Designing a Predictable Internet Backbone Network*

Rui Zhang-Shen, Nick McKeown
Computer Systems Laboratory
Stanford University
{rzhang, nickm}@stanford.edu

ABSTRACT

Designing a backbone network is hard. On one hand, users expect the network to have very high availability, little or no congestion, and hence little or no queueing delay. On the other hand, traffic conditions are always changing. Over time usage patterns evolve, customers come and go, new applications are deployed, and the traffic matrices of one year are quite different from the next. Yet the network operator must design for low congestion over the multiple years that the network is in operation. Harder still, the network must be designed to work well under a variety of link and router failures. It is not surprising that most networks today are enormously overprovisioned, with typical utilizations around 10%. In this paper we propose that backbone networks use Valiant Load-balancing over a fully-connected logical mesh. This is quite a radical departure from the way backbones are built today, and raises as many questions as it answers. But it leads to a surprisingly simple architecture, with predictable and guaranteed performance, even when traffic matrices change and when links and routers fail. It is provably the lowest capacity network with these characteristics. In addition, it provides fast convergence after failure, making it possible to support real-time applications.

1. INTRODUCTION

Network design can be formulated as an optimization problem where total cost is minimized subject to topology, demand, and performance constraints. The designer chooses where nodes and links are placed, their capacities, and how traffic is routed. On the face of it, this seems like a straightforward problem. We first determine the constraints on node and link location, and the expected demand, and then do our best to design an optimal network. In this paper, we will focus on designing a backbone network.

Once deployed, the expense of the infrastructure dictates that the backbone topology (i.e., the set of nodes and their inter-connectivity) doesn't change for several years. As traffic patterns change, and usage grows, a new topology will eventually need to be deployed, and the design process starts over. A well-designed network should support the traffic matrices presented to it for as long as possible. In the rest of this section, we will discuss some of the many challenges this presents to network designers today.

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

1.1 Obtaining a Traffic Matrix Estimation

The key to designing a network is to understand how it will be used. The process starts by measuring how the network is used today, and then extrapolating based on estimates of how traffic will grow and change over time.

Obtaining a traffic matrix – a matrix indicating the peak short-term average load between any pair of nodes in the backbone – is not an easy task. It is impractical to measure it directly, and the best estimation techniques (from link load measurements) give errors of 20% or more [1]. Moreover, demand fluctuates over time as a result of special events or even changes external to the network [2].

To design a network we need to know what the traffic matrix will be in the future – years after the topology is first deployed. Starting with an estimate of the traffic matrix, and then extrapolating, inevitably leads to a crude estimate of future demand. Typically, the traffic matrix is extrapolated based on historical growth rates, and adjusted according to marketing forecasts and decisions such as addition and elimination of applications and services. However, it is impossible to predict future growth rates and new applications. Even if total growth rate is estimated correctly, the growth rate doesn't take into account new applications which may change traffic patterns. For example, peer-to-peer traffic has demonstrated how quickly usage patterns can change in the Internet. The widespread use of voice-over-IP and video-on-demand may change usage patterns again over the next few years. What's more, the growth rate does not take into account large new customers bringing new demand.

1.2 Failures and Routing Convergence

A network must be designed to continue to operate when there are failures and service or maintenance in-

terruptions in the network.

Failures cause two problems: Loss of connectivity, which can take several seconds to recover from [3], and is too slow for real-time applications such as voice and gaming. Second, failures demand extra capacity to carry rerouted traffic. Today, rerouting leads to large, coarse flows suddenly traversing a link, requiring large chunks of spare capacity.

It is hard enough to design a predictable network when all links and routers are functioning correctly; it is even harder to ensure predictable behavior when there are unpredictable failures.

1.3 Network Design in Practice

While network design can be formulated as an optimization problem, networks are not designed this way in practice. Juggling the number of constraints, estimates, and unknowns means that network design is more of an art than a science, and is dominated by a number of rules-of-thumb that have been learned over the years. This explains why each network operator has built a very different backbone network.

Ad-hoc design makes it hard or impossible to predict how a network will perform over a wide variety of conditions. To offset the intrinsic inaccuracies in the design process, it is common to use traffic engineering in which operators monitor link utilization and route flows to reduce congestion [4], especially during failures [5].

Traffic engineering only works if the underlying network is able to support the current traffic matrix. With the complicated topologies deployed today, it is not generally possible to determine the set of traffic matrices that a backbone network can support.

