Percolation routing in a three-dimensional multicomputer network topology using optical interconnection

Ekpe Okorafor and Mi Lu
Department of Electrical Engineering, Texas A&M University, College Station, Texas 77843
ekpe@ee.tamu.edu; mlu@ee.tamu.edu

Received 3 January 2005; revised 3 February 2005; accepted 6 February 2005; published 4 March 2005

We investigate the network communication behavior of a three-dimensional (3D) multicomputer system using optical interconnection in which faulty nodes are left in place, a concept called “fail-in-place.” We call this the percolation problem in which various amounts of missing nodes fixed in position in the network may have a dramatic effect on the network’s ability to transport data effectively. As the number of failed nodes increases, data have to be rerouted through intermediate nodes creating potential “hot spots.” These hot spots become the bottleneck that degrades performance. The ability to absorb rerouted data without ejecting it from the network is critical in massively parallel computing systems. Optical technology is a promising solution for internode communication with extraordinarily quick response time supporting enormous bandwidth. To adopt it in multiprocessor systems, efficient routing techniques are needed. We adapt self-routing strategies for all-optical packet routing in 3D mesh networks and investigate the percolation properties. To achieve percolation routing, we incorporate the features inherent in optics to achieve decoding and routing capability in real time. The objective is to develop a dynamic communication environment that adapts and evolves with a high density of missing units or nodes, and by employing analytical, experimental, and simulation methods, show that optical interconnection in a dense 3D system reduces considerably this percolation problem. © 2005 Optical Society of America

OCIS codes: 060.4510, 200.4650.

1. Introduction

Interprocessor communication has become one of the most important issues affecting system performance. The topology, the communication medium, and the routing algorithm all have a great effect on network performance. While the routing algorithm and topology decide the path between two nodes involved in a one-to-one communication, the medium determines the volume and speed with which the routing can be accomplished. The current trend in multicomputer network design is to pack nodes more densely for efficient distribution of computing resources and uniform interconnection in 3D space. This has led to improved communication performance, scalability, and density. This trend toward placing more and more computing devices, accessories, power supplies, and the data communications linking these devices into ever smaller spaces has created a new and quite interesting percolation problem. The percolation problem deals with the ability of a system as a whole to continue its functions with some of its components missing or faulty. As individual nodes in a system get smaller and the packing gets denser, it becomes less desirable to try to fix problems that occur in individual nodes or accessories. Any attempt to fix a problem with
a node may result in making problems worse in the system as a whole. It then becomes increasingly important to leave these nodes in the system, a concept referred to as “fail-in-place.”

Many recent multicomputers and multiprocessors [1–3] use grid topology based on the mesh-connected topology [4, 5]. Mesh-connected topologies, also called $k$-ary $n$-dimensional meshes, have an $n$-dimensional grid structure with $k$ nodes in each dimension such that every node is connected to two other nodes in each dimension by direct communication. Mesh-connected topologies include $n$-dimensional meshes, tori, and hypercubes.

In large multicomputer systems, exchanging data at high speeds is increasingly becoming the bottleneck that limits the performance of such large-scale systems [6–9]. This creates the need to investigate new physical approaches to dense and high-speed interconnections at various levels of a system interconnection hierarchy. In recent years, extremely fast photonic networks [10–12] have been developed that have the potential to support large bandwidth interconnections, with an extraordinarily quick response time and low latency. However, the business model that enables optics to invade telecommunication as the medium of choice for communication is not applicable in computer systems. This has the effect of limiting the use of optical components in modern computing systems because of the high cost of optical devices.

Optics and optical interconnects present some solutions that can potentially address the increasingly complex communication requirements for multicomputer systems [13–15]. Time- and wavelength-division multiplexing are techniques employed in optical communication to increase the number of connections that can be simultaneously established in a network. This has the effect of reducing the frequency of control operations and thus limited network control overhead. Two approaches are used to establish source–destination connections. One approach called link multiplexing establishes connection on more than one communication link, possibly using different channels on each link. Conversely, path multiplexing uses the same channel on each of the links.

