Capacity Provisioning a Valiant Load-Balanced Network

Andrew R. Curtis and Alejandro López-Ortiz
Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
Email: {a2curtis, alopez-o}@uwaterloo.ca

University of Waterloo Technical Report CS-2009-02

***Abstract*—Valiant load balancing (VLB), also called two-stage load balancing, is gaining popularity as a routing scheme that can serve arbitrary traffic matrices. To date, VLB network design is well understood on a logical full-mesh topology, where VLB is optimal even when nodes or links can fail. In this paper, we address the design and capacity provisioning of arbitrary VLB network topologies. First, we introduce an algorithm to determine if VLB can serve all traffic matrices when a fixed number of arbitrary links fail, and we show how to find a min-cost expansion of the network—via link upgrades or installs or both—so that it is resilient to these failures. Additionally, we propose a method to design a new VLB network under the fixed-charge network design cost model. Finally, we prove that VLB is no longer optimal on unrestricted topologies, and can require more capacity than shortest path routing to serve all traffic matrices on some topologies. These results rely on a novel theorem that characterizes the capacity VLB requires of links crossing each cut, i.e., a partition, of the network's nodes.**

I. Introduction

New applications, such as VoIP, video on demand, and peer-to-peer, have helped create a wider range of traffic patterns on the internet. At the same time, higher quality-of-service demands are being placed on network traffic, due to the rise in importance of the internet. As a result, ISPs have started to deploy mechanisms to monitor network traffic and adapt routes to network traffic patterns if necessary. Dynamic optimization of routing is a difficult problem to solve, so interest has grown in *oblivious* routing schemes [25], which pre-provision forwarding circuits between each pair of nodes and do not update forwarding tables as a result of changing traffic patterns in the network. The traffic model adopted by this research is the *hose traffic model* [8], which requires that any traffic matrix possible under the nodes' ingress/egress rates, can be *served*, i.e., the traffic can be delivered without overloading the capacity of any link.

Recent work on oblivious routing has suggested Valiant load balancing (VLB) as an alternative to direct routing [15], [27]. VLB is also commonly known as two-stage load balancing, because it modifies routing to consist of two stages. In stage 1 of routing, a node splits a predetermined fraction of its ingress traffic to each node in the network. This load-balanced node is chosen randomly for each packet and does not depend on the packet's final destination. In stage 2, nodes forward all load-balanced packets they've received on to the packets' final destination.

Provisioning a logical full-mesh to serve all traffic matrices with VLB has been extensively studied. VLB is optimal in terms of the required link capacity to serve all traffic matrices on a homogeneous full-mesh topology [27], and this optimality remains when nodes can fail [4]. At the logical layer, VLB is *always* the best routing scheme for a homogeneous network. This optimality does not necessarily transfer well to the physical layer, however. We prove that a path is the worst-case topology for VLB and can require $\Theta(n)$ times the capacity of the lower bound for any routing scheme, a sharp contrast to VLB's optimal capacity requirement on a full-mesh. More generally, we show that VLB performs poorly on sparse topologies, where the ratio of links to nodes is low; however, we show that VLB's capacity requirements approach the theoretical optimum as the density of the topology increases. We view this as a step towards proving the viability of VLB. We emphasize that this is a worst-case analysis, VLB is an oblivious routing scheme, and it compares well to the theoretical lower bound for required capacity. In practice, evidence suggests that VLB performs very well, for instance, Kodialam et al.'s experiments [16] have shown VLB's throughput, the maximum utilization of any link, to be within 6% of optimal on ISP topologies, when VLB is optimized for that topology (which is not allowed in our worst-case analysis).

In this paper, we address the design and capacity provisioning of physical VLB networks as well. Network design is a difficult optimization problem; one would like to design a minimal cost network that meets several quality-of-service constraints. This problem is complicated by a number of factors, including difficulties in obtaining traffic estimates, node and link failures, and protocols that were not designed for efficient traffic load balancing. In practice, network design is an ad-hoc practice and is dependent on best practices passed down through the years. Unlike current routing practices, VLB routes traffic in a predictable fashion; VLB routing nodes are not required to update their routes due to congestion or failures. Instead, the goal of VLB network design is to design a minimal cost network with enough capacity to serve all traffic

matrices even under failures.

We show how to design an optimal VLB network under the fixed-charge cost model, which allows the network operator to estimate the expense of installing a link between each pair of nodes. We give an integer program (IP) that designs a minimum-cost VLB network under the fixed-charge cost model. We show that this IP formulation can also be used to find a min-cost upgrade—via link upgrades and/or installs—to an existing network that does not have enough link capacity to serve all traffic matrices with VLB.

The results we have described thus far rely on a theorem we give in Section III that characterizes the capacity of links crossing each cut, a partition of the network's nodes, in order to serve all traffic matrices with VLB. This theorem is the theoretical power behind the other results presented here.

II. Problems, Model, and our Approach

Previous work has left open important questions regarding VLB's behavior, including the following.

- *Can a network serve all traffic matrices with VLB when k arbitrary links fail?* Previous work has answered this question for the case when no failures occur by finding the maximum throughput of a network using VLB [16]. If the throughput is at least 1, then VLB can serve all traffic matrices. We give an algorithm to resolve the question when links can fail in Section III. The power behind this algorithm is a theorem we give that characterizes the required capacity of all links crossing each cut of a networks's nodes.
- *How much capacity does VLB require on arbitrary topologies?* An implication of a theorem of Babaioff and Chuang is that a homogeneous full-mesh is the only topology on which VLB is the optimal routing scheme when VLB load balances traffic evenly to all nodes in stage 1 [4]. Our work sets to determine what's the worst-case capacity requirements of VLB and what topologies VLB performs poorly on. We show that the topology which elicits worst-case VLB behavior is a path on n nodes. We show that as links are added to a VLB network, its worst-case required capacity decreases linearly with the number of additional links. These results are given in Section IV.
- *How to design a VLB network at the physical layer?* As mentioned, previous VLB network design work only applies to the logical layer, e.g., [4], [27]–[29], and there is no clear method to translate these results to the design of arbitrary topologies. We propose an approach to physical layer VLB network design in Section V.

Before giving the details of our results, we present the models and notation used throughout the remainder of this paper.

A. Traffic model

We assume that the amount of traffic entering and exiting the network from each node is fixed and that the two values are equal. We say that the amount of ingress/egress traffic at node i is the *rate* of i and we denote this value by r_i. We assume that the rate of a node is bounded by the sum of the ingress/egress links to that node; this is known as the *hose model* [8], which was originally used to specify the bandwidth requirements of a Virtual Private Network (VPN).

