Topology basics. Constraints and measures. Butterfly networks.

Lecture #2:	Monday, 7 April 2003
Lecturer:	Prof. William J. Dally
Scribe:	Debashis Sahoo
Reviewer:	Brian Towles

This class talked about the basics of topology, definition of important concepts, quantitative characterization of interconnection networks and few examples to illustrate the idea.

1 Nomenclature

The class goes through two examples and illustrates how the terminology fits in each of those.

An interconnection network I is specified by two sets C and N^*. C represents the set of channels. N^* represents the set of nodes. There is a distinguished set $N \subseteq N^*$, that represents the set of terminal nodes. We often refer to the number of terminal nodes in a network as N instead of $|N|$ and likewise for the number of channels.

In Figure 1, $N^* = N = \{0, 1, 2, 3, 4, 5, 6, 7\}$. This kind of network is termed as a direct network and N denotes the set of nodes. In a direct network every node is both a terminal and a switch.

In Figure 2, $N = \{0, 1, 2, 3\}$ and $N^* = \{0, 1, 2, 3, 0.0, 0.1, 1.0, 1.1\}$. Here N and N^* are different.

The degree of a node x is specified by δ, which is the sum of the number of input and output channels connected to the node. We often split δ by δ_I, the in degree and δ_O, the out degree.

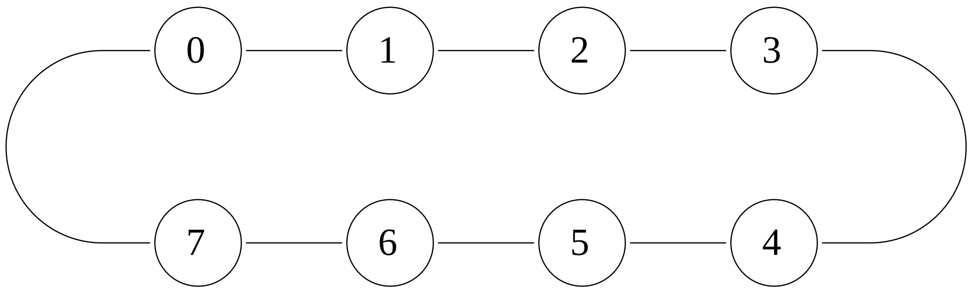

Figure 1: An eight node ring.

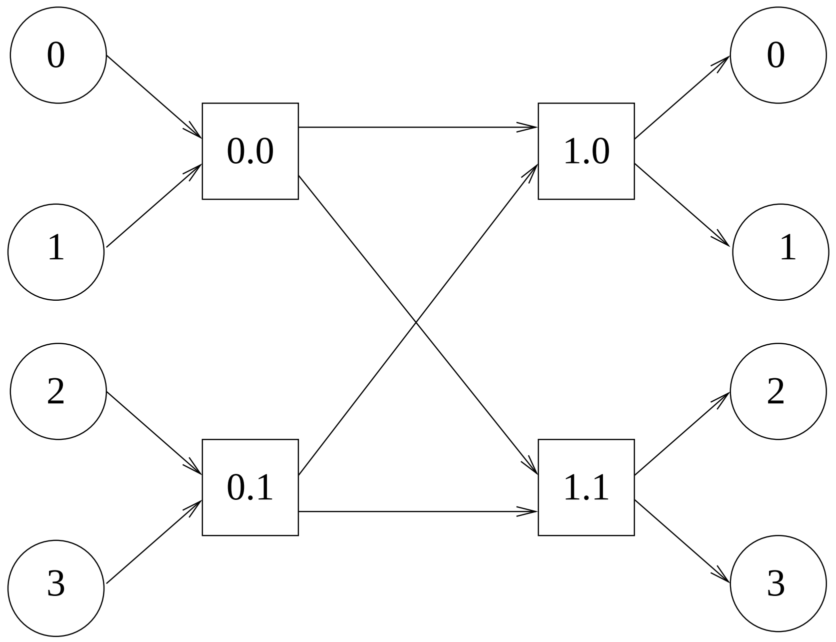

Figure 2: A 2-ary 2-fly network

The degree of node 0 in Figure 1 is 4. Because there are two bi-directional channels attached to it. The in-degree of the node 0, δ_I is 2 and the out-degree, δ_O is 2. The degree of the switching node 0.0 in Figure 2 is 4, with $\delta_I = 2$ and $\delta_O = 2$. Because they are unidirectional channels.

