Designing a Fault-Tolerant Network Using Valiant Load-Balancing

Rui Zhang-Shen
Department of Electrical Engineering
Princeton University
Email: rz@princeton.edu

Nick McKeown
Computer Systems Laoratory
Stanford University
Email: nickm@stanford.edu

***Abstract*—Commercial backbone networks must continue to operate even when links and routers fail. Routing schemes such as OSPF, IS-IS, and MPLS reroute traffic, but they cannot guarantee that the resulting network will be congestion-free. As a result, backbone networks are grossly over-provisioned—sometimes running at a utilization below 10% so they can remain uncongested under failure. Yet even with such large over-provisioning, they still cannot guarantee to be uncongested, sometimes even with just a single failure.**

With our proposed approach, a network can be designed to tolerate an almost arbitrary number of failures, and guarantee no congestion, usually with an extremely small amount of over-provisioning. In a typical case, a 50 node network can continue to run congestion-free when any 5 links or routers fail, with only 10% over-provisioning. The key to the approach is Valiant Load-Balancing (VLB). VLB's path diversity allows it to tolerate k *arbitrary* failures in an N node network, with over-provisioning ratio of approximately $\frac{k}{N}$.

I. Introduction

A. Background

In traditional intradomain routing protocols such as OSPF [9] and IS-IS [10], which route traffic along the shortest paths, failures cause at least two problems: First, new link state information needs to propagate throughout the network and new paths need to be found. The transient state before routing stabilizes can last up to 30 seconds [3], [4], unacceptable for real time applications such as voice over IP. Second, some links need to carry rerouted traffic and can become congested. Studies suggest that as much as 80% of congestion in the backbone is caused by failures [5].

Multi-Protocol Label Switching (MPLS) helps solve the first problem by quickly rerouting traffic from the primary path to a protection path [11], [14], [8]. But it still cannot guarantee to prevent congestion. So, to reduce the chance of congestion, backbone networks are hugely over-provisioned. While actual provisioning numbers are kept confidential, an August 2001 Merrill Lynch report states that at peak usage, US carriers were using only 6.4% of their total lit fiber capacity.[1]

In this paper we show that we do not need to run networks so inefficiently in order to make them congestion-free when links and routers fail. The key is to use Valiant load-balancing (VLB). In recent years several novel network design and traffic engineering approaches based on load-balancing [2], [6], [12], [15] have shown that efficient fault tolerance can be achieved. In [12] and [2] the authors consider link failures in multi-commodity flow networks; and in [6] the authors show how VLB can tolerate single node failures. In this paper we show that with VLB, a very small amount of over-provisioning is enough to tolerate a large number of link and router failures.

[1]Due to decreased capital expenditure in recent years, the utilization is expected to be higher than 6.4% but is unlikely to be significantly different.

As we will see, VLB has the following desirable properties when links and routers fail:

- In order to protect against k failures the fraction of extra capacity required is only $\frac{k}{N}$. This is extremely efficient compared to other fault tolerance schemes.
- All of the working paths between a pair of nodes are used all the time, and flows are load-balanced across all working paths. Most other schemes require protection paths that are idle during normal operation.
- VLB naturally protects against multiple failures. One can decide during design what failure scenarios the network should tolerate, such as k arbitrary link or node failures or a particular set of failure patterns.
- All paths operate all the time, so rerouting is instantaneous.

In what follows we first review Valiant Load-Balancing, and how it can support any traffic matrix. We then prove several results about its fault-tolerance, considering link and node failures separately, and then combining them. We consider both worst-case failures and random failures. Finally we discuss how to protect against a particular set of failures—perhaps a set of links and routers used to connect an important data-center or customer.

B. Basic Valiant Load-Balancing

Consider a hierarchical network with N backbone nodes, each connecting an access network to the backbone. Current backbone networks have about $N = 50$ nodes. We assume we know (roughly) the total capacity of each access network. We represent the traffic demand between the backbone nodes by a $N \times N$ traffic matrix, where $\lambda(i, j)$ is the average rate of traffic from node i destined to node j. We say the network can *support* a traffic matrix if the capacity between i and j (either directly or indirectly) is greater than $\lambda(i, j)$.

