# Fault-Tolerant Switched Local Area Networks *

Paul LeMahieu Vasken Bohossian Jehoshua Bruck

California Institute of Technology
Mail Code: 136-93
Pasadena, CA 91125

Email: {lemahieu,vincent,bruck}@paradise.caltech.edu

## Abstract

*The RAIN (Reliable Array of Independent Nodes) project at Caltech is focusing on creating highly reliable distributed systems by leveraging commercially available personal computers, workstations and interconnect technologies. In particular, the issue of reliable communication is addressed by introducing redundancy in the form of multiple network interfaces per compute node. When using compute nodes with multiple network connections the question of how to best connect these nodes to a given network of switches arises. We examine networks of switches (e.g. based on Myrinet technology) and focus on degree two compute nodes (two network adaptor cards per node). Our primary goal is to create networks that are as resistant as possible to partitioning.*

*Our main contributions are: (i) a construction for degree-2 compute nodes connected by a ring network of switches of degree 4 that can tolerate any 3 switch failures without partitioning the nodes into disjoint sets, (ii) a proof that this construction is optimal in the sense that no construction can tolerate more switch failures while avoiding partitioning and (iii) generalizations of this construction to arbitrary switch and node degrees and to other switch networks, in particular, to a fully-connected network of switches.*

* Supported in part by the NSF Young Investigator Award CCR-9457811, by the Sloan Research Fellowship, and by DARPA and BMDO through an agreement with NASA/OSAT.

## 1 Introduction

Given the prevalence of powerful personal computers/workstations connected over local area networks, it is only natural that people are exploring distributed computing over such systems. Whenever systems become distributed the issue of fault tolerance becomes an important consideration. In the context of the *RAIN* project (Redundant Arrays of Independent Nodes) at Caltech,we've been looking into fault tolerance in all elements of the distributed system (see Figure 1 for a photo). One important aspect of this is the introduction of fault tolerance into the communication system by introducing redundant network interfaces at each compute node and redundant networking elements. For example, a practical and inexpensive real-world system could be as simple as two Ethernet interfaces per machine and two Ethernet hubs.

**Figure 1. The RAIN System**

We have been working primarily with Myrinet networking elements [1]. This introduces switched networking elements where the switch cost happens to be low in comparison to interface costs so we can construct more elaborate networks of switches. With technology like this as motivation, we were faced with the question of how to connect compute nodes to switching networks to maximize the network's resistance to partitioning. Many distributed computing algorithms face trouble when presented with a large set of nodes which have become partitioned from the others. A network that is resistant to partitioning should lose only some constant number of nodes (with respect to the total number of nodes) given that we don't exceed some number of failures. After additional failures we may see partitioning of the set of compute nodes, i.e., some fraction of the total compute nodes may be lost. By carefully choosing how we connect our compute nodes to the switches we can maximize a system's ability to resist partitioning in the presence of faults.

The construction of fault-tolerant networks was studied in 1976 by Hayes [2]. This paper looked primarily at constructing graphs that would still contain some target graph as a subgraph even after the introduction of some number of faults. For example, the construction of $k$-FT rings was explored which would still contain a ring of the given size after the introduction of $k$ faults.

This work is complementary to the problems we are looking at. Specifically, some constructions that add fault-tolerance to networks of switches can be used to enhance the fault-tolerance of the final set of compute nodes connected to the network. Other papers that address the construction of fault-tolerant networks are [3] for fault-tolerant rings, meshes, and hypercubes, [4, 5, 6] for rings and other circulants, and [7, 8, 9] for trees and other fault-tolerant systems.

A recent paper by Ku and Hayes [10] looks at an issue similar to the one covered in this paper. In particular, it looks at maintaining connectivity among compute nodes connected by buses. This is equivalent to not permitting any switch-to-switch connections in our model. We are looking at permitting such switch-to-switch connections to allow the creation of useful switch topologies and then connecting compute nodes to this network of switches.

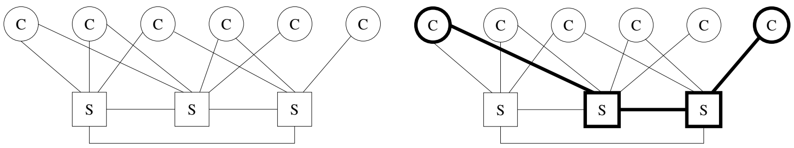
Our main contributions are: (i) a construction for degree-2 compute nodes connected by a ring network of switches of degree 4 that can tolerate any 3 switch failures without partitioning the nodes into disjoint sets, (ii) a proof that this construction is optimal in the sense that no construction can tolerate more switch failures while avoiding partitioning, and (iii) generalizations of this construction to arbitrary switch and node degrees and to other switch networks, in particular, to a fully-connected network of switches.

The structure of the paper follows closely the contributions listed above. In Section 2 we formally define the problem of creating fault-tolerant switched networks. We then, in Section 3, give our primary construction based on degree-2 compute nodes that are connected as "diameters" in a ring of switches and prove the correctness and optimality of this construction. The description of the generalized form of this construction follows in Section 4. Section 5 presents a systematic method for connecting compute nodes to a complete graph of switches (clique), and finally, Section 6 provides comments addressing other networks of switches as well as final conclusions.

## 2 Problem Definition

In this section we introduce the building blocks of our system and define its properties.

**Building blocks.** A distributed computing system is composed of a set of interconnected *switches* forming a communication network a set of *compute nodes* connected via switches. Figure 2 shows an example of a system with 3 switches and 6 nodes. Switches and nodes are characterized by their degree. We denote by $d_s$ the degree of




(a) (b) (c)

**Figure 2. (a) Example of a computing system composed of switches ($S$) and computing nodes ($C$). (b) Communication path between two nodes. (c) Another path between the same nodes.**

a switch (i.e., the number of network ports it has) and by $d_c$ the degree of each compute node (i.e., it's number of network interfaces).

