Designing a Predictable Internet Backbone with Valiant Load-Balancing

Rui Zhang-Shen, Nick McKeown

Computer Systems Laboratory
Stanford University
Stanford, CA 94305-9030, USA
{rzhang, nickm}@stanford.edu

Abstract. Network operators would like their network to support current and future traffic matrices, even when links and routers fail. Not surprisingly, no backbone network can do this today: It is hard to accurately measure the current matrix, and harder still to predict future ones. Even if the matrices are known, how do we know a network will support them, particularly under failures? As a result, today's networks are designed in a somewhat ad-hoc fashion, using rules-of-thumb and crude estimates of current and future traffic.
Previously we proposed the use of Valiant Load-balancing (VLB) for backbone design. It can guarantee 100% throughput to *any* traffic matrix, even under link and router failures. Our initial work was limited to homogeneous backbones in which routers had the same capacity. In this paper we extend our results in two ways: First, we show that the same qualities of service (guaranteed support of any traffic matrix with or without failure) can be achieved in a realistic heterogeneous backbone network; and second, we show that VLB is optimal, in the sense that the capacity required by VLB is very close to the lower bound of total capacity needed by any architecture in order to support all traffic matrices.

1 Introduction

A network operator would like their backbone network to serve customers' demands at all times. But most networks have grown in an ad-hoc fashion, which – when considering failures and maintenance that frequently change the network – makes it impractical to systematically provision links so they have the capacity to carry traffic both during normal operation and under failures. To compensate, network operators tend to grossly over-provision their networks (typically below 20% utilization), because it is hard to know where new customers will join, what new applications will become popular, and when links and routers will fail.

It would help if we knew the traffic matrices the network will have to carry throughout its lifetime. Though the usual practice is to measure the current demand, and then extrapolate it to the future, the approach does not work for many reasons.

First, Internet traffic is hard to measure. The number of entries in a traffic matrix is roughly quadratic in the number of nodes, so it is usually impractical to obtain all the measurements directly. To estimate the traffic matrix from incomplete measurements, the best techniques today give errors of 20% or more [6] [9]. More importantly, the traffic demand fluctuates over time, and it is hard to determine when the *peak* usage of the network is. In practice, the “peak” is determined in an ad-hoc manner – by inspection, or some rules-of-thumb.

Second, even if we could accurately measure the current traffic matrix, it is hard to extrapolate to the future. Typically, estimates are based on historic growth rates, and adjusted according to marketing forecasts. But it’s hard to predict future growth rates and what new applications will become popular. Even if the *total* growth rate is estimated correctly, the growth may not be uniform across the whole network, and the introduction of new applications 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 which may bring new demands. So network design has always used a wrong estimate of the future traffic matrix, and the designed network cannot guarantee to support the actual demand. It’s therefore not surprising that operators so heavily over-provision their networks.

In summary, existing networks, which have evolved in an ad-hoc fashion, have unpredictable performance. With current design techniques, it is hard to design a network with throughput guarantees because it is impossible to obtain a good estimate of the future traffic matrix. Once built, a network may have to work with a range of traffic conditions, but it is unknown to network operators as to how to design a network for a wide range of traffic matrices.

We recently proposed Valiant Load Balancing (VLB) [10], so that backbone networks can be designed to give strong guarantees on the support for an arbitrary set of traffic matrices, even under failure, *and* operate at much higher utilization (and hence higher efficiency and lower cost) than today. We assume that each backbone node in the network, or Point-of-Presence (PoP), has constrained capacity, such as the aggregate capacity of the access network it serves, and design a backbone network to guarantee 100% throughput for *any* traffic matrix.

The limitations of designing a network for a specific traffic matrix, and the necessity to design for a wide range of traffic matrices, have been realized by some researchers recently, and a few schemes have been proposed [1] [5] [10] [7]. Most of these schemes use some type of load-balancing.

The VLB architecture makes the job of estimating the future traffic much simpler. While obtaining a traffic matrix estimation is hard, it is easier to measure, or estimate, the total amount of traffic entering (leaving) a backbone node from (to) its access network. When a new customer joins the network, we add their 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

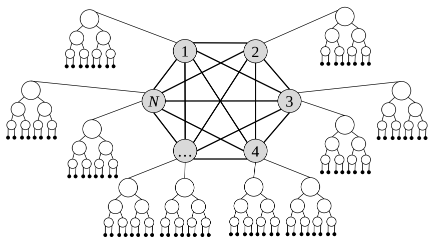

Fig. 1. A hierarchical network with N backbone nodes. The backbone nodes are connected by a (logical) full mesh, and each node serves an access network.

that the node serves. While still not trivial, it is a lot easier than estimating the traffic from every node to every other node.

The Valiant load-balancing architecture has simple and efficient fault tolerance so that only a small fraction of extra capacity is required to guarantee service under a number of failures in the network. The failure recovery can be quick because no new paths need to be established on the fly.

