Experiences in Building A Multihoming Load Balancing System

Fanglu Guo Jiawu Chen Wei Li Tzi-cker Chiueh
Computer Science Department, Stony Brook University, NY 11794
Rether Networks Inc., Centereah, NY 11720
{fanglu, jiawu, weili, chiueh}@cs.sunysb.edu

Abstract— **The growing popularity of consumer broadband connection technology, in particular cableTV and ADSL, has started a quiet revolution that will reshape the Internet connectivity solutions for commercial enterprises. One emerging theme is the replacement of dedicated lines based on frame relay or ISDN technologies with multiple inexpensive ADSL/cableTV links each of which potentially is subscribed from a different ISP. The key enabling technology for this revolution is multihoming load balancing, which spreads an enterprise's Internet traffic among multiple access links to increase the aggregate throughput, and diverts traffic away from non-functional links when they fail. Although there exist several commercial multihoming load balancing products in the marketplace, very little is published about the design tradeoffs and their performance implications. In this paper, we describe the design space of multihoming load balancing systems, discuss the tradeoffs involved in load balancing and fail-over implementation strategies, and present quantitative performance measurements collected on a commercial multihoming load balancing system.**

I. Introduction

The idea of using multiple access links to improve the aggregate bandwidth and the overall availability of Internet connectivity is not new to network administrators. When the access links are subscribed from different ISPs, this approach is called multihoming. With the advent of inexpensive and high-bandwidth broadband connection technologies for home users, multihoming is poised to emerge from a niche technique for large enterprises and become a dominant technology that underlies the Internet connectivity solutions for small to mid-sized businesses. The agent of change that triggers this evolution is the dramatic drop in broadband connectivity price, especially in East Asia. This trend is expected to happen in other parts of the developed world as well in the near future. For example, the monthly charge for a 500Kbps (downstream) ADSL link in Taiwan is less than $15 (US dollars) per month. By subscribing to four such ADSL links, it is possible for a site to enjoy a downstream bandwidth higher than a dedicated T1 line but at less than one fifth of the cost. Of course, the reliability and quality of service (QoS) of these ADSL links is not yet comparable to T1 lines. But the technology trend suggests that when broadband access links reach economies of scale, their reliability and QoS are expected to attain the same level as those based on frame relay and ISDN technologies. Before that day comes, multihoming load balancing is a technology that can convert a set of lower-bandwidth and somewhat unreliable broadband connections into a high-bandwidth Internet access link that is sufficiently robust for enterprise applications. Just as the RAID (Redundant Array of Inexpensive Disks) technology exploits commodity disks for higher performance and better availability, we believe multihoming load balancing systems will achieve the same benefits for Internet connectivity with commodity broadband connections.

At the conceptual level, a multihoming load balancing system must be able to determine the available bandwidth to a remote subnet through an access link, assign incoming and outgoing Internet traffic to the available access links, and detect failed access links and divert Internet traffic around them. From a user's standpoint, an ideal multihoming load balancing system should be plug-and-play, and requires no other modification to the existing network infrastructure than adding more access links.

However, perhaps the most important factor affecting the design of a multihoming load balancing system is whether the user site has its own set of public IP addresses that are assigned from the Internet Assigned Number Authority (IANA) rather than from the local ISPs. If a user site has a set of public IP addresses that are independent of the block of addresses owned by their ISPs, multihoming load balancing can be achieved through BGP peering [1][3][5][6], which requires the user site to set up a BGP router that can peer directly with the BGP routers of their ISPs, and thus exploit standard routing protocol to improve throughput and to bypass failed links. BGP peering requires a user site to acquire an AS (Autonomous System) number, build up a BGP peering relationship with ISPs, and to maintain BGP tables, etc. While this approach is conceptually straightforward, the administration overhead of BGP router maintenance is non-trivial and therefore only large enterprises or ISPs can afford it. For small and mid-size corporate users, neither obtaining a separate IP address range nor maintaining BGP routers is feasible.

In the case that a user site is assigned a distinct set of public IP addresses from each of its ISPs, the way to implement multihoming load balancing is through Network Address Translation (NAT) technique [7][8][9][11][10], which dynamically binds public IP addresses to internal hosts. Using NAT, a multihoming load balancing system can be deployed between a user site and its ISP, and works as a black box in a way transparent to the internal hosts and ISPs. Note that this approach is equally applicable to user sites that have their own ISP-independent IP addresses. Compared with the BGP peering approach, the only disadvantage of the NAT approach is that it can only exploit inter-connection parallelism, but not intra-connection parallelism, i.e., it is impossible to "stripe" the packets of a network connection across multiple access links.

Even though multihoming load balancing systems are gaining importance in commercial development and deployment, very few publications in the public literature discussed their practical design considerations and implementation tradeoffs. It is the goal of this paper to present the design space of multihoming load balancing systems, discuss the implementation issues in more detail, and quantitatively evaluate their performance implications based on a commercial multihoming load balancing system. With this paper, we hope to stimulate more future research on the development of more advanced multihoming load balancing systems.

The rest of this paper is organized as follows. Section II surveys previous research related to multihoming load balancing. Section III presents the design space of multihoming load balancing systems, their practical implementation issues and possible solutions. Section IV reports the results of a performance study on a commercial multihoming load balancing system based on instrumentation, measurement, and test-bed emulation using real traffic traces. Section V concludes the paper with a summary of its major contributions and an outline of future work.

II. Related Work

Along the BGP solution line, several solutions are proposed. RFC 2260 [1] proposes a BGP peering scheme to provide fault tolerance without increasing routing overhead in the "default-free" zone of the Internet. They assume enterprises can get their IP addresses via the "address lending" policy, i.e., get IP addresses from the IP pool of each ISP. Normally enterprises only advertise the IP prefixes to the ISP where they get those prefixes. This will allow ISPs to aggregate the prefixes. After aggregation, there is no need to advertise these prefixes to the "default-free" zone. When connectivity between enterprise and ISP goes down, the failed IP prefixes will be either advertised to the whole Internet (case 1) or only advertised to active ISPs (case 2). In case 1, Internet routing will achieve failover automatically. In case 2, ISP will forward the packets to enterprise via tunnelling. When enterprise want to have independent IP prefixes, RFC 1518 [2] proposes to assign the enterprise only one prefix that is independent from ISPs. This allows multihomed organization to aggregate the network within that organization to a single prefix.

Cisco in [3] shows how to use normal BGP techniques to achieve fault tolerance and load sharing. They can use multiple links to share the load of outbound traffic via static policy. For inbound traffic, the best that one can do is to divide the enterprise network to multiple subnets. Then the enterprise advertises one subnet to each ISP. This contradicts the suggestions in RFC 1518. It increases the number of route in the "default-free" zone. Since load can only be statically split, this technique is called load sharing instead of load balancing in Cisco's documents. In Cisco's term [4], load balancing means better load distribution among multiple paths. In the best case, load balancing can distribute the first packet to link 1 and the second packet to link 2. That means all links will be apply almost the same load simultaneously.