Several routing strategies that work in the presence of failed nodes for the mesh topology have been proposed in the literature [16–20]. The simplest routing algorithms are deterministic (oblivious) and define a single path between the source and destination. A message must wait for each busy channel in the path to become available. On the other hand, adaptive routing algorithms support multiple paths between the source and destination. Adaptive routing algorithms are either minimal or nonminimal. Minimal routing algorithms allow only shortest paths to be chosen, while nonminimal routing algorithms also allow longer paths. An adaptive routing algorithm is fully adaptive if it does not impose any restriction on the selection of nonfaulty profitable links, and is partially adaptive otherwise. Therefore, a fully adaptive algorithm can exploit all alternative optimal paths to well disperse local congestion, thus outperforming deterministic and partially adaptive algorithms. While most of these approaches are local-information-based (knowledge of only neighbor status), others are global-information-based. Local-information-based routing does not yield the shortest possible path in the presence of failures because insufficient information is available when the routing decisions are made. On the other hand, while global-information-based routing can achieve optimal or near routing, its overhead in maintaining up-to-date fault information at all network nodes is usually quite high. The main challenge is to devise a simple and effective way of efficiently routing information in a system that has a high degree of failed nodes that can be implemented using optical interconnection devices with limited global fault information.

Several header-processing systems have been proposed that can be used to implement an all-optical routing [21, 22]. In Ref. [21] an all-optical header processing using a terahertz optical asymmetric demultiplexer (TOAD) is demonstrated that has a bit rate of 250 Gbit/s. This TOAD-based header recognition operates at low energy and allows photonic integra-
A disadvantage with this approach is that the control pulse has to be synchronized with the header bits. To circumvent this, in Ref. [23] an asynchronous multioutput all-optical header processing technique based on the two-pulse correlation principle in a semiconductor laser amplifier in a loop optical mirror configuration (SLALOM) is presented. This concept was employed in an all-optical packet switch [24]. One disadvantage though in Ref. [23] is that the SLALOM configuration is too large to allow photonic integration.

In our implementation of an optical interconnection network that supports a percolation routing in a multicomputer system operating under the fail-in-place condition, we propose a dual-output header differentiation scheme that will allow a synchronous operation suited to photonic integration. The model we propose is an all-optical time-division-multiplexed transmission based on TOAD. Optical time-division multiplexing (OTDM) is an alternative to WDM for future networks that utilize a single wavelength at high (> 100 Gbit/s) data rates [25, 26]. In OTDM networks many signals are combined before being transmitted with a single wavelength. Each signal from a lower-bit-rate source is broken up into many segments (slots), each having short duration and multiplexed in a rotating repeating sequence (i.e., round-robin fashion) onto a high bit-rate transmission line. The use of short-duration (soliton) pulses allows information to be transmitted at very high bit rates (> 100 Gbit/s). An asset of OTDM is its flexibility; the scheme allows for variation in the number of signals being sent along the line and constantly adjusts the time intervals to make optimum use of the available bandwidth. Consequently, it is believed that OTDM networks are excellent candidates for meeting the future system requirements for massive ultrafast networks [27–29].

The rest of this paper is organized as follows. In Section 2, we introduce the percolation problem with emphasis on the fail-in-place disorder. In Section 3, we present our routing strategy that is suited for the optical interconnection network we propose. We present a feasible optical implementation of our routing method in Section 4. Simulation results on the performance of such a system are shown in Section 5, while we draw some conclusions in Section 6.

2. Percolation in Large Systems

The percolation theory as it describes computing with a faulty array of processors has been introduced in Ref. [30], while Ref. [31] looks at this phenomenon in large storage systems. In our context, percolation deals with the effects of varying the richness of interconnections present in a random system. The basic idea of percolation is the existence of a sharp transition at which the long-range connectivity of the system disappears (or, going the other way, appears). This transition occurs abruptly when some generalized density in this system reaches a critical value (percolation threshold). The percolation transition makes percolation a natural model for describing a diversity of phenomena. If we model failures as complete elimination of both data and communication in a node, then we have just described a percolation model. The central idea is that various amounts of missing elements fixed in position in a network may have a dramatic effect on the network’s ability to transport material, and in our case, information. Figure 1 is an illustration of a network with failed nodes. Every node in the network has a direct network connection with all its six neighbors. For nodes located on the surface of the network or adjacent to failed nodes, the number of neighbors is obviously less than six.

An important definition is the percolation threshold, which is a fraction of lattice elements, or in our case, good nodes, below which all the nodes remaining are connected to one another only as small clusters, not enough to span the whole structure or network. So if a node fails, availability requires that the repair be accomplished quickly. However, in a large system, trying to repair a bad node may ultimately introduce more problems, hence the fail-in-place concept. This means that in a cluster of nodes, there will increasingly be...
nodes that fail and are allowed to remain in place in order to reduce maintenance cost and avoid the introduction of more problems due to human error. In such an environment, the connectivity, bandwidth and data path lengths among others are adversely affected.