In this paper, we extend the result of [10] to networks with arbitrary node capacities. We will focus on deriving the optimal capacity allocation in this network and leave fault tolerance for future work. The rest of the paper is organized as follows: Section 2 introduces the VLB architecture and the notation used in this paper; Section 3 derives the lower bound on the required capacity to support all traffic matrices and two load-balancing schemes which achieve capacity requirements close to the lower bound; We then relate our work to others' and conclude the paper.

2 Valiant Load-balancing

The Valiant load-balancing architecture was first proposed by L. G. Valiant for processor interconnection networks [8], and has received recent interest for scalable routers with performance guarantees [2] [4]. Keslassy et al. proved that uniform Valiant load-balancing is the unique architecture which requires the minimum node capacity in interconnecting a set of identical nodes [3]. We applied the VLB architecture to designing a predictable Internet backbone which can guarantee throughput to all traffic matrices, and can tolerate a number of link and router failures with only a small amount of excess capacity [10]. We will re-introduce the architecture and re-state the relevant results here before extending it to the more general scenario in backbone network design.

2.1 Previous Results

Consider a network consisting of multiple backbone nodes, or PoPs, interconnected by long-haul links. The network is arranged as a hierarchy, and each PoP connects an access network to the backbone (see Figure 1). For now, assume that there are N backbone nodes, and all of them are connected to access networks of the same aggregate capacity, r.

A full mesh of logical links of capacity $\frac{2r}{N}$ are established among the N backbone nodes. Traffic entering the backbone is load-balanced equally across all N two-hop paths between ingress and egress. A packet is forwarded twice in the network: In the first stage, 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 at the application flow level. Assume we can achieve perfect load-balancing, i.e., we can split traffic at the exact ratio we desire, then each node receives $1/N$-th of every node's traffic in the first stage. In the second stage, all packets are delivered to the final destination.

Uniform load-balancing leads to a guaranteed 100% throughput in this network. We consider the two packet forwarding stages. 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 among N nodes of access capacity r.

This is perhaps a surprising result – a network where each pair of nodes are connected with a link of capacity $\frac{2r}{N}$ can serve traffic matrices where a node can send traffic to another node at rate r. This shows the power of load-balancing. Valiant load-balancing makes sure that each flow is carried by N paths, and each link carries a fraction of many flows, therefore a large flow is evened out by other small flows. If all the traffic were to be sent through direct paths, we would need a full-mesh network of link capacity r. Therefore load-balancing is $\frac{N}{2}$ times more efficient than direct routing. Although it may seem inefficient that every packet should traverse the network twice, it has been proved that in order to serve all traffic matrices in a network of identical nodes, the uniform Valiant load-balancing architecture provides the unique optimal interconnection pattern which requires the lowest capacity at each node [3]. An even more surprising result is that the VLB network only requires a small fraction of extra capacity in order to tolerate failures in the network [10].

A common concern with load-balancing is that packets may incur a longer delay. In a Valiant load-balanced network, propagation delay is bounded by traversing twice the network diameter. Within the continental US, it's been measured that this delay is well below 100ms, which is acceptable for all applications we know of. We believe that VLB gives a reasonable tradeoff of increased fixed propagation delay for improved predictability, and lower delay variance.

In this paper, we consider Valiant load-balancing in a more general and more realistic case – we remove the assumption that all access networks have the same capacity. Once the backbone network is no longer symmetric, the first stage load-balancing should no longer be uniform, and it is not clear what the optimal load-balancing scheme should be. In the rest of the paper, we will search for a scheme that is optimal in a similar sense as the uniform network case, i.e., a scheme which minimizes the interconnection capacity at each node.

2.2 Notations

We consider a backbone network consisting of $N \geq 3$ nodes of access capacities $r_1, r_2, \ldots, r_N$, where *access capacity* is the aggregate capacity of the access network a backbone node serves. This means, node i can initiate traffic to the other backbone nodes at a rate up to r_i, and can receive traffic from the other backbone nodes at a rate up to r_i. Without loss of generality, we assume that the nodes have been sorted according to decreasing access capacities, i.e., $r_1 \geq r_2 \geq \ldots \geq r_N$.

A traffic demand matrix Λ is an $N \times N$ matrix where the 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., $\sum_j \lambda_{ij} \leq r_i$, and $\sum_j \lambda_{ji} \leq r_i$, $\forall i$. We will only consider valid traffic matrices in this paper and our goal is to guarantee 100% throughput to *all* valid traffic matrices.

We start with a full-mesh interconnecting the N nodes, where the link capacity between node i and node j is represented by c_{ij}. Let C be the link capacity matrix $\{c_{ij}\}$. We are interested in finding the minimum values of c_{ij} that are required to serve all traffic matrices. If link$_{ij}$ is not needed, then we simply set c_{ij} to zero. We assume that a node can freely send traffic to itself without requiring any network resource, so we set $c_{ii} = \infty, \forall i$, and will not try to optimize them. Given any traffic matrix Λ, we can also set all its diagonal entries to zero.