In the homogeneous case where all the backbone nodes have the same capacity, r, a VLB network consists of a full mesh

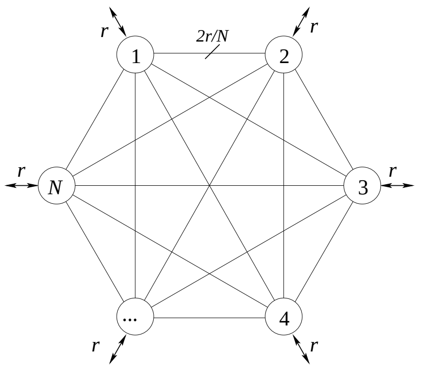

Fig. 1. Valiant Load-Balancing in a network of N identical nodes each having capacity r.

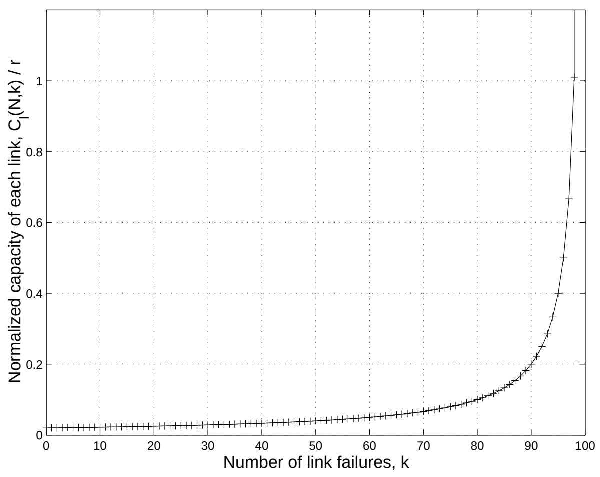

Fig. 2. The required link capacity vs. the number of link failures in a 50-node network as given by Equation (3).

of logical links with capacity $\frac{2r}{N}$, as shown in Figure 1. Traffic entering the backbone is load-balanced equally across all N one- and two-hop paths between ingress and egress. A packet is forwarded twice in the network: In the first hop, a node uniformly load-balances each of its incoming flows to all the N nodes, regardless of the packet destination. Load-balancing can be done packet-by-packet, or flow-by-flow, and each node receives $\frac{1}{N}$ of every flow in the first hop. In the second hop, all packets are delivered to the final destinations.

VLB has the nice characteristic that it can support all traffic matrices that do not oversubscribe a node. Since the incoming traffic rate to each node is at most r, and the traffic is evenly load-balanced to N nodes, the actual traffic on each link due to the first hop routing is at most $\frac{r}{N}$. The second hop is the dual of the first. Since each node can receive traffic at a maximum rate of r and receives $\frac{1}{N}$ of the traffic from every node, the traffic on each link due to the second hop is also at most $\frac{r}{N}$. Therefore, the full-mesh network (with link capacities $\frac{2r}{N}$) can support all traffic matrices. The advantage of VLB for the backbone operator is that they can design their network knowing only the capacities of the access nodes, without knowing anything about the traffic patterns or how they evolve over time. The cost is that the total network has twice the capacity needed, if we knew the actual traffic matrix. It is clear today that backbone operators have little idea what traffic matrices to expect, which explains (in part) why they use five or ten times the minimum capacity.

C. Fault Tolerance in VLB

In Valiant Load-Balancing a flow is load-balanced over all the N two-hop paths. The same basic mechanism works when there are failures and some flows have fewer than N working paths. The source node load-balances a flow evenly over the remaining *available* paths, which is easy to implement: A node only needs to keep track of the available paths for each flow originating from it, and no complex computation is needed. The network is self-healing because the paths are known prior to failures and traffic can be rerouted as soon as failure is detected.

