Chapter 2
Valiant Load-Balancing: Building Networks That Can Support All Traffic Matrices

Rui Zhang-Shen

Abstract This paper is a brief survey on how Valiant load-balancing (VLB) can be used to build networks that can efficiently and reliably support all traffic matrices. We discuss how to extend VLB to networks with heterogeneous capacities, how to protect against failures in a VLB network, and how to interconnect two VLB networks. For the readers' reference, included also is a list of work that uses VLB in various aspects of networking.

2.1 Introduction

In many networks the traffic matrix is either hard to measure and predict, or highly variable over time. In these cases, using Valiant load-balancing (VLB) to support all possible traffic matrices is an attractive option. For example, even though the traffic in the Internet backbone is extremely smooth due to high level of aggregation, it is still hard to measure. Accurately measuring the traffic matrix (e.g., using NetFlow) is too expensive to do all the time, and standard methods using link measurements give errors of 20% or more. Even if the current traffic matrix is satisfactorily obtained, extrapolating it to the future is fraught with uncertainty, due to the unpredictable nature of Internet traffic growth. Finally, since Internet traffic is dynamic, the traffic matrix can deviate from its normal values at any time, possibly causing congestion.

The traffic demand seen by a network can be represented by the *traffic matrix*, which indicates the rates at which each node initiates traffic to every other node. We say a network can *support* a traffic matrix if for every link in the network, the load caused by the traffic matrix is less than the capacity of the link. When a network cannot support the traffic matrix presented to it, at least one link in the network has a load higher than its capacity. Congestion occurs and backlog in the buffer builds up on the congested link(s), causing packet drops, increased delay, and high variations

R. Zhang-Shen (✉)
Google, Inc., New York, NY 10011
e-mail: ruizhangshen@gmail.com

G. Cormode and M. Thottan (eds.), *Algorithms for Next Generation Networks*,
Computer Communications and Networks, DOI 10.1007/978-1-84882-765-3_2,

in delay. Ideally, we would like to design a network that can support a wide range of traffic matrices, so that congestion occurs only rarely or not at all.

In this paper, we discuss the use of VLB in building networks that can efficiently support *all* traffic matrices which do not over-subscribe any node. We first briefly survey the wide use of VLB in various aspects of networking, and describe the basic scenario of using VLB in a network. Section 2.2 extends VLB from a homogeneous setting to networks with arbitrary capacities, Section 2.3 describes how to protect against and recover quickly from failures in a VLB network, and Section 2.4 proposes to use VLB to route traffic between two networks. Finally Section 2.5 discusses possible future work.

2.1.1 The Wide Use of VLB

In the early 1980s, Valiant [19] first proposed the scheme of routing through a randomly picked intermediate node en route to a packet's destination. He showed that in an N-node binary cube network, given any permutation traffic matrix, the distributed two-phase randomized routing can route every packet to its destination within $O(\log N)$ time with overwhelming probability. This was the first scheme for routing arbitrary permutation in a sparse network in $O(\log N)$ time. Since then, such randomized routing has been used widely, and is often referred to as (VLB), *randomized load-balancing*, or *two-phase routing*. VLB has many good characteristics. It is decentralized, where every node makes local decisions. This also makes the scheme scalable. VLB is agnostic to the traffic matrix because the randomness erases the traffic pattern, and different traffic matrices can result in the same load on the links.

Soon after its invention, was used in other interconnection networks for parallel communication, to improve delivery time [1], and to relieve effects of adverse traffic patterns [13]. In recent years, it was adapted for routing in torus networks [17, 18] in order to provide worst-case performance guarantees without sacrificing average-case performance. The key is to use VLB adaptively, based on the observation that under low load, load-balancing only a small amount of traffic is sufficient to avoid congestion.

VLB is also used in building network switches with great scalability and performance guarantee, without the need of a centralized scheduler. It was used in ATM switches [7], routers [4, 5], optical routers [3, 9], and software routers [2]. In particular, the scheme was rediscovered for designing router switch fabrics [4] to mitigate routers' scaling challenges, because it was difficult for centralized schemes to keep up with the increasing link speed. In this context, it was shown that splitting traffic in a round-robin fashion has the same effect on link load as random splitting [4], and that is the most efficient in terms of the total required interconnection capacity for supporting all traffic matrices [8].

