Guaranteeing Quality of Service to Peering Traffic

Rui Zhang-Shen
Department of Electrical Engineering
Princeton University
Email: rz@princeton.edu

Nick McKeown
Computer Systems Laboratory
Stanford University
Email: nickm@stanford.edu

***Abstract*—Network operators connect their backbone networks together at peering points. It is well known that the peering points are the most congested parts of the backbone network. Network operators have little incentive to provision them well, and have few tools to decide how best to route traffic over them.**

In this paper we propose how peering networks can be congestion free, so long as we know the total amount of traffic between them. In particular, we propose the use of Valiant Load-Balancing (VLB), which has been previously studied for individual backbone networks. In our approach, the backbone networks do not need to use VLB internally—they simply load-balance traffic over their peering links. Our analysis shows how the load-balancing should be done, and we conclude that no other method is more efficient than VLB in achieving a congestion-free network.

I. Introduction

A. Background

Today, most congestion in backbone networks takes place on the peering links between network operators [5], [1]. This is because peering links tend to be under-provisioned; *i.e.*, the network operators use links that are too small to carry all the traffic during peak periods. It might be surprising that operators do not just increase the capacity of each link — over-provision them — so the network will perform better. After all, each operator's backbone network is heavily over-provisioned – often by an order of magnitude or more [4]. Operators over-provision their backbone networks because of three main types of uncertainty: (1) **Future traffic**. When they deploy a network, it will have to operate for several years, even as an unpredictable number of new customers start to use the network, and as new applications create new traffic patterns; (2) **Failures and re-routing.** When links and routers fail, traffic is routed, and any link might have to carry additional traffic; and (3) **Queueing delay.** Customers do not like queueing delay and will frequently move to a new operator with lower delay and jitter performance. All of these factors provide ample incentive for an operator to over-provision their backbone network — at considerable additional cost — making the network easier to manage, have a longer deployment lifetime, and to keep their customers happy.

So why don't operators over-provision the peering links too? Apparently, they do not have enough of an incentive to do so. If a user's traffic is traversing two networks, and the performance is poor, the customer cannot tell which backbone network is at fault, or if the peering links are congested. Not knowing which network to blame, the user is unlikely to switch providers, and so there is little point in increasing the capacity of the links. Even if the operator wanted to increase the peering links, it is hard to know how large to make them. The size of the links depends not only on the future behavior of their own customers, it also depends on the number, behavior, and location of their peer's customers, which they are unlikely to be able to estimate. If they estimate badly, then some links will be swamped, while other links sit idle.[1]

In summary, it is not clear how a network operator could size their peering links so as to give good performance at a reasonable price, and in the absence of such a method they do not have much incentive to over-provision instead.

In this paper we propose a solution. First, we show a simple mechanism (based on a technique called "Valiant Load-Balancing", or VLB) that allows us to size peering links so as to prevent all congestion, regardless of the particular paths or traffic matrices between two networks. We just need to estimate the total amount of traffic between them. This then leads to a simple evolutionary model, in which new capacity can be added to any peering link and will improve the performance of the whole network. Second, we will show that this mechanism is the most efficient and cost-effective way to prevent congestion in peering networks.

B. Valiant Load-Balancing

In the early 1980s, L.G. Valiant proposed the idea of routing packets through random midpoints for the communication among sparsely connected parallel computers [14], [15]. In recent years, VLB was used to design Internet routers with performance guarantees [2], [3], [6], as well as in achieving high worst-case performance without sacrificing average- and best-case performance in interconnection networks [11]. Several groups independently applied the idea of Valiant Load-Balancing to backbone network design and traffic engineering, to efficiently support all possible traffic matrices [10], [9], [7], [8], [16], [17].

The Homogeneous Case. To illustrate VLB in a backbone network, consider the mesh of long-haul links (represented by the cloud) in Figure 1 that interconnect N backbone nodes. Current backbone networks have about $N = 50$ nodes. The network is hierarchical, and each backbone node connects

[1]Matters are made worse by the common and seemingly cheeky practice of *hot-potato routing* [12], [13], in which network operators push traffic to their peer's network as soon as they can, so as to minimize the load in their own network.