The bandwidth of a channel $c \in C$ is $b_c = f_c w_c$. f_c is the rate at which bits are transported on each signal. w_c is the number of parallel signals the channel contains. The latency of a channel $t_c = l_c/\nu$, where l_c denotes the length of the channel and ν denotes the propagation velocity.

A cut of the network is two sets N_1 and N_2 such that $N_1 \cup N_2 = N^*$ and $N_1 \cap N_2 = \phi$. A bisection of the network is a cut with $|N_1|$ approximately same as $|N_2|$. A minimal bisection is a bisection that cuts minimum number of channels. We define a channel bisection B_c as the number of channels cut by a minimal bisection of the network. For example in Figure 1 $B_c = 4$ and in Figure 2 $B_c = 2$. We are interested in bisection because it gives a bound on the throughput.

A path is an ordered set of channels $P = \{c_1, c_2, ..., c_n\}$ where $d_{c_i} = s_{c_{i+1}}$ for $i = 1...(n-1)$. The length or hop count of a path is $|P|$. A minimal path from node x to node y is a path with smallest hop count. We denote the set of all path from x to y as R'_{xy}. The set $R_{xy} \subseteq R'_{xy}$ is the set of all minimal path from node x to node y. $H(x,y)$ is the hop count of a minimal path from x to y. The diameter of a network H_{max} is the largest minimal hop count over all pairs of **terminal** node of the network.

$$H_{max} = \max_{x,y \in N} H(x,y)$$

In Figure 1 $H_{max} = 4$. In Figure 2 $H_{max} = H_{min} = 3$. If a network has N terminals and switches of degree δ, then $H_{max} \geq \log_{\delta/2} N$. The ring network doesn't achieve the lower bound on H_{max} asymptotically.

2 Traffic Patterns

The traffic pattern is represented by a matrix Λ, where each entry $\lambda_{s,d}$ gives the fraction of traffic sent from node s to node d.

$$\sum_s \lambda_{s,d} = 1 \qquad \sum_d \lambda_{s,d} = 1$$

A very trivial example of Λ is a permutation matrix Π, where each column has one 1 and each row has one 1. We will look at two special classes of permutation called bit permutation and digit permutation (bit permutation is a special case of digit permutation where digites are binary). In a digit permutation, each (radix-k) digit of the destination address d_x is a function of a digit s_y of a source address. A bit permutation $d_i = s_{b-i-1}$(b is size of the address), is called a bit reversal permutation. Each node sends its traffic to a destination with the address digit reversed. For example in Figure 1 the node 0 (Address 000) sends to 0 (Address 000), 1 (001) sends to 4 (100), 2 sends to itself, 3 sends to 6 and so on. Similarly we can also do digit permutation, where each node address is a radix k number.

3 Throughput

We define load on a channel c, denoted by γ_c, the amount of traffic that must cross the channel if each input injects one unit traffic according to a given traffic pattern. The maximum channel load $\gamma_{max} = \max_{c \in C} \gamma_c$. The channel corresponding to γ_{max} is a bottleneck channel and thus determines the throughput. Because we can put as many traffic as we want on input until the bottleneck channel saturate.

For example consider the eight node ring shown in Figure 1. Lets consider a random traffic pattern and minimal routing algorithm. We pick the channel $2 \rightarrow 3$ is called c. In this example node 7 contributes $\frac{1}{16}$ of its traffic to the load of channel c. Similarly, node 0 contributes $\frac{3}{16}$ (calculated $\frac{1}{8} + \frac{1}{16}$), node 1 contributes $\frac{5}{16}$, node 2 contributes $\frac{7}{16}$ of their respective traffic. All this sums to 1. Here γ_{max} is equal to 1. In Figure 2 $\gamma_{max} = 1$ for uniform random traffic pattern.

The ideal throughput of the topology Θ_{ideal}:

$$\Theta_{ideal} = \frac{b}{\gamma_{max}}$$

where b is the bandwidth of the bottleneck channel. If we apply b/γ_{max} amount of traffic to all the input, then the bottleneck channel will have load of b which is its capacity. So we can apply $\Theta * \Lambda$ as the load to all the inputs. That would exactly load the bottle neck channel to its capacity.

