Active Load Balancing Technique for Heterogeneous IP Networks

Satoru Ohta and Toshiaki Miyazaki
NTT Network Innovation Laboratories, NTT Corporation
e-mail: {satoshi,miyazaki}@exa.onlab.ntt.co.jp

Abstract *Real world IP networks are heterogeneous in terms of server/link capacities. Such networks need a new load-balancing technique to avoid congestion. This paper describes how the active network technology can satisfy this requirement effectively.*

1 Introduction

Server congestion must be avoided if IP (Internet Protocol) networks are to provide good service quality. Load balancing by setting many mirrored servers is a well-known way of decreasing the possibility of server congestion. Besides server congestion, link or network congestion also degrades the service quality provided by networks. Therefore, the best load balancing technique should prevent traffic from concentrating on particular servers or network resources.

Given the importance of load balancing, a considerable number of methods have been proposed in the literature and added to networks. The simplest method is to use round-robin address search in the DNS (Domain Name System) [1]. It is also possible to introduce proxy servers that pass a service request to an appropriate mirror server [2]. Reference [3] evaluates a method that uses a device called the HTTP scheduler, which balances server load by rewriting IP packet headers. One technique that can be used to balance loads over mirror servers is known as "anycast" [4, 5]. Several variations of the anycast technique have been also proposed [6, 7].

Real world IP networks are heterogeneous in terms of server and link capacities. In such networks, existing load balancing techniques may not work successfully. If the servers have different capacities and the load balancing mechanism employed offers the same load to each server, the servers with smaller capacities will be congested. Similarly, the network may contain links with a mixture of high and low capacities. In this situation, heavy loads should not be routed through low capacity links. Thus, a flexible and sophisticated load balancing mechanism is needed to deal with both forms of heterogeneity.

The active network technology [8] was applied to the load balancing problem in Ref. [7]. This approach is interesting because it distributes loads according to server performance through the power of active network technology. With this approach, however, each active node must measure and compute the performance of every mirror server. This is not easy when the network has many servers.

This paper presents a new anycast protocol; the protocol requires the use of the active network technology. Actually, the protocol is an extension of a multicast technique [9], which was implemented on the basis of the active network technology. The protocol can distribute loads across servers according to various criteria by executing different computational procedures at active nodes. By exploiting the most appropriate procedure, the method is able to balance the load to match the heterogeneous server and link capacities. Additionally, the control loads placed on active nodes do not increase greatly even if many mirror servers are used. These features are confirmed by computer simulations.

The paper first describes the proposed protocol in Sec. 2. The evaluation model is explained in Sec. 3. Section 4 presents the evaluation results. Finally, Sec. 5 describes our conclusions.

2 Protocol

The proposed protocol assumes a network composed of active nodes. Each active node acts as a transit node and maintains routing tables. A routing table has entries, each of which includes an element called "weight" as well as the address of the next hop. When a service request from a client arrives at an active node, the next hop is determined by selecting a table entry through some computation on the weight element values.

To manage routing tables and set weight element values appropriately, the protocol uses "keep alive" packets. A server sends keep alive packets to each active node periodically. A keep alive packet carries a field called "measure", which active nodes can use to compute the weight element values of the routing table entries. By entering server or link capacity information into the measure field of a keep alive packet, it becomes possible for an active node to relate the weight element to server and link capacities.

If an active node receives a keep alive packet, the node that sent the packet is recorded as the next hop in the routing table entry. Thus, the chain of next hops written in the tables from the keep alive destination to the source forms a route in the reverse direction. A distinct routing table is created and managed for each keep alive packet destination. The set of routes associated with a table forms a rooted and directed tree because the table includes keep alive routes destined for one particular active node. Since a tree can not contain loops, looping is prevented if a packet route is selected from the tree [10]. By contrast, existing multicast routing protocols have a problem of looping for multipath routing [5]. Simultaneously, the node calcu-

lates the weight element value from the measure field in the received keep alive packet.

The keep alive packet is forwarded if the active node is not the destination and if the period from the last keep alive packet is sufficiently long. This forwarding procedure prevents the keep alive load from increasing greatly with server number [10].

The above protocol can be the basis for various load balancing schemes by altering the measure field setting for keep alive packets, the weight computing procedure, and the next hop selection for service request packets. These schemes include the following.

