Allocation(Cont) and Network Performance Analysis

Lecture #12: 16 May 2001
Lecturer: Prof. Dally
Scribe: Gaurav Garg (gaurav.garg@stanford.edu)
Vaishali Kulkarni (vaishali.kulkarni@sun.com)
Reviewer: Kelly Shaw (kashaw@cva.stanford.edu)

Administrative Announcements:

Comments on Research Project Proposals:

The research project proposals should clearly outline the following four points:

- **Problem Description**: The problem that is sought to be solved in the project
- **Proposal**: The proposed solution for the problem
- **Methodology**: The methodology that is going to be followed to evaluate the proposed solution.
- **Hypothesis**: The expected results when the proposed solution is evaluated using the above mentioned methodology.

Due date for research paper submission: The research papers are due in class in the current lecture.

Canonical Allocation Problem:

The allocation problem is the optimal assignment of output ports to input ports when multiple input ports make simultaneous requests for multiple output ports. This can represented as a $m \times n$ matrix where m is the number of inputs and n is the number of output ports. A 1 in location (m, n) indicates that output port n is being requested by input port m. For example, consider the following request matrix where the rows correspond to the inputs and the columns correspond to the outputs. Several schemes have been devised to allocate outputs to inputs which try to maximize the utilization of the outputs.

outputs

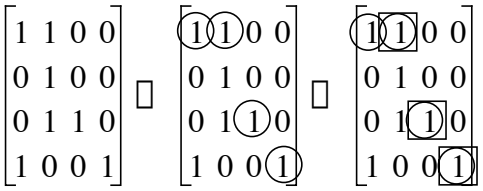
inputs

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

Request matrix

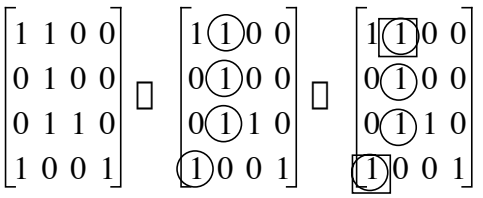
Parallel Allocation

Arbitration can either be performed row first or column first. Consider column first allocation in which one element from each column is selected first and then, for each row, one element from those chosen in the first step are picket. This is illustrated below:



Column first allocation

In this case input1, output0 is not being utilized. Another approach would be to select one element from each row first and then, for each column, select one of the elements selected in the previous step as illustrated below:



Row first Allocation

In this case output 3 and 4 are not being utilized at all.
There is no advantage to doing column first or row first allocation but usually column first allocation is done because it's easier to implement in hardware.

Iterative Allocation

The performance of the allocator can be improved if the process of matching inputs to outputs is repeated over several iterations. Two such schemes are:

- PIM: Parallel Iterative Matching
- SLIP

In such schemes, the process of request, grant and accept are repeated over several iterations. An analysis and comparison of such schemes can be found in the following paper.

Nick McKeown and Thomas E Anderson, "A Quantitative Comparison of Iterative Scheduling Algorithms for Input-Queued Switches",1997

Two observations can be drawn from this study:

- Even under heavy load conditions, utilization saturates at about 60% of the maximum because of conflicts after one iteration. In order to get complete assignment, everyone must pick a different row. The probability that nobody picks a row is $\left(1 - \frac{1}{k}\right)^{k-1}$, for a $k \times k$ matrix. SLIP per-

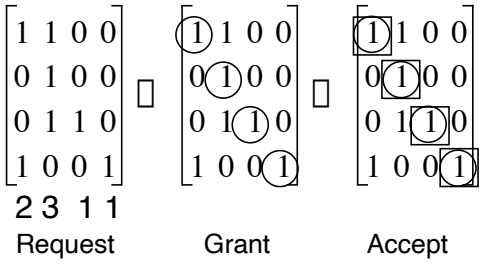
forms better than PIM and other schemes after one iteration because keeps state that spreads pointers out.
- Better results are obtained if multiple iterations are done. However, all the input queued switch allocation schemes converge to the same utilization after 4 iterations.