In this paper we determine how much link capacity is needed so as to support all traffic matrices under k_n arbitrary node failures[2] and k_l arbitrary link failures. Due to the symmetry of the topology, all links in the fault-tolerant network have the same capacity requirement, $C(N, k_n, k_l)$. We first consider node failures and link failures separately and then combine them. We use $C_n(N, k)$ and $C_l(N, k)$ to represent the capacity required to tolerate k arbitrary node failures and k arbitrary link failures, respectively. By definition, $C(N, 0, 0) = C_n(N, 0) = C_l(N, 0) = \frac{2r}{N}$.

We will show that to support *all possible* traffic matrices without congestion when there are k failures, each link needs a capacity of approximately $\frac{2r}{N-k}$ (compared to $\frac{2r}{N}$ when there are no failures). This is good news because the curve of $\frac{2r}{N-k}$ is very flat for small values of k (see Figure 2). So a small amount of over-provisioning goes a long way to make the network more fault tolerant. For example, if the links in a 50-node network are over-provisioned by just about 11%, the network can tolerate any five (node or link) failures. Compare this with existing backbone networks that are over-provisioned by several-fold, and yet are still unable to make such guarantees.

II. Notation

We use the following method to count the number of available paths between any pair of nodes: Let A be the $N \times N$ *connectivity matrix*: $A_{ij} = 1$ if the link from node i to node j is up and $A_{ij} = 0$ if the link is down. If a node is down, then all the links connecting to it are down. We assume that all links are bi-directional, so we have $A_{ij} = A_{ji}$.[3] We also assume that $A_{ii} = 1$ for all i, i.e., a node is always connected to itself. Let $P = A^2$, then $P_{st} = \sum_{i=1}^{N} A_{si}A_{it}$ is the number of available paths from node s to node t, and we call P the *path matrix*.

[2]If a node fails, we discard the traffic originating from or terminating at this node. Ideally, users should have backup routes, e.g., by multihoming to different backbone nodes or different routers in the same backbone node.

[3]The analysis can also apply to networks with uni-directional links.

With no failures the connectivity matrix A is an $N \times N$ matrix with all entries equal to 1. Thus $P_{st} = N$ for any (s,t) pair and every flow has N paths. If we look at the paths taken by flow (s,t), link (s,t) is used twice, in path s-s-t and path s-t-t, while link (s,i) and link (i,t), $i \neq s,t$, are used only once. In other words, the direct link between a pair of nodes accounts for two paths between the nodes, and these two paths are in fact single-hop.

With failures some entries of A are zero, and the number of paths between any two nodes is found from the path matrix $P = A^2$. For flow (s,t), node s keeps track of the value of P_{st} and which of the N paths are available. It sends $\frac{1}{P_{st}}$ of flow (s,t) over each of the available paths. Thus $\frac{2}{P_{st}}$ of flow (s,t) traverses link (s,t) if the link is up, and $\frac{1}{P_{st}}$ of flow (s,t) traverses link (s,i) and link (i,t), $i \neq s,t$, if the path is up.

III. Node Failures

Node failures are relatively easy to analyze. When a node fails, it takes down all the links connecting to it and stops sending or receiving traffic. The network becomes an $(N-1)$-node full-mesh network, and a link capacity of $\frac{2r}{N-1}$ is sufficient to support all traffic matrices. In general, when there are k node failures, the network becomes an $(N-k)$-node full mesh, so the required link capacity to tolerate k node failures is

$$C_n(N,k) = C(N-k,0,0) = \frac{2r}{N-k}. \tag{1}$$

IV. Link Failures

Link failures are a little more complicated. We consider two cases: the worst case (adversarial link failures), and the typical case (random link failures).

A. Adversarial link failures

Given the traffic matrix Λ and the path matrix P, we can obtain T_{ij}, the amount of traffic traversing link (i,j), in terms of Λ and P. This is likely different from λ_{ij}, the traffic demand from node i to node j.