A. Minimum Hop Scheme

With this scheme, a service request is sent to the nearest mirror server. To do this, a server sets the measure field of a keep alive packet to 0. Let m denote the measure field value of a packet. When a keep alive packet arrives at an active node, the weight of the associated table entry is set to $m + 1$. Before the keep alive packet is forwarded by the active node, value m is replaced by the minimum weight value found among all entries in the table. The service request packet is sent to the node specified in one of the table entries that share the minimum weight value.

B. Random Scheme

A service request is sent to one of the nodes listed in the table with equal probability. The values of measure and weight have no influence on packet forwarding in this scheme.

C. Server-Capacity-Oriented Scheme

This scheme apportions the service requests among the mirror servers; the probability of selection is proportional to mirror server capacity. Initially, a server sets measure value m of a keep alive packet to its own capacity. When an active node receives a keep alive packet, the weight element of the associated table entry is set to value m. When forwarding a keep alive packet, an active node sets m to the sum of the weight element values in the table. Meanwhile, an active node randomly routes a service request packet to the node indicated in an entry with probability proportional to its weight element value. Namely, if w_i is the weight element value of the i-th entry, the node shown in the entry is selected with probability $w_i / \sum_j w_j$.

D. Server-Link-Capacity-Oriented Scheme

This scheme issues a service request to a server upon considering the capacities of servers and links. A server sets value m of a keep alive packet to its capacity. This assumes that the link capacity is represented by the number of service request packets that can be transmitted per unit time. Thus, if link bandwidth is B bits/sec, and if the average size of a service request is S bits, the link capacity is B/S packets/sec. Upon the arrival of a keep alive packet, the active node compares value m with the capacity of the link from which the packet came. Next, the node sets the weight element value of the associated table entry to the smaller of value m and the link capacity. The remainder of this scheme follows the server-capacity-oriented scheme.

E. Alternative Server-Link-Capacity-Oriented Scheme

This scheme is the same as the server-link-capacity oriented scheme except for the computation of the weight element in the table entry. Namely, if an active node receives a keep alive packet, the weight element in the distribution table entry is set to value, w.

$$w = \frac{1}{1/m + 1/C_l}, \tag{1}$$

where C_l is the capacity of the link from which the keep alive arrived.

3 Evaluation Model

The effectiveness of the proposed technique was evaluated by computer simulations. The first purpose of the evaluation was to clarify the significance of considering both server and link capacities in a heterogeneous network. The second aim was to confirm that active node control load increases only slowly as more servers are added.

Two simulation scenarios were considered. The first scenario, called Scenario 1, compared the performances of the load balancing schemes shown in Sec. 2 in a heterogeneous network. Scenario 2 measured the control load as server number was gradually increased to match an increase in client number.

For Scenario 1, the simulation model had 8 servers and 24 clients. To approximate the heterogeneous server situation, server capacities were set to 100, 400 or 1000 packets per second. Link capacities were also heterogeneous; the capacities of links connected to servers were 1Mb/s, 10Mb/s, or 100 Mb/s. The other links had a capacity of 100 Mb/s. The service requests sent from each client followed a Poisson process. Each service request packet occupied 12000 bits. For this parameter setting, the average time taken from service request generation to service completion, denoted as "service completion time," was compared among the load balancing schemes. Service completion time will increase if a server or a link is congested. Thus, it is considered to be a good comparison measure.

Scenario 2 used a network model consisting of 9 servers and 24 clients. At the beginning of the simulation, only one client sent service requests. Namely, the other 23 client nodes do not participate in the service. Thereafter, one additional client node joined the service every 60 seconds and started sending service requests. The number of working servers was also one at the beginning. That is, only the original server kept sending keep alive packets and thus all service requests were sent to that server. The number of working servers increased with the demand; one additional server started sending keep alive packets every 150 seconds. This setting simulated the situation wherein servers are gradually introduced to meet an increase in demand. In this scenario, the simulation counted the rate of keep alive packets sent and received by a specified active node. Since this rate indicates the control load against the number of servers, we can evaluate the scalability of the method. For Scenario 2, each server had the capacity of 400 packets per second, while every link had the capacity of 100 Mb/s. Other conditions were the same as for Scenario 1.

4 Evaluation Results

The results of Scenario 1 clearly show that the minimum hop and random schemes do not perform successfully in the heterogeneous situation. With the minimum hop scheme, the server closest to a client node does not always have sufficient capacity to meet the demand. Thus, a server with small capacity is very likely to be congested. The random scheme has a similar characteristic. Thus, these schemes perform poorly due to the congestion of small capacity servers.