RouteScience [5], netVmg [6], etc go further to measure the performance of links to better manage the traffic distribution on the links. They can measure latency, loss to some prefixes, link utilization, etc to decide which link is better. To control the traffic distribution, they use BGP to update routers' routing table. So they can control the distribution of the outbound traffic on links in prefixes granularity. For inbound traffic, it seems they cannot go any further than Cisco can. The enterprise prefixes advertised is the same to all networks. They cannot control which network receives which kind of BGP advertisements.

Along the NAT line, Radware [7], F5 Networks [8], Nortel Networks [9], FatPipe [11], etc propose to use NAT to distribute traffic to multiple links. They also measure the performance of links as vendors such as RouteScience do. The difference lies in how to control the distribution of traffic to links. For enterprise initiated connections, the load balancing device will first decide which link to use. Once the link is selected, the public IP address of NAT should be the IP addresses assigned to that link. The inbound traffic of this connection will also go through the same link. For Internet initiated connections, the vendors above use DNS to dynamically return one IP address whose associated link is least loaded. This way they can also control which link to use for the Internet initiated connections.

Cisco in [12] proposes a different NAT scheme. It will translate both source IP address and destination IP address in a packet. Compared to normal NAT, it further translates the destination IP address of the packets for outbound traffic and the source IP address of the packets for inbound traffic. The purpose of the above two extra translations is to force packets in a connection to go to the same link. The scheme can control the IP address returned by DNS to choose which link to use. But this scheme doesn't consider the real-time load of links to control the return value of DNS. So it is only a kind of static load sharing.

Zhiruo Cao et al. [13] presented a comprehensive study on hashing based load balancing. They use hash to split traffic over multiple links. Their results show hashing does show good load balancing performance based on their traced flows.

Tao Zhou [15] presented the multiple default gateways/routers configuration in Win2K and NT 4.0. It reflects the efforts for network fault tolerance in the operating system field. But these multiple routers are not used for load balancing but fault tolerance. Windows can detect dead path to Internet [16]. So windows can switch to other available routers if possible. In [15], load balance is only achieved via static address division.

DNS Cycling (or DNS-based redirection) technology is widely employed in server selection, which serves for two major purposes: to do load balancing among servers based

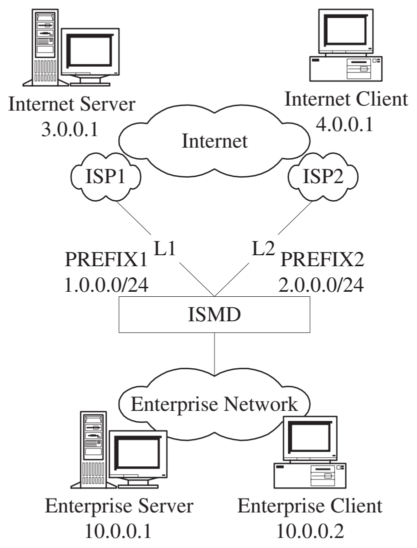

Fig. 1. The generic context in which a multihoming load balancing system works. Each ISP gives a specific IP prefix to an enterprise network, and only forward incoming (outgoing) packets whose destination (source) address uses the same IP prefix.

on their loads, and to choose the closest server to a client for best response time. In response to a DNS request for a specific server, the DNS server will set the value of the TTL field to be zero or small values (compared with the typical value of 24 hours) to prevent the client from caching the result. Anees Shaikh et al. [14] quantify the impact of low DNS TTL values on client-perceived latency. It reports that small TTL values allow fine-grained load balancing and rapid response to changes in server or network load, although disabling clients caching may cause latency and degrade the scalability of the DNS.

III. Design Space

A. Problem Formulation

Figure 1 shows the context in which a multihoming load balancing system operates. We will refer to such a system an Internet Service Management Device (ISMD) hereafter. An enterprise network connects to multiple ISPs via ISMD, which can balance the loads on the access links to these ISPs. In addition, ISP1 and ISP2 allocate an IP address prefix PREFIX1 and PREFIX2 to the enterprise network, respectively. ISP1 will only forward outgoing (incoming) packets whose source (destination) IP address is PREFIX1. ISP2 behaves similarly.

Because an ISP only forwards packets with source or destination address that falls into a proper range, ISMD can use an access link associated with an ISP only when the packets are marked with proper source or destination address. If an IP address is statically assigned to each internal host, then all traffic into and out of an internal host will follow the ISP link whose associated prefix is the prefix of its IP address. That is, the load distribution granularity is at the host level. To support finer-grained load distribution strategies, each internal host should be allowed to bind to different public IP addresses *simultaneously*. Although statically assigning multiple public addresses to an internal host is technically feasible, it leads to wastage of IP addresses, which small and mid-sized enterprises typically cannot afford. A more realistic solution is based on network address translation (NAT), which is able to assign each individual network connection a different IP address, thus achieving connection-level load distribution.

For an internally initiated network connection, its first packet comes from an internal host. When this packet reaches ISMD, ISMD uses NAT to change its original source address and port number to a public IP address and port number. The remote host of this connection will use the same public IP address and port number as destination address and port number in its response packets. As a result, once ISMD fixes the source address and port number of the first packet of an internally initiated connection, all subsequent packets in both directions will flow through the ISP link whose IP address range covers the chosen public IP address. As an example, assume an internal host whose IP address is 10.0.0.2 initiates a connection to a remote server whose IP address is 3.0.0.1. When the initiating packet reaches ISMD, ISMD changes its source address to 1.0.0.1, which has the same prefix as ISP1. Consequently, all subsequent packets, both inbound and outbound, will go through ISP1. Therefore, when ISMD uses NAT to modify the first packet of an internally initiated connection, it implicitly dispatches packets from that connection to a particular access link associated with an ISP.

For an externally initiated connection, ISMD cannot directly control the access link that the remote initiating host will use to reach an internal host. However, externally initiated connections typically target at internal hosts that are well-known servers. To access these servers, remote hosts usually need to issue a DNS query to resolve their host names to the corresponding IP addresses. ISMD can take advantage of this pattern by embedding load distribution into DNS query servicing. More concretely, each well-known server host is assigned multiple IP addresses using the port forwarding mechanism of NAT, one from each ISP's prefix. When a DNS query for a well-known server arrives, ISMD responds to it by choosing one of its assigned IP addresses according to its load balancing policy. Through this DNS re-direction mechanism, externally initiated connections can also be distributed evenly among the access links. As an example, assume there is an internal web sever whose private IP address is 10.0.0.1 and whose domain name is www.a.com. To allow this web server directly accessible from the Internet, it must have at least one public IP address. In this case, it has two: 1.0.0.2 and 2.0.0.2. To access the web server www.a.com, a remote host (4.0.0.1) first makes a DNS query to retrieve the IP address of www.a.com. If ISMD responds to this query with 1.0.0.2, all packets of this connection will go through ISP1.