Increasing the Dimension of the Switch

Increasing the number of inputs into the switch is one way of increasing the link utilization on the outputs. e.g. a 4×4 crossbar can be converted to a 8×4 crossbar. In the row-first parallel allocation process, two elements from each row can be selected followed by selection of one from each column. Good output link utilization can be achieved in this manner. As long as all outputs are utilized during each cycle, it doesn't matter that some inputs didn't get the switch.
As an interesting observation, switch scheduling is a bipartite matching problem whose complexity grows $O(V \times E)$.

Dally's Lonely Output Heuristic

This heuristic is based on the following observation: In the case of a conflict, output utilization is increased if the output with least number of bidders is allocated first because the other outputs will have alternate bidders. This scheme proceeds in the following manner:
- All inputs put in the requests for the output channels.
- Count the number of bidders for each output
- Follow row-first allocation. In each step during the grant phase, select the column in each row which has the least number of request in it.
- Select one element from each column.



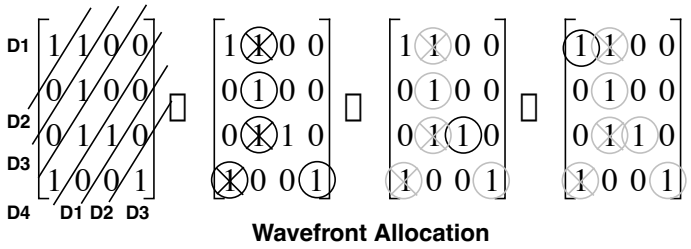
Dally's lonely output heuristic

This heuristic resolves conflicts by giving priority to the outputs which have the least number of bidders. It maximized output link utilization and performs better than other schemes over a single iteration especially for sparse matrices. It requires one more step as compared to a separable allocator.
The symmetric counterpart of this algorithm, in which the element in each row is selected according to which row has the least number of requests, doesn't perform as well on this matrix. In general, this is not necessarily true.

Wavefront Allocation

This algorithm is based on the observation that there can be no conflict among any elements in a diagonal so all of them can be selected simultaneously. This algorithm starts by selecting a diago-

nal and allocates to all the requesters in that row. All the elements in other diagonals which are either in the same row or column as any requester in this diagonal are removed from future arbitration. The algorithm proceeds by repeating this step for all the other diagonals.



Wavefront Allocation

Selection of the starting diagonal can be either deterministic or random. In deterministic scheme, the starting diagonal is advanced by one whenever a element in that diagonal wins. In the random case the starting diagonal is picked randomly.
The above depicted example achieves 100% utilization. It is interesting to note that if the starting diagonal is D4 then output utilization is only 50%.

Performance Analysis

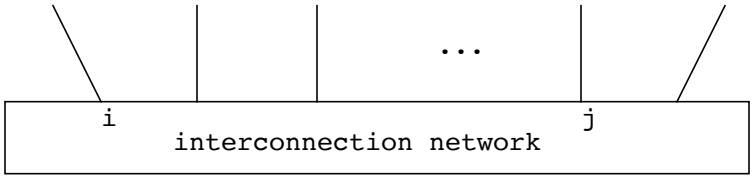


Figure 1

What it means to do a performance analysis? It is not just determining the delay and throughput. The delay Ti,j is not just the worst case delay from node i to node j. It highly depends on the load

from node i to node j, the routing function, the flow control used etc. Below shows an example latency Vs throughput curve.