There are two useful bounds that we can put on the ideal throughput. One of them is a bisection bound. In Figure 1 with uniform traffic pattern, half of the traffic has to cross the bisection. For N node ring $N/2$ traffic has to cross the bisection. So bound on γ is $N/2$ traffic has to cross B_c channels.

$$\gamma_{max} \geq \frac{N}{2B_c} = \frac{8}{2*4} = 1$$

The equality is achieved if the bisection is loaded uniformly and bisection is the bottleneck point. Another way of comming up to the bound is by using average hop count . For the figure 1 :

$$H_{avg} = \frac{0+1+2+3+4+3+2+1}{8} = 2$$

Each traffic will use 2 links. To send all the traffic we need $N * H_{avg}$ number of links. Hence :

$$\gamma_{max} \geq \frac{N * H_{avg}}{C} = \frac{8*2}{16} = 1$$

4 Latency

The latency of a network is the time required for a packet to traverse the network, from the time the head of the packet arrives at the input port to the time the tail of the packet departs the output port. There are two main components of the latency:

- **Head latency** : The time required for the head of the message to traverse the network. It has three main components.
  - The router delay t_r
  - The time of flight t_w
  - The contention delay
- **Serialization latency** : The time required for the rest of the message to catch up.

The Figure 3 describes the latency of a packet in the absence of contention. From the time message arrives at the node it takes some amount of time t_r before it leaves that node. This is called the router delay. When the message arrives, the router looks at part of the header to see how to route it, makes a decision how to route it, allocates resources to it and eventually it gets to leave the node x after time t_r. It takes some amount of time t_{xy}, the time of flight to reach the node y. Eventually it reaches the destination. Then rest of the message catch up in time L/b (serialization latency).

$$T = T_h + T_s$$

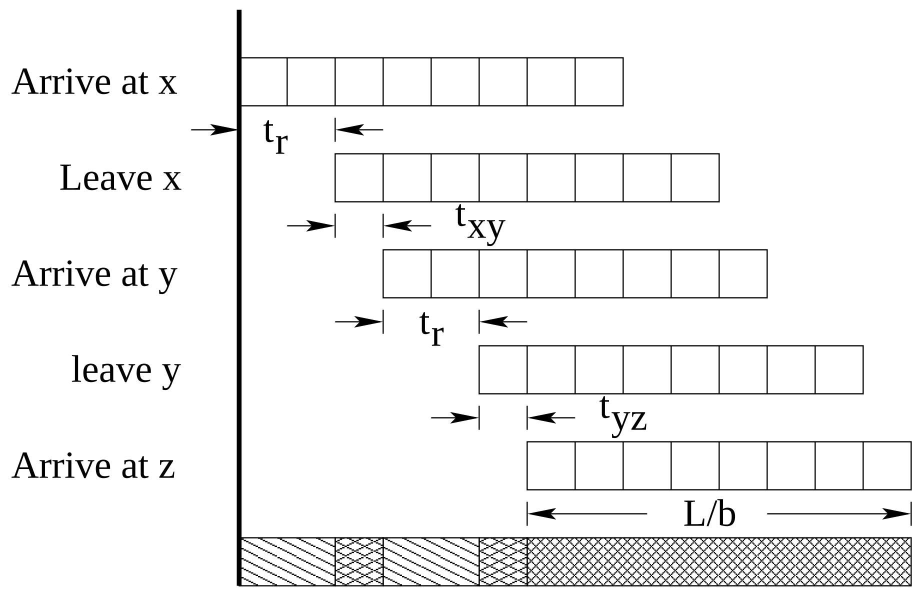

Figure 3: Gantt chart showing latency of a packet traversing two channels in the absence of contention

$$T_h = Ht_r + D_{avg}/\nu + T_c$$

$$T_s = L/b$$

The total latency is describe in the above equations. If we look at the equation without contention delay, then the latency is called zero other load latency. This is the time taken by a packet to reach its destination, if there are no other packets in the network. We can decrease serialization latency by widening the channel. We cannot increase the channel width indefinitely. The pin count(or packaging constraints) limits how wide a channel can be made. At every package boundary we have a limit on the bandwidth. Consider a ring network with bandwidth of a node, $B_n = 128Gbps$ and bisection bandwidth $B_s = 1024Gbps$. The bandwidth of the channel b_c satisfies following constraints:

$$b_c \leq \frac{B_n}{\delta} = \frac{128}{4} = 32Gbps$$

$$b_c \leq \frac{B_s}{B_c} = \frac{1024}{4} = 256Gbps$$

Hence this network is pin limited. We don't want our system to be pin limited, because we are throwing away lot of system bandwidth. We want to saturate the system bandwidth, that would give us the most global bandwidth with the given packaging technology. We dould like to saturate both node and system bandwidth. Usuallly we win by operating on the edge.