We wish to consider traffic matrices that respect the rates of each node, i.e., no node initiates or receives more than r_i traffic. A *traffic matrix* is an $n \times n$ matrix where the i, j entry indicates the amount of traffic node i is currently sending node j. For a traffic matrix D, we require

$$\sum_{j \in V, j \neq i} D_{ij} \leq r_i \text{ and } \sum_{j \in V, j \neq i} D_{ji} \leq r_i \quad \forall i \in V$$

where D is a traffic matrix of a network $G = (V, E)$ with node rates $r_1, \ldots, r_n$. As observed in [15], it is enough to consider only the traffic matrices where each node sends and receives at its maximum rate, i.e., $\sum_{j \in V, j \neq i} D_{ij} = r_i$ and $\sum_{j \in V, j \neq i} D_{ji} = r_i$, and we say that such a traffic matrix is a *valid traffic matrix* (VTM). We are interested in being able to serve any VTM, so we do not require that the traffic matrix of a network is static, only that it always remains valid.

B. Modeling VLB at the physical layer

Valiant load balancing (VLB) is also known as two-stage routing, because packet routing is done in two stages. Stage 1 is a load balancing step which sends packets to an intermediate nodes, and stage 2 forwards packets to their final destination. In detail, the two stages behave as follows.

- **Stage 1** Each node forwards a predetermined fraction of its ingress traffic to each node in the network; this forwarding is done without regard for each packet's final destination. The fraction of each node's traffic node j receives during stage 1 is specified by α_j.
- **Stage 2** Packets received during stage 1 are forwarded on to their final destination.

We call $\alpha_1, \ldots, \alpha_n$ the *load balancing parameters* of the network, and we require $\sum_{i=1}^{n} \alpha_i = 1$. We also consider a VLB variant where we require $\alpha_1 = \cdots = \alpha_n$, which we call *strict Valiant load balancing* (SVLB). In practice it only makes sense to use SVLB on a homogeneous network with a full-mesh topology; however, we use it here for theoretical analysis.

We assume that if node i is sending traffic to node j at rate D_{ij}, where D_{ij} is i, j entry in the current traffic matrix D, then node x receives exactly $\alpha_x D_{ij}/n$ packets destined for j during stage 1 of routing. In practice, it's unlikely that traffic for each destination would be load-balanced exactly evenly as in this assumption; however, if each packet is sent to a random node during stage 1 according to the distribution $\alpha_1, \ldots, \alpha_n$, then our results hold in expectation. Since we are dealing with traffic matrices where $\sum_{j \in V} D_{ij} = r_i$, node j receives exactly $\alpha_j r_i$ traffic from i during stage 1 and $\alpha_i r_j$ traffic from i during stage 2, so we have that i is sending a total of $\alpha_j r_i + \alpha_i r_j$ traffic to j.

The logical layer view of VLB is a full-mesh and each packet is forwarded along two-hop paths. First to an intermediate node, and then on to its final destination. However,

at the physical layer, this two-hop logical path may be much longer, so we must specify the paths that packets follow when sent from i to j. At the logical layer, specifying the load balancing parameters is enough to specify a solution to the VLB routing problem since all nodes are connected by a logical link; however, since we work with arbitrary topologies, we must specify the path P_{ij} packets sent from i to j follow.

In multi-path VLB, a node i can forward to j along multiple paths, which we define in terms of a flow. For a pair of nodes s, t, an *s-t flow* with rate $|f|$ assigns a value $f(P)$ to each path from s to t in G such that $\sum_P f(P) = |f|$. We denote the amount of traffic flow f places on a link e by $f(e)$.

The following are the VLB routing variants we consider.

- A solution to the *single-path VLB routing problem* consists of the set of traffic split ratios $\alpha_1, \ldots, \alpha_n$ together with a path P_{ij} for all $i, j \in V$ that indicates the path traffic forwarded from i to j follows.
- In the *multi-path VLB routing problem*, a solution consists of a set of traffic split ratios $\alpha_1, \ldots, \alpha_n$ along with a flow f_{ij} for each pair $i, j \in V$ such that $|f_{ij}| = \alpha_j r_i + \alpha_i r_j$, where r_i is the rate of node i; the set of all flows is denoted by $\mathcal{P} = \{f_{ij} : i, j \in V\}$.

We say that a solution to either VLB routing problem is a *feasible solution* if no link carries more traffic than its capacity.

C. Definitions and Notation

Let $G = (V, E)$ be a network with node set V and links E. We denote a link by e or by specifying its endpoints, so a link from i to j is denoted (i, j). We assume that links are bidirected, i.e., whenever $(i, j) \in E$ we also have $(j, i) \in E$. We use n to denote the number of nodes in a network, i.e., let $n = |V|$, and $m = |E|$ denotes the number of links in a network. The nodes connected to i by a link are called i's *neighbors*. We assume that all links in E have a capacity, which indicates the maximum number of bits they can carry at once. We denote the capacity of an link e by $c(e)$. We say that the *utilization* of a link is the amount of traffic it is carrying divided by its capacity. When studying link failures in this paper, we assume that no failure disconnects the network, that is, we assume there is always at least 1 path between all node pairs.

III. Does a Network Have Enough Capacity?

In this section, we give a combinatorial algorithm to find a feasible solution to the multi-path VLB routing problem. Our algorithm relies on a characterization of the capacity links crossing each of a network's cuts require, which we describe in Section III-A. This theorem is easily used to determine if a feasible multi-path VLB routing solution exists when up to k arbitrary links fail, which we describe in Section III-B. Before presenting either of these results, however, we describe the VLB routing problem as a multicommodity flow problem, which we use throughout the rest of this paper.

As mentioned, the case when no failures occur has been solved by Kodialam et al.. They have described a linear program (LP) which finds the maximum throughput multi-path VLB can obtain on a given network [16]. By finding the max throughput, they also determine if a feasible solution to the multi-path VLB routing problem exists, since any feasible solution must have a throughput that is at least 1. The single-path VLB routing problem is NP-hard; however, Kodialam et al. gave a fully polynomial-time approximation algorithm for the problem in [13], so it is possible to find an approximate single-path VLB routing that is arbitrarily close to the optimal single-path VLB routing in polynomial-time.