**Homogeneous system.** We look at *homogeneous systems* in which the switch degree $d_s$ is the same for all switches, and the compute node degree $d_c$ is the same for all nodes. That is not the case for the system of Figure 2. Figure 3 shows a homogeneous system.

**Connectivity.** For our purposes, we do not consider two compute nodes connected unless they are connected by a path solely through the switch network. In other words, compute nodes are not permitted to forward packets. Thus, $c_1$ connected to $c_2$ and $c_2$ connected to $c_3$ does not imply that $c_1$ is connected to $c_3$ as illustrated in Figure 4. Our goal is to connect the nodes to the network of switches in a redundant way, so that there exist more than a single communication path between two nodes, as shown in Figure 2. In the case of a fault one path may become inactive and another will be used.

**Faults.** We primarily consider switch faults, the failure of a switch as a whole. We also allow "lesser" faults such as link and node faults. A switch-to-switch link fault can always be looked at as a switch fault. Node faults and node-to-switch link faults are not especially interesting here since they are the most benign, having no affect on the connectivity of the other nodes.

**Non-locality.** Locality is defined over the network of switches. The distance between two switches is the number of links in the shortest path between them. Figure 5 illustrates the idea. As you'll see, the driving idea

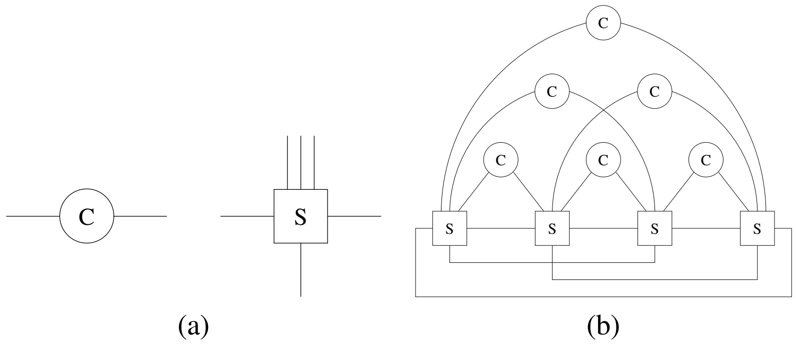



**Figure 3. (a) Computing node of degree $d_c = 2$ and switch of degree $d_s = 6$. (b) Homogeneous system made up of these components.**

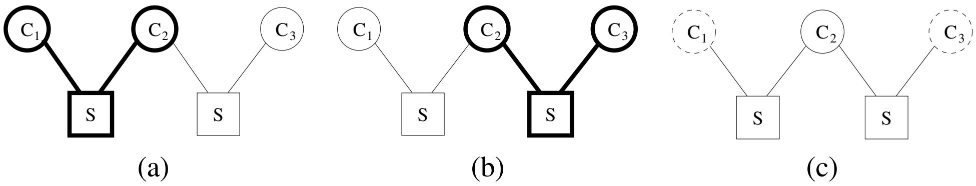



**Figure 4. (a) $c_1$ is connected to $c_2$. (b) $c_2$ is connected to $c_3$. (c) $c_1$ is not connected to $c_3$.**

behind our construction is to to connect the compute nodes to switches exhibiting *non-locality*.

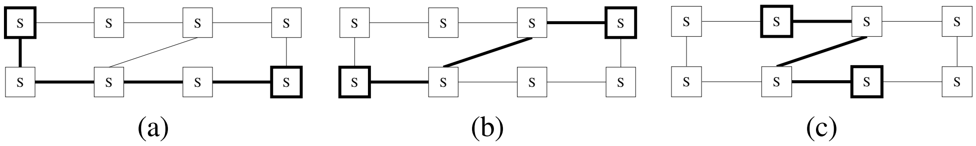



**Figure 5. Examples of switch to switch distance. (a) $d = 5$, (b) $d = 4$, (c) $d = 4$.**

Having defined the setting of our study we will look at two particular types of switch networks: the ring and the clique.

## 3 A Ring of Switches

We consider the following problem: Given $n$ switches of degree $d_s$ connected in a ring, what is the best way to connect $n$ compute nodes of degree $d_c$ to the switches to minimize the possibility of partitioning the compute nodes when switch failures occur? Figure 6 illustrates the problem.

### 3.1 A Naïve Approach

At a first glance, Figure 7a may seem a solution to our problem. In this simple construction we simply connect the compute nodes to the nearest switches in a regular fashion. If we use this approach, we are relying entirely on fault-tolerance in the switching network. A ring is 1-fault-tolerant for connectivity, so we can lose one switch without upset. A second switch failure can partition the switches and thus the compute nodes, as in Figure 7b. This prompts the study of whether we can use the multiple connections of the compute nodes to make the compute nodes more resistant to partitioning. In other words, we want a construction where the connectivity of the nodes is maintained even after the switch network has become partitioned.