The result of Scenario 1 also suggests that the server-capacity-oriented scheme performs slightly better than the minimum hop and random schemes, but does not offer the best performance. In the model examined, the capacities of the links connected to some servers are small compared to server capacities. Thus, if the load distribution is proportional to server capacities, these links become congested before the servers. Because of this link bottleneck, the server capacity is lower than recognized by the scheme. Thus, the server-capacity-oriented scheme is weak against heterogeneous link capacities.

The server-link-capacity-oriented scheme and the alternative server-link-capacity-oriented scheme offer the best performance. With these schemes, servers and bottleneck links are evenly loaded as the traffic increases so all resources are fully utilized. Since existing load balancing techniques are considered to have limitations similar to those of the minimum hop, random and server-capacity-oriented schemes, the proposed protocol is, implemented as the (alternative) server-link-capacity-oriented scheme, is obviously advantageous in a heterogeneous network.

As seen from the above argument, load balancing must consider both server capacities and link capacities. To do this, nodes must be equipped with some computational capability to determine the routing of service requests using some measure for server and link capacities. The active network technology is well suited to provide such nodes. We note that the computations performed will, most likely, change to meet changes in the network and services. The active network technology is essential to support these changes.

As demand grows, the number of mirror servers must be increased to avoid congestion. In such a situation, we might expect that the control load placed on the load balancing devices would increase. The result of Scenario 2 implies that the proposed method offers excellent scalability in this regard. Namely, although the number of servers increased nine times during the simulation, the result shows that the number of keep alive packets processed at the node increased only once. Except for that point, the control load was almost constant regardless of the addition of servers. This characteristic is favorable for the case where many mirror servers are employed in the network.

5 Conclusion

To achieve good load balancing, this paper presented an extended anycast protocol for active network implementation. It was shown that the protocol can yield several alternative load balancing schemes by modifying the computations performed. Since the proposed method distributes load considering both server capacities and link capacities, it is advantageous in fully utilizing the resource capacities of a heterogeneous network. This feature was confirmed through computer simulations. Another feature clarified by the simulations is that the control message load grows slowly against the server number. This result suggests that the proposed protocol offers excellent scalability.

References

[1] P. Mockapetris, "Domain names – concepts and facilities," RFC 1034, Nov. 1987.

[2] D. Andresen, T.Yang, V. Holmedahl, and O.H. Ibarra, "SWEB: towards a scalable World Wide Web server on multicomputers," in Proc. IPPS '96, pp.850–856, Honolulu, Apr. 1996.

[3] H. Bryhni, E. Klovning, and Ø. Kure, "A comparison of load balancing techniques for scalable web servers," IEEE Network, Vol.14, No.4, pp. 58–64, July/August 2000.

[4] C. Partridge, T. Mendez, and W. Milliken, "Host anycasting service," RFC 1546, Nov. 1993.

[5] E. Basturk, R. Engel, R. Haas, V. Peris, and D. Saha, "Using network layer anycast for load distribution in the Internet," in proc. Third Global Internet Mini-Conference, Sydney, Nov. 1998.

[6] S. Bhattacharjee, M.H. Ammar, E.W. Zegura, V. Shah, and Z. Fei, "Application-layer anycasting," in Proc. Infocom'97, pp.1388–1396, Kobe, 1997.

[7] M. Yamamoto, H. Miura, K. Nishimura, and H. Ikeda, "A network-supported server load balancing method: active anycast," IEICE Trans. Commun., Vol. E84-B, 6, pp.1561–1567, June 2001.

[8] D.L. Tennenhouse, J.M. Smith, W.D. Sincoskie, D.J. Wetherall, and G.J. Minden, "A survey of active network research," IEEE Communications Magazine, vol.35, 1, pp.80–86, Jan. 1997.

[9] S. Tani, T. Miyazaki, and N. Takahashi, "Adaptive stream multicast based on IP unicast and dynamic commercial attachment mechanism: an active network implementation," in Proc. IWAN 2001, Springer LNCS 2207, pp.116–133, Philadelphia, Oct. 2001.

[10] S. Ohta and T. Miyazaki, "Active load balancing for heterogeneous IP networks," Technical Report of IEICE, IN2002-66, pp. 79–84, Sept. 2002 (in Japanese).