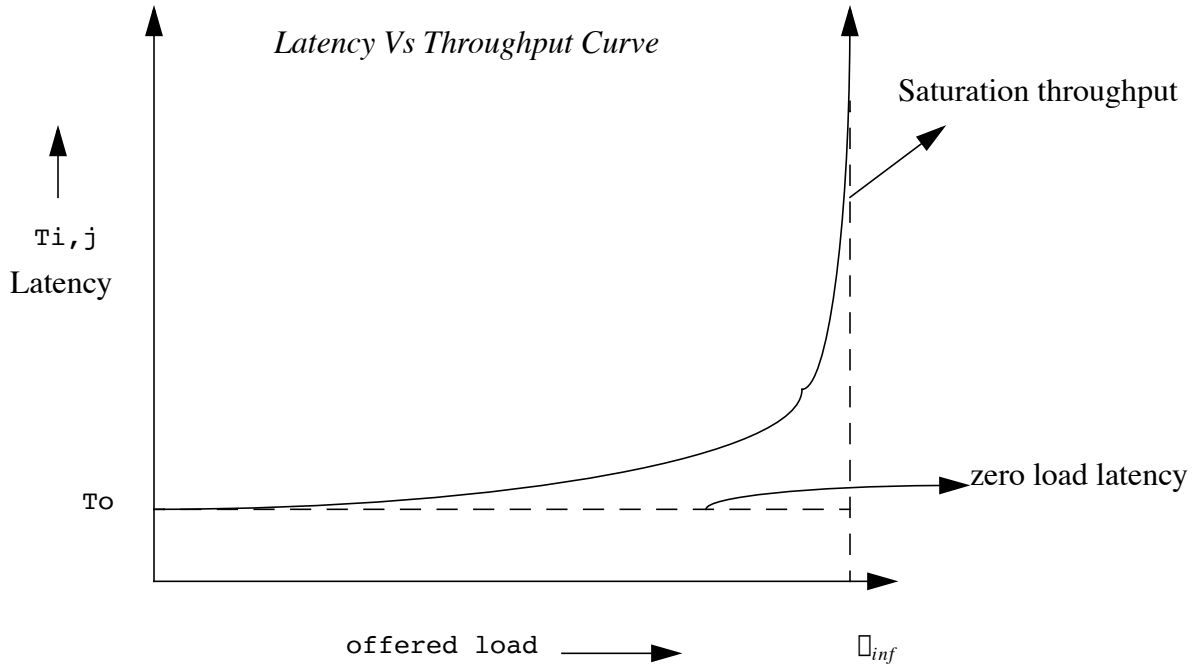


Figure 2

The question now is how these curves are plotted given a specification of a network including the routing function, traffic patterns, channel bandwidth, topology etc.There are two ways of doing performance analysis. One way is using a simulator and the other way is to use an analytical model. When we have to gather the performance numbers for a large number of topologies and for a large number of traffic patterns, then analytical model works better. But analytical models do not specially work well when the traffic patterns are too complex and most of the analytical models work using uniform traffic.

Performance Analysis of dropping flow control:

Consider a model (e.g. k-ary 4-fly) which is acyclic network with dropping flow control.

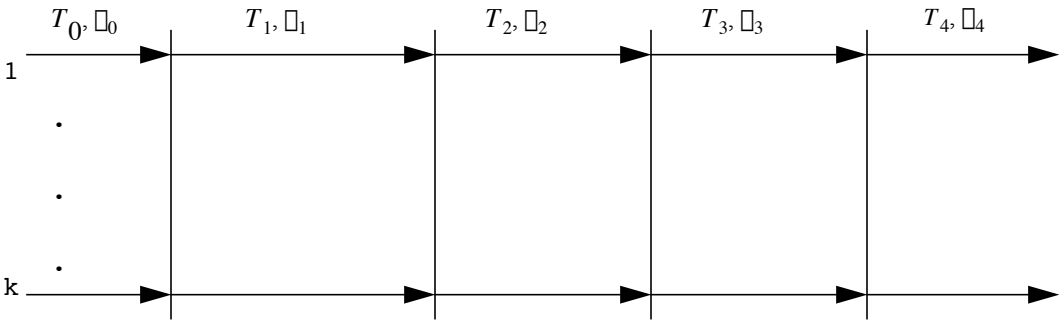


Figure 3

We will start analysis at the source and proceed to the destination.

