Theory of Network Communication

Fall 2002

Solutions to Assignment 3

Problem 8:

First, realize that for vertex $v = (v_{d-1}, v_{d-2}, \ldots, v_1, v_0)$, and for any $i \in [0, (d-1)]$, a bit adjustment path from $(v_{d-1}, \ldots, v_i, x_{i-1}, x_{i-2}, \ldots, x_0)$ to $(x_{d-1}, x_{d-2}, \ldots, x_i, v_{i-1}, v_{i-2}, \ldots, v_0)$ must go through v, assuming bit adjustment proceeds by flipping bits from right to left (since all $x_0 \ldots x_{i-1}$ will be flipped or not as needed, producing v, and then the upper bits will be flipped as needed).

Since this is a permutation routing problem, each vertex can only have one path, so we want to balance the set of vertices on either side of v, since for example if $i = 1$ then there are only two choices for input paths, and only two paths can go through v. But if $i = d/2$, then there are $2^{d/2}$ choices for appropriate input and output vectors, and each input vector can be matched to any output vector. No matter how that is done, each possible combination of the lower $d/2$ bits is routed through v to one of the combinations of the upper $d/2$ bits, causing $2^{d/2} = \sqrt{2^d}$ paths to pass through v.

Problem 9:

Let us use shortest paths to handle all routing decisions, where we route by shifting in the bits one at a time, using symmetry as we can to speed the process. We will specify the appropriate paths later, but note that because we are routing by shortest paths, dilation is at most d for a single path.

Now we will route every BMFP using Valiant's trick as follows: for every commodity and every source-destination pair using that commodity, route the commodity evenly to every other node in the network. Then route it back evenly from every node in the network to its destination. Note that this implies that dilation is at most $2d$ for any quantity, since it involves exactly two paths from source to destination, each of which is routed by shortest paths and therefore requires at most d edges.

Now for the congestion. There are $(2^d)^2$ source-destination pairs in the graph, since there are 2^d nodes, or 2^{2d} different commodities. Now consider half of Valiant's trick, the process of routing each source to each other vertex uniformly. The sum of the lengths of all paths is at most $d \cdot 2^{2d}$, since each of the 2^d vertices is sending 2^d commodities along at most d edges. The DeBruijn graph has at least 2^d edges, since each vertex has at least one edge (it's at most 2^{d+2} since vertex degree is at most 4, but a lower bound is sufficient). Then by symmetry (which is not directly applicable but can be assumed for the purposes of this assignment) one can say that the number of paths overlapping at an edge is at most $O\left(\frac{d \cdot 2^{2d}}{2^d}\right) = O(d \cdot 2^d)$. The demand for each path is given by the BMFP to be at most $\frac{c(v) \cdot c(w)}{c(V)} = \frac{4 \cdot 4}{4 \cdot 2^d} = \frac{4}{2^d}$. Then the product of the two is at most $O(d)$ ($4d$ if you assume symmetry exactly), which doubled (because of Valiant's trick requiring two overlapping routing problems) is still $O(d)$.

Problem 10:

Select the smallest hypercube that contains both s and t to route from s to t. Then one can more easily spread the flow than in a mesh, since each vertex in an i-dimensional hypercube is connected to each new vertex added to make it an $(i+1)$-dimensional hypercube. Therefore to go from a point to a line (0 to 1-dimensional) transfer half the current load. To go from a line to a square, vertex 0 which becomes 00 transfers half its load to vertex 10. Vertex 1 which becomes 01 transfers half its load to vertex 11; then each has load $\frac{1}{4}$. Then 00, 01, 10, and 11 become 000, 001, 010, and 011 respectively, and each gives half their load to the vertex which differs from them in the first bit (this is to route from $s = 000\ldots$; it could be extended for any s). Then clearly after dimension i is reached such that the i-dimension hypercube encompasses both s and t, the process can be reversed by compacting the data; when compacting from i to $(i-1)$, each vertex whose i^{th} bit differs from t will transfer all its load to its neighbor who matches t in the i^{th} bit. And when the process completes all load will have reached t.