The *interconnection capacity* of node i is the sum of the link capacities through which it connects to the other nodes, and is represented by $l_i = \sum_{j:j \neq i} c_{ij}$. The sum of all nodes' interconnection capacities is the *total interconnection capacity* of the network, and is represented by $L = \sum_{i=1}^{N} l_i = \sum_{(i,j):i \neq j} c_{ij}$. For convenience, let $R = \sum_{i=1}^{N} r_i$ be the *total access capacity* of all the nodes. We further define the *fanout* of node i to be $f_i = l_i / r_i$, the ratio of a node's interconnection capacity to its access capacity. Define the *network fanout* f to be the maximum fanout among all nodes, $f = \max_i f_i$.

Under these definitions, in the uniform network where every node has access capacity r, the optimal link capacities are $c_{ij} = \frac{2r}{N}$, for $i \neq j$. The interconnection capacity of a node is $l_i = \frac{2r}{N}(N-1), \forall i$, so the fanout of a node is $f_i = \frac{l_i}{r} = \frac{2(N-1)}{N}, \forall i$. Thus the network fanout is $f^u = \frac{2(N-1)}{N}$. The total interconnection capacity is $L = 2(N-1)r$ and the total access capacity is $R = Nr$.

3 Optimal Interconnect for a Heterogeneous Network

In this section we investigate the interconnection capacities required to serve any traffic matrix among N backbone nodes. To find the optimal interconnect, we can use the total capacity as the criteria and minimize L, or we can use the network fanout as the criteria and minimize f. Both may be of importance in network design. Minimizing the total capacity is more reasonable if the same amount of capacity costs the same in different places of the network. But by minimizing the network fanout, or the maximum fanout of all nodes, we try to make the fanouts of the nodes close to each other. This way, the interconnection capacity of a node is roughly proportional to its access capacity. This criteria is more sensible if the nodes in the network differ widely. The following lemma ties the two criteria together, and the minimum total capacity L gives a lower bound on the network fanout.

Lemma 1.

$$f = \max_i f_i = \max_i \frac{l_i}{r_i} \geq \frac{\sum_i l_i}{\sum_i r_i} = \frac{L}{R}$$

The inequality can be proved by induction and [11] has the details.

In the rest of this section, we will first derive the minimum total capacity required to serve all traffic matrices in any architecture, to give a lower bound on the network fanout. Then we propose a simple "gravity" load-balancing scheme which achieves a network fanout within a factor 2 of the lower bound. Finally we derive the minimum network fanout under oblivious load-balancing in the network, and show that it's within a constant factor 1.2 from the lower bound.

3.1 Minimum Total Capacity: Lower Bound on Network Fanout

Given a demand matrix Λ and we want to know whether an interconnection capacity matrix C can serve the demand. Load-balancing is allowed, so we do not require that $c_{ij} \geq \lambda_{ij}$ for all (i, j) pairs. For example, if there is not enough capacity from node s to node t to serve the demand, i.e., $c_{st} < \lambda_{st}$, we can send part of the flow through an alternative route, say, route s-k-t. Suppose we send amount x of flow$_{st}$ through route s-k-t, then the actual load carried on the network is Λ' which is obtained from Λ by subtracting x from λ_{st} and adding x to both λ_{sk} and λ_{kt}. We call this the *load-balancing transformation* of traffic matrix Λ, and the resulting matrix Λ' the *load matrix*. The traffic matrix Λ can be *served* by the capacity matrix C, if we can load-balance flows in Λ and obtain a load matrix Λ' such that $\lambda'_{ij} \leq c_{ij}$, $\forall i, j$.

Before we state the theorem of minimum total capacity, we prove the following lemma.

Lemma 2. *Load balancing cannot reduce the sum of the entries in the upper triangle of Λ, i.e., $\sum_{i<j} \lambda'_{ij} \geq \sum_{i<j} \lambda_{ij}$ for any load matrix Λ' obtained by load-balancing transformation of Λ. Same is true for the lower triangle.*

Proof. Suppose we route the amount x of flow$_{i_1 i_k}$, $i_1 < i_k$, through the route i_1-i_2-. . .-i_k. Then we subtract x from $\lambda_{i_1 i_k}$ and add x to each one of $\lambda_{i_1 i_2}$, $\lambda_{i_2 i_3}$, . . . , $\lambda_{i_{k-1} i_k}$ to obtain the load matrix Λ'. Since $i_1 < i_k$, and $i_1, i_2, \ldots, i_k \in \{1, 2, \ldots, N\}$, there must exist some $j : 1 \leq j \leq k-1$ such that $i_j < i_{j+1}$.

Thus the sum of the upper triangle of Λ does not decrease after the transformation, and we have $\sum_{i<j} \lambda'_{ij} \geq \sum_{i<j} \lambda_{ij}$. By induction, further load-balancing transformation will keep this inequality.