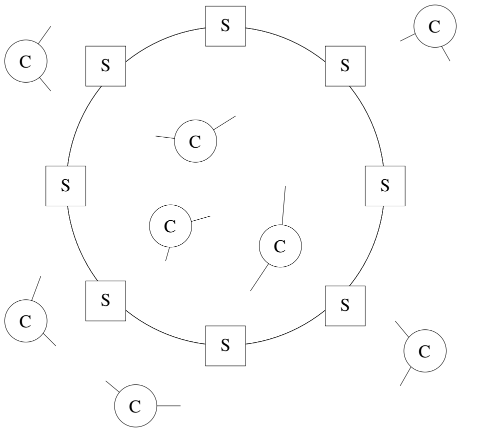



**Figure 6. How to connect $n$ compute nodes to a ring of $n$ switches?**

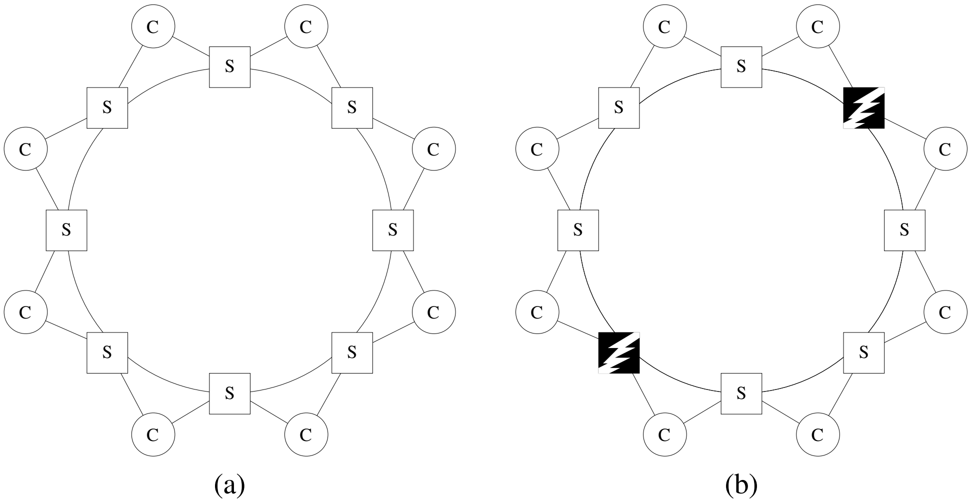


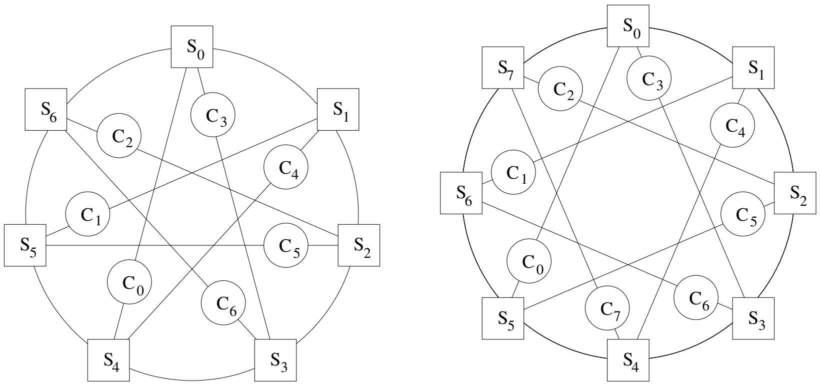

**Figure 7. (a) A naïve approach, $d_c = 2$. (b) Notice that it is easily partitioned with two switch failures.**

### 3.2 Diameter Construction $d_c = 2$

The intuitive, driving idea behind this construction is to connect the compute nodes to the switching network in as *non-local* a way as possible. That is, connect a compute node to switches that are maximally distant from each other. This idea can be applied to arbitrary compute node degree $d_c$, where each connection for a node is as far apart as possible from it's neighbors as possible.

We call this the *diameter solution* because the maximally distant switches in a ring are on opposite sides of the ring, so a compute node of degree 2 connected between them forms a diameter. We actually use the switches that are one less than the diameter apart to permit $n$ compute nodes to be connected to $n$ switches with each compute node connected to a unique pair of switches.

**Construction 1 (Diameters)** Let $d_s = 4$ and $d_c = 2$. $\forall i$, $0 \leq i < n$, label all compute nodes $c_i$ and switches $s_i$. Connect switch $s_i$ to $s_{(i+1) \bmod n}$, i.e., in a ring. Connect node $c_i$ to switches $s_i$ and $s_{(i+\lfloor n/2 \rfloor+1) \bmod n}$. See Figure 8 for an example for $n$ odd and $n$ even.




(a) (b)

**Figure 8. (a) Diameter construction for $n$ odd. (b) Diameter construction for $n$ even.**

**Note:** Although Construction 1 is given for an identical number of compute nodes and switches, we can add additional compute nodes by repeating the above process. In this case, we would connect node $c_j$ to the same switches as node $c_{j \bmod n}$. All the following results still hold, with a simple change in constants. For example, when we connect 10 nodes to 10 switches we have a maximum loss of 6 nodes at 3 faults. Tripling the number of nodes to $3n = 30$ triples the maximum nodes lost at 3 faults to 18. This is also true of the generalized diameters construction give in Section 4.1. The maximum number of lost nodes is still constant with respect to $n$, the number of switches. The addition of extra nodes to the ring constructions affects only this constant in our claims. The asymptotic results about resistance to partitioning are all still valid.

**Theorem 1** Construction 1 creates a graph of $n$ compute nodes of degree $d_c = 2$ connected to a ring of $n$ switches of degree $d_s = 4$ that can tolerate 3 faults of any kind (switch, link, or node) without partitioning the network. I.e., only a constant number of nodes (with respect to $n$) will be lost. In this case that constant is $\min(n, 6)$ lost nodes. This construction is optimal in the sense that no construction connecting $n$ compute nodes of degree $d_c = 2$ to a ring of switches of degree $d_s = 4$ can tolerate an arbitrary 4 faults without partitioning the nodes into sets of non-constant size (with respect to $n$).

The proof of this theorem is broken into two parts, given in the next two sections.

### 3.3 Proof of Correctness for $d_c = 2$ (Upper Bound)

Here we look at the *correctness* of the construction. We show it can tolerate 3 faults of any kind without partitioning the compute nodes.