Before stating the link failure duality theorem, we introduce one more definition. A *symmetric routing scheme* is one which is invariant under the re-numbering of nodes in a uniform network. The scheme of “uniform load-balancing over available paths” described in this section is one example of a symmetric routing scheme.

Theorem 1 *In a uniform full-mesh network with a symmetric routing scheme, the minimum link capacity required to support all traffic matrices under k arbitrary link failures is equal to the maximum load on link $(1,2)$ over all traffic matrices and under any k link failures, i.e.,*

$$C_l(N,k) = \max_{\Lambda,\ k \text{ link failures}} T_{12}.$$

The proof is straightforward and is omitted here. For simplicity of notation, we omit the variables over which T_{12} is maximized and simply write $\max T_{12}$, when it is clear from the context.

We first maximize T_{12} over all failure scenarios. Omitting the details, we arrive at

$$\max_{\Lambda} T_{12} = \max\left\{ \frac{2r}{P_{12}},\ r\left(\max_{i\geq 3} \frac{A_{2i}}{P_{1i}} + \max_{i \geq 3} \frac{A_{i1}}{P_{i2}} \right) \right\}. \tag{2}$$

Then we maximize Equation (2) over all path matrices representing k link failures. We only give the final result here and the details can be found in the extended version [16]. The amount of capacity required on each link to tolerate k arbitrary link failures, for $1 \leq k \leq N-2$, is

$$C_l(N,k) = \begin{cases} \frac{r}{N-2} + \frac{r}{N} & k = 1 \\ \frac{r}{N-k-1} + \frac{r}{N-1} & k = 2 \text{ or } N-2, \\ & \text{or } N \leq 6 \\ \frac{2r}{N-k} & \text{otherwise.} \end{cases} \tag{3}$$

For $k \geq N-1$, the network becomes disconnected in the worst-case failure scenario, therefore we cannot guarantee a congestion-free network.

B. Random link failures

The analysis above is for the worst-case scenario. If, instead, link failures happen randomly, we should be able to tolerate more failures.

If out of the $\frac{N(N-1)}{2}$ links in the network, k random links fail, then the probability that a particular link fails is

$$p = \frac{k}{\frac{N(N-1)}{2}} = \frac{2k}{N(N-1)}. \tag{4}$$

The probability that a two-hop path between nodes i and j fails is the probability that at least one of the two links fail, i.e., $2p - p^2$, and there are $N-2$ such paths between the two nodes. The probability for the one-hop path, i.e., link (i,j) to fail is p, and it is responsible for two paths. So the expected number of failed paths between node i and node j is $(N-2)\cdot(2p-p^2) + 2\cdot p = 2p(N-1) - p^2(N-2)$.

Substituting p from Equation (4), and taking into account the fact that the number of links is an integer, we obtain the expected number of available paths between Node i and Node j:

$$\mathbf{E}[P_{ij}] = N - \left\lceil \frac{4k}{N} - \frac{4k^2(N-2)}{N^2(N-1)^2} \right\rceil. \tag{5}$$

According to Equation (5), in order to tolerate 5 random link failures, only 2% of over-provisioning is needed in a 50-node VLB network. This is a factor of 5 smaller than what is given by Equation (3), which is for the worst-case failure scenarios. The extra resource required in a VLB network for fault tolerance is so small because load-balancing allows sharing of resource to protect failures. Without load-balancing, protecting against five arbitrary failures is unthinkable in today’s networks.

The case when link failures are random is the “best-case” scenario, because the failures do not concentrate on the paths between a particular node pair, as in the adversarial failure case. So we can conclude that an order of N failures can be tolerated in the worst case, and an order of N^2 failures can be tolerated in the best case. In real networks, typical failure scenarios will lie between these two extremes.

V. Combination of Node and Link Failures

Now assume that there are k_n node failures and k_l link failures. This is equivalent to having k_l link failures in a $(N-k_n)$-node network. So

$$C(N, k_n, k_l) = C_l(N-k_n, k_l) \approx \frac{2r}{N-k_n-k_l} = \frac{2r}{N-k}, \tag{6}$$

where $k = k_n + k_l$.