Almost simultaneously, several groups independently applied the idea of VLB to traffic engineering and network design for the Internet, in order to efficiently sup-

port all possible traffic matrices. Kodialam et al.'s two-phase routing [11, 12] is a traffic engineering method, where a full mesh of tunnels are set up over fixed capacity links and packets are sent in two phases (i.e., two hops) in the network. Winzer et al.'s selective randomized load-balancing [14, 16, 21] used VLB and its variants to design cost-effective optical networks. Their model assumes that a link's cost includes both the fiber and the terminating equipment, so there is incentive for having fewer links. In the optimal design, traffic is load-balanced only to a few intermediate nodes. Zhang-Shen and McKewon [23, 25] proposed using VLB over a logical full mesh in a backbone network to support all traffic matrices and to quickly recover from failures. In addition, VLB was used as an optical routing strategy in Ethernet LAN [20], for scheduling in metro area WDM rings [10], for circuit-switched networks [22], and for scaling and commoditizing data center networks [6].

A study on the queueing properties of a VLB network [15] found that VLB eliminates congestion in the network, and pseudo-random (e.g., round-robin) load-balancing reduces queueing delay. VLB was also shown to eliminate congestion on peering links when used to route traffic between networks [26].

2.1.2 A Simple VLB Network

Consider a network of N nodes, each with capacity r, i.e., a node can initiate traffic at the maximum rate of r, and can receive traffic at the same maximum rate. We assume that the network traffic satisfies such node aggregate constraint, because otherwise there is no way to avoid congestion. A logical link of capacity $\frac{2r}{N}$ is established between every pair of nodes over the physical links, as shown in Figure 2.1. We use the convention that a *flow* in the network is defined by the source node and the destination node, unless further specified. Every flow entering the network is equally split across N two-hop paths between ingress and egress nodes, i.e., a packet is forwarded twice in the network: In the first hop, an ingress node uniformly distributes each of its incoming flows to all the N nodes, regardless of

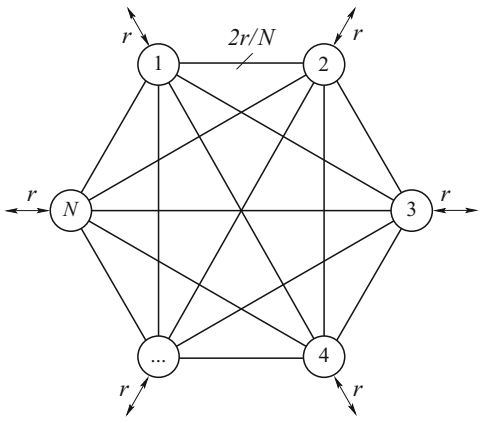

Fig. 2.1 VLB in a network of N identical nodes each having capacity r. A full mesh of logical links of capacity $2r/N$ connect the nodes

the destinations. In the second hop, all packets are sent to the final destinations by the intermediate nodes. Load-balancing can be done packet-by-packet, or flow-by-flow at the application flow level. The splitting of traffic can be random (e.g., to a randomly picked intermediate node) or deterministic (e.g., round-robin).

Assume we can achieve perfect load-balancing, i.e., can split traffic at the exact proportions we desire, then each node receives exactly $\frac{1}{N}$ of *every* flow after first-hop routing. This means, all the N nodes equally share the burden of forwarding traffic as the intermediate node. When the intermediate node happens to be the ingress or egress node, the flow actually traverses one hop (the direct link between ingress and egress) in the network. Hence, $\frac{2}{N}$ of every flow traverses the corresponding one-hop path.

Such uniform load-balancing can guarantee to support all traffic matrices in this network. Since the incoming traffic rate to each node is at most r, and the traffic is evenly load-balanced to N nodes, the actual traffic on each link due to the first-hop routing is at most $\frac{r}{N}$. The second-hop routing is the dual of the first-hop routing. Since each node can receive traffic at a maximum rate of r and receives $\frac{1}{N}$ of the traffic from every node, the actual traffic on each link due to the second-hop routing is also at most $\frac{r}{N}$. Therefore, a full-mesh network where each link has capacity $\frac{2r}{N}$ is sufficient to support all traffic matrices in a network of N nodes of capacity r.