Increasing the number of failures will drastically degrade the performance by creating hot spots and discontinuities in the network. As the number of failed nodes increases, we expect that the overall bandwidth will decrease. This decrease in bandwidth is due to fewer paths becoming available because traffic is rerouted around failed nodes. This also has the effect of creating hot spots. As a consequence, with half the nodes in operation, the bandwidth for data traffic is only 10% of the initial value. To see the effect of faulty nodes on overall bandwidth, we can calculate the maximum number of parallel paths that exist connecting opposite faces of an array with faulty nodes. Figure 2 shows the min-cut in both two-dimensional (2D) and 3D site percolation as a function of the fraction of good nodes. We define the min-cut as the smallest fraction of links that must be cut by a continuous, but not necessarily flat, surface passing normal to one axis of either a 2D or 3D lattice.

This graph is obtained by an algebraic analysis of a worst-case measure of overall bandwidth reduction due to node failure. Actual observed bandwidth may be even lower because of hot spots in the message traffic. As seen in the graph, when the number of faulty nodes is half the total number of bricks, the relative bandwidth available is reduced to 10% in the 3D site percolation. The increase in path length due to avoiding failed nodes is not particularly significant in our design. The increase becomes large when the number of faulty nodes is close to the percolation threshold. As mentioned above, we consider a design with 50% or more good nodes.

3. Percolation Routing with Optical Interconnectivity

In this section we now take a look at the percolation routing in a 3D mesh adaptable for optical interconnection networks. Our proposed scheme is based on the concept of limited global fault information and eliminates the need for lookup tables. By limited global information, we mean that individual nodes know which, if any, of its neighbors are faulty. This model is also known as the “one-step local information” model. As mentioned, a fault means that its entire communication links are faulty. Our emphasis is not so much the dynamic nature of the faults that occur in the system, but how well the system as a whole
handles steady-state faults in a fail-in-place environment.

The size of a lookup table for a network of a significantly large number of nodes presents a challenge to large-scale network design. The lookup times for such large address spaces are of the order of micro- or milliseconds. In addition to the large lookup times, an OEO (optical–electrical–optical) conversion has to be performed in order to carry out the lookup operation to determine the next hop in an optical system. The conversions that have to take place at each intermediate node will undoubtedly degrade the performance, and as such, the goal of using optics in the first place is defeated.

However, since optical logic is still in its infancy, designs that involve complex logic in decoding header information will not achieve the expected improvements in routing performance. We harness the features inherent in optics in our design to achieve header recognition and decoding, data routing capability, and contention resolution in real time. In our design we also eliminate the need for optical buffering, which is clearly unsuitable for large multiprocessor systems, by aggressively reducing the probabilities of data contentions and unavailability of outgoing links at each intermediate node.

In our scheme, the path $P$ between two nodes, source $S$ and destination $D$, is provided with an alternative path $A$. Consequently, every primary link is provided with an alternate physical one. These two paths can be switched back and forth depending on the availability of output links at each intermediate node. An address encodes a unique path from $S$ to $D$. A packet is encapsulated in layers of address markers corresponding to the action taken at an intermediate node $i$. After each marker is traversed, it is stripped from the address exposing the next marker. These markers may be defined on a per-destination basis, on an $S$-$D$ pair basis, or on a per-flow basis. However, we acknowledge that with a self-routing scheme it is difficult to implement congestion control and traffic engineering. For each destination address, each intermediate node has a preferred output link and an alternative

© 2005 Optical Society of America
JON 6178 March 2005 / Vol. 4, No. 3 / JOURNAL OF OPTICAL NETWORKING 161
output link should the preferred link be unavailable. If an alternative output link is taken, then the address markers have to change accordingly. In our design, an address is made up of two fields as shown in Fig. 3. At each node, the preferred output link is always chosen. This corresponds to field \( P \). If, however, an alternative link is taken then the field to which the link belongs becomes the preferred field \( P \).