VLB as a multicommodity flow The VLB routing problem can be described as a *multicommodity flow*, which generalizes the well-known maximum flow problem to have multiple source and destination pairs. A *commodity* is an s-t flow where node s sends traffic to node t at a specified rate r, and is denoted by (s, t, r). The multicommodity flow problem takes as input a set of commodities $\mathcal{W} = \{(s_i, t_i, r_i)\}$, and a solution to the multicommodity flow problem is a set of flows $\mathcal{P} = \{f_{s_i t_i} : (s_i, t_i, r_i) \in \mathcal{W} \text{ and } |f_{s_i,t_i}| = r_i\}$; finally, a solution to multicommodity flow problem $\mathcal{W}$ on network $G = (V, E)$ is *feasible* if for all $e \in E$, $\sum_{k \in \mathcal{W}} f_k(e) \leq c(e)$, where $f_k(e)$ is the amount of traffic sent on e by commodity k.

Viewed as a multicommodity flow problem, the VLB routing problem is a set of $2\binom{n}{2} = n(n-1)$ commodities, specified as follows.

$$\begin{aligned} \mathcal{W}_{\text{VLB}} = \{((s, i), \alpha_i r_s)\} \quad & \forall s, i \in V \quad \text{Stage 1} \\ \cup \{((i, t), \alpha_i r_t)\} \quad & \forall i, t \in V \quad \text{Stage 2} \end{aligned}$$

Since we have captured all flows between nodes, it's clear that the VLB routing problem with load balancing parameters $\alpha_1, \ldots, \alpha_n$ admits a solution if and only if the multicommodity flow $\mathcal{W}_{\text{VLB}}$ has a feasible solution.

Thus far, we have not precisely described the commodities in $\mathcal{W}_{\text{VLB}}$ since we have not specified values for $\alpha_1, \ldots, \alpha_n$. There are many ways could find values for these load balancing parameters, for instance, values for each α_i that maximizes the network's throughput, the maximum utilization of a link in the network, can be found in polynomial-time using an LP [16]. With values for $\alpha_1, \ldots, \alpha_n$ found by this LP, G can serve all VTMs with VLB if and only if the multicommodity flow $\mathcal{W}_{\text{VLB}}$ has a solution.

A. Characterizing the cuts of a VLB network

Finding a solution to the multicommodity flow problem $\mathcal{W}_{\text{VLB}}$ ensures that a network can use VLB to serve all VTMs; however, we do not gain any insight into the structure of networks with a feasible solution to $\mathcal{W}_{\text{VLB}}$ exists. We now give a combinatorial algorithm to find a solution to the multi-path VLB routing problem. It is based on a theorem we will give next that describes the necessary and sufficient capacity that each cut of a network must have in order to serve all VTMs with VLB.

The theorem is stated in terms of cuts. A *cut* is a partition of V into two disjoint sets, S and $V - S$, such that all pairs

of nodes $i, j \in S$ have a path between them that contains only nodes in S. We denote a cut by $(S, V-S)$. We say that an link (i,j) with $i \in S$ and $j \in V-S$ *crosses* the cut, and we denote the set of all links crossing the cut $(S, V-S)$ by $\delta(S)$. The capacity of a cut $(S, V-S)$ is the sum of capacities of link in $\delta(S)$, and we denote the capacity of $(S, V-S)$ by $c(S) = \sum_{e \in \delta(S)} c(e)$.

For convenience, we denote the rate of a set of nodes $S \subseteq V$ as $R_S = \sum_{i \in S} r_i$. Similarly, we denote the sum of α_i's in a set of nodes as $A_S = \sum_{i \in S} \alpha_i$.

The following theorem gives a necessary and sufficient condition for routing all VTMs regardless of the network's topology.

Theorem 1 (Necessary and sufficient capacity of a cut)**.** *A heterogeneous network G with node rates $r_1, \ldots, r_n$ and load balancing parameters $\alpha_1, \ldots, \alpha_n$ can serve all valid traffic matrices using multi-path VLB routing if and only if, for all cuts $(S, V-S)$ of G,*

$$c(S) \geq A_{V-S} R_S + A_S R_{V-S} = g(S)$$

where $R_S = \sum_{i \in S} r_i$ is the sum of node rates in $S \subseteq V$ and $A_S = \sum_{i \in S} \alpha_i$.

Proof: Necessity is not difficult to show by way of contradiction. We omit the details here; see, e.g., [19] for a proof that necessity holds in any multicommodity flow.

Assume that all cuts $(S, V-S)$ of G have capacity at least $g(S)$. To see that G can serve all VTMs, we will show that there exists a feasible solution to the multicommodity flow problem $\mathcal{W}_{\text{VLB}}$. We need to specify the rate of a commodity, so let $r(k) = r$ for a commodity $k = (s, t, r)$. And we denote the set of i's incoming links by $N^-(i)$ and i outgoing links by $N^+(i)$.

A directed graph is called *capacity balanced* if, for all $i \in V$, $N^+(i) + \text{demand}(i) = N^-(i) + \text{supply}(i)$, where $\text{demand}(i)$ is the sum of commodity rates with i as the target and $\text{supply}(i)$ is the sum of commodity rates where i is the source. Nagamochi and Ibaraki [20], [21] have shown that a feasible solution to a multicommodity flow problem $\mathcal{W}$ exists on a capacity balanced network if, for all its cuts $(S, V-S)$, $c(S) \geq \sum_{k \in \mathcal{W}_S} r(k)$, where $\mathcal{W}_S = \{(s,t,r) \in \mathcal{W} : s \in S \text{ and } t \in V-S\}$.

We assume $\sum_{k \in \mathcal{W}_S} r(k) = A_{V-S} R_S + A_S R_{V-S}$ for the multicommodity flow $\mathcal{W}_{\text{VLB}}$, since necessity holds, so if G is a capacity balanced network, then a feasible solution to $\mathcal{W}_{\text{VLB}}$ exists. Consider an arbitrary $i \in V$. we have $N^+(i) = N^-(i)$ by definition, since G is bidirectional. In a VTM D, we have $\sum_{j \in V, j \neq i} D_{ij} = r_i$ and $\sum_{j \in V, j \neq i} D_{ji} = r_i$, so $\text{supply}(i) = \text{demand}(i)$. Therefore, G is a capacity balanced network, and so a feasible solution to $\mathcal{W}_{\text{VLB}}$ exists. ∎

In the proof of Theorem 1, we show that the demands $\mathcal{W}_{\text{VLB}}$ are *capacity balanced*, so a combinatorial polynomial-time algorithm exists for the multi-path VLB routing problem [20]. However, this algorithm does not find optimal values for $\alpha_1, \ldots, \alpha_n$, so this algorithm is still dependent on the LP of [16] to find optimal settings for these load balancing parameters. We note that if one solves the LP of [16], it returns a solution to the multi-path VLB routing problem, so solving the VLB routing problem with this combinatorial algorithm is redundant.

