This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the ICC 2007 proceedings.

Load Balancing in Mobile IPv6's Correspondent Networks with Mobility Agents

Albert Cabellos-Aparicio, Jordi Domingo Pascual
Departament d'Arquitectura de Computadors
Universitat Politècnica de Catalunya
Barcelona, Spain
{acabello,jordid}@ac.upc.edu

Abstract— **A foreseeable scenario is where on the Internet Mobile IPv6 is deployed and a large percentage of the clients are mobile nodes. These mobile clients will communicate with large servers, which under the Mobile IPv6's point of view, will be Correspondent Nodes. Usually large servers operate in servers farms with a load balancer device. Mobile clients can communicate with these servers through their Home Agent (a sub-optimal path) or directly by using the built-in mechanisms of Mobile IPv6 Route Optimization. In this paper we detail an important incompatibility between the Mobile IPv6's Route Optimization and several load balancing techniques. This means that mobile clients need to revert to the sub-optimal path when communicating with these server farms. This issue reduces considerably the communications performance increasing the delay and the infrastructure load. Moreover it may be an important drawback when considering Mobile IPv6's deployment. In this paper we show which load balancing techniques are incompatible with Route Optimization and we propose a novel mobile entity that solves this issue for several load balancing techniques.**

Mobile Communications, Load Balancing, Mobile IPv6, Route Optimization

I. INTRODUCTION

Wireless technologies have rapidly evolved in recent years. IEEE 802.11 is one of the most used wireless technologies and it provides up to 54Mbps of bandwidth in an easy an affordable way. In current Internet status, a user can be connected through a wireless link but he cannot move without breaking the IP communications. That's why IETF designed Mobile IP which provides mobility to the Internet. With "mobility" a user can move and change his point of attachment to the Internet without losing his network connections.

The IETF has designed two versions of Mobile IP, one for IPv4 and another one for IPv6. Although Mobile IPv6 [1] is very similar to Mobile IPv4 it is more efficient and avoids some problems suffered by Mobile IPv4.

In Mobile IPv6 a Mobile Node (MN) has two IP addresses. The first one identifies the MN's identity (WHO) while the second one identifies the MN's current location (WHERE). The MN will always be reachable through its WHO IP address while it will change its WHERE IP address according to its movements. A special entity called Home Agent placed at the MN's home network will maintain bindings between the MN's WHO and WHERE addresses. The communications between the MN and its peers (Correspondent Nodes) will be routed through the Home Agent. Unfortunately packets routed through the Home Agent follow a sub-optimal path. If the MN wants to communicate directly (*Route Optimization*) it has to inform to its Correspondent Nodes (CN) about its location changes by using a special procedure called Return Routability. Obviously this requires some sort of support at the Correspondent Nodes.

A foreseeable scenario is where, on the Internet, Mobile IPv6 is deployed. In such this scenario a percentage of the clients are MNs. These mobile clients will communicate with large servers, which under the Mobile IPv6's point of view, will be Correspondent Nodes. Usually large servers operate into server farms with a load balancer device. Client requests are distributed by the load balancer among the servers in order to increase resource utilization and decrease computing delay. In this paper we detail an incompatibility between several load balancing techniques and the Mobile IPv6's Return Routability. In such this scenario, Mobile IPv6 clients cannot benefit from Route Optimization when communicating with these servers farms. The communications must be routed through the Home Agent using a sub-optimal path. This issue reduces considerably the communications performance increasing the delay and the infrastructure load. Moreover, it may be an important drawback when considering Mobile IPv6's deployment.

In this paper we present a new mobile entity called Mobility Agent which will act as a front-end for the different load balancing devices. The Mobility Agent will hide mobility related issues to the load balancers allowing Mobile IPv6 clients to communicate directly. This novel entity process Mobile IPv6 Return Routability's messages on behalf the load balancer and the servers. In this way mobile clients can communicate directly with the servers avoiding the problems suffered by sub-optimal paths. Moreover with our solution Mobile IPv6 can be deployed flawlessly because it does not require server (CN) support. Mobile IPv6's deployment for such servers is as easy as plug and play. Finally, it is important to remark that our Mobility Agent does not require modifying the CN's kernel, the MNs or the Mobile IPv6 standard.

In the following section a Load Balancing techniques overview is presented. Our motivation is discussed in section III. Section IV presents our Mobility Agent while section V

This work was partially funded by IST under contract IST-2006-NoE-0384239 (IST-CONTENT), MEC (Spanish Ministry of Education and Science) under contract TSI 2005-07520-C03-02 and the CIRIT (Catalan Research Council) under contract 2005 SGR 00481.