T_i here is the service time offered by each node and λ_i is the rate of packets on the link. λ_i is measured in packets per second or packets per cycle while Ti is measured in seconds or cycles. The product of these 2 terms is equal to the duty factor

$\rho = \lambda T$

The duty factor timing relation is shown below

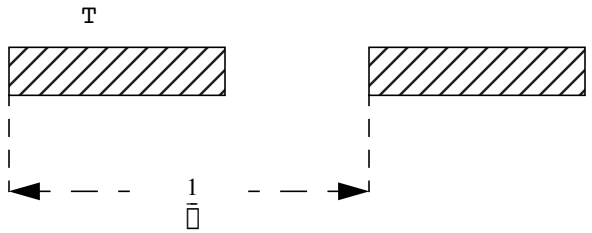


Figure 4

In the above diagram, $1/\lambda$ is the interval between arrivals.

Let's compute the probability of getting dropped at any stage.

For dropping flow control the T_i at various stages are same since packets never hold onto a channel longer than what is needed. But λ_{i+1} is less than λ_i because at each stages packets can be dropped.

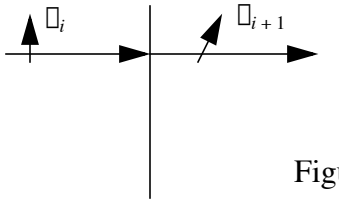


Figure 5

λ_{i+1} is the duty factor at the output link shown above and λ_i is the duty factor at the input link. P_{drop} is the probability that a packet will be dropped because the link is busy which is in fact the probability that the output link is busy due to one of the other inputs.

$$P_{drop} = \lambda_{i+1}\left[\frac{k-1}{k}\right]$$

The duty factor λ_{i+1} which is

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

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

Examples of dropping flow control is worked out for a butterfly using radix-8,4,2 switches. The table below shows the values of λ_i for a 4096 node network. As can be seen from the table, throughput drops to almost 22% for a radix 8 butterfly and to almost 14% for a radix-2 butterfly because as the number of stages increases the throughput drops. Also for higher radix switches it can be seen that at any stage, there is more collision with packets from other inputs resulting in smaller throughput. We will see the analysis for blocking flow control in the next section. In general, dropping flow control is bad.

Table 1:

K	8	4	2
(k-1)/k	0.88	0.75	0.50
p0	1.00	1.00	1.00
p1	0.53	0.57	0.67
p2	0.36	0.40	0.50
p3	0.28	0.31	0.40
p4	0.22	0.25	0.33
p5		0.21	0.29
p6		0.18	0.25
p7			0.22
p8			0.20
p9			0.18
p10			0.17
p11			0.15
p12			0.14

Performance analysis of blocking flow control:

Let's consider a flow control scheme where instead of dropping a packet, the packet is blocked until the channel is available. There are no buffers in this case. The diagram below shows the blocking a packet experiences and the timing in the presence of blocking.

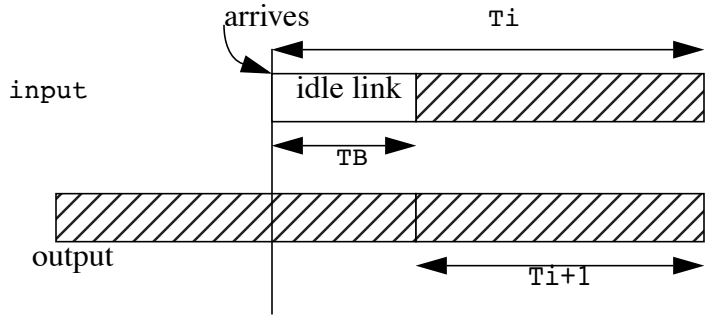


Figure 6

Consider how T_i and $\square_i$ are related for each stage in this case. Since nothing gets dropped, $\square_i$ and $\square_{i+1}$ are equal in this case. But T_i from one stage to another will be different with a relation of type $T_i > T_{i+1}$ since at each stage the probability that the packet gets blocked increases.