The proof for the lower triangle is similar. □

Corollary 1. *A necessary condition for capacity matrix C to be able to serve traffic matrix Λ is*

$$\sum_{i<j} c_{ij} \geq \sum_{i<j} \lambda_{ij} \text{ and } \sum_{i>j} c_{ij} \geq \sum_{i>j} \lambda_{ij}. \tag{1}$$

The proof follows from Lemma 2 and the details are in [11].

Remark: The intuition of the above lemma is, if we line up the nodes from left to right according to their node numbers, then the upper triangle of the traffic matrix Λ represents the traffic that needs to go from left to right. No matter how we load-balance the demand, the amount of traffic that needs to be sent from left to right does not decrease. The upper triangle of the capacity matrix C represents the capacity that can carry traffic from left to right, and this has to be at least the amount of traffic that needs to be sent from left to right. Same is true for the traffic that needs to go the other direction.

Theorem 1. (minimum capacity) *In order to serve any valid traffic matrix among N nodes of capacities $r_1 \geq r_2 \geq \ldots \geq r_N$, the minimum total interconnection capacity required is $2(\sum_i r_i - \max_i r_i) = 2\sum_{i=2}^{N} r_i$.*

Outline of proof. Necessity is shown by applying Corollary 1 to the following traffic matrix and its transpose:

$$\Lambda^{(1)} = \begin{pmatrix} 0 & r_2 & 0 & \ldots & 0 \\ 0 & 0 & r_3 & \ldots & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & r_N \\ 0 & 0 & 0 & \ldots & 0 \end{pmatrix}.$$

Sufficiency is shown by arranging the nodes in a "star" topology as in Figure 2. For details, please see [11]. □

Theorem 1 together with Lemma 1 gives a lower bound on the network fanout:

$$f \geq \frac{L}{R} \geq \frac{2(\sum_i r_i - \max_i r_i)}{R} = 2\left(1 - \frac{r_1}{R}\right). \tag{2}$$

This lower bound can be less than 1 if $r_1 > \frac{R}{2}$, or equivalently, if $r_1 > \sum_{i=2}^{N} r_i$. This is the case when the largest node can initiate more traffic than all the other nodes combined can receive. But if we only allow valid traffic matrices, the largest node cannot initiate traffic at its maximum rate because that would overload some of the other nodes. So we can replace the capacity of node 1 with $\min(r_1, \sum_{i=2}^{N} r_i)$ and will not change any property of the network. This is equivalent to introducing 1 as another lower bound for f. Since the network

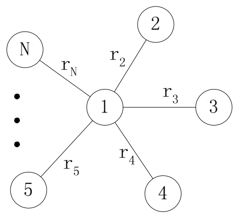

Fig. 2. The star topology which achieves the minimum total interconnection capacity. Node 1 has the highest access capacity.

fanout f is the maximum of all the nodes' fanouts, this lower bound of 1 can also be obtained by considering the smallest node, whose fanout has to be at least 1. So we now have the following:

Corollary 2. *In order to serve all valid traffic matrices, the network fanout of an interconnection network is lowered bounded by* $2\left(1-\frac{\max_i r_i}{R}\right)$ *and* 1*, i.e.,*

$$f \geq \max(1, 2\left(1-\frac{\max_i r_i}{R}\right)). \tag{3}$$

The lower bound on network fanout can be achieved in some cases. When $r_1 \geq \sum_{i=2}^{N} r_i$, a network fanout of 1 can be achievable by the star topology in Figure 2, where Node 1 has a fanout of at most 1 and the other nodes have a fanout of 1. But when $r_1 < \sum_{i=2}^{N} r_i$, the star topology does not minimize the network fanout. (The schemes presented in Sections 3.2 and 3.3 achieve better network fanouts.)

In the uniform case where all the nodes have the same capacity r, the lower bound in Equation (2) can also be achieved: $f^u = 2(1-\frac{1}{N})$. So the uniform full mesh architecture minimizes the network fanout.

Both the uniform full mesh and the star topologies have a total interconnection capacity of $2r(N-1)$. So in order to minimize the total interconnection capacity of the network, both uniform full mesh and star topologies are optimal. But if the goal is to minimize the network fanout, the full mesh is the only topology that is optimal, as proved in [3]. From the network design point of view, the full mesh topology is better, because the star topology has many single points of failure, and it requires a lot of processing power from the center node. Therefore, we will use network fanout as the optimization criteria in designing the backbone network.

3.2 Gravity Full Mesh: 2-Approximation of Optimal Fanout

The star topology in Figure 2 does not give a good network fanout, because the node fanouts are $f_1 = \sum_{i=2}^{N} r_i/r_1$ and $f_i = 1$ for $i > 1$, and the fanout of the center node can be large. For example, when all the nodes have the same capacity, we have $f_1 = N - 1$, which is much larger than the optimal network fanout $2(1 - \frac{1}{N})$.

