A dynamic traffic sharing with minimal administration on multihomed networks

Nariyoshi Yamai
Okayama University

Kiyohiko Okayama
Okayama University

Hiroshi Shimamoto
Okayama University

Takuji Okamoto
Okayama University

A Dynamic Traffic Sharing with Minimal Administration on Multihomed Networks

Nariyoshi Yamai   Kiyohiko Okayama   Hiroshi Shimamoto   Takuji Okamoto

Okayama University

3-1-1 Tsushima-naka
Okayama 700-8530 JAPAN

Abstract - Multihomed network is one of the most efficient configuration to improve response time of network services. However, it is hard to introduce or manage because the existing configuration methods have several problems in that they require much technical skill, involve administrative over-burden for the administrator and so on. In this paper, we propose a dynamic traffic sharing technique and a suitable backbone selection metrics to address some of these problems. Using the proposed technique, an appropriate backbone can be selected per connection with minimal technical skill and low administrative cost. In addition, the proposed metrics performs more efficient traffic sharing as compared to others techniques that were investigated.

1 Introduction

Recently, proliferation of the Internet causes a serious problem to the users in that the response time of the most Internet services, such as WWW, FTP and so on, has been getting degraded. As one technique to improve the response time, multihomed network, which is a kind of network connected to the Internet via more than one backbones, has attracted much attention.

However, the existing operation methods of multihomed networks have some problems.

For example, BGP routing[1], which is one of the most well-known operation method, requires much technical skill and administrative cost in exchanging the routing information with each backbone. In addition, since all the packets to the same destination go through the same backbone, traffic congestion may occur on a backbone while little traffic exists on the others especially when traffic to the same destination is outstanding.

Another well-known operation method is policy-based routing[2], which uses the pair of the source and the destination addresses for routing. With this method, outgoing traffic could be shared by more than one backbones even if traffic is imbalanced. However, since the incoming path is selected independent of the outgoing path, incoming traffic might be congested on a backbone even in the case. This method also requires more technical skill and administrative cost than those of BGP routing.

In this paper, we propose a dynamic traffic sharing method to solve these problems. In the proposed method, while establishing a TCP connection, the router connecting to the internal network and backbones probes the condition of each backbone and then selects an appropriate one through which all the following packets of the connection flow. Since our method does not depend on routing information from backbones, required technical skill and administrative cost are considerably low. Along with these features, the proposed method shares not only the outgoing traffic but also the incoming traffic with more than one backbones by introducing NAT (network address translation) function[3].

2 Outline of Our Method

As mentioned in the previous section, the existing operation methods of multihomed networks have some problems. Most of them are caused by the existing routing method based on the external routing information from backbones. To solve these problems and to share traffic with backbones efficiently, we introduce a routing method based on the current condition of backbones that the router can obtain by itself.

In this section, we describe the outline of our routing method.

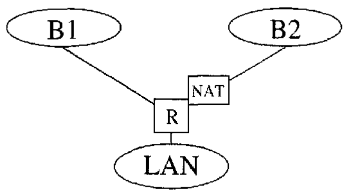


Figure 1: The configuration of the target network

2.1 Network Configuration

In the rest of this paper, we consider a LAN with two backbones, namely B1 and B2 respectively[1]. The method can be applied to a network configured as shown in Figure 1.

In this network, there exists a router R connecting to both backbones, where B1 is connected directly and B2 is connected with NAT function. Thus, the addresses used in the LAN are announced to the Internet by B1, and the addresses assigned by B2 are not used in the LAN. Note that the LAN is not a transient network, i.e. no packets from B1 are forwarded to B2 and vice versa.

2.2 Traffic Sharing Method

On the network configured as shown in Figure 1, this method shares traffic on TCP connections originating within the LAN. As for traffic on TCP connections originating outside the LAN, some well known techniques such as DNS round robin[4] could be made available, therefore we do not discuss this aspect in the rest of this paper. We do not consider any UDP traffic either; the current version of our program forwards all the UDP packets from the LAN to B1 and vice versa.

In the followings, we describe how to share the traffic along each direction.

2.2.1 Outgoing Traffic Sharing

As for outgoing traffic, the router R looks for all the packets with the SYN flag from the LAN, which request to establish new connections. Whenever R finds such a packet, R probes the current condition of each backbone and selects an appropriate one for that connection. Once a backbone is selected, the following packets of the connection flow on the same backbone.

