Solution to EE 7725 1997 Homework 3/4

Problem 1: A honeycomb network can be constructed from a 2-D grid by eliminating certain horizontal links; the nodes don't change. Therefore, any mesh labeling can be used. For example, consider a $k \times k$ mesh; the node labels would be $V = \langle k^2 \rangle$. When a node label is written as a 2-digit, radix-k number, the LSD might give the horizontal position and the MSD the vertical position.

Problem 2: Since the honeycomb is similar to a mesh, it's a good idea to see if any deadlock-free mesh routing schemes can be adapted. Consider dimension-order routing; because of the missing links horizontal traffic will have to zig-zag (move up and down by one), sharing vertical links with vertical traffic. Dimension-order routing is deadlock free in part because contention on a higher dimension cannot affect traffic on a lower dimension; in the honeycomb this separation is impossible because of the shared links. A solution is to divide the vertical links into two virtual channels, one used by "true" vertical traffic, the other by zig-zagging horizontal traffic.

Problem 3: Any request of the form $(012u_{(2)}u_{(1)}u_{(0)}{}_5, v_{(5)}v_{(4)}v_{(3)}v_{(2)}01_5)$ will pass through the cells, for example $(012432_5, 312401_5)$. The unspecified digits can be chosen in $5^5 = 3125$ ways.

Problem 4: In all networks the path from the source, u, to the destination, v, is obtained from the sequence of numbers constructed by repeatedly applying shuffles and exchanges to the n-digit, radix-m representation of u until the result is v:

$$
\begin{aligned}
& u_{(n-1)}u_{(n-2)}\cdots u_{(0)} \\
\sigma \rightarrow\ & u_{(n-2)}u_{(n-3)}\cdots u_{(0)}u_{(n-1)} \\
\chi \rightarrow\ & u_{(n-2)}u_{(n-3)}\cdots u_{(0)}v_{(n-1)} \\
\sigma \rightarrow\ & u_{(n-3)}u_{(n-4)}\cdots u_{(0)}v_{(n-1)}u_{(n-2)} \\
\chi \rightarrow\ & u_{(n-3)}u_{(n-4)}\cdots u_{(0)}v_{(n-1)}v_{(n-2)} \\
& \vdots \\
\chi \rightarrow\ & v_{(n-1)}v_{(n-2)}\cdots v_{(0)}
\end{aligned}
$$

The difference is in how the actual node or cell numbers are obtained from the sequence. In the shuffle exchange network each element in the sequence is a node number. (In the shuffle exchange the first shuffle isn't really needed.) In the de-Bruijn network only the elements following an exchange step are nodes. For the omega network elements following a shuffle step are used; unlike the shuffle exchange and de-Bruijn networks the element itself is not a cell number. The cell number is obtained by forming a two-tuple ($\langle y, x \rangle$) consisting of a stage number, y, and the list element with the least significant digit dropped, x. For example, the omega network cell corresponding to the first shuffle is: $\langle 0, u_{(n-2)}u_{(n-3)}\cdots u_{(0)} \rangle$.

Nodes may appear several times in paths in the shuffle exchange and de-Bruijn networks, but cells appear only once in omega network paths.

Define cost as the number of links. The shuffle exchange has almost $m^n(m+1)/2$ links; the de-Bruijn has almost $m^n(2m/2)$ links; and the omega has $nm^{n-1}m$ links. Crossbar cost limits m to small values leaving n unbound, and so above a certain size, $n > m$. Therefore, the shuffle exchange is least expensive and for sufficiently large networks the omega is most expensive.

In the omega network all paths are fixed at n stages, so its minimum and average path lengths, as well as its diameter are all n. The diameter of the de-Bruijn network is also n, the diameter of a shuffle exchange is $2n-1$. Being direct networks, the shuffle exchange and de-Bruijn each have single-link paths, since the degree of the de-Bruijn network is higher it has more single-link paths than the shuffle exchange. The average path length of the de-Bruijn network is lower than the shuffle exchange (derivations for the path lengths were not given in class).

Performance under traffic is determined (in part) by the total amount of traffic *per link* and how evenly that traffic is spread. Every message traverses n out of nm^n links in an omega network; in a shuffle exchange network a large fraction of messages traverse $2n+1$ out of $\frac{(m+1)}{2}m^n$ links, in a de-Bruijn they traverse n out of mm^n links. Then per link, for sufficiently large n, the omega has the least traffic and the shuffle exchange has the most.

Certain traffic patterns can overload a network that could handle the same volume of evenly distributed traffic. Consider element $u_{(n-2-y)} \cdots u_{(0)} v_{(n-1)} \cdots v_{(n-1-y)}$ in the sequence described above. This corresponds to a node or cell in the three networks; those will be congested if digits $n-2-y$ to 0 in the source and digits $n-1-y$ to $n-1$ in the destination of many messages match the corresponding positions in $u_{(n-2-y)} \cdots u_{(0)} v_{(n-1)} \cdots v_{(n-1-y)}$ (assuming no shortcuts are taken). Therefore there are traffic patterns that will congest all three. There are also traffic patterns that will congest one and not the other.

Problem 5: First, determine how to route through the network. One way to look at the network is as an ordinary omega with a stage added to the beginning (since stages are identical it doesn't matter where they're "added"). Then a message can take any output in the first (added) stage, then use the usual omega network routing for the remaining stages. Accounting for the new stage numbering (1 is really 0, etc.), in stage 0 use any cell output, in stage $i > 0$ use cell output $v_{(n-i)}$. A request (u, v) routed this way will pass through cell $u_{(n-2-y)} \cdots u_{(0)} p_0 v_{(n-1)} \cdots u_{(n-y)}$ in stage y, where p_0 was the cell output used in stage 0. The tapped cell will be encountered if $u_{(n-2-y)} \cdots u_{(0)} p_0 v_{(n-1)} \cdots u_{(n-y)} = x$; since the choice of p_0 was arbitrary cell x can be avoided by choosing $p_0 \neq x_{(y-1)}$. This restriction need only be imposed if the other source and destination digits match x: $x_{(0)} = u_{(n-y)}$, $x_{(1)} = u_{(n-y+1)}, \ldots\ x_{(n-2)} = v_{(n-2-y)}$. Otherwise, p_0 might be chosen to spread traffic evenly or to avoid congestion.

If the tapped cell is in stage zero or n, there is no way to avoid it, instead we pretend we don't know its tapped and feed the enemy misinformation.