**Proof:** We will look at switch failures since they are the worst-case situation. A node failure only results in the loss of a single node, and has no effect on the rest of the system. A link fault can always be simulated by the worse situation of a switch fault. So, we introduce 3 switch failures into the ring of $n$ switches. Two cases arise:

1. A segment of switches exists with $\lfloor n/2 \rfloor$ or more connected switches (three segments, one being at least $\lfloor n/2 \rfloor - 1$ hops in length).
2. All segments of switches have $\lfloor n/2 \rfloor - 1$ or less connected switches (three segments, each being at most $\lfloor n/2 \rfloor - 2$ hops in length).

**Case 1**. We have a "good" segment of $\lfloor n/2 \rfloor - 1$ or more hops in length. We can treat the remaining switches as a bad segment. By construction, all nodes span a distance of at most $\lfloor n/2 \rfloor + 1$ hops around the ring, measuring linearly around the ring in each direction. Thus, at most one node can have both connections in the bad segment, the one with a connection at each endpoint of the bad segment. All other nodes will have at least one endpoint in the good segment. Thus, at most one node can be lost.

**Case 2**. This corresponds to having faults interspersed more equally around the ring forming three segments of connected switches. First, we mark all nodes that have a connection to a faulty switch as removed. Thus, we mark at most 6 compute nodes as lost. All remaining nodes have two connections to functioning switches. No node can have both those connections in the same switch segment since all switch segments are of at most $\lfloor n/2 \rfloor - 2$ hops in length and by construction each node spans a distance of at least $\lceil n/2 \rceil - 1$ hops around the ring. All remaining nodes have a connection in two of the three switch segments. Thus, all but the 6 initially removed share at least one switch segment in common and are connected. □

The above really corresponds to the worst case situation. We'll make some further claims later corresponding to less severe fault situations. These claims will not be proven.

### 3.4 Proof of Optimality for $d_c = 2$ (Lower Bound)

Now we look at the *optimality* of the degree 2 diameters solution proposed above. We claim that our construction is optimal in the sense that no construction can do better: no construction can tolerate 4 switch faults with only a constant number of nodes lost. Namely, *any* construction can be partitioned with the introduction of 4 switch faults.

Our proof method is simple: we specify a systematic way to introduce faults that can be applied to an arbitrary construction. We show that 4 faults introduced in this way will partition any construction into sets of a size proportional to $n$.

**Proof:** We introduce 4 faults into the links between switches, breaking the ring into segments. For the purpose of clarity, we'll assume $n$ even. The case of $n$ odd is very similar. We introduce the faults as follows: Pick some switch-to-switch link and mark it as faulty. Proceed around the ring counting node-to-switch links. Introduce a switch-to-switch link fault every time you count $n/2$ node-to-switch links.

This should give a system with faults introduced evenly around the ring, so that each of 4 segments of switches has $n/2$ or less node-to-switch links going to it.

We can label each segment of switches from 1 to 4. We can classify the compute nodes by which switch segment(s) they're connected to. We'll use the notation $c_{ij}$ to indicate the number of nodes of the given type. For example, $c_{11}$ would be the number of nodes which have both connections in segment 1, and $c_{12}$ would be the number of nodes which have one connection in segment 1 and one connection in segment 2. All compute nodes will have two good connections since we've only introduced faults in the switch-to-switch links. See Figure 9 to see all classes of compute nodes, and see Table 1 for a table showing all the classes of compute nodes and how many connections each has per segment.

| Switch Segment | Node Type | | | | | | | | | | # Connections per Segment |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | $c_{12}$ | $c_{13}$ | $c_{14}$ | $c_{23}$ | $c_{24}$ | $c_{34}$ | $c_{11}$ | $c_{22}$ | $c_{33}$ | $c_{44}$ | |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | $n/2$ |
| 2 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 2 | 0 | 0 | $n/2$ |
| 3 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 2 | 0 | $n/2$ |
| 4 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 2 | $n/2$ |

**Table 1. Connections of node type per switch segment, and total connections per switch segment.**

For any connected set of nodes, we can show that the number of nodes in that set is less than or equal to $3n/4$.

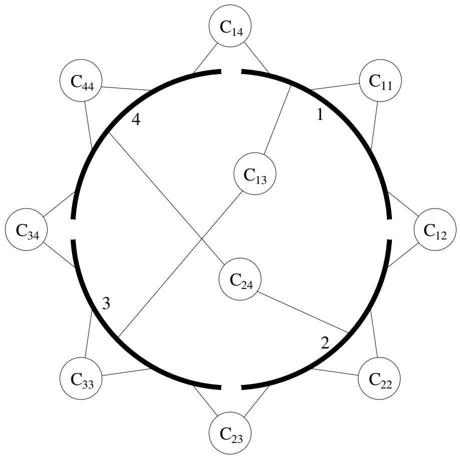



**Figure 9. Node types after the introduction of 4 faults in the ring. All nodes are classified by the switch segments they are connected to.**

This establishes that the compute nodes have been partitioned and the largest connected set will be separated from at least $1/4$ of the nodes. Without loss of generality, any set of connected nodes is symmetric to one of these two cases:

1. We include one type of node with both connections in the same segment. We'll look specifically at the set $c_{11}$, $c_{12}$, $c_{13}$, and $c_{14}$. See Figure 10a.

2. We don't include a type of node with both connections in the same segment. We'll look specifically at the set $c_{12}$, $c_{13}$, and $c_{24}$. See Figure 10b.