[1]Our method could be introduced to a LAN with more than two backbones.

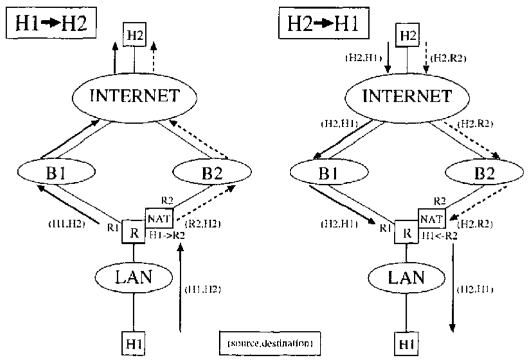


Figure 2: Paths of the outgoing and incoming packets

2.2.2 Incoming Traffic Sharing

Incoming traffic is shared by virtue of NAT function. Figure 2 shows how to share incoming traffic using NAT function, where there are some connections between two computers, namely H1 and H2.

At first, consider the case where R has selected B1 for a connection. As shown with solid lines in Figure 2, a packet sent from H1 on the connection is forwarded to B1 as it is. When H2 sends back a packet on the same connection, since the addresses used in the LAN, including the address of H1, are assumed to be announced by B1, the packet goes back to H1 through the same backbone B1.

Further, consider the case where R has selected B2 for a connection. As shown with dashed lines in Figure 2, a packet sent from H1 on the connection is received by R and forwarded to B2, with translation of its source address from H1 into R2, which is assigned by B2. When H2 sends back a packet on the same connection, since its current destination is not H1 but R2, it goes back to R though B2. Then R translates its destination address from R2 into H1, and sends it to H1.

Consequently, all the packets belonging to a connection flow through the same backbone regardless of the direction. This implies that backbones share not only the outgoing traffic but also the incoming traffic.

3 Backbone Selection

The metrics for backbone selection is very important in our method since it affects efficiency of traffic sharing. In this section, we propose a new metrics and show how it could be used to measure the current condition of backbones.

Since this metrics is used when establishing a connection, it should have the following properties.

(P1) It indicates the current condition, rather than the past condition, of backbones.

(P2) It can measure the condition of backbones in short period and at low cost.

(P3) It can be applied to any connections regardless of their destinations.

(P4) It never selects a backbone whose path reports some failure during selection.

Several researchers have proposed many methods to measure network condition, such as Bprobe[5], pathchar[6], TReno[7], and so on. However, since these methods send so many ICMP or UDP packets for precise measurement, they require much time and bandwidth. In addition, since ICMP or UDP packets to some destinations may be filtered out, they are not available to all connections. Thus, they have neither of properties P2 nor P3.

On the other hand, in order to share the load of a virtual server with several physical servers, LocalDirector[8] selects a host as the destination of each connection. Some selection criteria are available on LocalDirector, such as round robin, the number of connections, and so on. However, since these criteria are designed for LANs where all servers are connected in the same condition, they do not seem suitable for multihomed networks. In addition, they do not have properties P4 because they do not check the performance of each backbone.

To solve these problems, we propose a new metrics of backbone selection based on connection set-up time. With this metrics, the router tries to establish a connection through each backbone simultaneously and selects the first backbone through which a connection has been actually established. In the followings, we show the selection procedure in detail using Figure 2.

1. H1 sends a packet with the SYN flag (a SYN packet) to H2 for establishing a connection.
2. When R receives this packet, R duplicates it and sends its copy to each backbone. The source address of a copy for backbone B2 is translated appropriately with NAT function.