This is perhaps a surprising result – a network where any two nodes are connected with a link of capacity $\frac{2r}{N}$ can support traffic matrices where a node can send traffic to another node at rate r. It shows the power of load-balancing. In VLB, each flow is carried by N paths, and each link carries a fraction of many flows; therefore any large flow is averaged out by other small flows. In a static full-mesh network, if all the traffic were to be sent through direct paths, we would need a full-mesh network of link capacity r to support all possible traffic matrices; therefore load-balancing is $\frac{N}{2}$ times more efficient than direct routing.

2.2 VLB in Heterogeneous Networks

Real-life networks are often heterogeneous, i.e., the nodes in a network can have different capacities. In this section we discuss how to extend the result of uniform load-balancing to heterogeneous networks [24].

We first introduce notations. In a network of N nodes, the traffic matrix $\Lambda = \{\lambda_{ij}\}$ is an $N \times N$ matrix, where the entry λ_{ij} indicates the datarate at which Node i initiates traffic destined to Node j. The traffic rate is typically averaged over a long period of time so we consider it as constant. Typically in a network there are buffers to absorb short-lived traffic fluctuations. Suppose Node i has capacity r_i, i.e., the node can initiate traffic at the maximum rate of r_i, and can receive traffic at the same maximum rate. So in order for the traffic matrix to not over-subscribe any node, it must satisfy

$$\sum_j \lambda_{ij} \le r_i, \forall i \quad \text{and} \quad \sum_j \lambda_{ji} \le r_i, \forall i. \tag{2.1}$$

Without loss of generality, assume that the nodes have been sorted according to decreasing capacities, i.e., $r_1 \geq r_2 \geq \cdots \geq r_N$, so Node 1 is the largest node and Node N the smallest. Let R be the total node capacity, i.e., $R = \sum_{i=1}^{N} r_i$. We assume $r_1 \leq \sum_{i=2}^{N} r_i$ because even if $r_1 > \sum_{i=2}^{N} r_i$, Node 1 cannot send or receive traffic at a rate higher than $\sum_{i=2}^{N} r_i$, because that would over-subscribe some nodes.

Suppose that a full mesh of logical links are set up to connect these N nodes. Let c_{ij} represent the required link capacity from Node i to Node j and C the link capacity matrix $\{c_{ij}\}$. Having $c_{ij} = 0$ means that link (i, j) is not needed. The simple homogeneous network presented in Section 2.1.2 has that $c_{ij} = \frac{2r}{N}, \forall i \neq j$, and $r_i = r, \forall i$.

In a network with identical nodes, it is natural to load-balance uniformly. But uniform load-balancing seems too restrictive in a heterogeneous network because it does not take into account the difference in node capacities. A natural solution is to load-balance proportionally to the capacity of the intermediate node, i.e., Node i receives fraction $\frac{r_i}{R}$ of every flow. This is a direct generalization from uniform multicommodity flow in the homogeneous case to product multicommodity flow [16]. The required link capacity is $c_{ij} = 2r_i r_j / R$ [24].

We can further generalize VLB to allow *any* flow splitting ratio and let some external objective to determine the optimal ratios. We introduce a set of *load-balancing parameters* p_i such that $p_i \geq 0$, for all i, and $\sum_{i=1}^{N} p_i = 1$. An ingress node splits each flow according to $\{p_i\}$ and sends p_i of every flow to Node i. This gives us the freedom of, for example, letting the larger nodes forward more traffic than the smaller nodes, or not using some of the nodes as intermediates (by setting the corresponding p_i to zero). If there are some objectives to be optimized, there are N parameters (p_i, $i = 1, 2, \ldots, N$) that can be tuned, but if more freedom is needed, we can, for example, let each flow have its own set of load-balancing parameters.

We now find the required link capacity. The first-hop traffic on link (i, j) is the traffic initiated by Node i that is load-balanced to Node j, and the rate is at most $r_i p_j$. The second-hop traffic on the link is the traffic destined to Node j that is load-balanced to Node i, and the rate is at most $r_j p_i$. Therefore the maximum amount of traffic on link (i, j) is $r_i p_j + r_j p_i$, which is also the required capacity on link (i, j)

$$c_{ij} = r_i p_j + r_j p_i. \tag{2.2}$$

The required (outgoing) interconnection capacity of Node i is

$$l_i = \sum_{j:j\neq i} c_{ij} = r_i + R p_i - 2 r_i p_i,$$

and the total required interconnection capacity of the network is

$$L = \sum_{i=1}^{N} l_i = \sum_{i,j:i\neq j} c_{ij} = 2\left(R - \sum_i r_i p_i\right). \tag{2.3}$$