Every node in a 3D mesh has a direct network connection with all six of its neighbors. The exceptions are the nodes on the surface of the mesh and those adjacent failed nodes. The process of routing data from source to destination can be done in three phases. The first phase is to determine the primary and alternative intermediate nodes between source and destination and thereby forming the address. An address basically represents an action taken at each intermediate node. The action taken directs a packet to the appropriate output link. This means that the structure of an address for a packet while remaining unchanged throughout the transport process will illicit different actions at different nodes. The action is processed in real time. The node identifies an address and on the fly directs the packet to one of its output links. If the primary link is unavailable, the alternate link is chosen. The second involves path setup by assigning channels to each of the primary outgoing links at each intermediate node. Since we consider optical real-time decoding with little overhead, the header should not be extremely large or complicated. The destination address has information about the intermediate nodes, while these intermediate nodes make decisions on assigning channels to the outgoing link used to get to the next intermediate node. Third, the data are sent out by the source. As already mentioned, this self-routing scheme should support a 3D structure. Each node is represented by a 3-tuple \( N(\mathbf{x}, \mathbf{y}, \mathbf{z}) \), which completely describes the position of a node in a 3D mesh.

3.A. Address Formulation, Path Setup, and Channel Assignment

Given a source node \( S = (x_s, y_s, z_s) \) and a destination node \( D = (x_d, y_d, z_d) \), we define the address as the set \( \text{ADDR} = (S, i_2, \ldots, i_{n-1}, D) \) to give minimal distance routing, where \( i \) represent the intermediate links between the source and destination nodes, \( n \) number of nodes and there are \( n - 1 \) intermediate links. The primary route address \( \text{ADDR}_P = (S, i_2, \ldots, i_{n-1}, D) \) and alternate route address \( \text{ADDR}_A = (S, i_2, \ldots, i_{n-1}, D) \) are chosen from a set of equally unique routes that satisfy the following conditions:

1. \( \text{ADDR}_P \cup \text{ADDR}_A = \{S, D\} \)

2. \( \text{ADDR}_P \{i_x\} \) and \( \text{ADDR}_A \{i_x\} \) [where \( x = (2, 3, \ldots, n - 1) \)]; have only one of the \( (x, y, z) \) coordinates differing by only one in Hamming distance, as well known concept in channel coding is used.

Condition 1 means that no intermediate node address belongs to both the preferred and alternate node address set except the source and destination addresses. In condition 2, the Hamming distance, as well known concept in channel coding is used. Let \( a \) and \( b \) be two binary sequences or coordinates in space of the same length. The Hamming distance between these two addresses is the number of symbols that disagree. To further illustrate, we
give an example. If the third intermediate node of the preferred address is (2,3,4), the set of possible options for the alternate address will be \{(1,3,4), (3,3,4), (2,2,4), (2,4,4), (2,3,3), (2,3,5)\}. These obviously correspond to the six nearest neighbors. The conditions mentioned above try to guarantee that the minimal path length feasible is maintained when the alternate path is taken and in most cases only an additional link is traversed in the event that a link is faulty, especially near the destination. However, as the number of faults increases in the network, the path length will invariably increase. Our scheme is best described as a nonminimal adaptive routing scheme based on limited global information.

Having determined the intermediate nodes/links needed for data transfer, the sender \(S\) sends a header flit that contains information on the primary and alternate intermediate nodes (links) needed for the data transfer [ADDR\(_P\) = \((S, i_2, \ldots, i_{n-1}, D)\) and ADDR\(_A\) = \((S, i_2, \ldots, i_{n-1}, D)\)]. Each intermediate node has the task of assigning unused channels on the appropriate outgoing links. A hold state is placed on any channel assigned. If the header flit is successfully transmitted to the destination \(D\), the sender \(S\) then sends the data preceded by the control flit that contains information on the intermediate routes with channel assignments. A channel is released only when the final flit of data passes through it. In our implementation, we overlap the path setup/channel assignment phase with data transfer. The data lags behind to give time for the intermediate nodes to assign and place a hold on the appropriate channels.

In the next subsections, we outline the different strategies in our algorithm used to perform percolation routing. Emphasis is placed on the feasibility of implementing the algorithm using optical devices.

### 3.B. Routing Algorithms

In a 3D mesh array or nodes, there may exist up to six first hops that lead to a shortest path to a distant node. Two natural ways of selecting one of the alternative first steps for routing exist. The first follows a predetermined rule, and the second is random. The predetermined rule is traditionally used in nonadaptive mesh routing. However, the random rule is likely to cope with high levels of failed nodes more gracefully. In what follows, we describe these two approaches of our routing methodology.