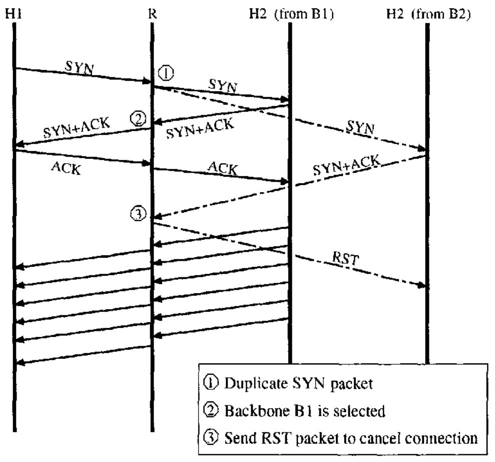


Figure 3: A packet transmission diagram on backbone selection

3. H2 receives each copy and sends back a packet with the SYN and the ACK flags (a SYN+ACK packet) respectively.
4. R receives the first SYN+ACK packet from a backbone and forwards it to H1, with destination address translation if necessary. Consequently, the connection is established and is associated with this backbone so that the following packets flow through the same path.
5. When R receives the second SYN+ACK packet from another backbone, R discards it and sends a packet with the RST flag (a RST packet) to H2 through the same path to cancel the connection.

An example of this procedure is depicted in Figure 3, where B1 is selected. Note that in this procedure, no routing information is used for backbone selection.

Connection set-up time is equivalent to round trip time (RTT) measured by sending an ICMP echo packet. In fact, both of them have properties P1, P2, and P4. However, as for P3, while ICMP packets may be filtered out, SYN packets cannot be filtered out whenever a connection can be established. Thus, our metrics has all desired properties described above, including P3.

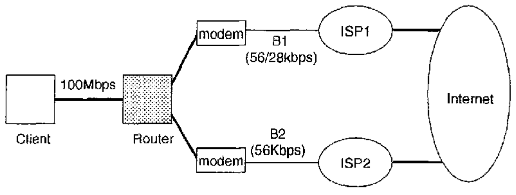


Figure 4: The configuration of the experimental environment

4 Performance Evaluation

4.1 Evaluation Environment

To evaluate the performance of the proposed metrics, we implemented a prototype router with traffic sharing function mentioned in the previous chapters. This router was implemented on a PC with 450MHz Pentium II running FreeBSD 2.2.7R, which had three network interfaces. The routing and traffic sharing functions were performed by a user process, which receives all packets with *divert* function and forwards them with BPF (Berkeley Packet Filter) interface, without kernel modification.

Performance of the prototype router was evaluated in the environment shown in Figure 4, where the router was connected to the Internet through two ISPs (Internet service providers), referred to as ISP1 and ISP2, respectively. The dial-up line between the router and ISP1, referred to as B1, had bandwidth of 56kbps (referred to as Sym) or 28kbps (referred to as Asym). Further, the line between the router and ISP2, referred to as B2, had bandwidth of 56kbps in both cases.

In this environment, the client generated several HTTP connections simultaneously. All HTTP connections were generated by *webjamma*[9]. Since the original webjamma was designed to keep a constant number of HTTP connections at any time, we modified it so that the interval of HTTP connection generation was exponentially distributed. However, it still had restriction on the maximum number of connections, which we set to 30. The average intervals used in the simulation experiments were 0.6, 1.2, and 1.8 seconds. In each simulation run, 3600 URLs were given to webjamma, which were extracted from the access log of the HTTP proxy in Computer Center, Okayama University.

In the simulation experiments, the average transmission rate per connection with the proposed metrics (referred to as Setup) was observed. Those with the number of connections (referred to as Nconn) and round robin (referred to as RR) as backbone selection criteria were also observed for comparison. As for Nconn, the backbone with the least number of connections was always selected even in Asymmetric condition, assuming that bandwidths of backbones were unknown.

For each combination of a backbone bandwidth (Sym and Asym), a backbone selection metrics (Setup, Nconn, and RR), and an average interval of connection generation (0.6, 1.2, and 1.8 seconds), two simulation runs were performed.

Table 1: Average transmission rate

average interval	met-rics	av. rate(kbps)	
		Sym	Asym
0.6	Setup	6.4	5.0
	Nconn	5.2	4.1
	RR	4.7	3.2
1.2	Setup	13.8	9.3
	Nconn	11.2	6.8
	RR	10.6	4.5
1.8	Setup	24.7	16.0
	Nconn	24.1	9.7
	RR	21.0	7.6

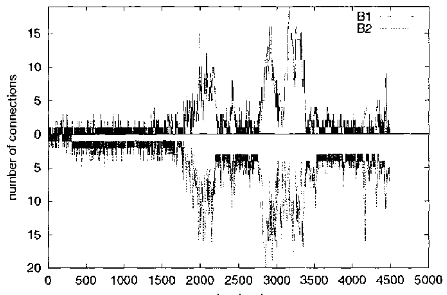
Table 2: Average rate of backbone selection