The statement of Theorem 1 gives an easy algorithm to determine if a feasible solution to $\mathcal{W}_{\text{VLB}}$ exists on a network G; however, the runtime of this algorithm is exponential. We present it here to show the usefulness of Theorem 1 and because we will modify it to account for link failures shortly. The algorithm follows.

1) Find values for $\alpha_1, \ldots, \alpha_n$ using the linear program to maximize throughput described in [16].
2) Enumerate all cuts of G. Let the set of all cuts be $\mathcal{C}$.
3) For each cut $(S, V-S) \in \mathcal{C}$, if $c(S) < g(S)$ then G cannot serve all VTMs.

Unlike the LP described earlier for determining if a network can serve all VTMs with VLB, the runtime of this algorithm is exponential in n, as a network may have exponentially many cuts. Many algorithms exist to enumerate all cuts [1], [11], [23] and they are able to find all cuts in time proportional to the number of cuts in the graph; however, this number may be exponential in n.

B. Serving all valid traffic matrices with link failures

We now show how Theorem 1 can be used to determine if a network can withstand link failures. We say that a network is *k link resilient* if it can serve all VTMs after k arbitrary links are removed.

We show that a slight modification of our combinatorial algorithm to check if a network can use VLB to serve all VTMs can also be used to check if a network is k link resilient. The observation behind the algorithm is that Theorem 1 holds regardless of the number of links crossing a cut, so it gives a necessary and sufficient condition for a network to serve all VTMs under link failures: if a set of links fail, the capacity of all cuts $(S, V-S)$ of the network must remain at least $g(S)$. Therefore, the following algorithm can be used to determine whether or not a network is k link resilient. As before, the algorithm's input is a network $G = (V, E)$ with link capacities, rates $r_1, \ldots, r_n$ for each node, and load balancing parameters $\alpha_1, \ldots, \alpha_n$.

1) For all cuts $(S, V-S)$ of G, let $e_1, \ldots, e_{|\delta(S)|}$ be the links in $\delta(S)$ ordered such that $e_1 \geq \cdots \geq e_{|\delta(S)|}$.
2) If $c(S) - \sum_{i=1}^{k} c(e_i) < g(S)$, then G cannot serve all VTMs under k link failures.

The worst-case runtime of this algorithm is again exponential since a network can have exponentially many cuts. Even so, it is practical to compute it for small networks. We enumerated the cuts of networks with size $n = 20$ in a less than a minute with a naive Python script; more sophisticated techniques exist for larger networks [1], [11], [23].

A weakness with this algorithm as specified above is that we do not update the load balancing parameters after links fail; changing these parameters may modify routes enough that the network could serve all VTMs after the link failures.

This can be resolved, at additional computational expense, by modifying step 2 above so that after removing the k highest capacity links, $e_1, \ldots, e_k$, from a cut, the load balancing parameters $\alpha_1, \ldots, \alpha_n$ are updated using the LP of [16]. Allowing the load balancing parameters to be updated after a failures raises questions about how to update the load balancing parameters online. In one approach, taken by [14], [17], a solution to the VLB routing problem is precomputed for a set of node or link failure scenarios. Then, in the event of a failure, nodes switch to these precomputed routes and settings for $\alpha_1, \ldots, \alpha_n$.

IV. Worst-Case Capacity Requirements of VLB

In this section, we study the necessary and sufficient capacity VLB needs to serve all VTMs on a topology G; we denote this capacity requirement by $L_{\text{SVLB}}(G)$. We begin by proving that VLB requires the most capacity when G is a path in Section IV-A. We next give an example of how $L_{\text{SVLB}}(G)$ decreases linearly as additional links are added to G in Section IV-B. Finally, we conclude this section with a brief comparison of SVLB and shortest path (SP) routing in Section IV-C.

For our analysis, we consider only homogeneous networks, where each node has rate r. We primarily analyze *strict Valiant load balancing* (SVLB), where $\alpha_1 = \cdots = \alpha_n$, no matter the topology since it is easier to analyze than VLB. Similarly to the definition of $L_{\text{SVLB}}(G)$, given a network G using SP routing to serve all VTMs, we denote the minimum necessary and sufficient sum of its link capacities by $L_{\text{SP}}(G)$.

A. Worst-case topology for SVLB is a path

We seek to find the topology which requires the most capacity to serve all VTMs with SVLB. We begin by showing that adding additional links to a network using VLB does not ever increase the network's necessary capacity.

Lemma 2. *Let $G = (V, E)$ be a network that can serve all valid traffic matrices using single- or multi-path VLB and let $G' = (V, E \cup F)$ be a network that is obtained by adding a set of links F to G. Then $L_{VLB}(G') \leq L_{VLB}(G)$.*

Proof: We can set $c(e) = 0$ for all $e \in F$ and G' can serve all VTMs using the links in E with their original capacities since G can serve all VTMs. ∎

This lemma implies that the worst-case topology for VLB, and consequently SVLB, must be a tree, a topology with exactly one path between each pair of nodes. This result contrasts a recent advance on direct routing, which shows that a tree is the optimal topology for single-path direct routing [10]. We will give an example of how adding additional links to a network can decrease its necessary capacity momentarily (Section IV-B); first, we show that a path is the worst-case topology for SVLB.

A *path* is a tree, denoted by $P_n = (V, E)$ where $V = \{0, 1, \ldots, n-1\}$ and each node i is neighbors with $i-1$ and $i+1$, except for when $i = 0, n-1$, then i has one neighbor, 1 and $n-1$ respectively. The following shows that a path is the worst-case topology in terms of necessary total link capacity when using SVLB.

Theorem 3. *For any homogeneous network G, $L_{SVLB}(G)$ is maximized when G is a path.*

Proof: We present a sketch for space reasons. By Lemma 2, we only have to show that $L_{\text{SVLB}}(G)$ is maximized when G is a tree. Suppose that the claim holds for trees with up to $n-1$ nodes; let $G_{n-1} = (V_{n-1}, E_{n-1})$ be a path on $n-1$ nodes, and let $n > 3$. Consider adding a node x to G_{n-1} such that the resulting graph $G_n = (V_{n-1} \cup \{x\}, E_{n-1} \cup \{(x, j), (j, x)\})$ is not a path, that is, $j \neq 1, n-1$. Using Theorem 1, we have that

$$L_{\text{SVLB}}(G_n) = \frac{4r}{n}\Big(\sum_{i=1}^{n-1} i(n-i) - j(n-j) + (n-1)\Big)$$