#### 3.B.1. Notation

Given a source node \(S = (x_s, y_s, z_s)\), and a destination node \(D = (x_d, y_d, z_d)\), we define \(C_{x,d}\) to be the smallest cube that includes both \(S\) and \(D\). Without faulty nodes, the algorithm will find a path from \(S\) to \(D\) within \(C_{x,d}\). It is worth mentioning that \(C_{x,d}\) is not unique for an \(S-D\) pair. With \(S\) as our focal point, let all the nodes \(n\) in \(C_{x,d}\) be labeled \(n_{x,y,z}\), where \((x, y, z)\) represent the \(x, y, z\) coordinates in space. In the XYZ rule, a canonical ordering, the approach is to always move first in the \(x\) direction if possible, then in the \(y\) direction if possible, then in the \(z\) direction. The pair of nodes \(S-D\) is represented as \((s_{x,y,z}, d_{x,y,z})\). If the number of virtual channels is \(C\), then the algorithm finds for each \(S-D\) pair an allocation \(C_{i}\) that is free. We define links as follows:

\[
l_x \in L(x_0, x_1, \ldots, x_{d-s}) = L_X,
\]

\[
l_y \in L(y_0, y_1, \ldots, y_{d-s}) = L_Y,
\]

\[
l_z \in L(z_0, z_1, \ldots, z_{d-s}) = L_Z.
\]

The value \(| d - s |\) represents the number of intermediate nodes between the source and destination. Thus, a path from \(S\) to \(D\) is represented as path \((S-D) = \{L_X, L_Y, L_Z\}\). Consequently, a channel allocation for an \(S-D\) pair is denoted by an integer value channel ±
Algorithm 1

Algorithm: XYZ rule (No faults-path multiplexing)
Inputs: (S-D) pair, channel_alloc(sx,y,z, dx,y,z), B_channel
Output: XYZ routing

begin
    initialize channel_alloc(sx,y,z, dx,y,z) = null for all links;
    for( each S-D pair requesting connection) do
        path(sx,y,z, dx,y,z) = {null} and B_channel = {null};
        x = sx;
        while (φ ≠ dx) do
            add lφ to path(sx,y,z, dx,y,z) and φ = φ + 1;
            if φ = dx, break; then φ = η;
            if η = z, break;
        end while
        B_channel = B_channel ∪ channel_alloc(sx,y,z, dx,y,z)
        channel = any number not in B_channel and less than C
        for (every link {LX, LY, LZ}in path(sx,y,z, dx,y,z)) do
            set C(sx,y,z, dx,y,z) = channel;
            channel_alloc(sx,y,z, dx,y,z) = channel_alloc(sx,y,z, dx,y,z) ∪ {channel};
        end for
    end for
end

We also represent a set of blocked channels that exist on the links that make up the path (S-D) as B_channel. To simplify the notation, we define two states φ = current state, and η = next state.

η = sx if φ = x = dx,

η = sy if φ = y = dy,

η = sz if φ = z = dz.

First, we discuss the XYZ routing for path and link multiplexing and then show an adaptation for faulty nodes implemented using optical devices.

3.B.2. Dimension Order Routing (XYZ Routing)

In the path-multiplexing algorithm above (Algorithm 1), channels are not assigned to connections as the path is constructed. Information about the channel availability is obtained first before a channel is assigned to a path. As a consequence, if a channel is not available in the preferred route from source to destination, a longer route may have to be taken, and data could be ejected from the network or some delay incurred while waiting for channel availability. Link multiplexing is better suited to alleviate this constraint. Therefore we can modify the algorithm above by eliminating “channel” and “B_channel” as shown below.

Let us take a look at the percolation routing of the XYZ rule. As mentioned above, C_{s,d} is not unique for an S-D pair. This property enables the system to circumvent faulty nodes or links by implementation of the algorithm for the alternate path. Information about the failed link is obtained during the path setup phase. It is important to note that link failure is different from link unavailability. The consequence being that link unavailability
Algorithm 2

**Algorithm:** XYZ rule (No faults - link multiplexing)
**Inputs:** (S-D) pair, channel_alloc(s_x,y,z, d_x,y,z)
**Output:** XYZ routing

begin
initialize channel_alloc(s_x,y,z, d_x,y,z) = null for all links;
for( each S-D pair requesting connection ) do
    path(s_x,y,z, d_x,y,z) = {null};
    x = s_x;
    while(φ =! d_φ) do
        add l_φ to path(s_x,y,z, d_x,y,z) and φ = φ + 1;
        channel_alloc(l_φ) = channel_alloc(l_φ) + 1
        if φ = d_φ, break; then φ = η;
    end while
end for
end