We can easily obtain a much better network fanout by extending the load-balancing scheme of the uniform case. Suppose all the nodes' access capacities are integer multiples of some capacity "granularity" r, i.e., $r_i = k_i r$ for some integer $k_i, \forall i$. Then node i can be treated as k_i nodes of capacity r located together. Now if we count all the imaginary "nodes" of capacity r, there are $M = \sum_i k_i = \frac{R}{r}$ of them. Between each pair of such "nodes" should be a link of capacity $\frac{2r}{M}$. Now between the real nodes i and j, which are clusters of k_i and k_j imaginary "nodes", there should be $k_i \times k_j$ links of capacity $\frac{2r}{M}$, i.e.,

$$c_{ij} = \frac{2r}{M} k_i k_j = \frac{2r^2}{R} k_i k_j = \frac{2 r_i r_j}{R}. \tag{4}$$

The link capacity between node i and node j is proportional to the product of capacities of the two nodes, therefore we call this the "gravity full mesh". Note that the capacity granularity r, which has dropped out of the expression in Equation (4), can be arbitrarily small, so we actually do not require any relationship among the nodes' capacities.

The load-balancing scheme in the gravity full mesh is: in the first stage traffic is spread proportionally to the capacity of the intermediate nodes, i.e., a proportion r_i/R of traffic is load-balanced to node i; in the second stage, traffic is delivered to the final destination. This network can be viewed as a uniform network of M nodes of capacity r, some of which have been clustered together, so the link capacities given in Equation (4) are sufficient to guarantee 100% throughput for any traffic matrix.

In the gravity full mesh, the fanout of node i is

$$f_i = \frac{l_i}{r_i} = \frac{\sum_{j \neq i} 2 r_i r_j / R}{r_i} = 2 \frac{\sum_{j \neq i} r_j}{R} = 2(1 - \frac{r_i}{R}).$$

So

$$f = \max_i f_i = 2(1 - \frac{\min_i r_i}{R}) = 2(1 - \frac{r_N}{R}) < 2. \tag{5}$$

The optimal network fanout f is at least 1 (Corollary 2), so we have obtained a 2-approximation to the optimal network fanout. In most cases, the approximation is much better than 2.

3.3 Minimum Network Fanout under Oblivious Load-Balancing

We will now directly minimize the network fanout f, in the cases when the optimal network fanout is not easily known. In Section 3.1 we have shown that the minimum fanout $\min f = 1$ when $r_1 \geq \sum_{i=2}^{N} r_i$, so we will only consider the case when $r_1 < \sum_{i=2}^{N} r_i$ in this subsection.

If we want to find the optimal interconnection to minimize f, the load-balancing scheme should take into account all system information, such as traffic matrix and node capacities. Let $p_i^{st}(\Lambda)$ be the portion of the flow from node s to node t that is load-balanced to node i when the traffic matrix is Λ. By definition, $p_i^{st}(\Lambda) \geq 0$ and $\sum_i p_i^{st}(\Lambda) = 1$. In general, the traffic matrix may not be easily available to the nodes, and even if it is available, it can change with time. So we are only interested in the load-balancing schemes that are independent of the traffic pattern, i.e., we let $p_i^{st}(\Lambda) = p_i^{st}$.

Given the load-balancing ratios p_i^{st} and the traffic matrix Λ, we can calculate the amount of outbound traffic from node n:

$$T_n(\Lambda) = \sum_{i \neq n} \lambda_{ni} + \sum_{i,j \neq n} p_n^{ij} \lambda_{ij}. \tag{6}$$

The first sum is due to the traffic that originates from node n destined for the other nodes. The second sum is the amount of traffic that "passes by" node n, i.e., the traffic that is load-balanced to node n from the other nodes which node n needs to forward to the destination. The inbound traffic to node n has similar properties as the outbound traffic so we only need to consider one of them.

The network should support any valid traffic matrix, so the outbound link capacity of node n needs to be at least $\max_\Lambda T_n(\Lambda)$. Thus $\max_\Lambda T_n(\Lambda)/r_n$ gives a lower bound on the fanout of node n, and the lower bound is in terms of p_i^{st}. Since $f = \max_i f_i$, the maximum of these lower bounds gives a lower bound on the network fanout f. So we can formulate an optimization problem to find the values of p_n^{ij} which give the best lower bound on f, and if the lower bound is also achievable, we have found the optimal network fanout:

$$\begin{array}{ll} \text{minimize} & f = \max_n \{\max_\Lambda T_n(\Lambda)/r_n\} \\ \text{subject to} & p_i^{st} \geq 0, \sum_i p_i^{st} = 1, \forall s \neq t \end{array}$$

where $T_n(\Lambda)$ is given by Equation (6). But the number of variables here is on the order of N^3, and the optimization problem is not easy to solve.

So we further simplify the load-balancing scheme and only consider *oblivious load-balancing*, where the load-balancing ratio is independent of the flow source and destination. That is, we set $p_n^{ij} = p_n$, and a proportion p_n of *every* flow is load-balanced to node n. We can find a closed form expression for $\min f$ under oblivious load-balancing schemes.