To compare this to the necessary capacity of a path, let P_n be a path on n nodes. We have that

$$L_{\text{SVLB}}(P_n) = \frac{2r}{n} 2 \sum_{i=1}^{n-1} i(n-i) \tag{1}$$

This gives

$$L_{\text{SVLB}}(G_n) = L_{\text{SVLB}}(P_n) - \frac{4r}{n}\Big(j(n-j) - (n-1)\Big)$$

Since $1 < j < (n-1)$ and $n \geq 4$, we have $j(n-j) > (n-1)$, giving $L_{\text{SVLB}}(G_n) > L_{\text{SVLB}}(P_n)$ as desired. ∎

We can now compute the worst-case capacity for SVLB using Eqn. 1.

$$L_{\text{SVLB}}(P_n) = 2 \sum_{i=1}^{n-1} \frac{2r}{n} i(n-i) = \frac{2(n^2-1)r}{3}$$

We now have that $2(n^2-1)r/3$ is an upper bound on the capacity required to serve all VTMs with SVLB; previous work [27] has shown that $2r(n-1)$ is a lower bound on the amount of capacity required by SVLB on a homogeneous full-mesh. We will discuss how these bounds compare with SP and optimal routing in Section IV-C. First, we give an example of how increasing the number of links in an SVLB network lowers its capacity requirements.

B. How $L_{SVLB}(G)$ is affected by additional links

Lemma 2 implies that adding links to a network reduces the capacity required by for the network to serve all VTMs with VLB, but it does not specify how much $L_{\text{SVLB}}(G)$ decreases (if any) when a new link is added to G. To get an idea for how additional links affect the necessary and sufficient capacity of an SVLB network, we'll show how $L_{\text{SVLB}}(G)$ changes as we add additional links to a cycle. Let C_n denote a network that is a cycle, i.e., if $V = \{0, \ldots, n-1\}$, then each node i has two neighbors $i+1 \mod n$ and $i-1 \mod n$.

Because of a cycle's structure, it's easy to compute $L_{\text{SVLB}}(C_n)$. For each link, we need to find the cut it crosses that maximizes $2r/n \cdot |S| \cdot |V - S|$. Since $(S, V-S)$ must

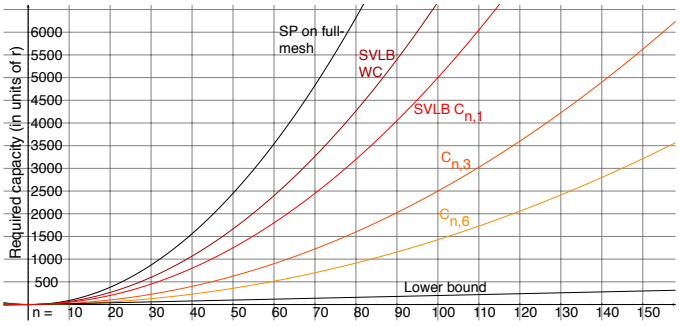

Fig. 1. Comparison of $L_{\text{SP}}(G)$ and $L_{\text{SVLB}}(G)$ for various topologies. From top to bottom, the curves represent the capacity requirements of SP routing on a full-mesh topology, SVLB on a path (the worst-case behavior of SVLB), SVLB on $C_{n,1}$, SVLB on $C_{n,3}$, SVLB on $C_{n,6}$, and the lower bound for any routing scheme.

partition V, we have $c(S) \leq 2r/n \cdot \lfloor n/2 \rfloor \cdot \lceil n/2 \rceil$. For simplicity of presentation, we assume that n is even. In an even cycle, each link crosses a cut $(S, V-S)$ where $|S| = n/2$ and $|V - S| = n/2$. Here, we have $|\delta(S)| = 2$, and the optimal routing strategy is to place 1/2 of the flow from S to $V - S$ on each link in $\delta(S)$. Then, Theorem 1 implies $c(e) = 1/2 \cdot 2r/n \cdot n/2 \cdot n/2 = rn/4$ for all $e \in E$. Therefore, we have

$$L_{\text{SVLB}}(C_n) = \frac{rn^2}{2}$$

when n is even since there are $2n$ links in C_n.

We'd like to be able to add more links around this cycle, so we define $C_{n,l}$ be a network where $V = \{0, \ldots, n-1\}$ and node i has neighbors $\{j \pm k \mod n : k \in \{1, \ldots l\}\}$ and $l < n/2$, so therefore $C_{n,1}$ has $2n$ links, $C_{n,2}$ has $4n$ links, $C_{n,3}$ has $6n$ links, and so forth. We omit the details here, but we can compute the required capacity for $C_{n,l}$, determining that

$$L_{SVLB}(C_{n,l}) = \frac{n^2 r}{k+1}$$

when n is even and $k < n/2$.

In Figure 1, we show the required capacity to route all traffic matrices with SVLB on $C_{n,1}$, $C_{n,3}$, and $C_{n,6}$. As can be seen in the figure, SVLB behaves very well while n is small on $C_{n,3}$, and $C_{n,6}$. As n increases, however, $C_{n,3}$, and $C_{n,6}$ pull away from the lower bound as their growth rate is $\Theta(n^2)r$. The required capacity to serve all VTMs with multi-path SP routing is also shown in the figure. As shown, $L_{\text{SP}}(G)$ when G is a full-mesh requires more capacity than the worst-case $L_{\text{SVLB}}(G)$ for any topology.

C. Comparison of SVLB, SP, and optimal routing

Finally, the following table summarizes our findings about the capacity requirements of SVLB and SP routing.

Routing scheme	**Worst-case**	**Best-case**
SVLB	$2(n^2-1)r/3$	$2r(n-1)$ [27]
topology	path	full-mesh
Shortest path	$\geq rn(n-1)$	$2r(n-1)$
topology	full-mesh	star

In the table, we give the best- and worst-case values for $L_{\text{SVLB}}(G)$ and $L_{\text{SP}}(G)$ on any topology G. For each routing scheme considered, we list the topology that brings about the best- or worst-case behavior.

V. VLB Network Design

In this section we show how the theoretical tools developed thus far can be applied to the design of VLB networks. We are interested in designing VLB networks at the physical layer so as to find a minimum-cost network that can serve all valid traffic matrices. We begin by showing that existing VLB provisioning tools can be adapted to account for the cost of sending traffic on each link, as is done in the VPN design literature; however, we argue that this is ineffective for physical layer network design due to its unrealistic cost model. We instead use the fixed-charge cost model, where each link has a maximum capacity $c(e)$, which upper bounds the amount of traffic that may be placed on the link, and a cost $\text{cost}(e)$ which is the expense required for installing e. This cost function can be the exact cost of installing e, allowing it to consider the physical geography between the endpoints of e. After we present the fixed-charge cost model, we describe the VLB network design problem in Section V-A, and then go on to develop an integer program (IP) that designs a minimum-cost VLB network. In Section V-B, we show how this IP can be modified to find the min-cost expansion of a network—via upgrading links, installing new links, or a combination of both—so that the upgraded network can serve all VTMs. Finally, we give an IP for constructing a VLB network that can serve all VTMs with up to k arbitrary link failures in Section V-C.

