SRC Technical Note

2000-001

January 6, 2000

Variations on a theme by Chuck Thacker: Hypercube routing with limited interconnections

Andrei Broder, Mark Manasse, Jim Saxe

COMPAQ
Systems Research Center
130 Lytton Avenue
Palo Alto, CA 94301
http://www.compaq.digital.com/SRC/

In a recent talk at SRC, Chuck Thacker described a routing scheme for a small hypercube network. In Chuck's talk, he described a dimension-4 network of processor/switches, with several interesting properties:

1. All packets are routed via a shortest path.
   - We can label nodes with length-d binary strings, where neighboring nodes differ in exactly one bit. The shortest distance between two nodes is equal to the number of 1-bits in the exclusive-OR of the labels. The bits which differ can be flipped in any order, but only those bits may be flipped, or else the path-length is increased.
2. No input port on a switch is connected to more than two output ports.
   - In Chuck's routing scheme, each switch has two input ports with exactly one output connection, and two inputs with two outputs. As we'll see later, this is optimal. Note that every input port might also represent a terminus for the arriving packet.
3. The routes are oblivious: once a packet arrives at a node, you can determine the next output port from the current node label, and the destination label.
   - Although Chuck didn't define it this way, we believe that Chuck's routing can be obtained by considering the set of bits to be flipped as a binary string, and flipping the rightmost 1 in the leftmost group of 1's. The symmetric choice would work equally well

4. The utilization of edges is balanced, when routing an all-pairs set of messages
    - That is, if every processor sends a message to every other processor, and the messages are routed as defined in Chuck's scheme, then all of the edges in Chuck's scheme will see (to within one) an equal number of packets routed over them, and that is true even when looking at utilization per step (if we imagine all the packets taking one step toward their destinations simultaneously).
5. An additional property, which we do not fully understand how to mimic in the extensions we'll describe, is Chuck's assertion that these properties can be preserved even in the presence of a switch failure.
    - In a subsequent conversation, Chuck agreed that he did not know how to do this while preserving all of the previously listed properties. We will show later that property 1 and 2 cannot both be maintained in our generalizations of Chuck's network in the presence of a failed switch.

Chuck's routing scheme viewed from the origin

$$\begin{tikzcd}
1010 & 1000 \arrow[l] \arrow[r] & 1001 \arrow[r] & 1011 \\
0010 \arrow[d] & 0000 \arrow[l] \arrow[r] \arrow[u] \arrow[d] & 0001 \arrow[r] & 0011 \arrow[d] \\
0110 \arrow[d] & 0100 \arrow[r] \arrow[d] & 0101 & 0111 \arrow[d] \\
1110 & 1100 \arrow[r] & 1101 & 1111
\end{tikzcd}$$

In this note, we explore some generalizations of Chuck's ideas. In particular, we offer a routing scheme in arbitrary dimensional hypercubes, prove that Chuck's properties 1, 2, and 3 are preserved in any dimension, and prove that property 4 holds overall independent of dimensionality, and on a per step basis if the number of dimensions is prime. We show that property 5 cannot hold in any odd dimension hypercube routing for which properties 1 and 2 both hold.

To define a routing for a k-dimensional hypercube, consider a packet at node s with destination node d, where s and d are represented as k-bit strings. s XOR d is a string in which the bits set to 1 represent the dimensions we must still change. Consider the rotations of s XOR d. There will be some numerically (or lexicographically; they're the same here, since all the strings have length k) least value which results. If k is prime, there will be exactly one rotation which is least, except for the all-0 and all-1 strings. If k is composite, there may be a rotation by a divisor of k which preserves the value of the string. In any case, take the shortest leftward rotation which produces that minimum value (it doesn't really matter which rotation you take, but you have to choose some way to break the symmetry). Flip the 1 which, when rotated, corresponds to the leftmost 1 in the chosen minimal rotation. Observe that, in the diagrams below, this changes the complement to Chuck's routing in dimension 4 ffor paths starting at the origin only for the path to 0101; he would arrive from 0001, while we would arive from 0100.

Chuck's routing scheme complemented (i.e., left-most 1 in right-most block) viewed from the origin

$$\begin{tikzcd}
0101 & 0001 \arrow[l] \arrow[r] & 1001 \arrow[r] & 1101 \\
0100 \arrow[d] & 0000 \arrow[l] \arrow[r] \arrow[u] \arrow[d] & 1000 \arrow[r] & 1100 \arrow[d] \\
0110 \arrow[d] & 0010 \arrow[r] \arrow[d] & 1010 & 1110 \arrow[d] \\
0111 & 0011 \arrow[r] & 1011 & 1111
\end{tikzcd}$$

Our routing scheme viewed from the origin, same labelling as above

$$\begin{tikzcd}
0101 & 0001 \arrow[r] & 1001 \arrow[r] & 1101 \\
0100 \arrow[u] \arrow[d] & 0000 \arrow[l] \arrow[r] \arrow[u] \arrow[d] & 1000 \arrow[r] & 1100 \arrow[d] \\
0110 \arrow[d] & 0010 \arrow[r] \arrow[d] & 1010 & 1110 \arrow[d] \\
0111 & 0011 \arrow[r] & 1011 & 1111
\end{tikzcd}$$

Clearly, we satisfy property 1. We only flip bits which need to be flipped.

We satisfy the generalization of property 2: each input port has outputs to at most k/2 ports.

Lemma

Given a source and destination, the route taken chooses a first bit to flip, and then flips bits from left-to-right, wrapping around as needed.

