Capacity Provisioning a Valiant Load-Balanced Network

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

***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 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 and installs—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

Recent work on *oblivious routing*—where all routes are static and do not change depending on congestion in the network—has suggested Valiant load balancing (VLB) as an alternative to direct routing [8], [11]. 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 [11], and this optimality remains when nodes can fail [2]. 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—these results are given in Section IV. 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. These results rely on a theorem we give in Section III that characterizes the necessary and sufficient capacity of links crossing each cut, or partition, of a network's nodes. This theorem also gives an algorithm to check if an existing network can serve all traffic matrices with VLB when a fixed number of links fail.

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. Unlike current routing practices, VLB routes traffic in a predictable fashion—using VLB, nodes are not required to update forwarding paths 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) formulation that designs a minimum-cost VLB network under the fixed-charge cost model. We propose this VLB network design approach in Section V.

II. Models and Definitions

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* [4], 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 i 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. As observed in [8], it is enough to consider only the traffic matrices where each node sends and receives at its maximum rate, i.e., for a traffic matrix D, we have $\sum_{j \in V, j \neq i} D_{ij} = r_i$ and $\sum_{j \in V, j \neq i} D_{ji} = r_i$, 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—we use it here for theoretical analysis.

In order to define the two variants of VLB we consider, we need some terminology. 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 simple 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 $n = |V|$. 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.

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 using multi-path VLB, can be found in polynomial-time using an LP [9]. It's NP-hard to find a feasible single-path VLB routing; however, there is a fully polynomial-time approximation algorithm for the problem [7]. Given optimal values for $\alpha_1, \ldots, \alpha_n$, 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 where 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 $c(S) = \sum_{e \in \delta(S)} c(e)$.

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: All proofs have been omitted due to space constraints. See the full version of this paper [3] for details. ■

In the proof of Theorem 1, we give a polynomial-time algorithm to find the max multicommodity flow for VLB networks with a fixed $\alpha_1, \ldots, \alpha_n$. As it cannot find optimal values for $\alpha_1, \ldots, \alpha_n$, this algorithm is still dependent on the LP of [9] to find optimal settings for these load balancing parameters. We note that if one solves the LP of [9], it returns a solution to the multi-path VLB routing problem, so solving the VLB routing problem with the algorithm given in the proof of Theorem 1 is redundant.

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 now give a simple algorithm, based on Theorem 1, 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. 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 $c(S) - \sum_{i=1}^{k} c(e_i) < g(S)$ for any cut $(S, V - S)$ of V, then G cannot serve all VTMs under k link failures.

The worst-case runtime of this algorithm is exponential since a network can have exponentially many cuts. Even so, the algorithm is practical for small networks. We enumerated the cuts of random networks with size $n = 20$ in less than a minute with a naive Python script; more sophisticated techniques exist for larger networks [1], [6], [10].

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 use SVLB, where $\alpha_1 = \cdots = \alpha_n$, no matter the topology since it is easier to analyze than VLB. Similar 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 $L_{VLB}(G') \leq L_{VLB}(G)$.*

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 [5]. 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.*

We can now compute the worst-case capacity for SVLB, which is:

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

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 [11] 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 or a ring.

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 $C_{n,l}$ has $2ln$ links. Computing the required capacity for $C_{n,l}$, we have

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

when n is even and $k < n/2$, showing that VLB performs very well as the number of links increases. While the full $n(n-1)/2$ links is necessary to obtain the optimality results of VLB, we have shown that VLB does not require too much additional capacity on networks with a moderate number of links vs. nodes.

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	$2(n^2-1)r/3$	$2r(n-1)$ [11]
topology	path	full-mesh
Shortest path	$\geq rn(n-1)$	$2r(n-1)$ [12]
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. Unlike previous work on provisioning a VLB network, we use the fixed-charge cost model (described below). 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.

The fixed-charge network design cost model 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.

Due to space constraints, we do not present the full VLB network design IP here—see the full version for details [3]. The goal of the VLB network design IP is to minimize the following objective function.

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

This objective function 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 $e \in E$ and hence e is in the min-cost network that can serve all VTMs with VLB.

As this is an IP, its runtime is exponential in the worst-case. This is a particularly difficult IP—in the full version of this paper, we discuss techniques to compute it.

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 output to the VLB upgrade problem

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.

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.

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.

$$c(e) \geq \frac{g(S)}{|\delta(S)| - k} \quad \text{for all } (S, V - S) \text{ where } e \in \delta(S) \quad (1)$$

While Theorem 4 implies that constraints (1) are enough to guarantee that the network found by the VLB network design IP with constraints (1) 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 (1), 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. Conclusions

The optimal load balancing scheme has yet to be found. 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; 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. 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.

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

[3] A. R. Curtis and A. López-Ortiz. Capacity provisioning a Valiant load-balanced network. Technical Report CS-2009-02, University of Waterloo, 2009.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

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

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

[12] 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.