Theorem 2. *Under oblivious load-balancing schemes, the minimum network fanout f_o is*

$$f_o = \min f = 1 + \frac{1}{\sum_{j=1}^{N} \frac{r_j}{R - 2r_j}}, \tag{7}$$

if $\max_{i=1}^N r_i < \frac{R}{2}$*; and* $f_o = \min f = 1$ *if* $\max_{i=1}^N r_i \geq \frac{R}{2}$*, where* $R = \sum_{i=1}^N r_i$.

Proof. We will first show that the expression in Equation (7) is a lower bound, and then show that it is achievable. Some details are omitted here and can be found in [11].

Replacing p_i^{st} with p_i, we can rewrite Equation (6) as

$$T_n(\Lambda) = \sum_{i \neq n} \lambda_{ni} + \sum_{i,j \neq n} p_n \lambda_{ij} = (1 - p_n) \sum_{i \neq n} \lambda_{ni} + p_n \sum_{j \neq n} \sum_i \lambda_{ij}.$$

We maximize $T_n(\Lambda)$ over all valid traffic matrices:

$$\max_{\Lambda} T_n(\Lambda) = \max_{\Lambda} \left((1 - p_n) \sum_{i \neq n} \lambda_{ni} + p_n \sum_{j \neq n} \sum_{i} \lambda_{ij} \right) = r_n + p_n(R - 2r_n).$$

From $\max_{\Lambda} T_n(\Lambda)$ we can obtain a lower bound on f_n, which we denote by g_n:

$$g_n = \frac{r_n + p_n(R - 2r_n)}{r_n} = 1 + p_n \frac{R - 2r_n}{r_n}. \tag{8}$$

The maximum of these lower bounds is a lower bound of the network fanout: $f = \max_n f_n \geq \max_n g_n$.

The load-balancing ratios p_n satisfy $\sum_n p_n = 1$, so combining with Equation (8), we have the following equation for the lower bounds g_n:

$$1 = \sum_n p_n = \sum_n (g_n - 1) \frac{r_n}{R - 2r_n}. \tag{9}$$

This means that the positive linear combination of g_n is a constant (note that we only consider the case when $R > 2r_1$), so to minimize $\max_n g_n$, we must have $g_1 = g_2 = \ldots = g_n = g$, and Equation (9) gives

$$g = 1 + \frac{1}{\sum_n \frac{r_n}{R - 2r_n}}. \tag{10}$$

Equation (10) is a lower bound on f. We now show that this lower bound is achievable. Equation (8) gives the load-balancing probabilities: $p_n = (g - 1)\frac{r_n}{R-2r_n}$. Consider the link from node i to node j. The traffic that traverses the link consists of two parts: the traffic that originates from node i and is load-balanced to node j of at most $r_i p_j$; and the traffic that is destined to node j and is load-balanced to node i of at most $r_j p_i$. So the link capacity required from node i to node j is $c_{ij} = r_i p_j + r_j p_i = (g - 1)\left(\frac{r_i r_j}{R-2r_j} + \frac{r_i r_j}{R-2r_i}\right)$.

With the link capacities given above, we can show $f_i = \frac{\sum_{j \neq i} c_{ij}}{r_i} = g$, which means $f = g$. This means that there exists a bandwidth allocation c_{ij} and oblivious load-balancing ratio p_n such that the lower bound g of the network fanout f is achieved. Therefore, under oblivious load-balancing schemes, $f_o = g = 1 + 1/\sum_j \frac{r_j}{R-2r_j}$.

Although we have assumed that $r_1 < \sum_{i=2}^{N} r_i$ in the above derivation, the case $r_1 \geq \sum_{i=2}^{N} r_i$ can be analyzed by the same method. We first set $r_1 = \sum_{i=2}^{N} r_i$ because we only consider valid traffic matrices. Then from Equation (8), we obtain $g_1 = 1$ and $g_n \geq 1$ for $n \geq 2$. To minimize $\max_n g_n$, the optimal solution is $p_1 = 1$ and $p_n = 0$ for $n \geq 2$, in which case we obtain $g_n = 1, \forall n$, and $g = \max_n g_n = 1$. This gives the star topology and $f_o = 1$. □

The network fanout given in Theorem 2 is always greater than one when $r_1 < \sum_{i=2}^{N} r_i$. This is because we want the network to support all traffic matrices.

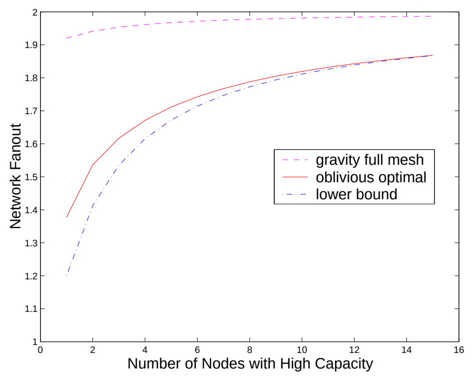

Fig. 3. A comparison of the optimal oblivious network fanout and the upper and lower bounds. The network has 16 nodes, and all nodes take either one of two capacities of ratio 10:1. The x-axis shows the number of nodes taking the higher capacity.