Equation (6) shows that to tolerate k arbitrary failures in the VLB network, each link is required to have a capacity of about $\frac{2r}{N-k}$. Compared to the capacity required without fault tolerance, $\frac{2r}{N}$, the increase is by a factor of $\frac{k}{N-k}$, which is roughly $\frac{k}{N}$ when k is much smaller than N. So in order to tolerate k failures in an N-node network, the extra resource required, relative to the total resource, is $\frac{k}{N}$. Intuitively, in a system with a characteristic number N, the fraction of extra resource required to tolerate k failures is expected to be $O(\frac{k}{N})$, i.e., $c \cdot \frac{k}{N}$ where c is a constant usually greater than 1 [13]. In the case of VLB, the constant is 1. This shows that VLB is as efficient a network architecture as one can achieve.

Using Equation (6), the network operators can design a VLB network so that the load on every link is below an operational target under all traffic matrices and a given number of failures. Suppose we want to build a national backbone of 50 nodes, each having an access capacity of 1Tb/s. So each logical link carries a load no more than 40Gb/s when there are no failures. To achieve a target utilization of 80% under normal operation, each link should have a capacity of 50Gb/s. If, under up to five arbitrary failures, we want the utilization to be blow 90%, then each link should have a capacity of 49.4Gb/s. Therefore, we should make each logical link 50Gb/s to guarantee that the link utilization is below 80% with no failures and below 90% with up to 5 failures.

VI. Designing for Any Failure Scenarios

Sometimes protecting against k arbitrary failures is not the right design goal. Instead, we may be given a set of failure scenarios, either because they occur more frequently, or because they affect network performance the most. For example, we can consider single *physical* link failures, each corresponding to a set of logical links that have failed. In this case, if we set k as the maximum number of component failures among all the scenarios and design for k arbitrary failures, we may unnecessarily over-provision the links. In this section we extend the method for provisioning for k arbitrary link failures to consider any set of failure scenarios.

Let S represent the set of operating scenarios we want to design for, including the failure scenarios as well as normal operation. We can find the required capacity for each link by finding the maximum possible load on this link over all scenarios and all valid traffic matrices. So for link (u, v), we have

$$c_{uv} = \max_{s\in S;\,\Lambda} T_{uv}.$$

Note that we have to consider each link separately because the failure scenarios may not be symmetric.

The load on link (u, v) is due to first- and second-hop traffic, so we have

$$c_{uv} = \max_{s\in S;\,\Lambda} \left(\sum_i \frac{\lambda_{ui}}{P_{ui}} A_{vi} + \sum_i \frac{\lambda_{iv}}{P_{iv}} A_{iu} \right). \tag{7}$$

One way to solve for c_{uv} is to maximize the link load over all traffic matrices for each scenario and then find the maximum load over all scenarios. For a particular scenario, maximizing c_{uv} over all traffic matrices is a linear programming (LP) problem, and an optimal solution is on the corner of the constraint set. Therefore, it is sufficient to consider only the corners, or the permutation-like traffic matrices. However, this can still turn out expensive because we have to solve the LP for each scenario.

We can simplify the calculation by making the following (slight) relaxation:

$$\begin{aligned} c_{uv} &= \max_{s\in S;\,\Lambda} \left(\sum_i \frac{\lambda_{ui}}{P_{ui}} A_{vi} + \sum_i \frac{\lambda_{iv}}{P_{iv}} A_{iu} \right) \\ &\le \max_{s\in S} \left(\max_{\Lambda} \sum_i \frac{\lambda_{ui}}{P_{ui}} A_{vi} + \max_{\Lambda} \sum_i \frac{\lambda_{iv}}{P_{iv}} A_{iu} \right) \\ &= r \max_{s\in S} \left(\max_i \frac{A_{vi}}{P_{ui}} + \max_i \frac{A_{iu}}{P_{iv}} \right). \end{aligned} \tag{8}$$

Now c_{uv} is much easier to solve for because we only need to consider the scenarios and not the traffic matrix any more. The relaxation from (7) to (8) is equivalent to giving each hop dedicated capacity and not allowing the first-hop and the second-hop to share resource.