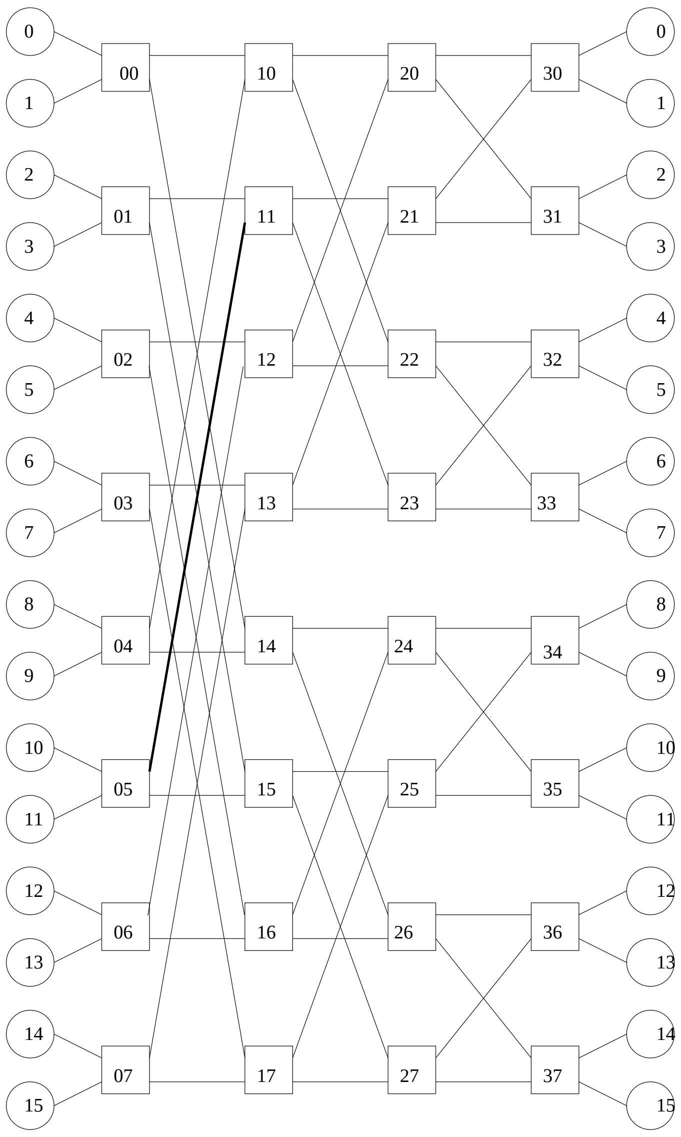

Figure 4: A 2-ary 4-fly network

5 Butterfly Network

The Figure 4 shows a binary 4-fly butterfly network. In general a k-ary n-fly butterfly network has n stages and k is the radix of each switch. Table 1 shows some information about a general k-ary n-fly network. Each terminal has an address of n-digit radix-k number, $\{d_{n-1}, d_{n-2}, ..., d_0\}$. Each channel at each stage has address of n-digit radix-k number. The address of a switch node is a $n-1$-digit radix-k number. The first $n-1$ digit of a terminal address corresponds to the switch, it is connected to. The wiring between the stages permutes the terminal address. Between stages $i-1$ and i (numbering starts

Number of stage	n
Degree of a switch, δ	2k
Total number of terminal, N	k^n
Total number of switch nodes	$n * k^{n-1}$
Diameter, H_{max}	$n+1$

Table 1: Information about a k-ary n-fly network

from 0) the wiring exchanges digits d_{n-i} and d_0. For example in the stage 0, the channel 10 (0.1010) goes to 3 (0.0011) as shown in the Figure 4. Similarly wire 6 of stage 1 goes to 3.

Channel load on every butterfly network under uniform traffic pattern is equal to one. All traffic evenly splits everywhere. There is no path diversity in a k-ary n-fly network. This can lead to significant degradation in throughput due to load imbalance across the channels when traffic pattern is not uniform.

6 Packaging

Consider packaging a butterfly network. We have two constraints on the packaging. Wiring constrains W_n per node and W_s across in the middle of the system. Channel width of a butterfly network:

$$W = \min(\frac{W_n}{\delta}, \frac{2W_s}{N}) = \min(\frac{W_n}{2k}, \frac{2W_s}{N})$$

In butterfly network half ($N/2$)the wire cross the bisection. We would like to satisfy both the constraints at the same time.

$$W_n/2k = 2W_s/N \quad \rightarrow \quad k = NW_n/4W_s$$

For example consider $W_n = 128$, $W_s = 1024$, $N = 256$. Table 2 shows width of a channel, W and the diameter, H in different networks. For $k = 2$ the diameter is

k	$W_n/2k$	$2W_s/N$	W	H
2	32	8	8	9
4	16	8	8	5
8	8	8	8	4
64	1	8	1	3

Table 2: Packaging a butterfly network

large. $k = 8$ satisfies both of the constraints simultaneously. For k=64 we are giving up throughput by not using all the system bandwidth and serialization latency would be large because of single bit channel.