A network fanout of 1 can be achieved if the exact traffic matrix is known and we provision just enough capacity on each link to directly route the traffic. But if the traffic matrix changes, such a network may not provide throughput guarantees. In order to serve *all* valid traffic matrices, we need a minimum network fanout of more than 1, as proved in Theorem 1. In fact, it is a nice surprise that the capacity required to carry all traffic matrices is only less than twice the absolute minimum capacity required in a network, given the traffic matrix.

Note that the gravity full mesh presented in the last subsection is also an oblivious load-balancing scheme, so we expect that the optimal network fanout under oblivious load-balancing given by Theorem 2, is between the network fanout obtained by the "gravity full mesh" given in Equation (5), which we treat as an upper bound, and the lower bound for any architecture given in Corollary 2. We can verify this. Since all the nodes are sorted according to their access capacity, we have $\sum_j \frac{r_j}{R-2r_j} \leq \frac{\sum_j r_j}{R-2r_1} = \frac{R}{R-2r_1}$. So

$$f_o = 1 + \frac{1}{\sum_j \frac{r_j}{R-2r_j}} \geq 1 + \frac{R-2r_1}{R} = 2(1 - \frac{r_1}{R}).$$

That is, the optimal network fanout under oblivious load-balancing is greater than the lower bound. We can similarly show that

$$f_o = 1 + \frac{1}{\sum_j \frac{r_j}{R-2r_j}} \leq 2(1 - \frac{r_N}{R}).$$

In the uniform network case where all nodes have the same access capacity r, the lower and the upper bounds of the optimal network fanout become the same, thus we have $f^u = 2(1 - \frac{1}{N})$.

Now let's consider another example, a network of 16 nodes. The nodes take either one of two access capacities of ratio 10:1. We change the number of nodes with high or low capacities and plot the network fanout in Figure 3. We can see that the oblivious optimal fanout is very close to the lower bound. In fact, as will be shown next, the ratio is bounded by 1.2. So restricting ourselves to oblivious load-balancing does not cause a big loss in capacity efficiency, and we can simplify the load-balancing scheme greatly.

3.4 Oblivious Optimal Network Capacity and the Lower Bound

In the optimal oblivious load-balancing scheme, all the nodes have the same fanout, which is equal to the network fanout, so the total interconnection capacity in the network is $L_o = Rf_o$, where f_o is given by Theorem 2. The lower bound on the total capacity required by any network in order to serve all traffic matrices is given by Theorem 1: $L_l = 2(R - r_1)$, where node 1 has the highest access capacity amongst all N nodes. When $r_1 \geq \sum_{i=2}^{N}$, we have $L_o = L_l$, so we will only consider the case $r_1 < \sum_{i=2}^{N}$. Let α be the ratio of the minimum capacity in oblivious load-balancing and the lower bound for any network:

$$\alpha = \frac{L_o}{L_l} = \frac{R(1 + \frac{1}{\sum_i \frac{r_i}{R-2r_i}})}{2(R - r_1)} = \frac{1 + \frac{1}{\sum_i \frac{r_i}{R-2r_i}}}{2(1 - \frac{r_1}{R})}. \tag{11}$$

We study the value of α in this section.

We have shown that $L_o = Rf_o < 2R$ and $L_l > R$, so α is between 1 and 2. We'd like to find out the maximum value of α. Equation (11) shows that α is a smooth function of r_i, so the maximum value of α is achieved when

$$\frac{\partial \alpha}{\partial r_i} = 0, \forall i. \tag{12}$$

We can solve for the values of r_i^* from the above set of equations. But note that the variables $r_i, i = 2, 3, \ldots, N$, are completely symmetric in the equations, so in the solution, they should have the same value. Since α is only a function of the *ratios* of the r_i's, we let $r_2^* = r_3^* = \ldots = r_N^* = 1$. Now Equation (11) becomes

$$\alpha = \frac{1 + \frac{1}{\frac{r_1}{N-r_1-1} + \frac{N-1}{r_1+N-3}}}{2\frac{N-1}{r_1+N-1}} = \frac{(N + r_1 - 1)(N - 2)}{N^2 - 2N + (r_1 - 1)^2}.$$

Now we solve for r_1^*: $\frac{\partial \alpha}{\partial r_1} = 0$ gives

$$r_1^* = 1 - N + \sqrt{2N(N-1)} \tag{13}$$

and

$$\alpha^* = \frac{(N-2)\sqrt{N(N-1)}}{2N(\sqrt{2}(N-1) - \sqrt{N(N-1)})}. \tag{14}$$

As N increases, α^* increases and when $N \to \infty$, $\alpha^* \to \frac{1}{2}(\sqrt{2} + 1) \approx 1.207$. At the same time, $r_1 \to (\sqrt{2} - 1)N = 0.414N$. Thus, the ratio of the capacity required by optimal oblivious load-balancing and the capacity lower bound is upper bounded by Equation (14), which is at most 1.2. This shows that in order to serve all valid traffic matrices, oblivious load-balancing requires a total link capacity very close to the minimum total capacity required by any architecture.