1.4 Problem Statement

In summary, it is extremely hard to design a network. First, it is nearly impossible to accurately estimate future demand, and how nodes will communicate with each other. Despite inaccurate information, network designers have to ensure the network will operate when links and routers fail unpredictably.

We propose a way to design backbone networks that are insensitive to the traffic matrix (i.e., that work equally well for all valid traffic matrices), and continue to provide guaranteed performance under a user-defined number of link and router failures.

We believe that Valiant Load-balancing is a promising way to design backbone networks. The approach was first proposed by Valiant for processor interconnection networks [6], and has received recent interest for scalable routers with performance guarantees [7] [8]. Here we apply it to backbone network design.

There are several benefits to Valiant Load-balancing. First, it can be shown that – given a set of boundary nodes with a fixed maximum capacity – the network will support any set of traffic matrices. It is provably the most efficient network design (in the sense that it minimizes total link capacity) that can support an arbitrary set of traffic matrices. Furthermore, the network can be designed to support any traffic matrix under a pre-defined number of link and router failures.

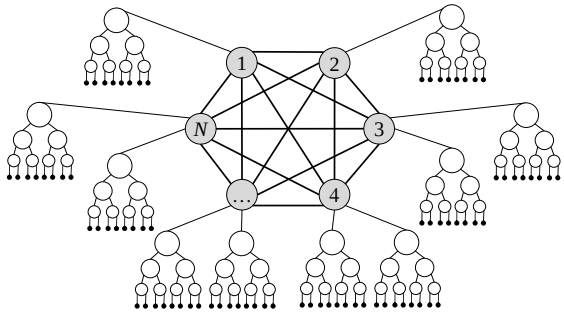

Figure 1: A hierarchical network with N backbone nodes, each serving an access network. The backbone nodes are connected by a logical full mesh.

In what follows, we will describe the general approach, and how Valiant Load-balancing can be applied to backbone network design. We'll first consider how it works when the network is homogeneous and all nodes have the same capacity, and then show how the network can be designed to work under an arbitrary number of failures.

In an accompanying paper [9], the authors independently arrived at the same conclusion and describe a slightly different scheme.

2. VALIANT LOAD-BALANCED NETWORKS

The motivation for Valiant Load-balancing is that it is much easier to estimate the aggregate traffic entering and leaving a node than to estimate the traffic matrix. The approach assumes a full-mesh topology among the nodes, and load-balances traffic over two-hop paths.

2.1 General Approach

Consider a backbone network consisting of multiple PoPs interconnected by long-haul links. The whole network is arranged as a hierarchy, and each PoP connects an access network to the backbone (see Figure 1).

Although traffic matrices are hard to obtain, it is straightforward to measure, or estimate, the total amount of traffic entering (leaving) a PoP from (to) its access network. When a new customer joins the network, we add its aggregate traffic rate to the node. When new locations are planned, the aggregate traffic demand for a new node can be estimated from the population that the node serves. This is much easier than trying to estimate the traffic rates from one node to every other node in the backbone.

Imagine that we establish a full mesh of logical links from each node to every other node. They are not nec-

essarily physical links; they could be implemented using tunneling or an overlay mechanism. Traffic entering the backbone is spread equally across all the nodes. Each packet traverses the backbone twice: once from the ingress node to an arbitrary intermediate node, and once again to reach the egress node. A flow is load-balanced across every two-hop path from its ingress to egress node. This means the capacity requirement on each link is lower than the node capacity, and we'll see that a network designed this way can serve any valid traffic matrix. What's more, as will be shown in Section 3, it is possible to calculate the exact amount of capacity needed on every logical link in the load-balanced network, making network design more systematic.

Since each flow is load-balanced over multiple paths through the network, only a small amount of excess capacity is required in order to restore from a small number of failures. More importantly, we can systematically determine how much extra capacity is needed to recover from a given number of failures.

Forwarding packets twice through the backbone network leads to increased average delay. However, we believe it is a reasonable tradeoff for improved predictability, and lower delay variance. And while many network customers claim they are sensitive to fixed delays, there are few, if any, applications that would really suffer from a two-hop traversal of the network. We believe the fixation on delay is only because it is easier to measure than bandwidth.

The VLB architecture can easily reroute around link and router failures. When a failure happens, the network doesn't need to rely on a complicated routing protocol to discover and converge to new routes. Failure recovery and restoration in a VLB network can be as fast as failure detection at a single router.