Thus if we want to minimize the total required interconnection capacity of the network, we need to maximize $\sum_i r_i p_i$, subject to the constraints on p_i. The optimal solution has the form that $p_i > 0$ only if $r_i = \max_j r_j$, i.e., traffic is only load-balanced to the largest node(s). Thus we have

$$\min L = 2(R - \max_i r_i). \tag{2.4}$$

We can show that Equation (2.4) is also the minimum total interconnection capacity required by *any network* to support all traffic matrices [24], and hence is the necessary and sufficient condition. One network that minimizes the total required interconnection capacity is a "star" with Node 1 at the center, i.e., $p_1 = 1$, $p_i = 0$ for $i \geq 2$. In the case where all nodes have the same capacity, to achieve minimum L, p_i can take any non-negative values as long as they sum up to 1. Thus the uniform load-balancing presented in Section 2.1.2 is optimal in terms of total required interconnection capacity. Splitting flows proportional to node capacities, i.e., $p_i = r_i/R$, is not optimal when nodes have different capacities.

In order to use the minimum amount of interconnection capacity, only nodes with the largest capacity can act as intermediate nodes. This can be limiting, especially if only one node has the largest capacity, because a star network is not good for fault tolerance. A star network is efficient but not balanced, because the center node acts as the intermediate node for all traffic. We need a scheme that is not only efficient but also balanced, and we found one by minimizing the *network fanout*.

The *fanout* of node i is $f_i = \frac{l_i}{r_i}$, the ratio of node i's interconnection capacity to its node capacity. Since the interconnection capacity is used both for sending traffic originated from the node and for forwarding traffic for other nodes, the fanout measures the amount of responsibility the node has to forward other nodes' traffic relative to its size. If the fanouts of the two nodes are the same, then the larger node forwards more traffic, which is a desired property. Thus, to have a balanced network, we minimize the maximum fanout over all nodes, which results in all nodes having equal fanout. The resulting load-balancing parameters and fanout are

$$p_i = \frac{\frac{r_i}{R-2r_i}}{\sum_k \frac{r_k}{R-2r_k}}, \quad i = 1, 2, \ldots, N,$$
$$f_i = 1 + \frac{1}{\sum_{j=1}^{N} \frac{r_j}{R-2r_j}}, \quad \forall i.$$

The optimal load-balancing parameters are almost proportional to the node capacities: $p_i \propto \frac{r_i}{R-2r_i}$. The parameter p_i is a strictly increasing function of r_i. Therefore a larger node has greater responsibility for forwarding traffic. We can further show that the total interconnection capacity used in this scheme is no more than $\frac{1}{2}(\sqrt{2}+1) = 1.207$ times the minimum total capacity [24]. Thus, the scheme of minimizing maximum fanout is not only balanced, but also efficient.

One can of course choose to optimize other criteria that are more suitable for the particular situation. Kodialam et al. [11] assume that the underlying physical links have fixed capacities, constraining the logical link capacities. They give efficient

algorithms for calculating or approximating the optimal load-balancing ratios which allow the network to support the most traffic. Shepherd and Winzer [16] use a realistic cost model that takes into account both the fiber cost and the equipment cost. They opt to load-balance to only a subset of the nodes so that a full mesh is not needed.

2.3 Fault-Tolerance in a VLB Network

Most networks need to accommodate planned and unplanned interruptions. As we show in this section, VLB networks, due to their rich connectivity and path diversity, have many advantages in terms of tolerating failures, such as

- All of the working paths between a pair of nodes are used all the time, and flows are load-balanced across all working paths. Most other schemes require protection paths that are idle during normal operation.
- All paths are used all the time, so there is no need to set up new paths upon failure.
- In order to protect against k failures,[1] the fraction of extra capacity required is approximately $\frac{k}{N}$. This is extremely efficient compared to other fault tolerance schemes.
- VLB naturally protects against multiple failures. One can decide during design what failure scenarios the network should tolerate, such as k arbitrary link or node failures or a particular set of failure patterns.

When there are no failures, each flow is load-balanced over all the N paths according to some ratio. The same mechanism can be used when there are failures and some flows have fewer than N working paths. We assume that a node keeps track of the *available* paths for each flow originating from it, and load-balances each flow over the available paths, according to the same ratios. The network is self-healing because the paths are known prior to failures and traffic can be rerouted as soon as failure is detected.