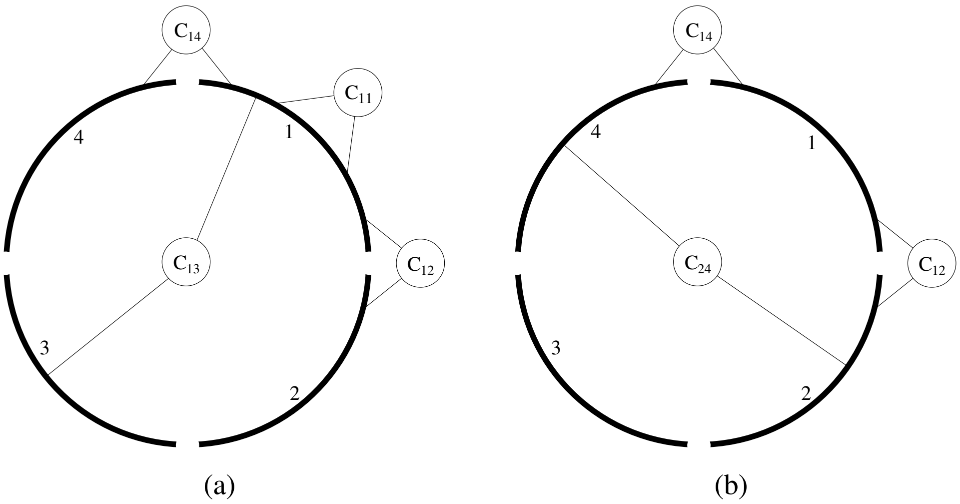



**Figure 10. (a) The connected set given that $c_{11}$ is in the set. (b) The connected set given that $c_{11}$ is *not* included.**

**Case 1**. See Figure 10a. Inclusion of a node type with both connections in the same segment forces us to consider only those node types with a connection in that segment. Specifically, we look at the set $c_{11}$, $c_{12}$, $c_{13}$, and

$c_{14}$. We can count up the number of node-to-switch links connected to this segment

$$2c_{11} + c_{12} + c_{13} + c_{14} = n/2$$

which gives us the following bound

$$c_{11} + c_{12} + c_{13} + c_{14} \leq n/2$$

**Case 2**. See Figure 10b. Here we consider a set of node types where all types have a connection in two different segments. Specifically, we look at the set $c_{14}$, $c_{12}$, and $c_{24}$. We can count up the number of node-to-switch links connected to the involved segments

$$\begin{aligned} c_{12} + c_{14} &\leq n/2 \\ c_{12} + c_{24} &\leq n/2 \\ c_{14} + c_{24} &\leq n/2 \end{aligned}$$

which gives us the following bound

$$c_{12} + c_{14} + c_{24} \leq 3n/4$$

In both cases we see we can bound the size of any set of connected nodes by a fraction of $n$. Thus, some fraction of the total nodes is forcibly lost when we choose a set to use as our connected set. □

The diameters construction also exhibits some nice properties when we don't suffer from the worst case of 3 lost switches. We make the following claim without proof:

**Claim 1** Construction 1 creates a graph of compute nodes and switches that can tolerate

1. 1 switch failure with no lost nodes
2. 2 switch failures with at most 1 lost node
3. 3 switch-to-switch link failures with no lost nodes
4. 3 node-to-switch link failures with at most 1 lost node
5. 3 link failures (any kind) with at most 1 lost node

## 4 Generalized Diameter Construction and Layout

### 4.1 Generalized Diameter Construction $d_c > 2$

The diameter construction can be extended to arbitrary compute node degree. Before giving the details of the construction, let's look at an example for $d_c = 3$ (which fixes $d_s = 5$) and $n = 8$. The constructions are really very simple. For this example, each of the $n$ compute nodes has connections spaced 3, 3, and 2 apart, i.e., as evenly spaced apart as possible. Figure 11 shows the connections pictorially. The compute nodes are connected as follows:

$$\begin{array}{lll} c_0 : \{s_0, s_3, s_6\} & & c_4 : \{s_4, s_7, s_2\} \\ c_1 : \{s_1, s_4, s_7\} & & c_5 : \{s_5, s_0, s_3\} \\ c_2 : \{s_2, s_5, s_0\} & & c_6 : \{s_6, s_1, s_4\} \\ c_3 : \{s_3, s_6, s_1\} & & c_7 : \{s_7, s_2, s_5\} \end{array}$$

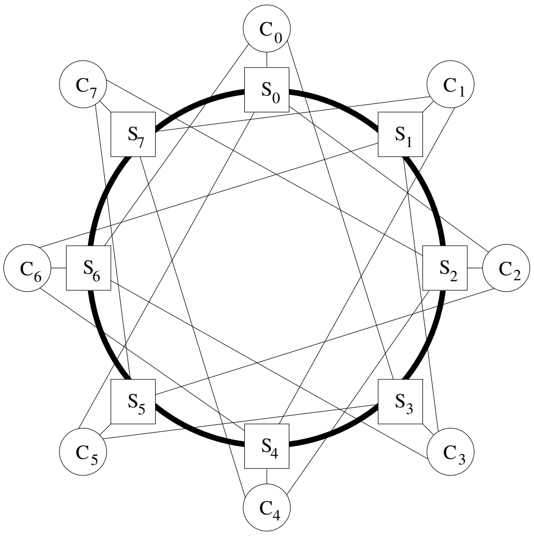



**Figure 11. Generalized diameters construction for $d_c = 3$ and $n = 8$.**

**Construction 2 (Generalized Diameters, $n$ *not* a multiple of $d_c$)** Let $d_s = d_c + 2$. Label all compute nodes $c_i$ and switches $s_i$, $\{i : 0 \leq i < n\}$. Define $q$ and $r$ such that $n/d_c = q\ remainder\ r$. Connect node $c_i$ to switches $s_\lambda$, $\{\lambda : 0 \leq j \leq r,\ \lambda = (i + j(q+1)) \bmod n\} \bigcup \{\lambda : r < j < d_c,\ \lambda = (i + r + jq)) \bmod n\}$.

**Construction 3 (Generalized Diameters, $n$ a multiple of $d_c$)** Let $d_s = d_c + 2$. Label all compute nodes $c_i$ and switches $s_i$, $\{i : 0 \leq i < n\}$. Define $q = n/d_c$. Connect node $c_i$ to switches $s_\lambda$, $\{\lambda : 0 \leq j \leq n-3,\ \lambda = (i + jq) \bmod n\} \bigcup \{\lambda = (i + (n-3)q + q + 1) \bmod n,\ \lambda = (i + (n-3)q + (q+1) + q - 1) \bmod n\}$.