VII. Discussions

So far we have assumed that all the nodes in the VLB network have the same capacity, but we can easily extend the results to the case of a heterogeneous network, in which the nodes may have different capacities. When all nodes have the same capacity, the ratio of the number of failures to the size of the network, namely $\frac{k}{N}$, characterizes the percentage of network resource that becomes unusable in the worst case. In a heterogeneous network, the amount of network resource taken offline by k failures, in the worst case, is $\frac{\sum_{i=1}^{k} r_i}{R}$, i.e., the capacity of the largest k nodes as a fraction of total capacity (recall that we sorted the nodes according to decreasing capacity). So we can generalize the results of this paper by replacing $\frac{k}{N}$ with $\frac{\sum_{i=1}^{k} r_i}{R}$ or, equivalently, by replacing k with $N \cdot \frac{\sum_{i=1}^{k} r_i}{R}$. Now the results also hold for heterogeneous networks.

Generalizing to the case of heterogeneous networks increases the amount of extra capacity required for fault tolerance because we always have

$$\frac{\sum_{i=1}^{k} r_i}{R} \ge \frac{k}{N}.$$

Let us look at the Abilene network as an example (for a map of the network please refer to Internet2 [1]). It has 11 nodes and the aggregate traffic rates of the nodes are shown in Figure 3

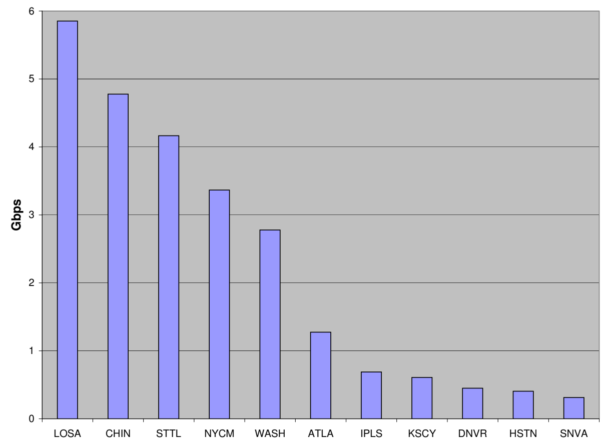

Fig. 3. Node aggregate traffic rates of the Abilene network sorted in decreasing order. The rates are the maximum of the daily average rates during the period of one year (from June 2006 to May 2007).

sorted in decreasing order. We can see that the rate of the largest node in this network is nearly 20 times that of the smallest node. In order to tolerate one arbitrary failure, the proportion of excess capacity required is roughly $\frac{r_1}{R} = 23.7\%$, much larger than $\frac{1}{N} = 9.1\%$.

We have limited ourselves to considering only logical links when considering link failures. This works if the underlying layers are protected and VLB only need to protect failures in the network layer; but if they are not, then a failure in a lower layer (such as the physical layer) can cause multiple simultaneous logical link failures. This means that even if there is only very few failures, the parameter k, the number of logical failures, can be large. However, these correlated failures may not form the worst case scenario, and therefore the design process should consider the possible failure patterns instead of the worst case k failures. We can also try to find the optimal mapping of the logical layer to the underlying layers so that the minimum amount of resource is required for fault tolerance. We leave this to future work.

Protection across layers is hard. Layering inherently means isolation, so the information of one layer may not be readily and accurately available to other layers. Shared Risk Link Group (SRLG) characterizes how failures in a lower layer propagates to an upper layer: The links that share a lower layer component form a SRLG and they would fail together if the lower layer component fails. However, the information of which lower layer components are used by a link is usually maintained manually, and is extremely prone to errors [7]. Automatic rerouting in certain layers also change the SRLGs. When outdated or inaccurate SRLG information is used, the network operator cannot be certain that the network can recover from failures. The ideal fault tolerance scheme should have coordination across layers to ensure reliable and fast recovery as well as efficient resource utilization.