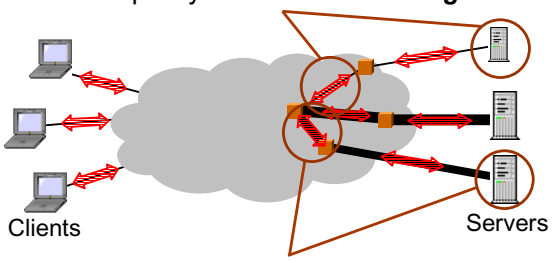
Active Load Balancing Technique for Heterogeneous IP Networks

Satoru Ohta and Toshiaki Miyazaki
{satoshi,miyazaki}@exa.onlab.ntt.co.jp
NTT Network Innovation Laboratories, NTT Corporation

Motivation

Heterogeneous IP Network

Small capacity servers & links: ***congested***



Large capacity servers & links: ***underutilized***

Good quality of service: load sharing among mirror servers

- Real world IP network: mixed server/link capacities.
- Existing load-balancing techniques are not appropriate in heterogeneous IP networks.
- New load-balancing technique is required.

Proposal

Anycast technique that utilizes active network technology

Protocol

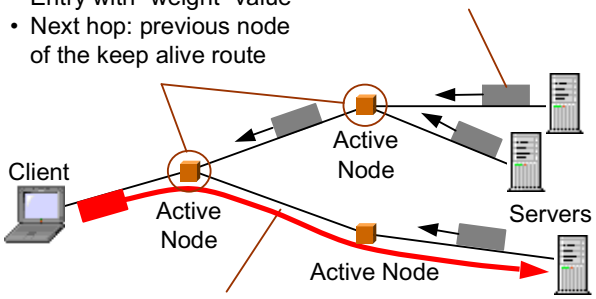
Processing at AN

Routing table:

- Entry with "weight" value
- Next hop: previous node of the keep alive route

Active keep alive packet:

- Sent from server to AN
- "Measure" field: capacity information



Service request: routed on the basis of weight values

- Forwarded to one server along a keep alive route
- Flexible route selection criteria through weight

Proposed Method: Features

- Different computational procedures at active nodes: various load balancing schemes
  - *A. Minimum Hop Scheme*
  - *B. Random Scheme*
  - *C. Server-Capacity-Oriented Scheme*
  - *D. Server-Link-Capacity-Oriented Scheme*
  - *E. Alternate Server-Link-Capacity-Oriented Scheme*

Effective congestion avoidance for heterogeneous server & link capacities

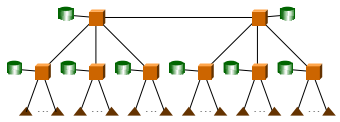
and also ...

- Keep alive forwarding mechanism → Slow increase of keep alive load against server addition
- Route management based on tree → Loop-free packet forwarding to one server

Evaluation Result

Simulation Scenario 1

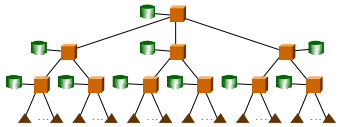
- Comparison among load-balancing schemes
- Server Capacities: 100-1000pps
- Link Capacities: 1-100Mbps



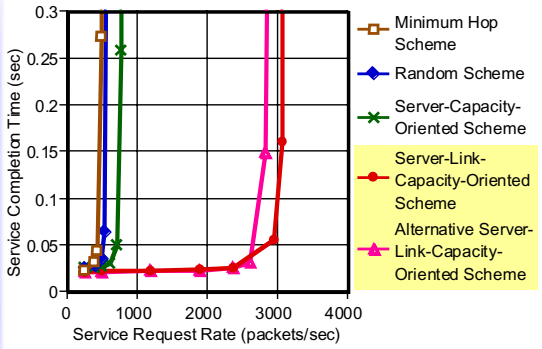
Service requests: Poisson arrival
■: Server ■: Active Node ▲: Client

Simulation Scenario 2

- Control load vs. server addition
- One additional server starts working every 150 sec

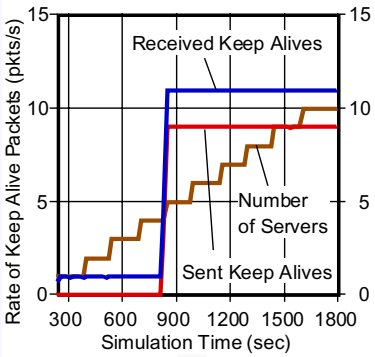


Scenario1: service completion time for various load-balancing schemes



Effectiveness of load-balancing that considers server and link capacities

Scenario2: Keep alive rate vs. server increase



Control load: insensitive to the increase of servers