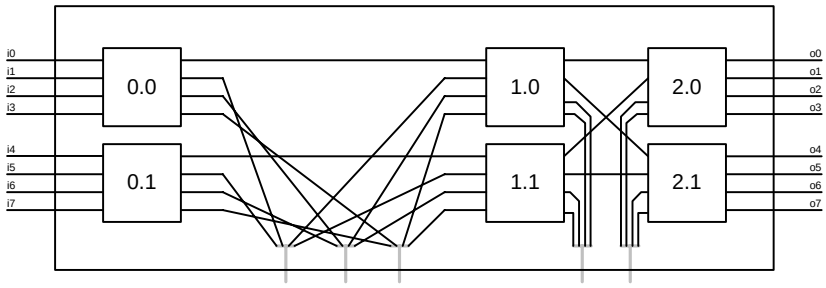

Figure 1: Packaging of a single board of a 4-ary 3-fly with 6 nodes per board.

Solutions to Homeworks 1, 2, and 3

2-1. Group switches 0.0,1.0,1.0,1.1,2.0, and 2.1 on a single board and similarly for the remaining groups of six switches. This gives the same number of chips and boards and the signal constraints of the boards are not violated. As shown in Figure 1, five cables are required for each pair of boards. There are 8 boards and therefore 20 cables. Thus, the total cost of the system is $48(200) + 8(200) + 20(50) =$ \$12,200.

3-9. (a) Six. (b) Maximum is six, minimum is four. For example, the permutation $[3142]$ gives a diameter of four.

4-2. For upper switch of the first stage, input 0000_2 sends to output 0000_2 (using the first output port of the switch), 0001_2 sends to 1000_2 (using the third output), 0010_2 sends to 0100_2 (using the second output), and 0011_2 sends to 1100_2 (using the forth output). The same pattern is repeated for each first stage switch and all middle channels are equally loaded with $\gamma_{\max} = 1$. So, the throughput is 100% of capacity.

4-3. Choose k using

$$k = \left\lceil \frac{NB_n}{4B_s} \right\rceil = 32.$$

Since $N = 1024$, this gives $n = 2$. The corresponding channel width is

$$w = \min\left(\frac{W_n}{2k}, \frac{2W_s}{N}\right) = 2 \text{ bits}.$$

Given $f = 1$GHz,

$$\Theta_{\text{ideal}} = \frac{wf}{\gamma} = 2 \text{ Gbits/sec}.$$

Zero-load latency is then

$$T_0 = \frac{L}{b} + H_{\text{avg}} t_r = \frac{512}{2} + 30(3+1) = 286 \text{ ns}.$$

5-3.

k	n	w_n	w_s	w	Θ_{ideal}
1	256	96	1500	96	3
2	16	48	93.75	48	24
4	4	24	23.4375	23	46
8	2	12	11.71875	11	44

The best throughput is achieved on a 4-ary 4-cube. Each node has $4(4)(23) = 368$ pins, so given the board constraint of 512 pins, the only packaging option is one node per board.

6-2. For $k = 3$, an (n, n, n) Clos network has n^2 ports. Then for $k = 5$, build a (n, n, n^2) Clos, which has n^3 ports. This pattern continues and in general a k stage Clos (odd k) can have $n^{(k+1)/2}$ ports.

7-1. First, to minimize cost only, we design for average-case bandwidth. Thus, $b_T = 500$Mbps and $b_N = Mb_T$. Therefore, each concentrator has a pin bandwidth of $2M$Gbps. Each node has a MGbps connection to its concentrator and a network bandwidth of $4nb$. The network channel width must be designed to support uniform traffic at the average rate, with $B_B = (0.5)N/2 = 1.024$Tbps. Thus, $b = 1024/4k^{n-1} = 1024kM/4(4096) = kM/16$Gbps. This gives a pin bandwidth of $nkM/4 + M = M(nk/4 + 1)$ for each node. The total pin bandwidth of the network is then $N(3+nk/4)$ — concentration makes no difference in reducing cost for a network designed for the average case. Therefore, concentration can be removed and any combination of n and k that minimizes nk (hop count) and meets the chip bandwidth constraint gives minimum cost (e.g. $n = 6$ and $k = 4$, giving a node bandwidth of 7Gbps).

If we want to also avoid any extra serialization latency, $b_T = 10$Gbps, $b_N = \max(10, M/2)$Gbps, and $b = 10$Gbps. The maximum concentration is $M = 4$ and requires 100Gbps per concentrator chip. Choosing $k = 32$ and $n = 2$ gives a bisection requirement of $10(4N)/k = 10(4)(1024)/(32) = 1.28$Tbps with $b = 10$Gbps channels. The node bandwidth requirement is $10(4n + 2) = 100$Gbps.

8-1. (a) Deterministic. It's shortest ignoring any contention. (b) Deterministic. Uniform is already load balanced and any additional load balancing effort will only degrade performance. (c) Adaptive. While weighted will do well, it won't be able to compete with adaptive on patterns with high locality.