2.2 Details of the Approach

Consider a homogeneous network of N identical nodes, each with capacity r, where capacity is the aggregate capacity of the access network that the backbone node serves. This means a node can initiate traffic to, or receive traffic from, other backbone nodes at an aggregate rate up to r. The analysis can be extended to heterogeneous networks but space limitations preclude us from including it here.

The network has a full-mesh of links among the nodes. Let the capacity requirement on the link between node i and node j be c_{ij}. A flow is defined by its source and destination nodes, so there are a total of N^2 flows.

A traffic demand matrix Λ is an $N \times N$ matrix where the $(i,j)^{th}$ entry λ_{ij} represents the traffic rate from node i to node j. A *valid* traffic matrix is one such that no node is over-subscribed, i.e., $\sum_j \lambda_{ij} \leq r$ and $\sum_j \lambda_{ji} \leq r$, $\forall i$. We will only consider valid traffic matrices in this paper and our goal is to guarantee 100% throughput to *all* valid traffic matrices.

What is the optimal interconnection pattern that can guarantee 100% throughput to any valid traffic matrix for this homogeneous network? Keslassy et. al. [10] show that a uniform full-mesh with each link having capacity $\frac{2r}{N}$ is the unique optimal interconnection pattern which requires the lowest link capacity at each node.

Uniform load-balancing leads to guaranteed 100% throughput in this network. Packets are forwarded in two stages. First, each node uniformly load-balances its incoming traffic to all the N nodes, regardless of the packet destination. Load-balancing can be done packet-by-packet, or flow-by-flow. Either way, each node receives $1/N$-th of every node's traffic. In the second stage, all packets are delivered to the final destination. 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 stage routing is at most $\frac{r}{N}$. The second stage is the dual of the first stage. Since each node can receive traffic at a maximum rate of r, and it receives $1/N$-th of the traffic from every node, the actual traffic on each link due to the second stage routing is also at most $\frac{r}{N}$. Therefore, a full-mesh network where each link has capacity $\frac{2r}{N}$ is sufficient to guarantee 100% throughput for any valid traffic matrix.

It may seem counter-intuitive that this is the most efficient network. After all, almost every packet is forwarded twice in the network. But note that this network can serve a traffic matrix where a node sends traffic at the maximum rate r to another node, while the link capacity required between any pair of nodes is only $r \cdot \frac{2}{N}$. If all the traffic were to be sent through direct paths, we would need a full mesh network of link capacity r. So load-balancing is $\frac{N}{2}$ times more efficient when the goal is to serve all valid traffic matrices.

3. FAULT-TOLERANCE IN A VALIANT LOAD-BALANCED NETWORK

It is easy to design a VLB network to perform predictably in the presence of failure. In an N node full-mesh network, node failures will result in a smaller full-mesh network. When there are no node failures, we need at least $N-1$ link failures to make a cut in the network. In this sense it has higher tolerance to faults than any other architecture. In this section, we again assume a homogeneous network of N identical nodes of capacity r, and derive the fault tolerance requirements in the Valiant Load-balanced network.

We will derive the link capacities we need so as to guarantee 100% throughput under k_1 arbitrary node failures[1] and k_2 arbitrary link failures[2], where k_1 and

[1]If a node fails, we discard the traffic originating from or terminating at this node, because there is no way this traffic can be served.

[2]Here we refer to the logical links. If the underlying physical layer is unprotected, then a failure in the physical layer could cause multiple simultaneous failures in the logical network.

k_2 are design parameters. Due to the symmetry of the topology, all links in the fault-tolerant network have the same capacity. Let $C(N, k_1, k_2)$ denote the minimum capacity required on each link in an N-node full mesh network such that 100% throughput is guaranteed under k_1 arbitrary node failures and k_2 arbitrary link failures. We similarly define $C_n(N, k)$ and $C_l(N, k)$ to be the capacity required to tolerate k node failures and k link failures in an N-node network, respectively. Obviously, C, C_n, and C_l are increasing functions of k_1, k_2, and k. And we have, by definition, $C(N, 0, 0) = C_n(N, 0) = C_l(N, 0) = \frac{2r}{N}$.

In what follows, we derive the link capacity needed so as to tolerate node failures and link failures separately. Then, we combine the two conditions and give the capacity requirement for general failure scenarios.

3.1 The Path Matrix

Under normal operation, every flow is uniformly load-balanced over N paths. When there are failures, some flows have fewer than N paths. Assume that each source node evenly spreads a flow over all the *available* paths for the flow, no matter how many of these paths there are. This scheme is easy to implement at the node: A node only needs to keep track of the available paths for each flow originating from it, and no complex computation is needed. Such a network is also self-healing because traffic can be rerouted as soon as failure is detected.