1-4244-0353-7/07/$25.00 ©2007 IEEE

shows an evaluation. The related work is detailed in section VI, finally section VII is devoted to the conclusions of our work.

II. LOAD BALANCING TECHNIQUES OVERVIEW

This section presents an overview of the different existing load balancing techniques. Although these techniques can be applied for any service in this paper we focus on web services load balancing techniques.

Web server administrators face the challenge of increase web server capacity as the Internet grows up. The first option is to add more hardware resources or improve the web server itself. While these strategies relieve short-term pressure it is neither a cost-effective nor long-term solution. A more appealing solution is to deploy a distributed web server with multiples nodes. Some system component is needed to distribute client requests among the servers. The multiple web servers are loosely coupled and under the client's point of view act as a single server. Depending if this virtualization is extended to the IP level or not there are two different techniques. In the following subsections these techniques are detailed.

A. Distributed Web Systems

A distributed web systems consists of a set of web servers whose IP addresses are visible to client applications and thus, the virtualization is not extended to the IP level. A client request is routed to a single web server that belongs to distributed web system by using two different approaches. The first one uses the DNS servers while the second one uses web servers to route incoming client requests.

1) DNS-Based Techniques: The DNS-based technique uses DNS servers to route incoming client requests to a target web server. This technique was initially presented in [2] and it is intended to geographically distributed web systems. DNS-routing is performed during the client lookup procedure. DNS Servers reply to DNS requests not with a single IP address but with a list of IP addresses (the servers' IP addresses). The list's order follows a certain policy. Usually the first returned IP address belongs to the nearest available server or to the less busy server. Basic DNS clients simply use the first entry and discard the rest.

This technique has the main drawbacks from the DNS hierarchy itself and TTL (Time to live) values [26]. Firstly DNS servers usually cache DNS replies since DNS information changes very little. This means that even if a server becomes unavailable some DNS servers may continue redirecting traffic to it. Secondly this technique may not distribute traffic uniformly just because O.S's do not usually make requests to the authoritative name servers but to their pre-configured name servers. Those name servers then forward the requests to the authoritative DNS servers and cache the reply. Finally, new information on the DNS hierarchy takes a while to propagate. This issue does not allow a site to quickly increase its capacity.

2) Web-Based Techniques: The second approach uses web servers to route client's requests. In this approach a single web server receives all the incoming clients' requests and redirects them to other web servers through the *HTTP redirection* [3] message or the *URL rewriting* mechanism [4]. The main drawback for these approaches is that they increase delay as every redirection requires the client to initiate a new TCP connection. Even more, the web server that redirects incoming clients' requests may be overloaded adding extra delay.

B. Cluster-Based Web Systems

Cluster-based web systems extend the virtualization to the IP level. In this technique a set of web servers that are interconnected through a high-speed network and in a single location can be viewed as a single computer. The cluster system is accessible under a single IP address, known as virtual IP address. This virtual IP address is configured at a front-end node that will handle all the incoming clients' requests. The front-end node, known as load balancer, intercepts the servers' communications to the Internet making the whole system transparent both to the clients and to the servers. The load balancer device is able to identify all the servers through a private IP address or a layer-2 address. This load balancer will distribute the inbound packets to a target server according to a certain policy. Mainly, there are two types of load balancers. The first type uses layer 4 information to make the routing decision while the second type uses the whole protocol stack to make the decision.

1) Layer 4 Load Balancers: Layer 4 Load Balancers assign packets that belong to the same TCP connection to the same server persistently. Thus clients are identified by a source IP address and port. There are different mechanisms to redirect the packets to the selected server.

The first mechanism, *Packet Rewriting* [5], is based on the IP Network Address Translation [6] (NAT) and it is implemented by many commercial products. The *Packet Rewriting* load balancer consists of a virtual server which has a virtual IP address. Clients will always send their requests to the virtual IP address. In turn, the load balancer will rewrite the destination IP address of the client's packet to the IP address of a server according to a given policy. Next, the load balancer will forward the packet. Then the server will process the packet. Server's responses will flow through the load balancer that will rewrite the packet's source IP address to its virtual IP address. In this way, clients will receive packets as they were sent from the virtual IP address. As it has been said before, when a given client has been redirected to a given web server further client's requests must be redirected to the same server.