8-6. This is direction order routing: route in the increasing directions ($+x$, $+y$, and $+z$) before the negative directions.

8-7. Consider a source sending to it's neighbor in the positive y direction. If the channel directly connecting these nodes is faulty, dimension order routing is forced to route the long way around in the negative y direction to reach the node. However, in direction order routing, a sidestep in $+x$, followed by a step in $+y$, and ending with a hop in $-x$ reaches the destination in only 3 hops.

9-2. First we show a lower bound on the worst-case of *all* minimal routing algorithms by considering a 1-d tornado pattern: each node (i, j) sends to $((i + \lfloor k/2 \rfloor) \mod k, j)$. Because there is only one minimal path between each source-destination pair for this pattern, the load is the same for all minimal algorithms and is $\lfloor k/2 \rfloor$. Considering e-cube routing, for any channel in a row (column) of the torus, only sources (destinations) in that row can add load to that channel. Moreover, only half of the nodes in each row (column) can load a particular channel — the other half sends the other way around the ring because the algorithm is minimal. Therefore, the worst-case of e-cube is at most $k/2$. This is roughly equal to the lower-bound, thus e-cube is optimal (within an additive factor). However, this argument does not extend to meshes with $n > 2$ because sources (destinations) outside the row (column) can place load on the channels in that row (column).

10-1. For the transpose pattern, it's possible for minimal routes to be selected such that the channel load is exactly one. Since the minimal adaptive routing algorithm is in the steady-state, we assume this is the case. Minimal oblivious loads the center channels of the mesh the most heavily. Routing from (1,2) to (2,1) adds 1/2 of a unit load to the east going channel of node (1,1) (assuming the origin is in the northwest corner). Routing from (0,2) to (2,0) contributes another 1/6 load and routing from (1,3) to (3,1) adds 1/3. Finally routing from (0,3) to (3,0) adds 1/8 load. The total load is therefore $1/2 + 1/3 + 1/6 + 1/8 = 9/8$, which is greater than the minimal adaptive algorithm's load.

11-2. (a) A RAM table would require an entry for each destination or 64 entries. (b) A CAM table can be optimized to far fewer entries:

CAM Address	Direction
00X XXX	W
010 XXX	W
1XX XXX	E
011 0XX	N
011 101	S
011 11X	S
011 100	X

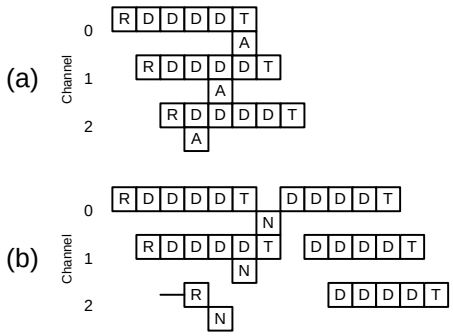

Figure 2: Optimistic circuit switching. (a) A successful optimistic transmission of the data with an acknowledgment to the source. (b) A failed transmission, the nack received by the source indicates the packet must be resent.

12-2. The minimum timeout is the latency of the forward traveling packet's header plus the latency of the acknowledgment. Serialization latency is already accounted for by starting the timeout once the entire packet has been sent from the source. Thus, the timeout is:

$$2H_{\max} + 1$$

assuming one flit time to "think" and t_r is equal to a one flit time.

12-4. As shown in Figure 2, optimistic circuit switching can potentially reduce the idle time of the channels by a round-trip delay.

14-1. (a) This not deadlock-free because all 8 turns are allowed. (b) The rules eliminate the $+y$ to $+x$ turn and the $-x$ to $-y$ turn. Using the turn model, this is enough to ensure deadlock freedom. (c) While no single route can create a cycle, no clockwise (right) turns have been disallowed and therefore cycles can be created between several packets.

14-4. To stay within the constraints of the $C \times N \rightarrow C$ routing relation, the injection channel at the source of a packet can be used to differentiate packets and better balance load. For example, when routing a packet from $4 \rightarrow 2$, it can be injected on VC 1 to spread load. Likewise, shifting the $3 \rightarrow 2$ and $1 \rightarrow 2$ routes to VC 1 helps balance load. More balance can be achieved by routing from $2 \rightarrow 3$ on VC 0.

14-8. One approach is to use an increasing VC for each dimension traversed. So, for example, an XYZ traversal pattern would use VC 0 for X, VC 1 for Y, and VC 2 for Z. Similarly, an YZX traversal uses 0 for Y, 1 for Z, and 2 for X. This

is deadlock-free because there are no dependencies between dimensions within a VC and the VC numbers always increase — there is no possibility for a cycle. Another approach is to increment the VC (starting from 0) when you turn from a negative direction to a positive direction. Within a VC, positive going channels can be enumerated with their distance from the origin and negative going channels can be enumerated with $|C|$ minus their distance from the origin. This prevents cycles within a VC. Since the VC's are only incremented, no inter-VC dependencies can introduce cycles. This scheme requires $\lfloor n/2 \rfloor + 1$ virtual channels.