We derive how much link capacity is needed so as to support all traffic matrices under k_n arbitrary node failures[2] and k_l arbitrary link failures. For simplicity, we assume the *simple* network model where all nodes have the same capacity r. Due to the symmetry of the topology, all links in the fault-tolerant network have the same capacity requirement, represented by $C(N, k_n, k_l)$. We first consider node failures and link failures separately and then combine them. By definition, $C(N, 0, 0) = \frac{2r}{N}$.

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

[1] We focus on failures in the logical topology, and since several logical links can share a physical link, a physical failure can correspond to multiple logical failures.

[2] If a node fails, we discard the traffic originating from or terminating at this node.

for supporting all traffic matrices. In general, when there are k node failures, the network becomes an $(N-k)$-node full mesh, so the link capacity required to tolerate k node failures is

$$C(N,k,0) = C(N-k,0,0) = \frac{2r}{N-k}. \tag{2.5}$$

Link failures are a little more complicated, as they can destroy the symmetry of the topology, so that it is no longer a full mesh. We only consider the worst-case failure scenarios (adversarial link failures) here. We omit the details and only give the final result. The amount of capacity required on each link to tolerate k arbitrary link failures, for $1 \le k \le N-2$, is

$$C(N,0,k) = \begin{cases} \frac{r}{N-2} + \frac{r}{N} & k = 1 \\ \frac{r}{N-k-1} + \frac{r}{N-1} & k = 2 \text{ or } N-2, \text{or } N \le 6 \\ \frac{2r}{N-k} & \text{otherwise.} \end{cases} \tag{2.6}$$

For $k \ge N-1$, the network becomes disconnected in the worst-case failure scenario, therefore we cannot guarantee a congestion-free network. The significance of Equation (2.6) is that all entries on the right-hand side are very close to $\frac{2r}{N-k}$. Figure 2.2 plots Equation (2.6) for $N = 50$, and as we can see, the curve is very flat for small values of k.

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

$$C(N,k_n,k_l) \approx \frac{2r}{N-k_n-k_l} = \frac{2r}{N-k}, \tag{2.7}$$

where $k = k_n + k_l$. This means that the curve for Equation (2.7) is roughly the same as that for Equation (2.6), shown in Figure 2.2. So we conclude that in a VLB network, a small amount of over-provisioning goes a long way to make the network

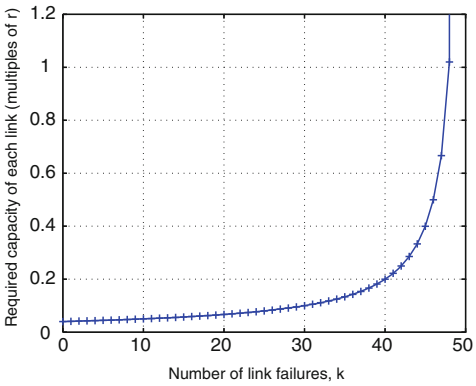

Fig. 2.2 The required link capacity vs. the number of link failures in a 50-node network

fault tolerant. For example, if the links in a 50-node network are over-provisioned by just about 11%, the network can tolerate any five (node or link) failures.

2.4 VLB for Peering Traffic

Today, most congestion in Internet backbone takes place on the peering links connecting them. When a link between two networks is congested, very likely some other peering links between these two networks are lightly loaded. That is, the peering links between two networks are usually not evenly utilized. We propose to use a technique similar to VLB to route peering traffic, so as to eliminate congestion on the peering links. We show that if traffic is load-balanced over all the peering links between two networks, there will be no congestion as long as the total peering capacity is greater than the total peering traffic. Even though the two networks using VLB to route their peering traffic do not need to use VLB internally, we assume they do and analyze how the peering scheme affects how the networks run VLB.

Suppose two VLB networks are connected by a subset of their nodes (the peering nodes), as shown in Figure 2.3. For the ease of description, we use the same numbering for the peering nodes in both networks. (Note that this convention is different from Section 2.2.) The traffic exchanged between the two networks is called *peering traffic* and assume the total amount is no more than R_p in each direction.

In the network of N nodes, we introduce the *peering load-balancing parameters* $q_i, i = 1, 2, \ldots, N$, such that a portion q_i of the peering traffic between the two networks is exchanged at node i. Naturally, $q_i = 0$ if i is not a peering node.