Algorithm 3

**Algorithm:** XYZ rule (Percolation)
**Inputs:** (S-D) pair, channel_alloc(s_x,y,z, d_x,y,z)
**Output:** XYZ routing

begin
initialize channel_alloc(s_x,y,z, d_x,y,z) = null for all links;
for( each S-D pair requesting connection ) do
    path(s_x,y,z, d_x,y,z);
    x = s_x;
    while(φ =! d_φ) do
        if (l_φ =! Ψ) 
            add l_φ to path(s_x,y,z, d_x,y,z) and φ = φ + 1;
            channel_alloc(l_φ) = channel_alloc(l_φ) + 1;
        else
            φ = η and add l_φ to path(s_x,y,z, d_x,y,z);
            channel_alloc(l_φ) = channel_alloc(l_φ) + 1;
            if φ = d_φ, break; then φ = η;
        end if
    end while
end for
end

is transient and not permanent, so the information is treated as such. The differentiation between link failure and unavailability is done at the hardware sense level. The objective of the routing algorithm will be to minimize the number of additional steps needed to circumvent a faulty node. Our heuristic percolation routing follows the XYZ rule, albeit only loosely. If a fault is detected in the x axis, move a step in the y axis toward the destination node. The same rule applies to faults in both y and z axes. In both cases move in the z and x axes respectively. This method is described as cyclic-XYZ rule. We define Ψ to denote a faulty link.
3.B.3. Probability-Based Percolation Random Routing

In this approach each intermediate node determines the next step on the basis of its current position, calculated probability of success, and the destination address. The current node chooses a next step in order to produce a minimal path to the destination. Routing at each node is based on a calculated probability vector \( P = (P_N^1, \ldots, P_N^n) \). Each intermediate node determines a set of faulty neighbors and updates its faulty set \( F_{x,y,z} \). With this it calculates the probability vector \( P_1 \). It then performs an exchange with its neighbors to determine the rest of the vector elements, \( P_N^l \), for all \( 2 \leq l \leq n \). \( P_N^1 \) represents the probability that a destination at distance \( l_{(x,y,z)} \) cannot be reached from the current node \( N_{(x,y,z)} \) using a minimal path due to a faulty intermediate node. In summary, each node runs the following basic steps of this algorithm:

1. Determine \( F_{x,y,z} \) of unreachable neighbors; compute probability vector \( P = (P_N^1, \ldots, P_N^n) \) based on \( F_{x,y,z} \) and exchanged information from neighbors.
2. Determine primary and alternate route to destination based on probability vector and encode information in header and data flits.

A path is faulty if it includes at least one faulty or unreachable node. Since there are at most six neighbors to each node and \( f \) is defined as the number of faulty neighbors, then

\[
P_N^1 = f / 6.
\]

If we define \( Q_N^{(i)} \) to be the probability of reaching a destination at distance \( l \) from \( N \) via its neighbor \( N^{(i)} \), then the probability \( P_N^l, l \geq 2 \), can be expressed as

\[
P_N^l = \prod_{i=1}^{6} \left(1 - Q_N^{(i)}\right),
\]

where

\[
Q_N^{(i)} = \begin{cases} 
0 & \text{if node } N^{(i)} \text{ is faulty} \\
\frac{1}{6} \left(1 - Q_{N^{(i-1)}}\right) & \text{otherwise}
\end{cases}
\]

After the probability vector \( P_N^l \) is determined, a source node \( S \) selects two paths at random that have the least \( P_N^l \). The source node then encodes these two addresses onto the header and data flits as the primary and alternate addresses. See Algorithm 4.

To show how these two approaches perform in various degrees of number of failed nodes, we first define a traffic model. In the traffic model, each node is either a compute node or a storage node. We have an \( n \times n \times n \) network, with \( n = 10 \). Of the 1000 nodes, 100 are compute nodes. Each compute node reads data from all the storage nodes, performs some specified computation on the collected data, and writes the result to all the storage nodes. Next, for each storage node, its data are mirrored to another node in the network. An acknowledgement is also returned to the original compute node. The idea is to generate as much traffic as possible to every node. The compute nodes are distributed uniformly throughout the whole network. This traffic model is simulated using various degrees of failed nodes. We simulate the traffic model for both types of routing rules. We also note that of particular interest to us is in understanding the hot spots that form under load. The performance of the network is more adversely affected by the bottlenecks and worst-case loadings rather than the average loadings. We do not consider single link failures; rather, a node failure means that all six outgoing links are no longer available for routing.
Algorithm 4