In summary, by associating a public IP address/link with an internal host when a connection is established, ISMD is able to dispatch that connection's packets to a particular ISP link. For an internally initiated connection, this binding occurs through NAT when its first packet reaches ISMD. For an externally initiated connection, this binding occurs through DNS redirection if its set-up is preceded by a DNS query. Once the internal host of a network connection is bound to a public IP address, all subsequent packets of that connection in both directions will go through the access link whose ISP's prefix covers the chosen the public IP address. In this scheme, the load balancing granularity is individual network connections.

In addition to dispatching network connections to specific access links, ISMD performs two more functions. First, ISMD determines the degree of availability of each access link, i.e., whether each link is available and how much bandwidth is remaining in each link. Second, ISMD chooses which access link to use and balances the loads among the access links and to avoid failed links whenever they are detected. In the following subsections, we will discuss these two issues in more detail.

B. Determining Link Availability

In the ideal world, ISMD should be able to determine, for each access link, whether it is possible to reach each network subnet represented by a BGP prefix, and the bandwidth available to each such subnet if it is reachable. Armed with this information, it is straightforward for ISMD to decide which access link to use for a new network connection. In practice, however, ISMD (and other Internet routers) is rarely able to obtain such a clear picture of the Internet at any point in time, because Internet traffic conditions fluctuate constantly. Therefore, some compromises must be made in terms of the scope (only some prefixes rather than all prefixes) and the accuracy (only reachability rather than bandwidth/latency characteristics). The other end of the spectrum is to determine whether an access link is available and therefore usable in shouldering a user site's Internet traffic. The focus here is limited to the availability of the link between a user site and its ISP.

There are two general approaches to Internet traffic condition determination: *passive monitoring* and *active probing*. Passive monitoring is unobtrusive, requires less network resource, but is less accurate. Active probing provides more flexibility, requires more network resource, but is also more accurate. To determine whether an access link is still available, passive monitoring is sufficient in most cases. Because most network connections are bi-directional (at the transport or application level), there should be some inbound traffic on an access link as long as at least one connection is using that link. If there is only inbound traffic on an access link, passively monitoring the inbound traffic only may fail to detect failure of the outbound path. One way to address this problem is to check if the incoming packets are either pure TCP ACKs or TCP DATA segments with increasing ACK sequence number. The disadvantage of this approach is that it requires per-connection state.

When there is only outbound traffic but no inbound traffic, or when there is neither inbound nor outbound traffic, passive monitoring alone cannot ascertain whether an access link is available or not. In particular, it cannot distinguish between the case that there is no incoming traffic, and the case that the access link is down. In these cases, some sort of active probing methods need to be brought to bear on this problem. Table I summarizes the limitations of passive monitoring. Note that it is still important to determine a link's availability even when there is no inbound or outbound traffic because the information is needed for supporting externally initiated connections.

TABLE I

PASSIVE MONITORING INFERENCE RULES. NORMALLY, THE EXISTENCE OF INBOUND TRAFFIC WILL REPRESENT GOOD HEALTH. THE LACK OF INBOUND TRAFFIC SHOULD REMIND US TO DOUBT THE HEALTH.

Outbound	**Inbound**	**Link health**
No	No	Possibly bad, need to check
No	Yes	Available
Yes	No	Possibly bad, need to check
Yes	Yes	Available

There are two main design issues in the use of active probing to determine link availability. First, which set of hosts should be used as probe targets? Because the goal is to determine access link availability, it is important the chosen probe targets be at the other end of an access link. In addition, the chosen probe targets should be able to handle a large number of probe requests. The second issue of active probing is what type of probe packets should be used. Because TCP connection set-up is too expensive, typically UDP or ICMP packets are preferred. However, many firewalls are set up to drop ICMP packets. Therefore, UDP packets seem to be more appropriate. Taking into account these two considerations, ISMD chooses a set of DNS servers on the Internet as the probe targets and uses DNS queries as probe packets.

To reduce the traffic load induced by active probing on the network and probe targets, ISMD uses active probing mainly as a fall-back mechanism to passive monitoring. Since passive monitoring works when there is inbound traffic, active probing will be triggered only in the absence of incoming traffic. If active probing finds that it is a false positive, the probing will stop until passive monitoring report possible outage again. If it is a true positive, the probing will continue indefinitely. But if the link is down, such probing will not impose additional load to the Internet. Figure 2 shows a state machine that uses a combination of passive monitoring and active probing to determine the availability of an access link. The POSSIBLY DOWN state is used to eliminate false positives of passive monitoring. In this state, the link is still treated as UP. But we will start active probing to ascertain if it is UP or DOWN.

In some cases, an access link may be available but it cannot be used to reach certain remote subnets because, for example, the associated peering link between the access link's ISP and its upstream ISP is highly congested or simply goes down. To address this problem, ISMD needs to maintain availability state for each BGP prefix, which is clearly not feasible in practice. A more realistic approach is to maintain availability state for a subset of BGP prefixes, and the choice of this subset

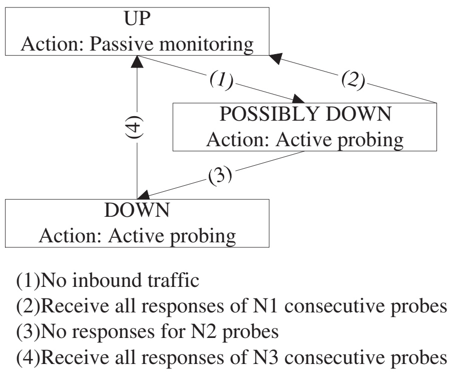
UP
Action: Passive monitoring
(1)
(2)
POSSIBLY DOWN
Action: Active probing
(4)
(3)
DOWN
Action: Active probing
(1)No inbound traffic
(2)Receive all responses of N1 consecutive probes
(3)No responses for N2 probes
(4)Receive all responses of N3 consecutive probes

Fig. 2. A state machine that determines link availability using passive monitoring and active probing.

is based on either their geographic locations [8] or based on how often a user site has interacted with them in the past. One can take one step further to determine the packet latency to these selective subnets so that load distribution can also take into account performance factor. This can be achieved via DNS queries [17], or passively monitoring of TCP ACKs [18].