The second mechanism is actually a set of mechanisms known as *One-way* architectures. In *One-way* architectures inbound packets pass through the load balancer device while outbound packets flow directly from the servers in order to avoid that the load balancer becomes the bottleneck of the whole system. There are different proposals of *one-way* architectures such as *Packet Tunneling* [7] and *Packet Layer-2 forwarding* [12].

2) Layer 7 Load Balancers: Layer 7 Load Balancers distribute client's requests according to information from the application level (HTTP). This way the load balancers device, acting as a TCP proxy, establishes a separate TCP connection with the client and with the target server in order to receive the whole HTTP request. In this case the load balancer can distribute

different HTTP requests from the same client to different servers because HTTP is a stateless protocol [3]. This technique is called *TCP Gateway* (which actually is a simple proxy).

The *TCP Splicing* [8] technique is an enhancement of the *TCP Gateway* technique where IP packets are forwarded from one endpoint to the other one without having to cross the TCP layer. Once the client-to-server binding has been established, the load balancer handles the subsequent packets by changing the IP and TCP headers so that the process is transparent for the client and for the server.

Layer 7 Load Balancers also work in *One-Way* architectures where outbound packets flow directly from the server to the clients. Approaches such as *TCP Handoff* [9] and *TCP Connection Hop* [10] (a proprietary mechanism) are good examples. With these approaches, the load balancer "hand offs" the TCP connection endpoint to the selected server. This mechanism is transparent to the client as data sent by the servers appear to be coming from the load balancer.

III. Problem Statement

The following subsections present a discussion of the incompatibility between MIPv6 and the different load balancing techniques.

A. Distributed Web Systems

Distributed Web Systems are compatible with MIPv6's RR because they do not extend the virtualization to the IP layer. The communications are always established between a MN and a single server without any further packet processing.

B. Layer 4 Load Balancers

MIPv6's RR is not compatible with any Layer 4 Load Balancing technique because it requires that some state is stored at the CN. The CNs must store a list of bindings between the MN's Home Addresses and the MN's CoA in a structure called Binding Cache [1].

Layer 4 Load Balancers are required to establish client-to-server bindings. In this way, each client has an assigned target server. Packets sent by the client are forwarded always to the same server. Upon a client connection establishment, the load balancer will identify the client according to its source port and IP address and will create the appropriate binding. Subsequent packets will be forwarded to the selected server according to this binding.

With MIPv6, data packets flow with the CoA (the temporal IP address) as source address. This address will change according to the MN's movements. Thus, load balancers cannot identify clients by inspecting the packet's source address.

Load balancers should identify MNs by their Home Address. This address will not change even if the MN changes its point of attachment. Each MIPv6's data and signaling packet includes the Home Address except for the Care-of Test Init message.

MNs send the Care-of Test Init message when they start the RR procedure due to a connection establishment or a handover. The message is used by the MN to request to the CN a "care-of keygen token". This token, combined with the "home keygen token" (requested trough the Home Test Init message) provides the binding key used to authenticate the Binding Update.

According to the information contained in this message the load balancer will not be able to identify the MN (client). This message includes the new Care-of Address that will be used by the MN and a "care-of init cookie" which is a newly generated random number. The reserved field has not been yet standardized and the MIPv6 RFC [1] does not define any Mobility Options for such message.

This information is not enough to relate it neither to the client nor to the stored state at the server. In other words, the load balancer cannot relate the Care-of Test Init message with any pre-established client-to-server binding. This means that the load balancer is unable to process this message and thus, the RR procedure will fail forcing the MN to communicate through the HA (sub-optimal path). An obvious option would be to forward the Care-of Test Init message to all the servers. In this case each server would reply to the MN with its own "care-of keygen tokens" leading to an authentication failure.

C. Layer 7 Load Balancers

The *TCP Gateway* technique is compatible with the MIPv6's RR because the load balancer creates separate TCP connections with the clients and with the servers. In this case the MN would perform the RR procedure with the load balancer. The RR's state would be stored at the load balancer. However the *TCP Splicing* technique is not compatible. As it has been explained in section II.B this technique also creates separate TCP connections but, in order to improve the performance of the *TCP Gateway* technique, it forwards IP packets from one endpoint to another without crossing the TCP layer. Packets are forwarded directly from one connection to another changing its IP and TCP headers appropriately. This means that the required MIPv6's RR state is stored at the server instead of at the load balancer. Once again, when a Care-of Test Init message arrives, the load balancer will be unable to identify the client and relate it to the appropriate client-to-server binding. *TCP Handoff* and *TCP Connection Hop* are not compatible with the RR procedure. These techniques also create client-to-server bindings and they forward the TCP connection state to the selected server. Even if these protocols were updated to forward also the required MIPv6's RR state the load balancers would fail to identify the client when a Care-of Test Init message arrived.