B1 bandwidth	met-rics	selection %	
		B1	B2
Sym	Setup	45.8	54.2
	Nconn	46.3	53.7
	RR	50.0	50.0
Asym	Setup	35.7	64.3
	Nconn	41.7	58.3
	RR	50.0	50.0

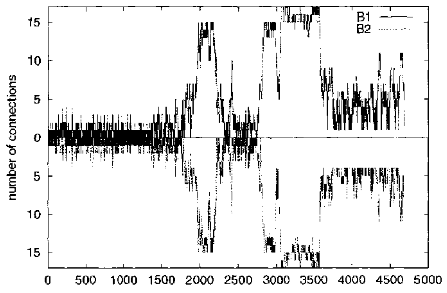
4.2 Simulation Results

The average transmission rate and the average rate of backbone selection of simulation experiments are shown in Table 1 and Table 2, respectively. The transition of the number of connections on each backbone selection criteria in the asymmetric condition is shown in Figure 5.

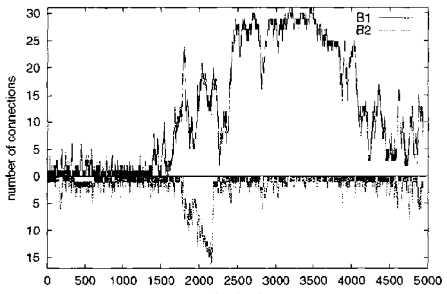
In all combinations, Table 1 shows that the average transmission rate of metrics Setup is larger than those of others. These results could be explained by the following facts. As for metrics RR, B1 and B2 are selected alter-



(a) In case of Setup



(b) In case of Nconn



(c) In case of RR

Figure 5: The transition of the number of connections in the asymmetric condition

natively regardless of their conditions. Since a connection tends to reside on the slower backbone in a long time, so many connections are on the same (slower) backbone as shown in Figure 5(c), and this causes the deterioration of performance. As for metrics Nconn, since both backbones are equally used at any time as shown in Figure 5(b), its performance is better than that of RR. However, since the faster backbone is selected not so assertively, it does not show the best performance. On the other hand, since metrics Setup selects the faster backbone assertively as shown in Table 2 and Figure 5(a), it shows the best performance.

5 Conclusion

In this paper, we have proposed a dynamic traffic sharing method with NAT function and a backbone selection metrics based on connection set-up time. The proposed method is easy to implement and performs efficient traffic sharing.

However, our method currently does not support UDP traffic sharing function. Further works include development of this function and appropriate selection metrics for UDP traffic.

References

[1] Rekhter, Y. and Li, T.: *A Border Gateway Protocol 4*, RFC1771 (1995).

[2] Cisco Systems, Inc.: *Policy-Based Routing*, http://www.cisco.com/warp/public/cc/techno/protocol/tech/plicy_wp.htm (2000).

[3] Egevang, K. and Francis, P.: *The IP Network Address Translator(NAT)*, RFC1631 (1994).

[4] Cisco Systems, Inc.: *Scaling the Internet Web Servers*, http://www.cisco.com/warp/public/cc/pd/cxsr/400/tech/scale_wp.htm (2000).

[5] Carter, R. L. and Crovella, M. E.: *Measuring Bottleneck Line Speed in Packet-Switched Networks*, Tech. Rep. BU-CS-96-006, Boston University (1996).

[6] Jacobson, V.: *pathchar — A Tool to Infer Charasteristics of Internet Paths, MSRI*, ftp://ftp.ee.lbl.gov/pathchar/msri-talk.pdf(1997).

[7] Mathis, M. and Mahdavi, J.: *Diagnosing Internet Congestion with a Transport Layer Performance Tool*, Proc. of INET'96 (1996).

[8] Cisco Systems, Inc.: *LocalDirector in the Data Center*, http://www.cisco.com/warp/public/cc/pd/cxsr/400/tech/ldir_wp.htm.

[9] Wooster, R. P. and Abrams, M.: *Proxy Caching that Estimates Page Load Delays*, Proc. 6th World Wide Web Conference, pp.325–334 (1997).