**Algorithm**: Random rule (Percolation)

**Inputs**: (S-D) pair, f, channel_alloc

**Output**: Random percolation routing

```
begin
  initialize channel_alloc(sx,y,z, dx,y,z) = null for all links;
  for each S-D pair requesting connection do
    path(sx,y,z, dx,y,z) = {null};
    \( p^S_f = \frac{L}{6} \) /* determine probability vector */
    for(l=2 to n) do 
      for(i = 1 to n) do
        if (\( N^0 \) is faulty)
          \( Q^0_i = 0 \);
        else
          \( Q^0_i = \frac{L}{6}(1 - Q^{N^0}_{i-1}) \);
          \( P^N_i = P^n_i(1 - Q^0_i) \);
      end for
    end for
    Randomly select two minimal routes based on least \( P^N_i \);
    Encode primary and alternate address such that \( ADDR_P \cup ADDR_A = \{S,D\} \);
    while (\( N =! D \) do
      if (\( l_{N(P)} =! \psi \) /* \( l_{N(P)} \) denotes the primary link on current node */
        add \( l_{N(P)} \) to path(sx,y,z, dx,y,z);
        channel_alloc(\( l_{N(P)} \)) = channel_alloc(\( l_N \)) + 1;
      else
        \( l_{N(P)} = l_{N(A)} \) and add \( l_{N(A)} \) to path(sx,y,z, dx,y,z);
        channel_alloc(\( l_{N(A)} \)) = channel_alloc(\( l_N \)) + 1;
      end if
    end while

end
```

Figure 4 shows that with all the nodes working, the predetermined rule produces a highly regular and balanced arrangement. By contrast, the random routing scheme produces a tail of heavily loaded buffers when all nodes are working. However, as the number of failed nodes increases, the random routing scheme is progressively better than the predetermined case up to 50% as shown in Fig. 5.

4. Feasibility of Optical Implementation

For a practical implementation of our all-optical header processing system utilized in devising an all-optical packet switch, the header processor should be scalable, have low power consumption, be high speed, and be photonic integrated on a chip. In particular, scalability defines the capacity of the header processor to recognize a large amount of header information and eventually to update the system easily to recognize more headers. High-speed operation is required for matching the line rate of the optical transmission system so that no bottleneck is generated. Low power consumption and photonic integration guarantee large-scale production, low cost, and integration with other functionalities on the same chip.

Optical implementation of our routing scheme is based on the following:
Fig. 4. Distribution of link loading on traffic model for 0% faulty nodes.

Fig. 5. Distribution of link loading on traffic model for 50% faulty nodes.
• Address recognition
• Ability to decode optical data in real time
• Generation of switching control signal
• Contention detection/resolution

In this section we demonstrate the feasibility of implementing our routing scheme and the ability to accomplish all the goals stated above by simulation. The architecture is based on a system that has as its inputs an OTDM packet containing header and payload information. Figure 6 shows a block diagram of the simulated OTDM transmission system. It consists of an OTDM packet generator, additive white Gaussian noise (AWGN) channel, optical router, matched filter, and detector. Figure 7 shows the format of two consecutive packets at the output from the OTDM packet generator. Framing bits indicate the inter-packet boundaries thereby providing a synchronization mechanism. The address bits indicate the destination port to which the payload information is routed. A value of 1 (0) results in the payload information being routed to link 1 (link 2) of the OTDM router.

To demonstrate all-optical address recognition and single-bit self-routing, a single node of the OTDM router is constructed from two TOADs. As shown in Fig. 8, the switching node consists of an all-optically controlled routing switch (TOAD #2 with $\Delta x = T c/2$, where $T = 20$ ps is the width of switching window), an ultrafast controller (TOAD #1 with $\Delta x = T c/2$, where $T$ is the width of the window equals the duration of the address bit), and a buffer. The ultrafast controller all-optically sets the states of the routing switch (TOAD #2) in a switched or unswitched state, and the optical buffer matches the delays of the input packet to the processing delay of the routing controller [32].
and then used as the control signal of TOAD #1. A portion of the packet is split off and sent to TOAD #1 before entering the buffer. TOAD #1 reads the packet destination address bit and uses it as the optical routing control for the routing switch (TOAD #2). In a single-bit routing scheme, packets with address bit of value 1 are routed to output link 1, while packets with an address bit of value 0 are routed to output link 2. Thus photonic packets are self-routed through an all-optical ultrafast switch without the need for optoelectronic conversion.