D. Summary

As we have shown, Distributed Web Systems techniques are compatible with mobile clients while the only Cluster Based Web System compatible technique is the *TCP Gateway* mechanism. In this paper we present a novel entity that will act as a load balancer front-end that will allow Route Optimized connections with the most common approaches, the *Packet Rewriting* and the *TCP Splicing* techniques.

IV. PROBLEM STATEMENT

This section presents our novel mobile entity.

A. Load Balancing Architecture

Fig. 1 presents our novel load balancer architecture which has two modules, the first one is called Mobility Agent. A Mobility Agent is a new mobile entity placed at the Correspondent Network that it is able to perform the RR procedure on behalf the Correspondent Nodes (i.e large servers). The second module is a regular load balancer device (a *Packet Rewriting* or a *TCP Splicing* device). We have not introduced any modification on these devices.

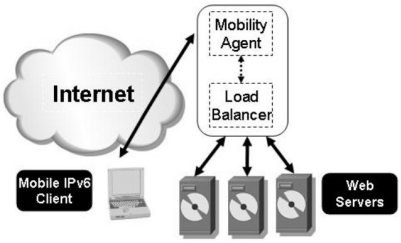

Figure 1. Proposed Load Balancer Architecture

Clients will address their requests to the server's network. The novel architecture will receive these packets that will be initially processed by the Mobility Agent module. If the packet belongs to a non-MIPv6 client (i.e it does not use Mobility Extension Headers) the Mobility Agent module will forward it to the load balancer device. In this case the load balancer will process the packet as usual. It will identify the client by inspecting the packet's source address and it will forward the packet to the selected server according to the client-to-server binding.

If the client is a MN that wants to establish a Route Optimized connection with the servers it will start the RR procedure. As it is detailed in subsection IV.B, the Mobility Agent module will process the RR's signaling on behalf the servers, hiding mobility related issues. This way the required MIPv6's RR's state will be stored at the load balancer and it will be able to reply to Care-of Test Init messages with its own "care-of keygen tokens".

Once the RR procedure has finished the MN will start to send data packets using the Extension Headers [1]. For each data packet the Mobility Agent will replace the MN's CoA for the MN's Home Address and it will process the Extension Headers hiding mobility issues to the load balancer (subsection IV.C). In this way the load balancer can process the packet as usual, as if the MN was a fixed node or at home. It can identify the MN by inspecting the packet's source address, in this case the Home Address.

B. Mobility Agents Operations

Fig. 2 shows how the Mobility Agent module performs the RR procedure on behalf the servers.

The Mobility Agent will act as a transparent proxy for the MIPv6 protocol, receiving and processing all the signaling messages. When the MN's Binding Update has been authorized it will store it and it will reply with a Binding Acknowledgement. The functionalities required by the Mobile Agent module are exactly the same to those provided by Correspondent Nodes as defined in Section 9 of the MIPv6 RFC [1].

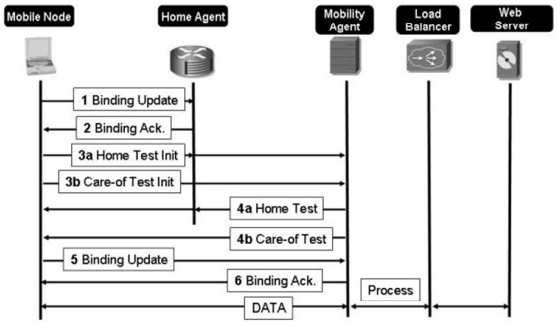

Figure 2. Mobility Agents interaction with RR

C. Mobility Agents Signaling Interaction

When the MN sends Route Optimized packets to the load balancer it includes the Home Address Option. Fig. 3 shows how they are processed by the Mobility Agent.

