delay by simply assuming that a packet blocks on every other packet that can reach its current node for every stage in the network. However, the reason why results sometimes show an infinite latency is that they simulators use an infinite queue at the injection channels of the network and the latency that the simulator measures is from the start of this infinite queue rather than the actual injection channel to the network. The students were encouraged to keep this in mind as they read papers on network performance models. Figure 6 demonstrates this difference.

6 Limited Queue Model

Memory is often the dominant cost in real computer systems. Although the price per bit has dramatically decreased over time, the number of bits in a computer has increased at nearly the same quadratic rate! Thus, queues in a real system are never relatively infinite.

Figure 7 shows the model we use for a limited queue node. We designate two waiting times at each node of this network. $T_{i,x}$ is the waiting time at the input to the node. $T_{i,b}$ is the waiting time from the output of the queue to the input of the switch. We know that $T_{i,x} < T_{i,b}$.

As a packet moves through the network in a system with limited queues, the packet is spread out over multiple nodes. When the head blocks, there is some amount of time before the network communicates that fact to the tail. The tail just keeps on chugging along until the buffers on the nodes of the packet are all filled. The student can envision this as a *scrunching* effect. The packet is strung out during normal transmission, and *scrunched* into buffers on the nodes during waiting periods. The buffer size determines the amount of *scrunching* that a packet can take. Figure 8 illustrates this effect on waiting time.

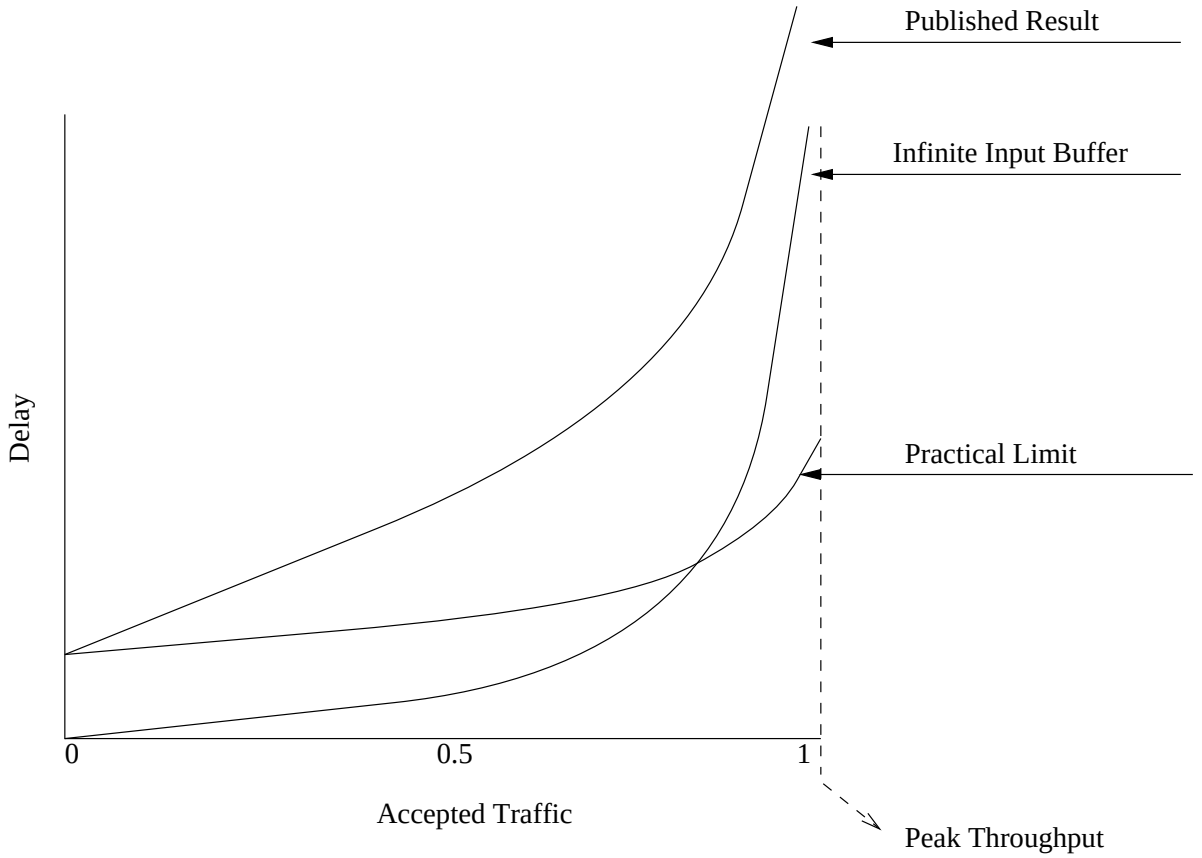

Figure 6: An example of a false graph due to simulation

7 Virtual Channel Model

Figure 9 provides a diagram for virtual channel communication. The probability that a packet occupies a virtual channel is given by $P_{i,l}$. Since virtual channels are filled in order, the probability that the last VC is filled is equal to the probability of all the space being used up. Thus $P_{i,l}$ is also the probability that an input packet will wait for a virtual channel.