**Theorem 2** Constructions 2 and 3 create a graph of compute nodes and switches that can tolerate $k = 2d_c - 1$ faults of any kind (switch, link, or node) without partitioning the network. I.e., only a constant number of nodes (with respect to the number of nodes in the configuration) will be lost. In this case that constant is $\min(n, kd_c)$ lost nodes.

**Proof:** As before, we will look at switch failures since they are the worst-case situation. We introduce $k = 2d_c - 1$ switch failures into the ring of $n$ switches. Two cases arise:

1. A segment of switches exists with $\lfloor n/d_c \rfloor$ or more connected switches ($k$ segments, one being at least $\lfloor n/d_c \rfloor - 1$ hops in length)
2. All segments of switches have $\lfloor n/d_c \rfloor - 1$ or less connected switches ($k$ segments, each being at most $\lfloor n/d_c \rfloor - 2$ hops in length)

**Case 1**. We have a "good" segment of $\lfloor n/d_c \rfloor - 1$ or more hops in length. We can treat the remaining switches as a bad segment. By construction, a node spans a distance of at most $\lfloor n/d_c \rfloor + 1$ hops around the ring. By this we mean that from one switch to which a node is connected to the next to which it is connected, there are at most $\lfloor n/d_c \rfloor + 1$ hops. Thus, only those nodes whose connections completely span the good segment will be lost: those with one connection at each end of the bad segment. Since there are $d_c$ ports on each switch for node connections, $d_c$ is a loose upper bound on the number of nodes lost. All other nodes will have at least one endpoint in the good segment. Thus, at most one node can be lost if we have a connected segment of $\lfloor n/d_c \rfloor$ or more switches.

**Case 2**. This corresponds to having faults interspersed more equally around the ring forming $k$ segments of connected switches. First, we mark all nodes that have a connection to a faulty switch as removed. Thus, we mark at most $kd_c$ compute nodes as lost. All remaining nodes have $d_c$ connections (all those with fewer connections have already been marked as lost). No node can have both those connections in the same switch segment since all switch segments are of at most $\lfloor n/d_c \rfloor - 2$ hops in length and by construction each node spans a distance of at least $\lceil n/d_c \rceil - 1$ hops around the ring. All remaining nodes have a connection in $d_c$ of the $2d_c - 1$ switch segments. Thus, all but the $kd_c$ initially removed share at least one switch segment in common and are connected. □

Again, the above really corresponds to the worst case situation. We make the following claim corresponding to less severe failure situations without proof:

**Claim 2** Constructions 2 and 3 create a graph of compute nodes and switches that can tolerate

- $k = 2d_c - 1$ switch-to-switch link failures with no lost nodes
- $k = 2d_c - 1$ node-to-switch link failures with at most 1 node
- $k = 2d_c - 1$ link failures (any kind) with at most 1 lost node

### 4.2 Layout Issues

The layout of the standard diameter construction may initially raise some concerns, in particular the long link lengths required to connect compute nodes across the diameter of the ring. Some of these problems can be overcome by modifying the layout of the graph to eliminate long lines. The layout goal will differ depending on the physical constraints of the system. Figure 12 shows an example where the long node-to-switch links are eliminated. This might be useful, for example, in a situation where the switches are centrally located in a building and compute nodes are in more remote locations. See [3] for some related layout work for fault-tolerant meshes.

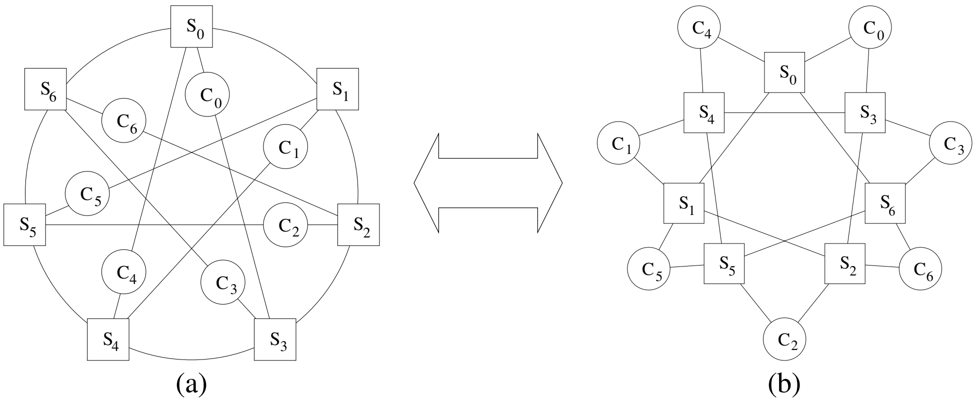



**Figure 12. (a) Normal construction layout. (b) A new layout for the same graph with short node-to-switch links.**

## 5 A Clique of Switches

A clique is a fully connected graph, i.e., there is a link between any two switches. Figure 13 shows a few examples.

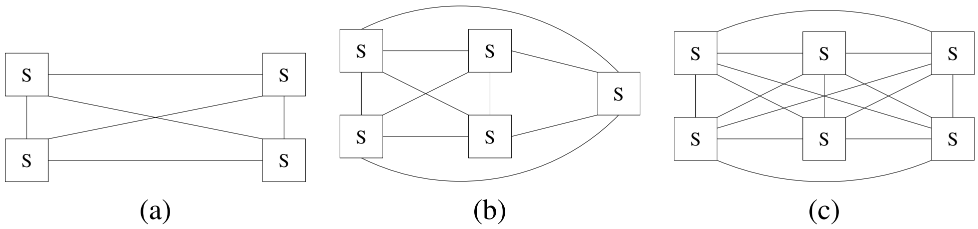



**Figure 13. Examples of switch cliques with 4, 5 and 6 nodes.**

**Connecting compute nodes to a clique of switches.** Notice that the maximum distance between two switches in the clique is 1 and the minimum distance 0. In terms of connecting compute nodes to the clique, we only need to make sure that a node is connected to different switches, in order to satisfy the idea of non-locality mentioned above. Figure 14 shows two constructions.