5. Simulation Results

This section presents simulation results for the 3D mesh in a fail-in-place situation with varied amounts of failed nodes. The simulator is a 10,000-line C++ program capable of modeling multicomputer nodes, routing, and interconnection network properties already discussed in the preceding sections. We compare the results of both electrical and optical interconnects using the same topology but based on their known properties for various degrees of faulty nodes, to show the strength of our routing methodology. The following parameters and values listed in Table 1 are used in modeling the optical header decoding and routing subsystem.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit period</td>
<td>4 ps</td>
</tr>
<tr>
<td>Control pulse width FWHM</td>
<td>2 ps</td>
</tr>
<tr>
<td>Control pulse wavelength</td>
<td>1500 nm</td>
</tr>
<tr>
<td>Data signal width FWHM</td>
<td>2 ps</td>
</tr>
<tr>
<td>Data signal wavelength</td>
<td>1550 nm</td>
</tr>
<tr>
<td>SOA length</td>
<td>300 µm</td>
</tr>
<tr>
<td>SOA active area</td>
<td>$0.2 e^{-12}$ m²</td>
</tr>
<tr>
<td>SOA carrier density</td>
<td>$10^{24}$ m⁻³</td>
</tr>
<tr>
<td>SOA confinement factor</td>
<td>0.3</td>
</tr>
<tr>
<td>SOA position $\Delta x$</td>
<td>2 ps</td>
</tr>
</tbody>
</table>

The electrical interconnection network is modeled using the gigabit Ethernet specifications approved by the IEEE 802.3z Gigabit Task Force in 1996. Throughput, packet transmission delay (or network latency), and worst-case loading effects for various degrees of faulty nodes are used as the indices of performance. The two networks are driven with the following traffic loads: uniform random traffic, saturation traffic, and hot-spot traffic.

In uniform random traffic mode, every node generates messages with exponentially distributed arrival times and uniformly distributed destination nodes. Figure 9 shows the channel width for both types of interconnects as a function of dimension. Optics has a wider channel in all cases. As shown in Fig. 10, optical interconnects achieve lower latency...
and are closer to achieving constant minimal network latency for various degrees of faulty nodes for a 3D mesh.

Figure 11 illustrates the effect of faulty node degree on saturation throughput for both electrical and optical interconnection networks. The destination nodes for each source are uniformly distributed, and fully adaptive routing is used. Each source is constantly injecting a message into the network. This is done to maintain 100% utilization of the network and to keep the network saturated. The figure shows that the optical interconnection network achieves a much higher saturation throughput and is less affected by the number of faulty nodes. Both interconnection networks have high bisection bandwidth and a large number of routing options; however, as the number of faulty nodes increases, the number of routing options decreases. This decrease is more apparent in the electrical case owing to its smaller network link capacity.

![Channel width](image1)

**Fig. 9. Channel width.**

![Network latency (10\times10\times10)](image2)

**Fig. 10. Network latency for a 10 \times 10 \times 10 3D mesh network with arbitrary fixed message size under uniform random traffic.**
In hot-spot traffic, the network is over saturated. The idea is to identify bottlenecks and worst-case loading in an effort to understand the hot spots that form under this traffic mode. The next set of figures show curves for worst-case loading. The hot nodes (bottlenecks) will increasingly have to deal with a lot more load being diverted through them as the number of faulty nodes increases. In Fig. 12 the maximum number of packets passing per unit time is plotted as the degree of faulty nodes is increased for both electrical and optical networks. Figure 13 shows the number of hot spots created for a worst-case loading as the number of faulty nodes is increased for both electrical and optical cases. The result shows that with optical interconnects, the number of hot spots created is relatively small compared with electrical. As a consequence, more routing options are available, resulting in an increased throughput and a balanced load distribution.

Fig. 11. Effect of faulty node degree on saturation throughput.

Fig. 12. Effect of faulty node degree on worse case loading.
6. Conclusions

We have shown that in a large system, there will increasingly be nodes that fail and are allowed to remain in place. We have also shown through simulation analysis that bandwidth is the most critical factor affected by the so-called fail-in-place problem. We demonstrated that optical interconnection is the solution to this bandwidth need created by the percolation problem that exists in large computing systems. We introduced a routing strategy for all-optical packet-switched networks that harnesses the features inherent in optics to achieve header recognition/decoding and data routing in real time. We used analytical study and simulations to examine the feasibility of such a network. We also presented a simple OTDM system model based on the optical router developed in Ref. [32]. The model successfully recognizes the address bits and routes the data at speeds up to 0.25 Tbit/s. The all-optical communication system overcomes the bottleneck of optoelectronic conversion because of its ultrafast switching capability. This communication environment is able to adapt and evolve with a high density of missing units or nodes.

References and Links