C. Assigning Traffic to Links

The first question in assigning traffic to access links is the granularity. Finer granularity tends to achieve better load balance and fault tolerance, but may require more infrastructural support. Packet-by-packet link assignment is only possible for network connections whose internal hosts have a public IP address that is covered by the BGP routers of the ISPs, and therefore is not compatible with NAT-based multihoming load balancing systems. Connection-by-connection link assignment is feasible for NAT-based multihoming load balancing systems in theory, but does not always work in practice. The reason is that some network applications require setting up several network connections that should have the same IP addresses as end points. For instance, in Figure 1, the enterprise client 10.0.0.2 wants to connect to a remote FTP server 3.0.0.1. FTP needs to set up a control connection and a data connection. Assume the control connection is assigned to L1 and the public IP address of the enterprise client for this connection is 1.0.0.1. If the data connection is assigned to L2, the public IP address of the enterprise client for this connection will be in the range of 2.0.0.0/24. As a result, the remote FTP server may reject the data connection because it finds the remote IP address for the data connection is different from that of the control connection. Therefore, it is essential that both connections of a FTP session be assigned to the same access link. There are other network applications that have a similar requirement, e.g., RTP/RTCP, P2P file sharing, etc. For them, assigning the set of network connections that belong to a logical session to the same link is required for correct operation.

However, identifying all connections in a session for all applications is difficult, if not impossible. A simpler way to address the problem is to assign packets to links based on their source and destination IP address only, rather than the four-tuple connection identifier that also includes source and destination port number. The disadvantage of this simple session-by-session link assignment approach is that its load distribution granularity becomes coarser, and may lead to suboptimal performance in some cases. For instance, it is quite common for a client to download data from a web server through multiple simultaneous HTTP connections. With session-level link assignment, these HTTP connections will be assigned to the same link, and therefore cannot benefit from the availability of multiple access links. To optimize for these common cases, ISMD uses connection-level link assignment for specific cases like these and leave all other connections session-level link assignment.

Which access link to be used for a network connection should be determined at the beginning of the connection. Given the connection ID (four-tuple) or session ID (two-tuple), there are two possible approaches to the link assignment problem. The first approach is stateless, and applies random hashing on the connection/session ID to obtain the link ID [13]. Since the hashing result is invariable for a given connection/session ID, there is no need to keep any state. However, there are two problems associated with this approach. First, it is not possible to use hashing to determine the access link for an externally initiated connection, because the choice of access link must be made before the choice of the public IP address for the connection's internal host. The only way to resolve this problem is to assign all the traffic to and from a well-known internal host to only one access link, but this denies the benefits of multiple access links to such hosts. The second problem with the hashing approach is that it does not take into account dynamic traffic load characteristics. As a result, it is possible that the loads distributed among the access links may not be balanced because the hash function is biased for a given traffic load.

The second link assignment approach is stateful, and explicitly takes into account such factors as latency, throughput, and availability. This stateful approach typically requires a look-up table to record the dispatch decision so that all the subsequent packets of a network connection will follow the same link assignment decision made at the beginning of the connection. One possible variant of this approach is to assign link based mainly on latency, in the hope to decrease the user response time. The assumption of latency-based link assignment is that throughput is not an issue. For example, the total download time of a web page is mainly affected by the latency instead of throughput of the access path, because typical web page size is small. Several commercial route control products [8], [5] are based on this approach.

Unfortunately, latency-based link assignment may not be always feasible. First, most of the traffic on an enterprise access link is inbound traffic. This makes it difficult to determine the latency to a remote subnet based on passive monitoring. Second, because of the limited bandwidth on the access links, there is every incentive to fully utilize them. As a result, throughput may well become the dominant factor that affects the user response time, and it thus makes sense to assign links based on the traffic load on the access links. This load-based link assignment approach requires a way to accurately estimate the load on an access link. Because most network connections are TCP-based, and a single TCP connection is capable of using up all the bandwidth of a link as long as the connection's lifetime is sufficiently long, there is no point of keeping track of the link load based in terms of bytes/sec. Instead, it is better to estimate the load on a link based on the number of connections assigned to that link. Because different connections may carry different amount of traffic, a weighted sum of these connections should be more accurate than a simple count. However, it is not clear how to derive the weights since the traffic volume of each network connection cannot be easily determined before the connection is terminated.

A related issue of load-based link assignment is traffic volume asymmetry. In many cases, the load on an access link can be heavily biased toward a particular direction because the connections assigned to the link are biased. If load-based link assignment ignores the directionality, it is possible to choose an access link whose total load is smaller but whose load on the direction towards which the new connection is biased is actually higher than other access links. One way to resolve this problem is to balance the traffic load on each direction separately for the access links. However, this again requires a priori knowledge of the bias direction of each network connection. While it is relatively easy to infer the directionality of some network connections such as HTTP connections, it is not clear how to infer this knowledge for arbitrary connections in general. In the case traffic volume asymmetry of a connection cannot be inferred, link assignment decision can be made based on the load of the links' bottleneck direction.

D. Other Design Issues

1) Bandwidth Guarantee and Multihoming: There is a subtle interaction between bandwidth guarantee and multihoming. When an enterprise has only one access link, providing bandwidth guarantee for a group of connections is relatively straightforward, because all connections will go through the same link. However, when there are multiple access links, there is the possibility that the sum of the residual capacities of the access links can support a reservation, but none of each individual link can support the reservation on its own. In this case, if there is only one network connection in that reservation, this connection cannot use up all the reservation's bandwidth because a network connection cannot use more than one access link. Essentially this is an instance of the resource fragmentation problem due to the indivisibility of a connection's traffic.

As an example, suppose the bandwidth guarantee for group0 and group1 are both B, and there are two access links whose maximum bandwidth is also B. Let's assume further group1 already consumes all its reserved bandwidth and spreads its traffic between the two access links evenly. Therefore, each access link now has a residual capacity of B/2. When the first connection from group0 comes along, it can be assigned to either access link, and consume at most B/2 bandwidth, even if it is the only connection from group0. This clearly violates the bandwidth guarantee requirement but is inevitable because of the indivisibility of the traffic associated with a connection or session.

One way to solve this problem is to increase the amount of bandwidth reservation of each group of traffic to accommodate the loss in fragmentation. Assume we can cap the maximum throughput requirement of each network connection at M for a traffic group whose total bandwidth reservation is R (R $\geq$ M). Then it can be shown that on an N-link system, as long as the total reservation for a traffic group with an original bandwidth reservation of R is R + M * (N - 1), the traffic group is guaranteed a bandwidth reservation of R. In practice, it is not clear how to derive M in advance. On the other hand, the fragmentation problem arises only in specific scenarios and does not occur all the time. Therefore, a good compromise is to reserve $\alpha * B$ for a logical reservation of B, where α is an empirical parameter that depends on the number of access links and the dynamic traffic distributions.