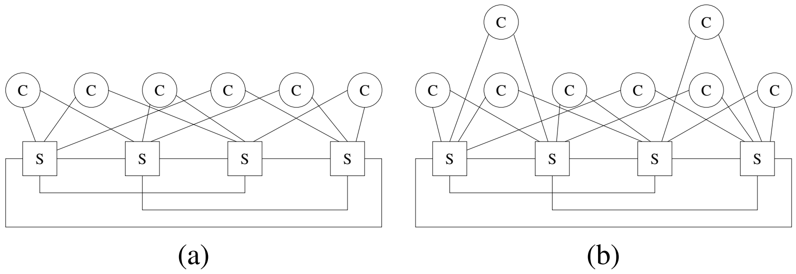



**Figure 14. Examples of homogeneous clique-based systems with (a) 4 switches and 6 nodes, (b) 4 switches and 8 nodes.**

**Fault-tolerance of a clique-based system.** Given a homogeneous clique-based system, as defined above, with $s$ switches and degrees, $d_s$ and $d_c$, one can compute a tight upper bound on the number of lost nodes as a function of the number of lost switches, $l$.

$$f(l) = \left\lfloor \frac{l(d_s - s + 1)}{d_c} \right\rfloor$$

Where $f(l)$ is the maximum number of nodes lost provided that $l$ switches have failed. To derive the above equation notice that $d_s - s + 1$ is the number of switch-to-node links coming out of a switch and use a counting argument. Figure 15 shows two examples of switch failures and the resulting node loss. One of them achieves the upper bound.

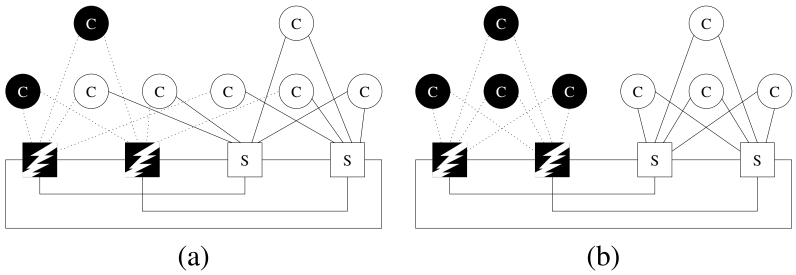



**Figure 15. Examples of faults and the related node loss. In (b) the upper bound is achieved.**

**Improved fault-tolerance.** To avoid the situation depicted in Figure 15b, one must connect the nodes in a *uniform* way. Given a system with $s$ switches of degree $d_s$ and nodes of degree $d_c$, there are $\binom{s}{d_c}$ ways of connecting a node to the system. The idea is to use all $\binom{s}{d_c}$ combinations evenly. Formally, given a subset of switches $A$ with $|A| = d_c$, define $N(A)$ as the number of nodes connected exactly to the switches in $A$. A *uniform system* is such that for any two subsets of switches $A$ and $B$, with $|A| = |B| = d_c$, $|N(A) - N(B)| \leq 1$. The systems of Figure 14 are uniform. They are described by the connectivity matrices shown in Figure 16

$$\underset{(a)}{\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}} \quad \underset{(b)}{\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \end{pmatrix}}$$

**Figure 16. Connectivity matrix (row=switch, column=node) for the graph in (a) Figure 14a and (b) Figure 14b.**

Each row corresponds to a switch, each column to a node. A 1 at location $(i, j)$ indicates a connection between switch $i$ and node $j$. To connect a new node to the system corresponds to adding a column to the connectivity matrix. Each column contains exactly $d_c$ 1's. The idea is to add the column that appears the fewest times in the matrix. Figure 17 shows an example.

$$\underset{(a)}{\begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}} \Longrightarrow \underset{(b)}{\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & \mathbf{0} \\ 1 & 0 & 0 & 1 & 1 & \mathbf{0} \\ 0 & 1 & 0 & 1 & 0 & \mathbf{1} \\ 0 & 0 & 1 & 0 & 1 & \mathbf{1} \end{pmatrix}}$$

**Figure 17. Connectivity matrix (row=switch, column=node) showing (a) initial configuration (b) adding a new node (last column) in a uniform manner.**

What is the fault-tolerance of a uniform system?

**Theorem 3** Given the loss of $l$ switches, the number of nodes lost is at most:

$$f(l) = \begin{cases} 0 & \text{if } l < d_c \\ \binom{l}{d_c} \left\lceil \frac{c}{\binom{s}{d_c}} \right\rceil & \text{otherwise} \end{cases}$$

where $s$ is the total number of switches, $c$ the total number of nodes and $d_c$ the node degree.