The peering load-balancing parameters, q_i, together with the maximum peering traffic between the two networks, R_p, determine the sizes of the peering links: the required capacity of the peering link at node i is $R_p q_i$. Suppose the peering links have the required capacities, then if the peering traffic is load-balanced across the peering links according to the proportions q_i, and the total amount of peering traffic between the two networks does not exceed R_p, there will be no congestion on the peering links.

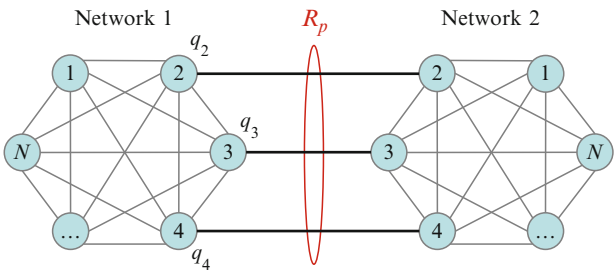

Fig. 2.3 Two VLB networks connect at a set of peering nodes. The total amount of traffic exchanged between the two networks is no more than R_p, and a portion q_i of the peering traffic is exchanged at node i.

The extra requirement of routing peering traffic may result in higher capacity requirements inside the networks. If we treat the peering traffic that originates from the network as traffic destined to the peering nodes, and the peering traffic that enters the network as traffic originated from the peering nodes, then the peering traffic may have to traverse two hops in each network.

Alternatively, we can load-balance peering traffic over the peering points only, instead of all the nodes. Thus, we require that peering traffic traverses at most one hop in each network, and at most two hops altogether.

Suppose the peering load-balancing parameters are fixed, for example, through negotiation between the two networks. Then we can vary p_i, i.e., how non-peering traffic is routed internally, to minimize the required link capacities. We observe that R_p is likely to be bigger than the node capacities r_i. R_p is the total amount of traffic the two network exchanges and can be a large portion of the network's total traffic R, while the node capacities are likely to make up only a small fraction of R, on the order of $\frac{R}{N}$.

If we assume that $R_p \geq r_i$ for all i, then we have

$$c_{ij} = r_i \max(p_j, q_j) + r_j \max(p_i, q_i),$$

and the minimum c_{ij} is achieved for all links when $p_i = q_i$ for all i. So the optimal capacity allocation in a network with peering traffic is

$$c_{ij} = r_i q_j + r_j q_i. \tag{2.8}$$

Since q_i is zero if node i is a non-peering node, $c_{ij} = 0$ if both Node i and Node j are non-peering nodes. The network is now a two-tiered one: in the center is a full mesh connecting the peering nodes, and on the edge are the non-peering nodes, each connected to all the center nodes.

Setting local load-balancing parameters to be the same as peering load-balancing parameters means that only the peering nodes will serve as intermediate nodes to forward traffic. Peering nodes are often the largest nodes in the network, so they should have larger responsibilities in forwarding traffic. The analysis shows that the optimal way is to *only* let the peering nodes forward traffic. This has the additional benefits of requiring fewer links and reducing network complexity.

2.5 Discussions

Using VLB to route traffic in a network has many advantages, such as efficiency and fast failure recovery. There are some open questions as well. Some examples are

- Sending packets through two hops may increase the propagation delay, while sending packets through the direct link may cause increased capacity requirement. There should be a tradeoff between packet delay and required capacity.

- Different paths that a flow traverses may have different delays, and hence packets in the flow may not arrive at the destination in the original order. This is usually not a problem in the Internet, since a flow may consist of many application-level flows, and can be split accordingly for load-balancing. But it can be a problem if a flow cannot be easily split.
- It is unclear whether VLB can be incrementally deployed in a network.

Despite the challenges, VLB is rapidly gaining popularity in networking research. This survey paper can be a starting point for those interested in applying VLB to other situations.