2) Preventing Denial of Service: In addition to incurring higher performance overhead, the stateful load balancing approach is also vulnerable to denial-of-service attacks. A remote user can exhaust the state memory of ISMD by initiating a large number of network connections. Moreover, it can evade ISMD's detection by IP address spoofing. When the state memory is used up, no more new connections can be serviced. A simple way to address this problem is to monitor the state memory usage, and to give up the load balancing capability when the state memory is about to be exhausted. This way, only the load balancing capability of ISMD is lost in the presence of a denial of service attack, but not its connectivity. To prevent IP spoofing, we can further check the three-way handshake of each TCP connection, because hosts that use IP spoofing typically cannot finish three-way handshake. State allocated to connections that cannot finish three-way handshake is timed out and reclaimed quickly.

A more refined approach is to bound the state memory consumption of each traffic group, and thus isolate the state memory usage of one group from the others. When the state memory allocated to a traffic group is used up, only this traffic group loses the load balancing functionality. With this approach, ISMD can at least better prevent those traffic groups that have their own resource reservations.

3) Link and Node Fault Tolerance: To provide highly available Internet connectivity, both ISMD itself and the access links have to be highly available. While multihoming load balancing increases the failure resilience of the access links, it needs to be integrated with ISMD's fault tolerance mechanism to form an end-to-end availability solution for Internet connectivity. ISMD is built on a reusable fault tolerance implementation framework called Duplex [20], which is designed to provide fault tolerance specifically for edge network devices.

The Duplex framework provides two alternative configurations for a pair of fully redundant edge network devices,

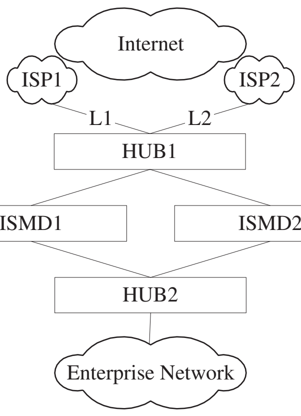

Fig. 3. Parallel dual-device configuration for ISMD's own fault tolerance.

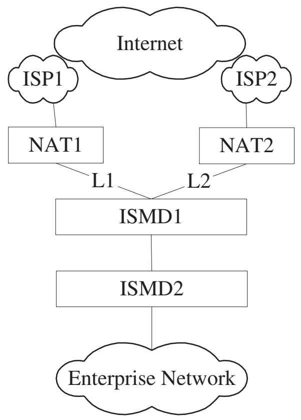

Fig. 4. Serial dual-device configuration for ISMD's own fault tolerance.

one serving as the master while the other as the slave. The first configuration is a parallel dual-device configuration as shown in Figure 3. In this configuration, the two ISMD devices are connected with hubs, and work in promiscuous mode. So both devices can see all the packets. Because both devices run the same link assignment algorithm and carry out the NAT functionality, they will make the same decision and eventually reach the same internal state. In the beginning, it is possible that the two devices may not have the same state. But since the NAT state and load balancing state is soft state, gradually the inconsistencies will time out and be removed. When the master fails, the slave takes over automatically. Because both the NAT and link assignment state remain unchanged, no connections are disrupted and thus the fail-over is completely transparent.

One problem with the parallel dual-device configuration is that it relies on the hubs to duplicate the packets, which limit the traffic to be half duplex. In the second configuration in the Duplex framework, the serial dual-device configuration as shown in Figure 4, the two ISMD devices are connected in series, work in promiscuous mode, and each have a watchdog-based hardware bypass. Moreover two routers (NAT1 and NAT2) are added to perform NAT functionality. ISMDs only needs to balance the load on the access links, L1 and L2. At any point in time, both devices are running the same algorithm and thus having the same state. When the master dies, the slave takes over. When both devices fail, the bypass hardware will turn the ISMD pair into a passive cable, and the traffic still goes through but without load balancing. Although this configuration provides an additional level of fault tolerance when both devices fail, it does so at the expense of additional latency because both devices need to perform the same link assignment algorithm even though only one of them takes effect. The other disadvantage of this configuration is that the NAT functionality needs to be moved to the edge routers. However, this requirement is not absolutely essential if tolerance for double device failures is not required.

IV. PERFORMANCE EVALUATION

In Section III, we discussed the design issues of a multi-homing load balancing system and their possible solutions. In this section, we will evaluate the effectiveness of some of these solutions using a commercial multihoming load balancing system from Rether Networks Inc. called ISMD, which is a Linux-based edge network appliance using a 850-MHz PentiumIII as the main CPU. We used a trace-driven emulation approach to evaluate the performance of these schemes whenever possible. We collected packet traces in a random week from the access router of an engineering company that has an Internet access link running at 500Kbps. This company represents a typical small to mid-sized business that is likely to use an NAT-based multihoming load balancing system to improve the aggregate throughput and fault tolerance of its Internet connectivity.

A. Effectiveness of multihoming fault tolerance and latency based link selection

An ideal latency-based link selection algorithm will keep track of the latency to each of the remote subnets with which a user site interacts in the recent past. To determine whether it is feasible to maintain such a list, we count the number of network prefixes that show up in our week-long trace according to a default-free BGP routing table available from Routeview [21], which has 141325 entries in total. From this analysis, we found the engineering company accessed about 4632 different hosts on the Internet, which fall into 1541 different network prefixes. In this case, the visited network prefixes during one week is only 1.07% of the prefixes in the routing table.

Among those subnets visited in our trace, the access frequency is also highly biased, as shown in Figure 5. That is, most of the accessed subnets concentrate on a small subset of the visited prefixes. For example, the top most 10% most popular prefixes account for more than 60% of the Internet access. These results suggest that a multihoming load balancing system can reap most of the benefit of latency-based link selection by maintaining a small number of prefixes.

To measure the potential gain of latency-based link selection, we extract prefixes from a trace that corresponds to one randomly chosen day from the tracing week. Then we probe

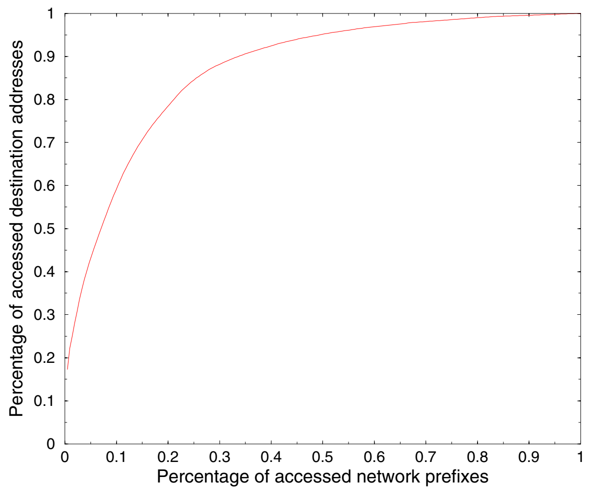

Fig. 5. Each point (X, Y) in the curve means that the top X% of network prefixes cover Y% of the destination addresses in a week long trace. The prefixes are sorted from larger destination coverage to smaller.