Now we introduce a method of counting the number of available paths between any pair of nodes given a failure pattern. Let A be the *connectivity matrix*, that is, entry $A_{ij} = 1$ 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. Since all links are bi-directional, we have $A_{ij} = A_{ji}$.[3] We also assume that $A_{ii} = 1, \forall i$, 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*.

Under normal operation with no failures, the connectivity matrix A is an $N \times N$ matrix of all ones. Thus $P_{st} = N$ for any (s, t) pair and a flow is load-balanced over N paths. If we look at the paths taken by the flow$_{st}$, link$_{st}$ is used twice, in path s-s-t and path s-t-t, while link$_{si}$ and link$_{it}$, $i \neq s, t$, 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. This is true even when some paths fail.

With failures, some entries of A become zero, and the number of paths between any two nodes can be obtained from the path matrix $P = A^2$. For flow$_{st}$, node s knows the value of P_{st} and which of the N paths are available. So it sends $1/P_{st}$ of flow$_{st}$ over each of the available paths. Thus $2/P_{st}$ of flow$_{st}$ traverses link$_{st}$, if the link is up, and $1/P_{st}$ of flow$_{st}$ traverses link$_{si}$ and link$_{it}$, $i \neq s, t$, if the path is up.

[3]The analysis can also apply to networks with unidirectional links.

3.2 Node Failures

When a node fails, it takes down all the links connecting to it and stops sending or receiving traffic. So the network becomes an $(N-1)$-node uniform full mesh network, and a link capacity of $\frac{2r}{N-1}$ is sufficient to guarantee 100% throughput. In general, when there are k node failures, the network becomes an $(N-k)$-node uniform full mesh network, so the required link capacity to tolerate k node failures is

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

3.3 Link Failures

Analyzing link failures is more complicated, because they break the symmetry of the network. In this subsection, we assume there are no node failures, and only consider link failures.

Given the traffic matrix Λ and the path matrix P, we can obtain T_{ij}, the amount of traffic traversing link$_{ij}$, in terms of Λ and P. Note that this can be very 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 guarantee 100% throughput for any valid traffic matrix under k arbitrary link failures, is equal to the maximum load on link$_{12}$ over all valid traffic matrices and under any k link failures, i.e.,*

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

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

PROOF. Clearly we require $C_l(N, k) \geq \max T_{12}$; otherwise link$_{12}$ would be overloaded and we would not be able to guarantee 100% throughput for some valid traffic matrices under some failure scenarios. Thus, a capacity of $\max T_{12}$ is necessary. Due to the symmetry of the network and routing scheme, the maximum traffic on any link under all valid traffic matrices and any k link failures is at most $\max T_{12}$. Therefore, a capacity of $\max T_{12}$ is sufficient. Thus we have $C_l(N, k) = \max T_{12}$. □

Now we derive $\max T_{12}$. Assume link$_{12}$ does not fail because otherwise there will be no traffic on it. The

traffic traversing link_{12} either originates from node 1, or terminates at node 2, or both. Node 1 spreads flow_{1i} over route 1-2-i only if link_{2i} is up. Similarly, node i spreads flow_{i2} over route i-1-2 only if link_{i1} is up. We can also assume $\lambda_{ii} = 0$ because this traffic does not need to traverse the network. Thus, we have

$$T_{12} = \frac{2\lambda_{12}}{P_{12}} + \sum_{i\geq 3} \frac{\lambda_{1i}}{P_{1i}} A_{2i} + \sum_{i\geq 3} \frac{\lambda_{i2}}{P_{i2}} A_{i1}. \quad (2)$$

Since we are only considering valid traffic, the row and column sums of Λ are bounded by r. It is easy to see that

$$\begin{aligned} T_{12} &\leq \frac{r}{\min_{i\neq 1} P_{1i}} + \frac{r}{\min_{j\neq 2} P_{j2}} & (3)\\ &\leq \frac{2r}{\min_{i\neq 1, j\neq 2}(P_{1i}, P_{j2})}. & (4)\end{aligned}$$

This gives an easy upper bound on T_{12}.