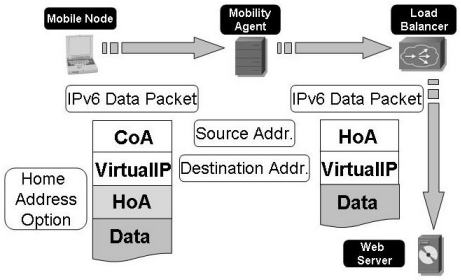

Figure 3. Home Address Option processing

When a data packet including a Home Address Option is received by the Mobility Agent module it will first check if it has a binding between the packet's source address (CoA) and the Home Address. If it has a binding it will remove the extension header and it will replace the packet's source address CoA for the MN's Home Address included into the Home Address Option. The Mobility Agent module will also set the IPv6's Next Header field according to the new headers. In this way the load balancer module will receive a packet from the MN's Home Address and it will process it as usual. This procedure is very similar to the MIPv6 CN's support. It is very important to remark that the TCP checksums must not be recomputed by our module. In fact, the MIPv6 RFC [1] states that these checksums must be computed with the Home Address instead of with the Care-of Address.

When the server sends packets to the MN in MIPv6 it includes a Routing Header, however with our novel load balancer the servers do not have MIPv6 support and thus, they send the packets as stated by the IPv6 RFC [11]. First the packet will be received by the load balancer that will process it as usual. Next, as shown in fig. 4, the packet will be received

by the Mobility Agent module that it will check if it has a binding for the packet's destination address (the MN's Home Address). If it does not have a binding it will forward it as defined in the IPv6 standard. However if a binding exists it will replace the packet's destination address with the MN's CoA and it will add the Routing Header Type 2 Extension Header. This extension header will include the MN's Home Address. The Mobility Agent will also set the Next Header field according to the new headers. This procedure is very similar to the MIPv6 CN's support. Once again, the TCP checksum must not be recomputed because the server has computed it with the Home Address. Moreover the load balancer has recomputed them to match to the Virtual IP address. The MIPv6 protocol states that the MN will verify this checksum with its Home Address and not with the actual packet source address, the Care-of Address.

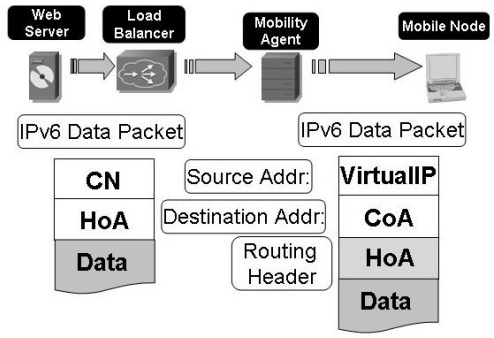

Figure 4. Routing Header processing

V. LOAD BALANCER EVALUATION

This section presents an evaluation of our novel load balancer architecture as well as the benefits that it provides.

A. Load Balancing Techniques Compatibility

Our Mobility Agent module provides compatibility for the *Packet Rewriting* and the *TCP Slicing* load balancing techniques with the MIPv6's RR. While Distributed Web Systems are yet compatible, our Mobility Agents provide compatibility for the many existing Cluster-Based Web Systems. Table II and III present a classification of several products that provide Layer 4 and Layer 7 Load Balancers. Tables are based on [12,26] and have been updated to reflect recent changes.

As we can see many commercial products use the *Packet Rewriting* technique and they can benefit from our Mobility Agent module. Many other products use the *Packet Forwarding* technique, unfortunately this is a *One-Way* approach where our Mobility Agent cannot provide MIPv6's RR compatibility.

As the table III shows the *TCP Splicing* technique is also used by many major Layer 7 Load Balancers vendors. These products can also benefit from our Mobility Agents.

TABLE I. LAYER-4 LOAD BALANCERS

Packet Rewriting	Packet Tunneling	Packet Layer-2 Forwarding
Cisco's LocalDirector [13] *LinuxVirtualServer [14]* *F5's BIG/IP [15]* *Foundry Networks ServerIron [16]* *IBM WebSphere Edge Server Network Dispatcher [17]* *Coyote Point Equalizer [18]* *Allot NetEnforcer [19]*	*LinuxVirtualServer [14]*	*IBM WebSphere Edge Server Network Dispatcher [17]* *Cisco's LocalDirector [13]* *LinuxVirtualServer [14],* *F5's BIG/IP [15]* *Foundry Networks ServerIron [16]* *IBM WebSphere Edge Server Network Dispatcher [17]* *Nortel Networks Application Switch [20]* *Radware's AppDirector [21]*

TABLE II. LAYER-7 LOAD BALANCERS

TCP Gateway	TCP Splicing	TCP Handoff	TCP Connection Hop
IBM WebSphere Edge Server Network Dispatcher [17]	*F5's BIG/IP [15]* *Foundry Networks ServerIron [16]* *Nortel Networks Web OS [20]* *Radware's AppDirector [21]* *Lucent Web Switch [22]* *Cisco CSS [13]* *Zeus ZXTM-LB [23]* *IBM WebSphere Edge Server Network Dispatcher [19]* *TCPSP [25]*	*TCPHA [24]*	*Resonate's Central Dispatch [10]*

Regarding Distributed Web Systems they are usually used in combination of some sort of Cluster-Based Web Systems for geographically distributed architectures. DNS servers redirect users to the closest web cluster. In this way load balancing is performed in two separate levels. First geographically in a coarse-grained approach and then selecting the "best" web server from the cluster.