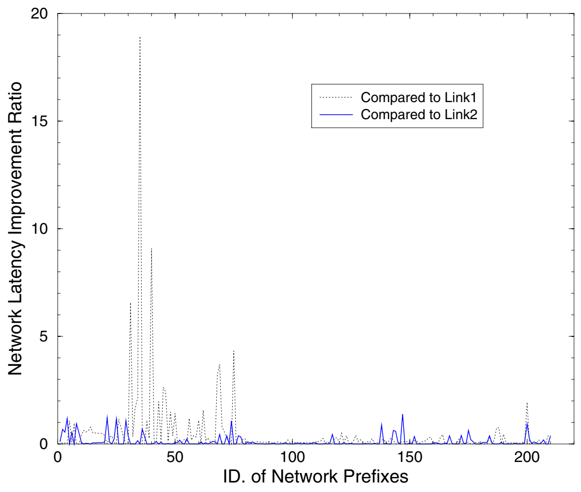

Fig. 6. The network latency improvement between single-linked and multi-homing solution: the improvement ratio is (S-M)/M, where S is the latency through a certain access link and M is the latency through a multihoming system that uses latency-based link selection algorithm.

one host in each of these prefixes one by one simultaneously from the engineering company and from a nearby university.

These two probing sites use different ISPs and thus together represent a typical multihoming site. We started the latency probing tasks on both sites at the same time. Each probe transaction uses TCP SYN and TCP SYN//ACK to measure the round-trip time and thus the network latency to the probed subnet. If the TCP SYN/ACK response doesn't come back within 3 seconds after the corresponding TCP SYN is sent, the test program will retransmit the SYN packet up to twice, and claims the particular destination subnet is unreachable. For latency analysis, unreachable subnets/prefixes are ignored. For each reachable subnet/prefix, we have two measurements, one from each probing site. An ideal latency-based link selection algorithm should always choose the faster of these two access links, and therefore its latency should be the smaller of the two.

We started a probing run every five minutes on both sites at the same time, and conducted this latency probing test for one full day. The final latency value for each subnet/prefix is calculated based on the average of these latency measurements. Figure 6 shows the ratios between the latency of a multihoming solution based on an ideal link selection algorithm and that of the company's link and that of the university's link, respectively, for all the prefixes tested. From the figure, there is plenty of room of optimization for a latency-based link selection algorithm because the latency ratio in some cases could be more than an order of magnitude.

The number of unreachable subnets in this test gives a hint on the usefulness of a multihoming system as a network fault tolerance solution. In the full-day test, there are two subnets which could sometimes be reached via ISP1 and sometimes via ISP2. Because the test is far from comprehensive, it is difficult to draw any definitive conclusions from this result, but it does suggest that a multihoming system can indeed provide some form of tolerance against faults or serious congestion on the ISP or Internet links.

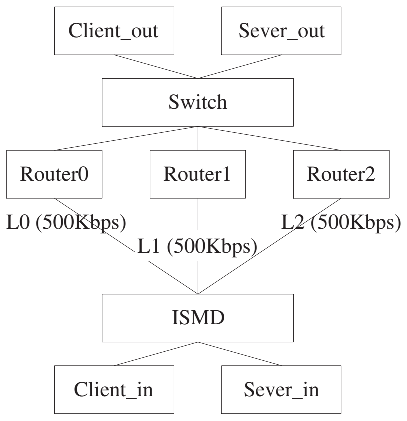

Fig. 7. The topology of the emulation testbed for evaluating the effectiveness of load balancing algorithms

B. *Effectiveness of load balancing*

1) Testbed setup: To evaluate the effectiveness of the load balancing algorithms discussed in Section III, we set up a testbed to emulate an enterprise network connected through a multihoming system such as ISMD to multiple ISPs, as shown in Figure 7. Router0, router1 and router2 represent three ISPs. Server_out and client_out represent Internet servers and Internet clients. Server_in and client_in represent enterprise servers and clients. All the above machines run Linux 2.2.5. The speed of each link, L0, L1 and L2 is 500Kbps, the same link speed as the link where the trace was collected. ISMD uses NAT to allow client_in to use any of the three links to initiate connections and communicate with server_out. To allow externally initiated connections, ISMD uses DNS and static port mapping to allow client_out to obtain one of the IP addresses of server_in, and thus uses one of the three links to connect and communicate with server_in.

TABLE II

THE NUMBER OF CONNECTIONS AND VOLUME OF DATA FOR THE TWO TRACES USED IN THE TEST OF LOAD BALANCING ALGORITHMS. BOTH TRACES ARE COLLECTED ON A 500 KBPS INTERNET ACCESS LINK ON A NORMAL WEEKDAY. THE DURATION OF EACH TRACE IS 4 HOURS.

	Number of connections	Total data volume (MBytes)
Trace 1	4,774	218
Trace 2	6,020	153

We used two four-hour traces in this study, whose characteristics are shown in Table II. From these traces, we extracted the start time and traffic volume in both directions for each TCP connection, and replayed them using real TCP connections on the testbed. Therefore, data is exchanged between two ends of a TCP connection as fast as TCP and the network link allow. To speed up the emulation process, we compress the four-hour tracing period into one hour, but keep the volume of each TCP connection unchanged.

2) Asymmetric traffic: hashing based vs. load based link selection algorithm: In this test, we only use client_in to create connections with server_out. So most traffic is from client_in to server_out and the traffic is asymmetric. The purpose of this experiment is to simulate the case when the traffic on an access link is highly asymmetric. We run the two traces against three configurations. In first configuration, there is only one access link through router0. This represents the base case. The second configuration represents a multihoming site with two access links, L0 and L1, both of which run at 500Kbps, and their corresponding routers are router0 and router1, respectively. The third configuration represents a multihoming site with three access links, L0, L1, and L2, all of which run at 500Kbps, and their corresponding routers are router0, router1, and router2, respectively.

We test three load balancing algorithms. The first algorithm is DRR (Deficit Round-Robin) for each packet. There is no NAT in this case. The returning traffic of each connection will go through one router: router0. But the returning traffic is small. So it doesn't affect the performance results. This algorithm represents a packet-granularity load balancing algorithm and thus corresponds to the best load balance result one can achieve. The second algorithm is hashing based selection algorithm. The hashing function is: selected_link = (source_ip_addr + source_port + destination_ip_addr + destination_port) % number_of_links. The third algorithm is load based selection algorithm. When a connection needs to be assigned to one link, we always select the least loaded link. Both hashing based algorithm and load based algorithm represent flow-granularity or connection-granularity load balancing algorithms. Even though we have only two servers and two clients, the above test set-up faithfully emulates the dynamics of these load balancing algorithms in the real world.