Most previous VLB network design results would have one build a full-mesh topology. This strategy is certainly simple, and the resulting network is optimal in terms of link capacity, but it's hardly a practical design for a network that covers a large geographic region. On the other extreme, constructing a network with as few links as possible does not necessarily minimize cost either—each link may need an excessively high amount of bandwidth; additionally, a single link failure might partition such a network. Our aim here is to design the best network with regards to resiliency requirements, the distance between nodes, and available link technology.

Designing a VLB network with the VPN cost model Kodialam et al. described an LP that can find a minimal assignment of link capacities so that a network can use VLB [15]. This LP modifies a standard LP for solving multicommodity flow [2] to include the load balancing parameters $\alpha_1, \ldots, \alpha_n$ as variables. Their LP can easily be modified to include a cost, denoted $\text{cost}(e)$, to send a unit of traffic on link e. Given a topology $G = (V, E)$, their LP finds a solution to the multi-path VLB routing problem that minimizes the necessary total link capacity used. Their LP does not give a cost to routing flow on a link; however, can be made to do so simply by

modifying their objective function to minimize the following.

$$\text{Minimize} \quad \sum_{e \in E} \Big(\text{cost}(e) \cdot \sum_{k \in \mathcal{W}_{\text{VLB}}} f_k(e) \Big)$$

This objective function now charges a fixed-cost per link e to send a unit of traffic on e. This is primarily the cost model used by the literature on provisioning a VPN [8]; this cost model is appropriate in the VPN provisioning setting because the physical network already exists and so the only cost associated with sending traffic on a link is increased congestion on that link. When designing a new VLB network, however, we assume that we are only given the set of nodes, so the VPN cost model is inadequate for our purposes. The VPN cost model does not take into account the distance between nodes, nor does it take into account economies of scale that occur because of different cable types. For instance, the cost per Megabit (Mb) of bandwidth used on a cable that can handle 100 Mb/s is the same, whether it carries 1 Mb or 100 Mb; however, the cost of sending 101 Mb/s of traffic on this link is infinite because of the physical limitations of the cable. As such, we eschew this cost model and instead use the fixed-charge cost model, which is suitable for designing a physical network.

The fixed-charge network design cost model We require a cost model that is flexible and provides accurate cost estimates. In the fixed-charge cost model, we are given a set of cable types, each with a maximum capacity, that can be used to connect nodes. For each pair of nodes i and j, the network planner estimates the cost of installing the link (i, j) with each cable type. This estimate could be as simple as multiplying the cost per unit length of a cable with the distance between i and j, or could be more sophisticated, e.g., one's cost estimate could take into account the type of terrain the link will traverse (installing a link across rugged mountains is more costly than across a flat plain). Anytime we describe a network design problem using the fixed-charge cost model, let $F = V \times V$ be the set of all candidate links such that each $e \in F$ has a cost of installation $\text{cost}(e)$ and a maximum capacity $c(e)$.

A. Designing a new VLB network

We now show how to design a new VLB network under the fixed-charge cost model, that is, we give an integer program (IP) formulation of the *VLB network design problem*, which takes as input a set of nodes V, each with a rate r_i, and a set of candidate links F where each link has a maximum capacity $c(e)$ and a fixed cost $\text{cost}(e)$. A solution to the VLB network design problem is a network $G = (V, E)$ where $\sum_{e \in E} \text{cost}(e)$ is minimal among all possible networks whose link set is a subset of F that can serve all VTMs.

The VLB network design problem is easily seen to be NP-hard. It can be reduced to the generalized Steiner tree problem and the knapsack problem, see, e.g., [5].

The following IP solves the VLB network design problem. This IP is an adaption of [15]'s LP to find the minimal assignment of link capacities for a VLB network. We use $N^+(i)$ to denote i's outgoing links and $N^-(i)$ to denote its incoming links. For a commodity k, we denote its source node by $s(k)$ and it's destination node by $d(k)$. Here, $\text{cost}(e)$ is the cost of routing a unit of traffic on link e. Recall that $\mathcal{W}_{\text{VLB}}$ is the multicommodity flow that expresses VLB's routing demands and that $f_k(e)$ is the amount of traffic placed on link e by commodity k.

VLB Network Design IP

$$\text{Minimize} \quad \sum_{e \in F} \text{cost}(e) x(e)$$

subject to

$$\sum_{e \in N^+(i)} f_k(e) = \sum_{e \in N^-(i)} f_k(e) \quad \forall i \neq s(k), d(k), \forall k \in \mathcal{W}_{\text{VLB}} \tag{2}$$

$$\sum_{e \in N^+(i)} f_k(e) = \alpha_{s(k)} r_{d(k)} + \alpha_{d(k)} r_{s(k)} \quad i = s(k), \forall k \in \mathcal{W}_{\text{VLB}} \tag{3}$$

$$\sum_{i \in V} \alpha_i = 1 \tag{4}$$

$$x(e) \in \{0, 1\} \quad \forall e \in F \tag{5}$$

The objective function here minimizes the cost of links that are selected for use. The indicator variable $x(e)$ for each link indicates whether or not e is selected for use, i.e., if $x(e) = 1$, then $e \in E$. These integer constraints are expressed in (5). Inequalities (2–3) solve the multicommodity flow problem, taking into account $\alpha_1, \ldots, \alpha_n$.

As this is an IP, its runtime is exponential in the worst-case. This is a particularly difficult IP, so we will now discuss methods of computing a solution to it.

Computing the VLB network design IP

The VLB network design IP (NDIP) a computationally intense problem. A useful heuristic to speed the computation is to find an approximate solution and seed the VLB NDIP with this solution. For the approximate problem, we suggest the fixed-charge network design problem (FCDP). The FCDP returns a minimal cost solution to a candidate network $G = (V, F)$, labeled as described by the fixed-charge cost model.

The VLB network design problem is a slight generalization of the FCDP. The FCDP does not take into account the balancing parameters $\alpha_1, \ldots, \alpha_n$, however. To transform a VLB NDIP input $G = (V, F)$ into an instance of the FCDP, one must specify fractional values for $\alpha_1, \ldots, \alpha_n$. We suggest setting $\alpha_i = r_i / R_V$ for all $i \in V$. Choosing $\alpha_1, \ldots, \alpha_n$ in this way is guaranteed to result in a network with a max throughput that is at least 1/2 the throughput of the optimal routing scheme (see the proof of Theorem 1 in [16]).