For a given path matrix P, we can maximize T_{12} over all valid traffic matrices Λ; i.e., under the constraints $\sum_i \lambda_{1i} \leq r$ and $\sum_i \lambda_{i2} \leq r$. From Equation (2),

$$\begin{aligned} &\max_{\Lambda} T_{12} \\ =\ &\max_{\lambda_{12}} \left(\frac{2\lambda_{12}}{P_{12}} + (r - \lambda_{12}) \left(\max_{i\geq 3} \frac{A_{2i}}{P_{1i}} + \max_{i\geq 3} \frac{A_{i1}}{P_{i2}} \right) \right) \\ =\ &\max \left\{ \frac{2r}{P_{12}},\ r \left(\max_{i\geq 3} \frac{A_{2i}}{P_{1i}} + \max_{i\geq 3} \frac{A_{i1}}{P_{i2}} \right) \right\}. \quad (5)\end{aligned}$$

Now we can maximize Equation (5) over all path matrices representing k link failures. We omit the details here and only give the final result. The amount of capacity required on each link to tolerate k arbitrary link failures, for $1 \leq k \leq N-2$, is

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

For $k \geq N-1$, in the worst case failure scenario, the network becomes disconnected, therefore no throughput guarantee can be given.

Equation (6) shows that the amount of capacity needed to tolerate k link failures is on the order of $\frac{2r}{N-k}$. This is good news because the curve of $\frac{2r}{N-k}$ is very flat for small values of k (see Figure 2). This means that a small amount of overprovisioning goes a long way to make the network more fault tolerant. For example, if the links in a 100 node network are overprovisioned by just about 5.3%, the network can tolerate any five logical link failures. Compare this with existing backbone networks that typically use significantly more overprovisioning, yet are unable to make any guarantees.

3.4 Combination of Node and Link Failures

Now assume that there are k_1 node failures and k_2 link failures. This is equivalent to having k_2 link failures in a $(N-k_1)$-node network. So

$$C(N, k_1, k_2) = C_l(N - k_1, k_2), \quad (7)$$

where C_l is given by Equation (6).

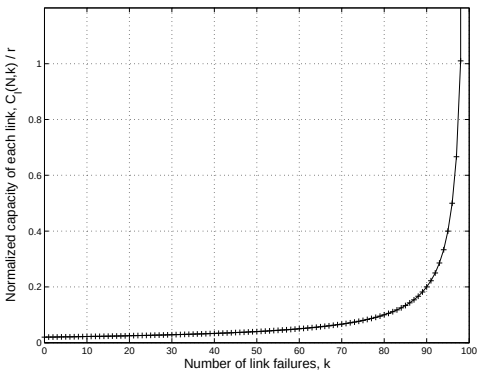

Figure 2: The required link capacity vs. number of link failures in a 100-node network as given in Equation (6).

4. DISCUSSION

Equation (7) has some interesting consequences. A network operator can deploy links so that under normal operation the load on each link never exceeds α, and with up to k_1 node failures and k_2 link failures, the load never exceeds β, where $0 < \alpha < \beta < 1$. Unlike in a traditional network in which maximum load is determined by the current traffic condition, and can reach a very large value, the operator can constrain the load on every link to be below a threshold. For example, an operator can keep the load on every link below the "knee" of the queueing delay vs. utilization curve, even under failure. Unlike traditional network design methods, explicit results like Equation (7) enables the design of a Valiant Load-balanced network with provable performance characteristics.

In this paper we've restricted our consideration to a homogeneous network, in which all N nodes are identical and connect to access networks of equal capacity. Our results apply to a more general and realistic network in which each node has a different capacity. In that case, load-balancing is no longer uniform, and the optimal load-balancing ratios can be derived.

A common concern with load-balancing is that the traffic takes longer paths than strictly necessary and so typically experiences longer delay. In a Valiant Load-balanced network, the delay is bounded by twice the network diameter. However, the tradeoff of delay vs. throughput is always present in network design. This network architecture is intended for regional networks, such as the Internet backbone of the United States. In such a network, the maximum propagation delay due to two-hop routing is well below 100ms, which is acceptable for most applications. For the applications to

which low latency is important, if these packets account for no more than $2/N$-th of each flow, we can choose to send these packets via the direct route. In fact, one can envisage a scheme in which traffic is only load-balanced over multiple paths between a pair of nodes when traffic between them exceeds $\frac{2r}{N}$. On the other hand, delay jitters are usually much more important for most applications. In the VLB architecture, any burst in traffic is absorbed by multiple paths, mediating the effect of the burst, and thus reducing possible jitters in delay.

Another concern with load-balancing is packet reordering. When packets from the same flow travel through different paths with different delays, they may arrive at the destination out of order. But in the Internet backbone, where tens of thousands of application level flows share a link, each taking up bandwidth much smaller than backbone link capacities, it is possible to ensure that no single application flow is sent over multiple paths. This can be done, for example, by flow hashing as in [11].