VIII. Summary and Conclusion

One of the biggest advantages of Valiant Load-Balancing is that it can efficiently tolerate and quickly recover from failures. There are N paths between any pair of nodes, so when some paths fail, the source node only needs to send more on the paths that are still available. In order to tolerate k arbitrary failures, the network is required to increase its link capacities by a fraction of approximately $\frac{k}{N}$. (When link failures are random, the extra capacity required to tolerate failures is even smaller.) This is much more efficient when compared to today's networks. We also extend our result to tolerating an arbitrary set of failure scenarios. This general formulation is useful when the frequency or importance of different failure scenarios is known.

IX. Acknowledgment

This research was funded by NSF under ITR award ANI-0331653, the Stanford Clean Slate Program, and the Lillie Family Stanford Graduate Fellowship.

References

[1] The Abilene network, Internet2. http://abilene.internet2.edu/.
[2] D. Applegate, L. Breslau, and E. Cohen. Coping with network failures: routing strategies for optimal demand oblivious restoration. In *Proc. ACM SIGMETRICS*, pages 270–281, 2004.
[3] A. Basu and J. Riecke. Stability Issues in OSPF Routing. In *Proc. ACM SIGCOMM*, pages 225–236, New York, NY, USA, 2001. ACM Press.
[4] G. Iannaccone, C. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot. Analysis of link failures in an IP backbone. In *Proc. of ACM SIGCOMM Internet Measurement Workshop*, November 2002.
[5] S. Iyer, S. Bhattacharyya, N. Taft, and C. Diot. An approach to alleviate link overload as observed on an IP backbone. In *Proc. IEEE INFOCOM*, 2003.
[6] M. Kodialam, T. V. Lakshman, J. B. Orlin, and S. Sengupta. Pre-Configuring IP-over-Optical Networks to Handle Router Failures and Unpredictable Traffic. In *Proc. IEEE INFOCOM*, April 2006.
[7] R. Kompella, J. Yates, A. Greenberg, and A. Snoeren. IP fault localization via risk modeling. In *Proc. USENIX Symposium on Networked Systems Design and Implementation*, 2005.
[8] J. L. Marzo, E. Calle, C. Scoglio, and T. Anjali. QoS Online Routing and MPLS Multilevel Protection: A Survey. *IEEE Communications Magazine*, pages 126–32, October 2003.
[9] J. Moy. RFC 2328: OSPF Version 2. *Internet RFCs*, 1998.
[10] D. Oran. RFC 1142: OSI IS-IS Intra-domain Routing Protocol. *Network*, 1990.
[11] E. Rosen, A. Viswanathan, and R. Callon. RFC3031: Multiprotocol Label Switching Architecture. *Internet RFCs*, 2001.
[12] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg. COPE: traffic engineering in dynamic networks. *Proc. ACM SIGCOMM*, 36(4):99–110, 2006.
[13] S. Wei and G. Lee. Extra group network: a cost-effective fault-tolerant multistage interconnection network. In *Proceedings of the 15th Annual International Symposium on Computer architecture (ISCA)*, pages 108–115, 1988.
[14] X. Xiao, A. Hannan, B. Bailey, and L. M. Ni. Traffic engineering with MPLS in the Internet. *IEEE Network*, 14(2):28–33, April 2000.
[15] R. Zhang-Shen and N. McKeown. Designing a Predictable Internet Backbone Network. In *HotNets III*, November 2004.
[16] R. Zhang-Shen and N. McKeown. Designing a Fault-Tolerant Network Using Valiant Load-Balancing. Technical Report TR07-HPNG-070207, Department of Electrical Engineering, Stanford University, July 2007.