4 Related Work

The papers by Applegate et al. [1] and Kodialam et al. [5] both studied using load-balancing to guarantee throughput for a wide range of traffic matrices, from a traffic engineering point of view. Given a capacitated network, they both compared the efficiency of providing service for all traffic matrices vs. for a specific traffic matrix. Applegate et al. [1] used maximum link utilization ratio as the performance measure and optimization criteria; Kodialam et al. [5] used the maximum traffic which can be carried by the network. The two measures are equivalent and a ratio of less than 2 was achieved in all their experiments.

Our paper looks at the problem from a network design point of view and tries to find the optimal capacity allocation. We analytically derived that in order to guarantee throughput for all traffic matrices, about twice as much capacity is required compared to the case when the exact traffic matrix is known. This result gives theoretical justification to the upper bound of 2 which has appeared in experiments.

Our work is also complimentary to the above two pieces of work. When designing a network from scratch, or expanding a network's capacity, our results provide guidelines to the optimal capacity allocation, and can lead the network to grow in an optimal direction. If a network has been built and cannot be changed on a short time scale, the traffic engineering approach can help utilize the network resources more efficiently and alleviate congestion in the network.

5 Conclusion

At first glance, it appears that the VLB architecture is inefficient (because it is based on a full mesh) and introduces long delays (because each packet traverses the network twice). But we believe these fears are probably unfounded. First, we can show that the network is surprisingly efficient – in fact, essentially the most efficient network that can support 100% throughput. We suspect that the actual deployed link capacity of such a network could be much lower than current networks that can make no such guarantees of service. Second, we believe that the additional delay is unlikely to be a problem to end-user applications. The additional delay is a fixed propagation delay, and so adds nothing to delay variation. In fact, given the efficiency of the network, the queueing delay (and hence delay variation) is likely to be lower than today.

We believe the Valiant Load-Balancing architecture opens up a new dimension in network design. It enables us to design efficient networks that can guarantee 100% throughput for any traffic matrix, and can continue to do so under a number of element failures. The load-balancing scheme is simple and easy to implement. What's more, as demonstrated in [10], the amount of excess capacity required for fault tolerance is surprisingly small. With VLB, we can build networks to efficiently provide high availability under various traffic conditions and failure scenarios.

6 Acknowledgment

We would like to thank Kamesh Munagala, Isaac Keslassy and Huan Liu for their insights and very helpful inputs.

References

1. D. Applegate and E. Cohen. Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs. In *Proceedings of the ACM SIGCOMM '03 Conference*, 2003.
2. C.-S. Chang, D.-S. Lee, and Y.-S. Jou. Load balanced Birkhoff-von Neumann switches, Part I: One-stage buffering. In *Proceedings of IEEE HPSR '01*, May 2001.
3. I. Keslassy, C.-S. Chang, N. McKeown, and D.-S. Lee. Optimal load-balancing. In *Proceedings of IEEE Infocom 2005*, March 2005.
4. I. Keslassy, S.-T. Chuang, K. Yu, D. Miller, M. Horowitz, O. Solgaard, and N. McKeown. Scaling Internet routers using optics. *Proceedings of ACM SIGCOMM '03, Computer Communication Review*, 33(4):189–200, October 2003.
5. M. Kodialam, T. V. Lakshman, and S. Sengupta. Efficient and robust routing of highly variable traffic. In *HotNets III*, November 2004.
6. A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. Traffic matrix estimation: Existing techniques and new directions. In *Proceedings of ACM SIGCOMM '02*, Pittsburgh, USA, Aug. 2002.
7. G. Prasanna and A. Vishwanath. Traffic constraints instead of traffic matrices: Capabilities of a new approach to traffic characterization. *Providing quality of service in heterogeneous environments: Proceedings of the 18th International Teletraffic Congress*, 2003.
8. L. G. Valiant. A scheme for fast parallel communication. *SIAM Journal on Computing*, 11(2):350–361, 1982.
9. Y. Zhang, M. Roughan, C. Lund, and D. Donoho. An information-theoretic approach to traffic matrix estimation. In *Proceedings of ACM SIGCOMM '03*, pages 301–312. ACM Press, 2003.
10. R. Zhang-Shen and N. McKeown. Designing a predictable Internet backbone network. In *HotNets III*, November 2004.
11. R. Zhang-Shen and N. McKeown. Designing a predictable Internet backbone with Valiant load-balancing (extended version). *Stanford HPNG Technical Report TR05-HPNG-040605*, April 2005.