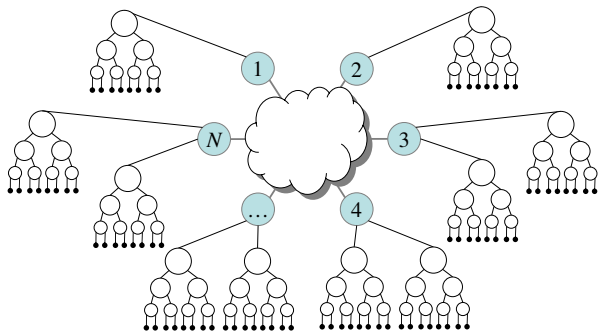

Fig. 1. A hierarchical network with N backbone nodes

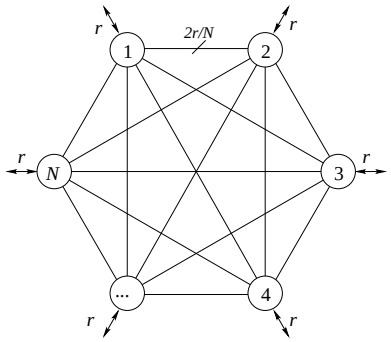

Fig. 2. Valiant Load-Balancing in a network of N identical nodes each having capacity r.

an access network to the backbone. We assume we know (roughly) the total capacity of each access network.

We represent the traffic demand between the backbone nodes by a $N \times N$ traffic matrix, where $\lambda(i,j)$ is the average rate of traffic from node i destined to node j. We say the network can *support* a traffic matrix if the capacity between i and j (either directly or indirectly) is greater than $\lambda(i,j)$.

We will start with the simple (but unrealistic) homogeneous case where all the backbone nodes have the same capacity, r. In this case, a VLB network consists of a full mesh of logical links with capacity $\frac{2r}{N}$, as shown in Figure 2. Traffic entering the backbone is load-balanced equally across all N one- and two-hop paths between ingress and egress. A packet is forwarded twice in the network: In the first hop, a node uniformly load-balances each of its incoming flows to all the N nodes, regardless of the packet destination. Load-balancing can be done packet-by-packet, or flow-by-flow, and each node receives $\frac{1}{N}$ of every flow in the first hop. In the second hop, all packets are delivered to the final destinations.

VLB has the nice characteristic that it can support all traffic matrices that do not oversubscribe a node. 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 is the dual of the first. Since each node can receive traffic

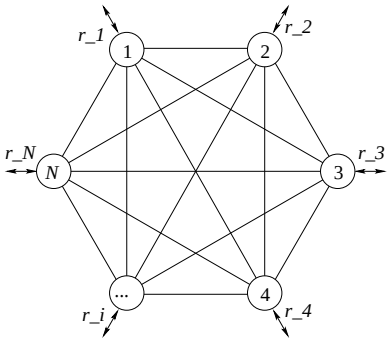

Fig. 3. Valiant Load-Balancing in a heterogeneous N-node network.

at a maximum rate of r and receives $\frac{1}{N}$ of the traffic from every node, the traffic on each link due to the second hop is also at most $\frac{r}{N}$. Therefore, the full-mesh network (with link capacities $\frac{2r}{N}$) can support all traffic matrices. The advantage of VLB for the backbone operator is that they can design their network knowing only the capacities of the access nodes, without knowing anything about the traffic patterns or how they evolve over time. The cost is that the total network has twice the capacity needed, if we knew the actual traffic matrix. It is clear today that backbone operators have little idea what traffic matrices to expect, which explains (in part) why they use five or ten times the minimum capacity. As we have shown elsewhere, VLB networks can be very easily designed to continue working when links and nodes fail, with much lower capacity requirements than existing backbone networks [18].

The Heterogeneous Case. Of course in practice, the capacity of each access network is different. VLB can be extended quite easily to the heterogeneous case [17]. Uniform load-balancing is no longer the best solution, and it is better to load-balance by sending different amounts of traffic to each node, as a function of the size of the nodes. To illustrate this, consider the N-node network shown in Figure 3. The access capacities of the nodes are $r_1, r_2, \ldots, r_N$, and c_{ij} is the link capacity from node i to node j.[2]

The *interconnection capacity*, l_i, is the total capacity of all the links between node i and other nodes, i.e.,

$$l_i = \sum_{j:j\neq i} c_{ij}. \tag{1}$$

The total capacity of the network, L, is simply

$$L = \sum_{i=1}^{N} l_i = \sum_{i,j:i\neq j} c_{ij}. \tag{2}$$

The maximum amount of traffic that all the access nodes can bring to the network, R, is given by

$$R = \sum_{i=1}^{N} r_i. \tag{3}$$

[2]We assume that a node can send traffic to itself without using any network resource, so we set $c_{ii} = \infty$. Equivalently, we can set the diagonal entries of any given traffic matrix to zero.