**Proof:** Notice that because of the uniform construction the maximum number of nodes of a particular type is

$$\left\lceil \frac{c}{\binom{s}{d_c}} \right\rceil$$

Given $l$ switch faults, $\binom{l}{d_c}$ types of nodes are lost, so the maximum number of nodes lost is

$$\binom{l}{d_c}\left\lceil \frac{c}{\binom{s}{d_c}} \right\rceil$$

□

**Cost of a clique-based system.** A clique of switches offers a high level of fault tolerance at the expense of a growth in $d_s$, the number of connections per switch. In other words, a clique composed of $s$ switches requires the latter to have at least $s$ network ports ($s-1$ for switch-to-switch connections and at least 1 to connect to a computing node). Given a system with $c$ compute nodes of degree $d_c$, the required switch degree satisfies:

$$d_s \geq s - 1 + \left\lceil c\frac{d_c}{s} \right\rceil$$

Table 2 shows some numerical values of $d_s$ for systems with node degree $d_c = 2$.

| system | $d_s$ |
|---|---|
| $s = 4,\ c = 6$ | 6 |
| $s = 11,\ c = 55$ | 20 |
| $s = 21,\ c = 210$ | 40 |
| $s = u,\ c = \binom{u}{2}$ | $2(u-1)$ |

**Table 2. Number of ports per switch for different clique-based systems. The node degree $d_c = 2$, $s$ is the number of switches and $c$ the number of compute nodes.**

## 6 Extensions and Conclusion

We introduced the problem of connecting computing nodes to switching networks, and looked at two extreme cases of switch networks (in terms of the amount of redundancy): a ring and a clique. We used the ideas of non-locality and uniformity to design fault-tolerant systems. These can be applied to different kinds of graphs. In particular, for regular graphs such as the torus the results of Section 3 hold. A fault is no longer the failure of a single switch but the failure of a ring of switches, producing a cut in the torus, see Figure 18. For nodes of degree two, up to three such cuts can be tolerated using the diameters construction over the torus. In general up to $2d_c - 1$ cuts can be tolerated. For non-regular graphs the idea of non-locality can be applied using the distance measure defined in Section 2.

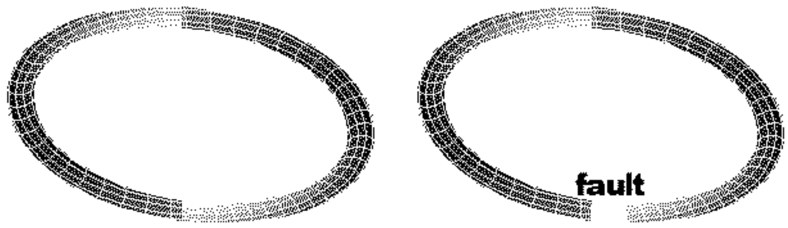



**Figure 18. (a) Torus. (b) A fault is the failure of a ring of switches.**

It is instructive to look at the performance of the ring vs. the clique at this point to summarize the merits of the two solutions. The performance of the two networks depends heavily on the cost model. Let's consider a cost model where the total cost of the network is the number ports in the switch network. For example, a ring of 45 switches of degree 4 would have a cost of 180 ($45 \times 4$). A clique of 10 switches of degree 18 would also have a cost of 180 ($10 \times 18$). For $d_c = 2$, 45 degree-4 switches in a ring can support 45 compute nodes, and 10 degree-18 switches in a clique can also support 45 compute nodes. These two systems are comparable since they create networks with the same number of compute nodes connected at the same cost. Table 3 shows how they behave as we introduce faults.

| # faults | Ring: lost nodes | Clique: lost nodes |
|---|---|---|
| 1 | 0 | 0 |
| 2 | 1 | 1 |
| 3 | 6 | 3 |
| 4 | $\mathcal{O}(n/2)$ | 6 |

**Table 3. Compute node loss: Ring vs. Clique**

In general, the clique performs better than the ring. Why did we spend so much time on the ring solution if a simple clique does better?

The problem with the clique is really scalability, which is not well represented in this simple cost model. A simple sum-of-ports cost doesn't take into account that switch cost probably doesn't scale linearly in the number of ports. In fact, large switches (say, greater than 16 ports) may be very expensive or non-existent. The strength of the ring comes from its ability to perform and scale well using small switches (i.e., only 4 ports). For a small configuration or the availability of large switches, the clique is the best solution. For larger configurations or restrictions on switch size, the ring shows it's merit.

## References


[1] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W. K. Su, "Myrinet: A gigabit per second local area network," *IEEE-Micro*, vol. 15, pp. 29–36, February 1995.

[2] J. P. Hayes, "A graph model for fault-tolerant computing systems," *IEEE Transactions on Computers*, vol. 25, no. 9, pp. 875–884, 1976.

[3] J. Bruck, R. Cypher, and C. T. Ho, "Fault-tolerant meshes and hpercubes with minimal numbers of spares," *IEEE Transactions on Computers*, vol. 42, pp. 1089–1104, September 1993.

[4] F. T. Boesch and A. P. Felzer, "A general class of invulnerable graphs," *Networks*, vol. 2, pp. 261–283, 1972.

[5] F. T. Boesch and R. Tindell, "Circulants and their conenctivities," *Journal of Graph Theory*, vol. 8, pp. 487–499, 1984.

[6] F. T. Boesch and J. F. Wang, "Reliable circulant networks with minimum transmission delay," *IEEE Transactions on Circuits and Systems*, vol. CAS-32, no. 12, pp. 1286–1291, 1985.

[7] S. Dutt and J. P. Hayes, "On designing and reconfiguring k-fault-tolerant tree architectures," *IEEE Transactions on Computers*, vol. 39, no. 4, pp. 490–503, 1990.

[8] S. Dutt and J. P. Hayes, "Designing fault-tolerant systems using auto-morphisms," *Journal of Parallel and Distributed Computing*, vol. 12, no. 3, pp. 249–268, 1991.

[9] S. Dutt and J. P. Hayes, "Some practical issues in the design of fault-tolerant multiprocessors," *IEEE Transactions on Computers*, vol. 41, pp. 588–598, May 1992.

[10] H. K. Ku and J. P. Hayes, "Connective fault tolerance in multiple-bus systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 8, pp. 574–586, June 1997.

[11] G. W. Zimmerman and A. H. Esfahanian, "Chordal rings as fault-tolerant loops," *Discrete Applied Mathematics*, vol. 37, pp. 563–573, July 1992.