We use the sum of the durations of the TCP connections in a trace as the comparison metric because each TCP connection's duration typically represents the user waiting time. Table III shows the resulting duration sum for the two traces and under different configurations and load balancing algorithms. Surprisingly, despite its stateless nature, hashing based algorithm exhibits a similar performance to load based algorithm, except when they use flow granularity in the two-ISP configuration under Trace2. Moreover, as the number of access links increases, the performance difference between hashing based and load based algorithms decreases, because hashing tends to produce more evenly distributed results when the hash table is larger. Finally, the performance difference between the packet-granularity DRR and the two flow-granularity load balancing algorithm is not that significant, under 15% for all cases except the three-ISP configuration under Trace2. This result shows that being able to exploit inter-connection parallelism in many cases can already achieve pretty good performance without exploiting intra-connection parallelism.

Figure 8 shows the duration of the top 100 longest connections in the three configurations under Trace1, when the flow-granularity load balancing algorithm is used. This figure shows that multiple access links can indeed dramatically decrease the

TABLE III

TOTAL DURATION OF ALL CONNECTIONS, THE SMALLER THE VALUE, THE BETTER THE AGGREGATE THROUGHPUT

Test cases	Trace1 (seconds)	Trace2 (seconds)
1-ISP, 500 Kbps	79,400	6,957
2-ISP, packet, DRR	29,932	2,059
2-ISP, flow, hash based	30,924	3,357
2-ISP, flow, load based	30,618	2,222
3-ISP, packet, DRR	14,553	1,408
3-ISP, flow, hash based	16,596	2,019
3-ISP, flow, load based	17,531	1,930

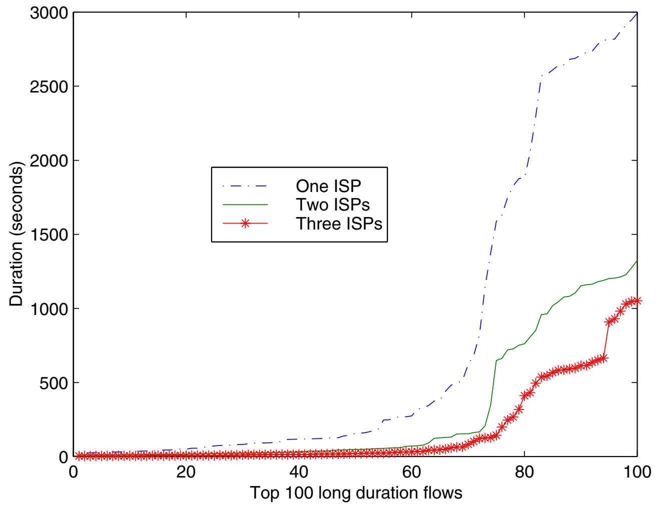

Fig. 8. The durations of the top 100 longest flows in the one-ISP, two-ISP and three-ISP configurations under Trace1. For two-ISP and three-ISP configurations, we use the flow-granularity load-based link selection algorithm.

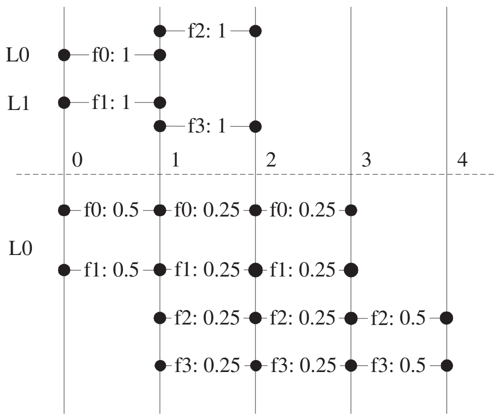

Fig. 9. Example to demonstrate the duration decrement factor can be greater than 2. The magic lies in the fact that the duration will be affected by when there are concurrent active connections and how many concurrent active connections are.

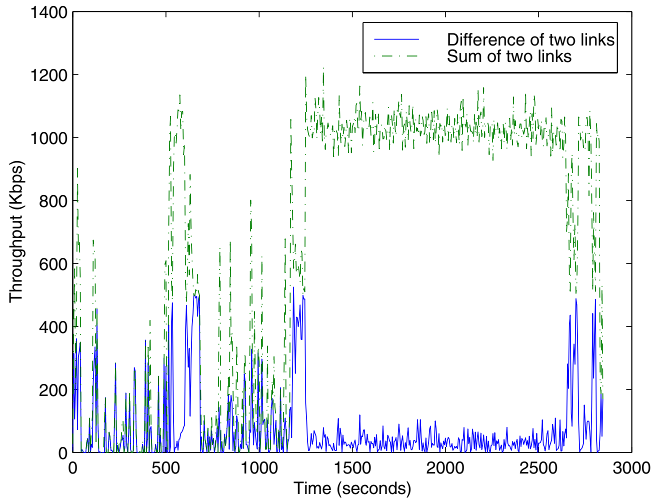

Fig. 10. The load difference between two access links (solid line) under a flow-granularity load-based link selection algorithm is small, especially when the sum of the loads on the two links (dashed line) is high. When the sum of the two links's loads is low, the load difference of these two links matters less because one link can already serve all the requests.

duration of long-life-time connections, sometimes by a factor that is more than the number of access links, in this case 2. For example, under Trace1, the total sum of connection durations is 79K seconds in the one-ISP configuration and 30K seconds in the two-ISP configuration. The decrement factor is 2.6. This result is somewhat counter-intuitive, but can be explained by the fact that connections may overlap in time and an overlapped period is counted by as many times as the number of connections that cover it.

Figure 9 showed a concrete example to demonstrate why the sum of connection durations can be decreased by a factor of more than 2 for a 2-ISP configuration. Two cases are considered here, the one-link case and the two-link case. The link speed and the traffic load in both cases is the same. There are totally four connections, each with the same amount of data to be sent. The duration of a connection is 1 if the connection can use the whole link. Two connections, f0 and f1, start at time 0, and two other connections, f2 and f3, start at time 1.

Assume we can achieve perfect load balance in the two-link case. Two connections, f0 and f1, start at time 0, and are assigned to L0 and L1, respectively. At time 1, f0 and f1 are done and f2 and f3 start. At time 2, f2 and f3 are done. The duration of each connection is 1 and the sum of connection durations is thus 4. In the one-link case, at time 1, f0 and f1 only finish 0.5 of their task since they two share one link. Flow f2 and f3 will start just as in the two-link case. Now four connections share one link. At time 2, f0 and f1 finish 0.75 of their task. Flow f2 and f3 finish 0.25 of their task. At time 3, f0 and f1 are done. Flow f2 and f3 finish 0.5 of their task. After time 3, only f2 and f3 use the link. So at time 4, f2 and f3 will be done too. The duration of each connection is 3 and the sum of connection durations is 12. Therefore, the ratio between the sums of connection durations of these two cases is 3. Interesting enough, if f2 and f3 start at time 0 or start at time 2, the ratio between these two cases will be only 2.