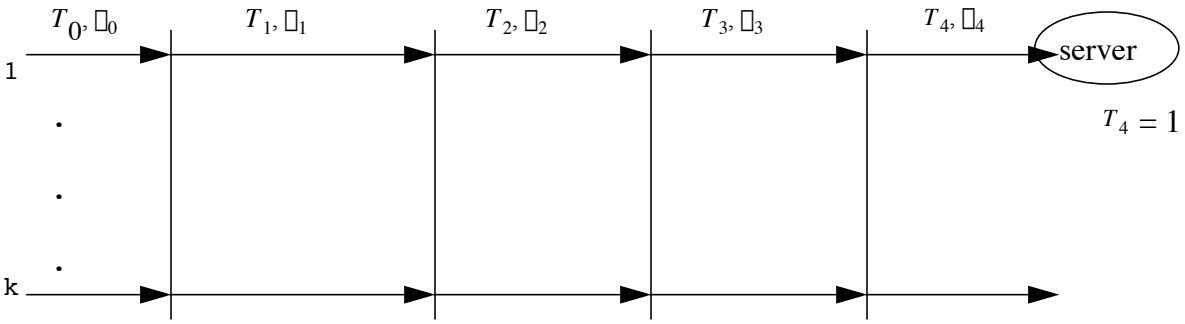


Figure 7

Let's start the analysis starting from the destination and proceeding to the source. We assume that T_4 is 1 (assume there is server at the last link with a service time of 1 time unit)
The probability of collision with other packets can be written as

$$P_{collision} = P_{drop} = \left(\frac{k-1}{k}\right)^{T_i + 1}$$

T_{i+1} is used in this case since service time on the output links matters.
If we collide on some input, the expected value of blocking time is

$$E(T_B|collision) = \frac{T_{i+1}}{2}$$

Assuming that the packet length is quite long, the expression for T_i can be written as shown below. If the packet length is not long, there will be terms in the expression which is a function of packet length.

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

$$= T_{i+1} + P_{coll} \cdot E(T_B|collision)$$

$$T_i = T_{i+1} + \left[\frac{(T_{i+1})^2}{2}\right]\left[\frac{k-1}{k}\right]$$

There is a similar analysis done using a table for this type of flow control. Prof. Dally forgot to bring those numbers but will be shown in the next lecture. This performs better than the dropping flow control but once again it saturates for a small fraction of capacity for this size network. The network saturates when $T_0 > \frac{1}{}$ which means the duty factor is greater than 1 (the network saturates when $T_0 = 1$)

Performance analysis of blocking flow control using buffers:

For the case of blocking flow control, there can be no buffers, infinite buffers or a small finite number of buffers. An analysis of blocking flow with no buffers has been done in the last section. An analysis with infinite buffers can be easily done but it is not a realistic case. In this section, we will do an analysis of blocking flow control which includes buffers. Let's do the analysis for small finite number of buffers first. Assume that the packet length is quite large (say 1000 flits) and the buffer length Q is 4 flits.