The advantage of using the fixed charge network design problem to approximate VLB network design is that techniques exist for solving the fixed charge network design problem exactly. Fixed-charge network design has been the subject

of much work, see surveys [6], [7] for instance. Increased computational power has allowed for larger instances of the FCDP to be solved. As far back as 1999, fixed charge network design problems on networks with 200 nodes and over 10,000 links could be solved near-optimally by heuristics [12]. More recently, the authors of [24] considered a generalization of the fixed charge network design problem, and used a branch-cut-and-price algorithm to obtain optimum solutions to the generalized problem on networks with over 300 nodes.

B. Upgrading an existing network

We now show how to find a min-cost upgrade to an existing network so that it can serve all VTMs with VLB—a problem we call the *VLB upgrade problem*. The VLB upgrade problem takes as input a network $G = (V, E)$ with link capacities $c(e)$ for all $e \in E$ and rates for each node $r_1, \ldots, r_n$ and a set F of candidate links, each $e \in F$ with a max capacity $c(e)$ and cost $\text{cost}(e)$ to install. Presumably G is an existing network that does not have enough capacity to serve all VTMs with VLB.

The VLB upgrade problem can be solved by reformulating it as a VLB network design problem. The idea is to, for each link $e \in E$, add e to the set of candidate links F with $c(e)$ set to e's existing capacity and $\text{cost}(e) = 0$ so that e is free to use. This way, all existing link capacity is free to use.

This approach allows one to upgrade an existing link by increasing its capacity. For instance, if a trunk has several strands of dark fiber, lighting that fiber may be considerably less expensive than installing a new trunk between its end-points.

C. Designing a fault-tolerant VLB network

We now show how to design a k link resilient VLB network. Ideally, we would like to be able to specify the necessary capacity of each link in the network so that the network can serve all VTMs with up to k link failures; unfortunately, Theorem 1 cannot be used to find such a bound for individual links—only cuts, which typically contain many links. We can, however, use it to find a sufficient capacity of each link.

Theorem 4 (Sufficient link capacity under link failures)**.** *Let $G = (V, E)$ be a heterogeneous network with node rates $r_1, \ldots, r_n$ that uses VLB with multi-path routing and load balancing parameters $\alpha_1, \ldots, \alpha_n$. If each link $e \in E$ has*

$$c(e) \geq \frac{g(S)}{|\delta(S)| - k}$$

for all $S \subseteq V$ where $e \in \delta(S)$ and $g(S) = A_{V-S}R_S + A_S R_{V-S}$, then G can serve all valid traffic matrices with up to k link failures that do not disconnect G.

Proof: Assume that, for each link $e \in E$, $c(e) \geq g(S)/(|\delta(S)|-k)$ for all $S \subseteq V$ containing e. Let $(S, V-S)$ be a cut of G and let $l = |\delta(S)|$. We have $c(S) \geq \sum_{i=1}^{l} c(e_i) = l \cdot g(S)/(l-k)$. Suppose that k links in $\delta(S)$ fail. After this failure, we have $c(S) \geq (l-k) \cdot g(S)/(l-k) = g(S)$. Since $c(S) \geq g(S)$, by Theorem 1 G can serve all VTMs. ∎

An immediate consequence of this lemma is that adding following set of constraints to the VLB network design IP ensures that the resulting network is k link resilient.

$$c(e) \geq \frac{g(S)}{|\delta(S)| - k} \quad \text{for all } (S, V-S) \text{ where } e \in \delta(S) \quad (6)$$

While Theorem 4 implies that constraints (6) are enough to guarantee that the network found by the VLB network design IP with constraints (6) added is k link resilient; however, there is no guarantee that the resulting network will be optimal in terms of link capacity.

There is exponentially many constraints in (6), making the IP with these constraints impractical to compute except on very small problem instances. As an alternative approach, we could find a minimum capacity k link resilient network by adding additional constraints to the VLB network design IP to ensure that the capacity of all cuts remains at least $g(S)$ when any set of k links are removed from the network. This approach finds an optimal network in terms of link capacity; however, it's complexity grows exponentially in k, as the number of subsets of k links grows exponentially in k.

VI. Related Work

We have already discussed closely related work in Sections I and II. Our work here continues the line of research which aims to design networks that can serve any traffic matrix, an area of study that rose to importance after Duffield et al. introduced the hose model for provisioning a VPN [8]. Much work has followed on provisioning a VPN, including optimal multi-path direct routing [9] and optimal single-path direct routing [10].

Theoretical work on oblivious routing was initiated by Valiant and Brebner who described how to efficiently serve packets on a hypercube [26]. Their scheme maximizes throughput, and is $O(\log n)$-competitive with respect to the offline routing, i.e., the best possible throughput that can be obtained by an offline algorithm is at most a logarithmic factor higher than their oblivious algorithm. Oblivious routing on arbitrary undirected graphs was studied by Räcke who described an oblivious routing scheme with a polylogarithmic competitive ratio [22]. His algorithm to find such a routing took exponential time; later work has found a polynomial-time algorithm to construct the optimal oblivious routing [3].

Previous work has found VLB to have the following advantages over direct routing.

- VLB uses network resources efficiently—on a full-mesh topology, VLB is optimal in terms of the total link capacity needed to serve all VTMs [27]; on an arbitrary topology, the throughput of multi-path VLB is at least 1/2 the throughput of the optimal routing scheme, which can adapt to the current traffic matrix [16]
- VLB is optimal in terms of required capacity to serve all VTMs when nodes and/or links can fail [4], [29]
- Optimal multi-path VLB routing can be precomputed for a set of router [14] or link [17] failure scenarios

- VLB packets can be touched by exactly 1 router and then remain in the optical layer for the remainder of transit [13]
- Can be used to efficiently load-balance traffic over peering links [30]

We do not expound on these strengths of VLB. More details can be found in [18].

We view our work as complementary to Kodialam et al.'s striking result that VLB has throughput at least 1/2 that of the optimal scheme [16]. Here, however, we study the total capacity VLB needs to serve all traffic matrices, rather than VLB's throughput, and in the worst-case at least, VLB's required capacity is not within a constant factor of optimal.