The Valiant Load-balanced network enables a simple routing scheme in which the destination of a packet is looked up only once. The ingress node can forward a packet immediately to an outgoing link based, for example, on a hash of its address. Only the intermediate node needs to know where the packet is headed, so that it can deliver it to the correct egress node and onto the correct access network. Or, alternatively, a tunnel can be set up for each *two-hop* path, so only the ingress router needs to look up the destination of a packet, upon which it will send the packet onto an appropriate path. Unlike multi-hop routing, where routing protocols are complex, fragile and converge slowly, this network allows a simpler, and more robust routing protocol. The logical links, which may traverse multiple physical links, can be established with some tunneling mechanism much simpler than IP. So the VLB scheme can lower the performance requirement imposed on routers today, enabling the possibility of simpler and cheaper routers.

The reader may recognize similarities between the proposed load-balanced network and an MPLS network. But they are fundamentally different. The proposed network architecture requires no dynamic routing, and the network is effectively self-healing through load-balancing. In an MPLS networks, there is rarely any load-balancing. A label-switched path needs to be established and torn down on the fly, requiring a complex protocol.

The architecture raises an interesting question of economic policy. It is not obvious how a Valiant Load-balanced network can support multiple competing network operators. For example, how could they peer so as to maintain end-to-end performance guarantees? Perhaps one or more nodes could be devoted to peering points, with known, bounded amounts of traffic between network operators. Or perhaps it would be better to build a single national shared grid of wavelengths to which all network operators connect and compete by building their own nodes, and share the capital cost of the grid. These, and many more economic and social questions all need to be addressed, and are beyond the scope of this paper.

5. CONCLUSION

The goal of our work is to develop methods for network design, so that backbone networks can be more reliable, operate more predictably, and be provisioned more efficiently over the lifetime of deployment. We strongly believe that some form of explicit and managed load-balancing goes a long way to achieving this. It improves availability, removes hot-spots and reduces congestion. Load-balancing is, in general, simple and intuitive. The appealing characteristic of Valiant Load-balancing is that it provides a way to quantify, analyze and then guarantee performance.

There are many ways to load-balance, and many alternatives to consider; and there is much more work to be done to answer some of the questions raised here. Our goal here is to point out the potential power of load-balancing, and to start a discussion.

6. REFERENCES

[1] Y. Zhang, M. Roughan, N. Duffield, A. Greenberg, "Fast accurate computation of large-scale IP traffic matrices from link loads," *Proc. ACM SIGMETRICS*, June 2003.

[2] R. Teixeira, A. Shaikh, T. Griffin, J. Rexford, "Dynamics of hot-potato routing in IP networks," *Proc. ACM SIGMETRICS*, June 2004.

[3] Anindya Basu, Jon Riecke, "Stability issues in OSPF routing," *Proc. ACM Sigcomm*, August 2001.

[4] B. Fortz, J. Rexford M. Thorup, "Traffic engineering with traditional IP routing protocols" *IEEE Communications Magazine*, 40(10):118-124, 2002.

[5] A. Nucci, B. Schroeder, S. Bhattacharyya, N. Taft, C. Diot, "IGP Link Weight Assignment for Transient Link Failures," *18th International Teletraffic Congress*, 2003.

[6] L. Valiant, G. Brebner, "Universal schemes for parallel communication," *Proc. of the 13th Annual Symposium on Theory of Computing*, pp. 263277, May 1981.

[7] C.-S. Chang, D.-S. Lee, Y.-S. Jou, "Load balanced Birkhoff-von Neumann switches, part I: one-stage buffering," *IEEE HPSR* '01, Dallas, May 2001.

[8] I. Keslassy, S.-T. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, N. McKeown, "Scaling Internet Routers Using Optics," *Proc. of ACM SIGCOMM '03*, Karlsruhe, Germany, August 2003. Also in *Computer Communication Review*, Vol. 33, No. 4, pp. 189-200, October 2003.

[9] M. Kodialam, T.V. Lakshman, S Sengupta, "Efficient and Robust Routing of Highly Variable Traffic," HotNets III, San Diego, Nov. 2004.

[10] I. Keslassy, C.-S. Chang, N. McKeown, D.-S. Lee, "Optimal Load-Balancing", *Infocom 2005*, Miami, Florida.

[11] G. H. Gilmer, "Composite Links - A Tool for Bandwidth Optimization," Avici white paper. http://www.avici.com/technology/whitepapers/