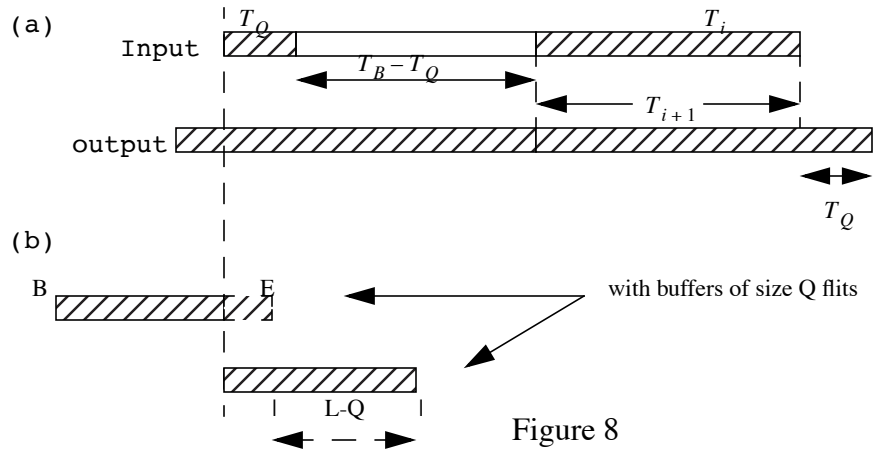


Figure 8

Figure 8(a) above shows the input and output links when there are buffers of size Q flits and shows the idle time on the input links. when a packet arrives at the input, the input link can receive the first Q flits since they can be saved in the buffer. The corresponding time is shown as T_Q . If the output link is busy, the input link will remain idle for $T_B - T_Q$ because no more flits can be received on the input link. Once the output is available, the buffered Q flits and the rest of the flits can be sent on the output link. The total time the output link will be used is T_Q for sending the buffered Q flits plus T_{i+1} which is the time to send the rest of flits in a packet which traversed through the router after the output link was available for use.

Figure 8(b) shows the amount of time flits of a packet arriving at the input link will have to wait to traverse through the output link. This wait time depends on how many flits of the previous packets have traversed through the output link when the flits arrived at the input link. This is addressed as *collision point* of the arriving flits at the inputs with respect to the flits traversing the outputs at that time. In the figure 8(b), B indicates that the output link just began sending the first flit of the packet while E indicates the output link just sent the last flit of the packet. L is the length of the packet and Q is the number of flits that can be buffered at the input.

Figure 9 shows below shows the waiting time T_{wait} Vs the collision point when there are no buffers and when there are buffers of length Q. Without any buffers, when the collision point is right at the beginning (B), the waiting time is L and when the collision point is at the end (E), there is no wait time. The two points are shown below in the plot for ‘no buffer’ case and between these two points the plot is probably going to be a straight line. With buffers of length Q, when the collision point is the beginning (B), the waiting time reduces to L-Q because until time T_Q , no bandwidth is lost on the input link. For the case when there are buffers and when the collision point is at the end (E) or even Q flits before the end (E-Q), the waiting time is zero. This is shown below in the plot for ‘buffers of length Q’ case.

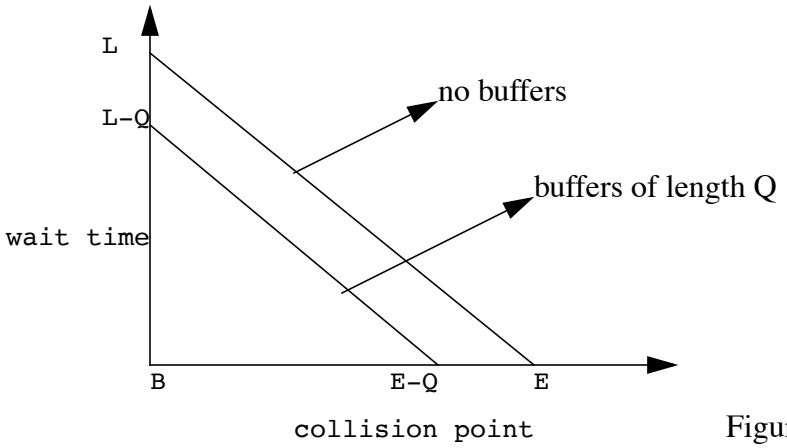


Figure 9

Based on the above graph, it can be seen that the expected blocking time when there are no buffers is L/2
But the expected value of blocking time when there are buffers of length Q is

$$\left(\frac{L-Q}{2}\right)/\left(\frac{E}{E-Q}\right)$$

Instead of using E this can be written down in terms of packet length L as

$$\frac{0.5(L-Q)^2}{L}$$

So as Q goes to L, the expected blocking time goes to zero and as Q goes to zero, the expected blocking time goes to 0.5L.
This case of small finite buffer holds true only when Q is less than length of packet and at any time there are no more than two packets in a buffer. The other cases of intermediate lengths will be covered in the next lecture.