VII. Conclusions

The optimal load balancing realization has yet to be discovered. VLB doubles the round trip time of packets in the worst-case and, as we've shown here, can require more capacity than shortest path routing. VLB is especially ineffective on sparse topologies, i.e., topologies with a low ratio of links to nodes. However, we've shown that as the density of a topology increases, VLB's worst-case required capacity decreases linearly with the number of links beyond the first $n-1$ links required to connect all nodes. Our future work includes determining classes of topologies where VLB requires less capacity than shortest path routing.

Theorem 1, which characterizes the capacity of cuts in a VLB network, is the power behind these VLB provisioning results. We view this theorem as a step towards understanding the structure of VLB networks. Unfortunately, Theorem 1 is not extendable to case where the ingress rate is not equal to the egress rate at each node. Theorem 1 is also not extendible to generalized load balancing parameters [15], which allow ingress traffic to be load-balanced based on both its source and destination nodes, i.e., there is a α_{ij} for each pair of nodes $i, j \in V$ which specifies the fraction of i's ingress traffic that it load-balances to j. Further, no similar theorem exists for direct routing because we cannot guarantee that nodes in S always send traffic to nodes in $V - S$ at a certain rate.

We have also shown that the predictable nature of traffic in a VLB network allows for VLB network design problems to be accurately stated and computed. VLB is a powerful network design framework, and we've shown it can facilitate network design so that operators have rigorous tools, rather than best practices, available for designing and extending their networks.

References

[1] A.R. Abdelaziz. A new approach for enumerating minimal cut-sets in a network. *The 7th IEEE International Conference on Electronics, Circuits and Systems (ICECS '00)*, 2000.

[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[3] Y. Azar, E. Cohen, A. Fiat, H. Kaplan, and H. Räcke. Optimal oblivious routing in polynomial time. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing (STOC '03)*, pages 383–388, 2003.

[4] M. Babaioff and J. Chuang. On the optimality and interconnection of valiant load-balanced networks. In *IEEE Infocom*, 2007.

[5] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. McGraw Hill, Boston, 2001.

[6] Alysson M. Costa. A survey on benders decomposition applied to fixed-charge network design problems. *Computers & Operations Research*, 32(6):1429–1450, 2005.

[7] T. G. Crainic, A. Frangioni, and B. Gendron. Bundle-based relaxation methods for multicommodity capacitated fixed charge network design. *Discrete Applied Mathematics*, 112(1–3):73–99, 2001.

[8] N. G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. K. Ramakrishnan, and J. E. van der Merive. A flexible model for resource management in virtual private networks. In *ACM SIGCOMM*, 1999.

[9] Thomas Erlebach and Maurice Rüegg. Optimal bandwidth reservation in hose-model VPNs with multi-path routing. In *IEEE INFOCOM*, 2004.

[10] N. Goyal, N. Olver, and F. B. Shepherd. The VPN conjecture is true. In *Proceedings of the 40th annual ACM symposium on Theory of computing (STOC '08)*, 2008.

[11] L. Khachiyan, E. Boros, K. Elbassioni, V. Gurvich, and K. Makino. Enumerating disjunctions and conjunctions of paths and cuts in reliability theory. *Discrete Appl. Math.*, 155(2):137–149, 2007.

[12] Dukwon Kim and Panos M. Pardalos. A solution approach to the fixed charge network flow problem using a dynamic slope scaling procedure. *Operations Research Letters*, 24(4):195–203, 1999.

[13] M. Kodialam, T. V. Lakshman, J. B. Orlin, and S. Sengupta. A versatile scheme for routing highly variable traffic in service overlays and IP backbones. In *IEEE Infocom*, 2006.

[14] M. Kodialam, T. V. Lakshman, J. B. Orlin, and S. Sengupta. Preconfiguring IP-over-optical networks to handle router failures and unpredictable traffic. *IEEE J. on Sel. Areas in Comm. (JSAC)*, 2007. An earlier version appeard in IEEE *Infocom*, 2006.

[15] M. Kodialam, T. V. Lakshman, and S. Sengupta. Efficient and robust routing of highly variable traffic. In *Third Workshop on Hot Topics in Networks (HotNets-III)*, 2004.

[16] M. Kodialam, T. V. Lakshman, and S. Sengupta. Maximum throughput routing of traffic in the hose model. In *IEEE Infocom*, 2006.

[17] M. Kodialam, T. V. Lakshman, and S. Sengupta. Throughput guaranteed restorable routing without traffic prediction. In *IEEE ICNP*, 2006.

[18] M. Kodialam, T. V. Lakshman, and S. Sengupta. Advances in oblivious routing of internet traffic. In *Performance Modeling and Engineering*. Spring, 2008.

[19] H. Nagamochi. *Studies on Multicommodity Flows in Directed Networks*. PhD thesis, Department of Applied Mathematics and Physics, Kyoto University,, 1988.

[20] Hiroshi Nagamochi and Toshihide Ibaraki. Max-flow min-cut theorem for the multicommodity flows in certain planar directed networks. *Electronics and Communications in Japan, Part 3*, 72(3):58–71, 1989.

[21] Hiroshi Nagamochi and Toshihide Ibaraki. On max-flow min-cut and integral flow properties for multicommodity flows in directed networks. *Information Processing Letters*, 31:279–285, 1989.

[22] Harald Räcke. Minimizing congestion in general networks. In *Proceedings of the 43rd Symposium on Foundations of Computer Science (FOCS '02)*, 2002.

[23] Y. Shen. A new simple algorithm for enumerating all minimal paths and cuts of a graph. *Microelectronics and Reliability*, 35(6):973–976, 1995.

[24] T. Thomadsen and T. Stidsen. The generalized fixed-charge network design problem. *Computers and Operations Research*, 34(4):997–1007, 2007.

[25] L. G. Valiant. A scheme for fast parallel communication. *SIAM J. Comput.*, 11(2):350–361, 1982.

[26] L. G. Valiant and G. J. Brebner. Universal schemes for parallel communication. In *Proceedings of the thirteenth annual ACM symposium on Theory of computing (STOC '81)*, 1981.

[27] R. Zhang-Shen and N. McKeown. Designing a predictable internet backbone network. In *Third Workshop on Hot Topics in Networks (HotNets-III)*, 2004.

[28] R. Zhang-Shen and N. McKeown. Designing a predictable internet backbone with Valiant load-balancing. In *Thirteenth International Workshop on Quality of Service (IWQoS '05)*, 2005.

[29] R. Zhang-Shen and N. McKeown. Designing a fault-tolerant network with Valiant load-balancing. In *IEEE Infocom Mini-Conference*, 2008.

[30] R. Zhang-Shen and N. McKeown. Guaranteeing quality of service to peering traffic. In *IEEE Infocom*, 2008.