Proof

Consider the minimal-valued rotation of the exclusive-OR. The route we choose flips the leftmost 1 bit, which produces a longer run of zeroes at the left end, resulting in a numerically smaller value which is the unique minimal-valued rotation of the resulting string.

Given that, the generalization is easy: the number of outputs needed from an input port corresponds to the number of positions in which the next 1 bit might be located. But the intervening bits must all be 0; if more than half the total bits in the string were zeroes, then that block of zeroes would be longer than the string of zeroes preceding the bit we just flipped, contradicting minimality.

Moreover, we get equality only when k is even (obviously), and when there are only two bits to flip, equally spaced (e.g. 1010, or 00010001).

Our routing trivially satisfies property 3: we defined the route by looking at the minimal rotation of the XOR of the current node and the destination: the origin is irrelevant.

We had to break ties in our definition of minimality and look at rotations, but ties can happen only at the very first step, because that step extends the longest run of zeroes in the exclusive-OR by at least one, making it the unique longest run of zeroes. In fact, using the lemma above, the route is quite trivial to compute a slightly non-oblivious way: if each input port knows which bit in the original string it corresponds to, each input port routes a packet to the cyclically-next bit that needs to be flipped. Only the origin node needs to consider rotations.

Property 4 also holds, when k is prime. We'll show in that case that the load is perfectly balanced on every routing step, if we ignore the messages which have to traverse every edge. Even without primality, we'll show that the aperiodic traversals are perfectly balanced per step, and that the total load for every edge is balanced.

The total load is balanced, because

1. The dimension-i input ports on all switches have equal utilization per step, so the total utilization of every dimension-i input port is the same across all switches.
    - If the path from source s to destination d utilizes input port i on intermediate node n on step t, then the path from s XOR m XOR n to d XOR m XOR n utilizes input port i on intermediate node m on step t.
2. In looking at the set of XORs of node values, the total number of 1s in any bit position is the same. Therefore, the total utilization of all dimension-i input ports is the same as the total utilization of all dimension-j input ports.

Let's look at item 1 above slightly differently, to introduce some notation and the idea of routing from the origin and then translating. Consider any node ID n. If there are m ones in n, then the path from 0^k reaches n in m steps. Suppose that, at step j, we arrive at some node i via input port d. We can use input port d of every switch at step j by XORing both the source and destination nodes by the XOR between i and the target node.

We next want to show that every direction of every switch is equally utilized at every step when the XOR of the source and destination nodes is aperiodic. This is easy: if we start at node 0^k, heading for n, and arrive at node i via input port d at step j, we can find a corresponding destination which uses input port d+1 at step j by rotating our original destination one bit leftward. This changes the intermediate node, but the translation properties in the previous step allow us to shift both source and destination to restore the intermediate node.

This proof strongly uses the aperiodicity of the XOR: if the XOR were periodic, then rotating leftward by one might cause the first output port to leap rightward by the period. Consider, for example, the antipodal traversal: the XOR is 111...11 (i.e., 1^k) and so in any rotation the first step to be taken would be in direction k (i.e., the left-most bit).

Fortunately, for prime k, there are only two periodic XORs: 0^k and 1^k. 0^k uses no edges, so it doesn't affect the balance. 1^k gives us a family of paths of length k which never use the same switch at the same time, so they use 1 edge in each of the switches at each step (adding a slight imbalance on a per-step basis) and they use every edge in every switch exactly once (leading to total balance).

Given the limited number of periodic paths, it's possible that a slightly more sophisticated way to break ties would allow us to prove per-step near balance in composite dimension, but we'll leave that for the sequel.

We finally turn our attention to property 5, the resilience of the network to single-switch failure.

First, as a warm-up, let's show that our ceiling((k-1) / 2) edges per input port is optimal. To do this, consider all of the paths of length 2. Each of them can be routed in only two ways, and contributes one edge from an input port to an output port. Every switch is the origin of k * (k-1) / 2 length-2 paths. Therefore an average switch has k * (k-1) / 2 edges from inputs to outputs, and so some switch has at least that many. That switch has k inputs, so the average input has at least (k-1) / 2 outputs. Since the number of outputs must be an integer, some input has ceiling((k-1) / 2) edges.

Now, suppose we color the switches red and black based on their total parity. A length-2 path starting from a red node uses an edge in a black switch to arrive at a red node. Now, every red switch is the origin of k * (k-1) / 2 length-2 paths, which induce the same number of edges in black switches. If a black switch fails, we have 2^{k-1} red nodes producing paths, but only 2^{k-1}-1 black switches. The average black switch therefore has 2^{k-1} * k * (k-1) / (2 * (2^{k-1} -1)) edges. It has k input ports, so the average input port in a black switch has 2^{k-2} * (k-1) / (2^{k-1} - 1) edges, i.e. ((k-1)/2) * 2^{k-1}/(2^{k-1}-1). Since the second term is strictly greater than 1 for k > 1, this product is strictly greater than (k-1)/2.

For odd k, therefore, some input port must exceed ceiling((k-1)/2) outputs. For even k, we cannot show this: for k=4, for example, the average rises from 3/2 to 12/7, which is still less than 2. This simple counting argument is insufficient to prove even that some switch must have 8 connections between inputs and outputs in dimension 4, since 7 switches with 7 connections each yields more connections than 8 switches each with 6 connections. We do not know of an explicit construction of a satisfying fault-tolerant routing topology, with 7 connections, however.