Figure 10 shows the load imbalance between two access links for a flow-granularity load based link selection algorithm in the two-ISP configuration. When the sum of the two links' loads is high, the load difference between the two links is small. Significant load imbalance occurs mostly when the sum of the links' loads is small. However, load imbalance does not affect the overall performance that much in these scenarios, because a single link is sufficient to carry the load. This result explains why flow-granularity load balancing algorithm can achieve similar performance as packet-granularity DRR algorithm.

3) Symmetric traffic: direction known vs. direction unknown: To determine the importance of load balancing along each traffic direction separately, we test two load balancing algorithms, one maintaining a separate load statistic for each traffic direction on each access link and requiring the knowledge of the dominant traffic direction of each TCP connection, and the other maintaining a single load statistic for each access link ((outgoing_load + incoming_load)/2) and not requiring any knowledge of the dominant traffic direction of each TCP connection. To generate traffic, we start the same traffic from client_out to server_in and from client_in to server_out. Server_in has two public IP address and one public name. ISMD will return one IP address to client_out based on the load of the links. There are two routers in the test. Client_out will query ISMD for the IP address of server_in via DNS each time it create connections with server_in because ISMD answers the query with TTL as zero.

Table IV shows the sum of all connection durations in each direction under Trace1 and Trace2 when the two load balancing algorithms are used. The result indicates that a load balancing algorithm that works on each traffic condition separately is only 11% better than a bi-directional load balancing algorithm under Trace1, but is 36% better under

TABLE IV

THE SUM OF CONNECTION DURATIONS ASSUMING THE DOMINANT TRAFFIC DIRECTION OF EACH CONNECTION IS KNOWN OR UNKNOWN. THE COMPARISON SHOWS THE KNOWLEDGE OF TRAFFIC DIRECTION ONLY HELPS A LITTLE BIT IN DECREASING THE SUM OF CONNECTION DURATIONS.

Algorithm	Trace1 Incoming (seconds)	Trace1 Outgoing (seconds)	Trace2 Incoming (seconds)	Trace2 Outgoing (seconds)
Bi-Directional	45,462	34,924	4,474	3,885
Uni-Directional	36,475	35,034	2,623	2,685

Trace2. Although it requires additional information on the dominating traffic direction of each connection, uni-directional load balancing algorithms appear to be worthwhile in practice.

V. CONCLUSION

The advent of inexpensive broadband connections motivates the development of multihoming load balancing technology to convert a set of access links that are somewhat unreliable and do not have predictable performance into a high-performance and robust Internet connection, much like a RAID controller turns a set of inexpensive PC-based disks into a high-performace disk subsystem that is comparable in reliability to custom-made mainframe-class disks. Despite its promising role in future Internet connectivity solutions for small and mid-sized enterprises, very little is published in the open literature about its design tradeoffs and implementation considerations. The first contribution of this paper is a comprehensive analysis of the design space of multihoming load balancing systems.

The second contribution of this paper is that it presents the first quantitative measurements to evaluate the effectiveness of various design decisions of load balancing algorithms. These measurements are collected by running a real packet trace against a commercial multihoming load balancing system. Among the important lessons from this study are

- Although hashing based link selection algorithm is stateless, its effectiveness is comparable to load based link selection algorithm,
- Performance difference between connection-granularity and packet-granularity load balancing algorithms is relatively modest, suggesting NAT-based load balancing schemes are quite adequate in practice, and
- Balancing the loads on the access links separately for each traffic direction is worthwhile, although it requires additional information on the dominating traffic direction information for each connection.

As for future work, we plan to carry out a more thorough study on the performance behavior of other aspects of a multihoming load balancing system, including the comparison between active probing and passive monitoring, the tradeoff between effectiveness and performance overhead of latency-based link selection algorithm, the performance cost of integrating node and link fault tolerance, etc.

ACKNOWLEDGEMENTS

Thanks to the other team members of the ISMD groups. They are Sheng-I Doong, Jie Zhang, Ryan Jan, Nikolai Joukov, Aparajita Choudhury and Kalyan Mergu.

REFERENCES

[1] T. Bates, Y. Rekhter, “Scalable Support for Multi-homed Multi-provider Connectivity”, RFC2260

[2] Y. Rekhter, T. Li, “An Architecture for IP Address Allocation with CIDR” RFC1518

[3] Cisco, “Sample Configurations for Load Sharing with BGP in Single and Multihomed Environments” http://www.cisco.com/warp/public/459/40.html

[4] Cisco, “How Does Load Balancing Work?” http://www.cisco.com/warp/public/105/46.html

[5] RouteScience http://www.routescience.com

[6] netVmg http://www.netvmg.com/

[7] Radware http://www.radware.com/

[8] F5 Networks http://www.f5.com/

[9] Nortel Networks http://www.nortelnetworks.com/

[10] Rether Networks Inc. http://www.rether.com/

[11] FatPipe http://www.fatpipeinc.com/

[12] Cisco, “Enabling Enterprise Multihoming with Cisco IOS Network Address Translation” http://www.cisco.com/warp/public/cc/pd/iosw/ioft/ionetn/tech/emios_wp.htm

[13] Zhiruo Cao, Zheng Wang, and Ellen Zegura, “Performance of Hashing-Based Schemes for Internet Load Balancing” Proc. of IEEE INFOCOM ’00

[14] A. Shaikh, R. Tewari, and M. Agrawal, “On the effectiveness of DNS based server selection,” Proc. of IEEE INFOCOM ’01

[15] Tao Zhou, “Build Redundant IP Routing” Windows 2000 Magazine Jul. 2000 http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnw2kmag00/html/Redundant2.asp

[16] Microsoft, “TCP/IP Dead Gateway Detection Algorithm Updated for Windows NT” http://support.microsoft.com/default.aspx?scid=kb;EN-US;q171564

[17] Krishna P. Gummadi, et al., “King: Estimating Latency between Arbitrary Internet End Hosts” Internet Measurement Workshop 2002

[18] Prashant Pradhan, Tzi-cker Chiueh, Anindya Neogi, “Aggregate TCP Congestion Control Using Multiple Network Probing” International Conference on Distributed Computing Systems (ICDCS), April 2000

[19] R. Braden, “Requirements for Internet Hosts - Communication Layers” RFC1122

[20] Srikant Sharma, et al. “Duplex: A Reusable Fault Tolerance Extension Framework for Network Access Devices” Proc. of 2003 International Conference on Dependable Systems and Networks (DSN 2003), June 2003

[21] Route Views project http://antc.uoregon.edu/route-views/