The actual time a packet will wait for a VC is given by

$$X_i = P_{i,l} T_{i,l}$$

where $T_{i,l}$ depends on the amount of time a packet waits for the switch. This also depends on the number of VCs that are attached to the input port of the switch and how many of them are allocated. The total waiting time is given by

$$T_{w,i} = W_i \sum_{i=1}^{n} X_i$$

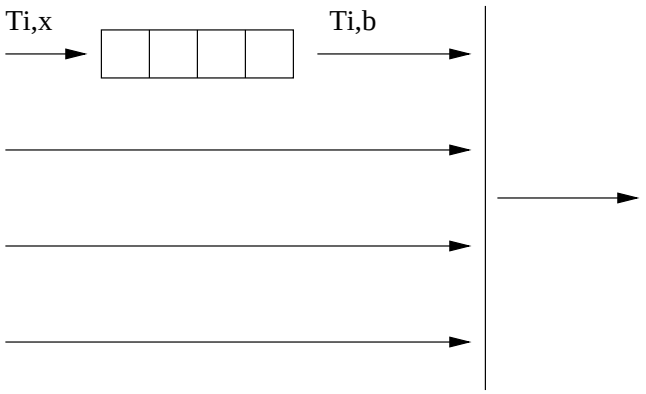

Figure 7: A limited queue node

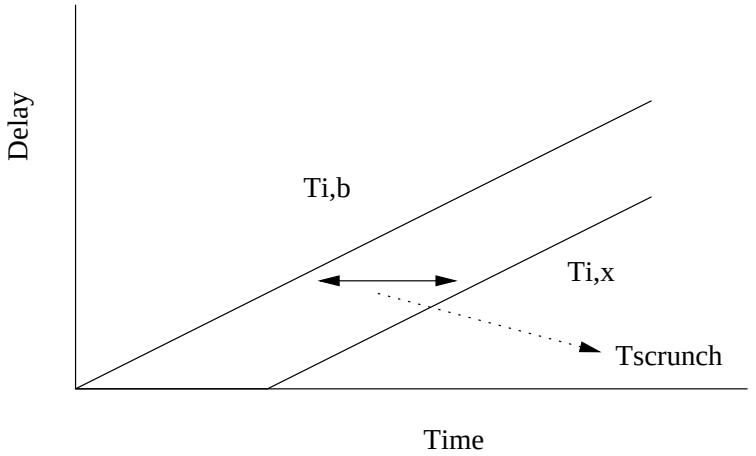

Figure 8: The offset of the delay through the buffer

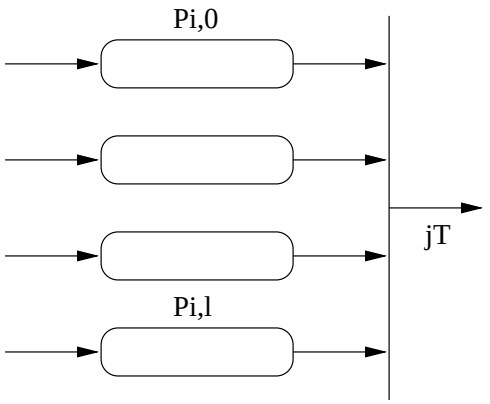

Figure 9: A node using virtual channels

We can also model the idle time on an output port of the node by

$$b_{i,j} = (\frac{W_j + 1}{jT})^j$$

where jT is the amount of time to send j packets over a channel. The student can understand the expression in the following manner: The formula minus the power term represents the time that each VC waits upon the output channel. When we multiply the time that each channel waits times all the other channel's waiting times, we get the probability that every VC is waiting on the output channel. This time is the time wasted using VC flow control.

8 Ring Topologies

When analyzing the topologies such as a ring, the student can use the same tools that we used in analyzing a regular network. However, there is not a clear cut start and stop; thus, there is no clear ordering of dependencies as with above. For example, with dropping flow control, we establish an ordering on λ and iteratively solve the problem using this ordering. Here, we can pick a start and stopping place in the network and then apply symmetry to solve the problem using the superposition of every other node's effect at the same link in the network.

9 Simulation vs. Analysis

The advantages of paper and pencil analysis using these models are that we gain an intuition of how the network operates. We can perform the analysis faster than in a simulation (provided we can generate correct models quickly), and we can more easily discard unnecessary parts from the model, i.e., we can simplify the simulation.

Simulation is useful in verifying the analysis model and verifying that our assumptions are correct. We use simulation when we can't develop analytical models for irregular networks or for bizarre network flow control strategies.