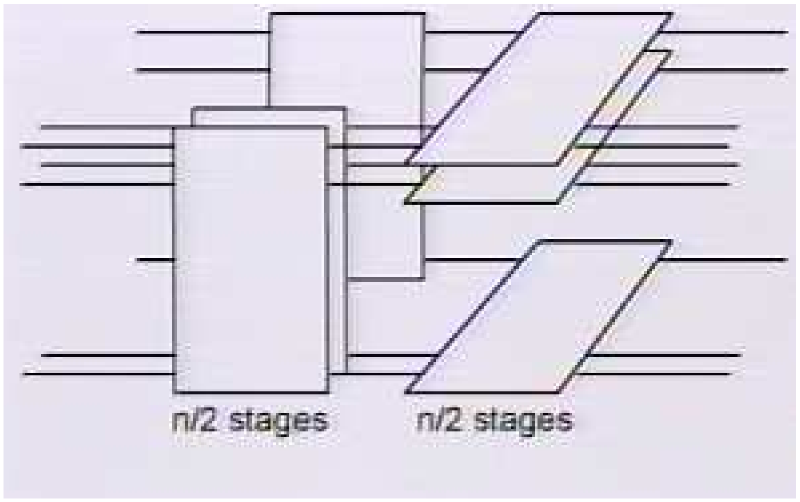

Figure 5: 2-dimentional view of a butterfly network

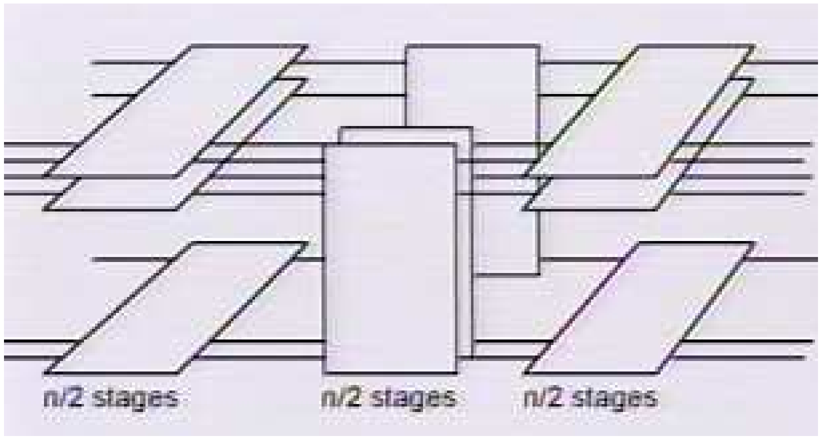

Figure 6: 2-dimentional view of a butterfly network with extra stages

7 Path diversity

In butterfly network there are no redundant paths. Therefore an adversary can choose a particular traffic pattern to reduce the throughput. For example in the butterfly network shown in Figure 4 if 0, 1, 8, 9 send their traffic to 0, 1, 2, 3 then the link 1.0000 is used by all these channels. If we rotate one bit left in 0, 1, 8, 9, we get 0, 1, 2, 3. Here $\gamma_{max} = 4$. In general if we apply "rotate one bit left" traffic pattern then we get $\gamma_{max} = \sqrt{N}$. This is a very serous degradation in performance if the network is very large.

This traffic concentration is easy to visualize if we draw out butterfly network in two dimensions as illustrated in Figure 5. The figure shows that a k-ary n-fly can be thought of as $2^{n/2}$ y-axis switching planes, each with $2^{n/2}$ switches per stage followed by an equal number of z-axis switching planes with the same number of switches per stage.

If all the nodes in vertical plane sends their traffic to all the nodes in one horizontal plane then a single channel would carry all this traffic. The easy solution to this problem is to add extra stages as shown in Figure 6. The extra stages is used to distribute the traffic uniformly over the channels. However congestion can still occur entirely within the vertical plane. We can for example divide each vertical plane into two $n/4$ stage network and concentrate the traffic at the junction of these two networks.

The load balance problem for butterflies can be solved by duplicating all n stages of the network. The resulting $2n$-stage network is called a Beneš network.