References

1. R. Aleliunas. Randomized parallel communication (preliminary version). In *PODC '82: Proceedings of the first ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, pages 60–72, 1982.
2. K. Argyraki, S. Baset, B.-G. Chun, K. Fall, G. Iannaccone, A. Knies, E. Kohler, M. Manesh, S. Nedevschi, and S. Ratnasamy. Can software routers scale? In *PRESTO '08: Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow*, pages 21–26, 2008.
3. P. Bernasconi, J. Gripp, D. Neilson, J. Simsarian, D. Stiliadis, A. Varma, and M. Zirngibl. Architecture of an integrated router interconnected spectrally (IRIS). *High Performance Switching and Routing, 2006 Workshop on*, pages 8 pp.–, June 2006.
4. C.-S. Chang, D.-S. Lee, and Y.-S. Jou. Load balanced Birkhoff-von Neumann switches, Part I: One-stage buffering. *Computer Communications*, 25(6):611–622, 2002.
5. C.-S. Chang, D.-S. Lee, and C.-M. Lien. Load balanced Birkhoff-von Neumann switches, Part II: Multi-stage buffering. *Computer Communications*, 25(6):623–634, 2002.
6. A. Greenberg, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. Towards a next generation data center architecture: scalability and commoditization. In *PRESTO '08: Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow*, pages 57–62, 2008.
7. M. Henrion, K. Schrodi, D. Boettle, M. De Somer, and M. Dieudonne. Switching network architecture for ATM based broadband communications. *Switching Symposium, 1990. XIII International*, 5:1–8, 1990.
8. I. Keslassy, C.-S. Chang, N. McKeown, and D.-S. Lee. Optimal load-balancing. In *Proc. IEEE INFOCOM*, 2005.
9. 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.
10. I. Keslassy, M. Kodialam, T. Lakshman, and D. Stiliadis. Scheduling schemes for delay graphs with applications to optical packet networks. *High Performance Switching and Routing (HPSR)*, pages 99–103, 2004.
11. 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 *Proc. IEEE INFOCOM*, April 2006.
12. M. Kodialam, T. V. Lakshman, and S. Sengupta. Efficient and robust routing of highly variable traffic. In *HotNets III*, November 2004.
13. D. Mitra and R. A. Cieslak. Randomized parallel communications on an extension of the omega network. *J. ACM*, 34(4):802–824, 1987.
14. H. Nagesh, V. Poosala, V. Kumar, P. Winzer, and M. Zirngibl. Load-balanced architecture for dynamic traffic. In *Optical Fiber Communication Conference*, March 2005.

15. R. Prasad, P. Winzer, S. Borst, and M. Thottan. Queuing delays in randomized load balanced networks. In *Proc. IEEE INFOCOM*, May 2007.
16. F. B. Shepherd and P. J. Winzer. Selective randomized load balancing and mesh networks with changing demands. *Journal of Optical Networking*, 5:320–339, 2006.
17. A. Singh. *Load-Balanced Routing in Interconnection Networks*. PhD thesis, Department of Electrical Engineering, Stanford University, 2005.
18. A. Singh, W. J. Dally, B. Towles, and A. K. Gupta. Locality-preserving randomized oblivious routing on torus networks. In *SPAA '02: Proceedings of the fourteenth annual ACM symposium on parallel algorithms and architectures*, pages 9–13, 2002.
19. L. G. Valiant. A scheme for fast parallel communication. *SIAM Journal on Computing*, 11(2):350–361, 1982.
20. R. van Haalen, R. Malhotra, and A. de Heer. Optimized routing for providing Ethernet LAN services. *Communications Magazine, IEEE*, 43(11):158–164, Nov. 2005.
21. P. J. Winzer, F. B. Shepherd, P. Oswald, and M. Zirngibl. Robust network design and selective randomized load balancing. *31st European Conference on Optical Communication (ECOC)*, 1:23–24, September 2005.
22. R. Zhang-Shen, M. Kodialam, and T. V. Lakshman. Achieving bounded blocking in circuit-switched networks. *IEEE INFOCOM 2006*, pages 1–9, April 2006.
23. R. Zhang-Shen and N. McKeown. Designing a Predictable Internet Backbone Network. In *HotNets III*, November 2004.
24. R. Zhang-Shen and N. McKeown. Designing a predictable Internet backbone with Valiant Load-Balancing. *Thirteenth International Workshop on Quality of Service (IWQoS)*, 2005.
25. R. Zhang-Shen and N. McKeown. Designing a Fault-Tolerant Network Using Valiant Load-Balancing. *Proc. IEEE INFOCOM*, pages 2360–2368, April 2008.
26. R. Zhang-Shen and N. McKeown. Guaranteeing Quality of Service to Peering Traffic. *Proc. IEEE INFOCOM*, pages 1472–1480, April 2008.