B. Load Balancer Performance Evaluation

Without Mobility Agents MNs must communicate through the sub-optimal route with the servers. This subsection presents the main drawbacks of sub-optimal paths [27].

First, the longer route increases delay and infrastructure load. When the CN and the MN are close to one another but relatively far from the Home Agent the increase in delay is very large. Such increase may not be tolerated by time sensitive applications. Moreover such increase may affect the TCP protocol performance since the sending rate depends on the round-trip-time. Moreover, the total network resource utilization is higher due to the longer path.

Second the MN encapsulates [7] packets to the Home Agent and thus, the sub-optimal path leads to an increased packet overhead. This tunnel may also lead to an increase of the processing delay due to packet's encryption/decryption and

other verifications. This tunnel may also increase the chances of the packets being fragmented due to the increased packet size.

Third, as the sub-optimal path is longer it is less robust against link-failures. And finally, since all the packets to and from MNs are forwarded through the Home Agent, the Home Agent itself or the Home Link may be overloaded. This means that the Home Agent or the Home Link may become the bottleneck of the whole system. Moreover, a congested Home Agent can lead to additional packet delay or even packet loss.

VI. Related Work

At the best of the authors' knowledge the incompatibility between several load balancing techniques and the RR procedure has not been addresses so far. Nevertheless, several papers have presented solutions that run the RR procedure on behalf the Correspondent Nodes. In [31] the authors present an agent-based route optimization for Mobile IPv4. In their proposal, a special entity located at the Correspondent Network border router achieves Route Optimization on behalf the Correspondent Nodes. Data packets are tunneled between the special entity and the MNs.

In [32] authors propose a bi-directional route optimization for Mobile IPv4. With the authors' solution, a special entity called Correspondent Agent is placed at the correspondent network border router. This entity also achieves Route Optimization on behalf the Correspondent Nodes. Another special entity (Foreign Agent) is placed at the MN's visited network. The Correspondent Agent establishes a bi-directional tunnel with the Foreign Agent to send and receive data packets.

Our Mobility Agents is intended for MIPv6 instead of Mobile IPv4. Moreover, with our solution packets are not tunneled but sent using the mobility extension headers. These headers provide less overhead than the traditional tunneling technique.

VII. Conclusions

In this paper we have shown an incompatibility between several load balancing techniques and the MIPv6's Return Routability procedure. Due to this incompatibility mobile clients cannot benefit from Route Optimization connections leading to a sub-optimal path. We have detailed the loss of performance of sub-optimal paths which includes increased delay, round-trip-time and infrastructure load among others. We have also presented which load balancing techniques are affected by this incompatibility.

We have presented a novel entity called Mobility Agents that solves this incompatibility for two load balancing techniques. Our Mobility Agent acts as a front-end for the load balancing devices hiding mobility related issues to the load balancer itself and to the servers. Moreover, our Mobility Agent reduces the MIPv6's deployment cost because it provides Correspondent Node support (and Route Optimization) without modifying the servers. Finally, our solution is transparent both to the MNs and to the servers and it does not require modifying the MIPv6 standard.

Finally it is important to remark that we have shown that many existing commercial product use the *Packet Rewriting* and the *TCP Splicing* load balancing technique and they can benefit from our Mobility Agent module.

References

[1] D. Johnson et al. "*Mobility Support in IPv6"*. RFC 3775, June 2004

[2] T. Brisco "*DNS Support for Load Balancing*" RFC 1794, April 1995

[3] T. Berners-Lee, R. Fielding and H. Frystyk. "*Hypertext Transfer Protocol – HTTP/1.0*" RFC 1945, May 1996

[4] Q. Li et al. "*Distributed Cooperative Apache Web server*" WWWC, 2001.

[5] P.Srisuresh, D.Gan "Load Sharing using IP Network Address Translation (LSNAT)". RFC 2391, August 1998

[6] K. Egevang and P. Francis "*The IP Network Address Translator (NAT)*". RFC 1631, May 1994.

[7] C. Perkins. "*IP encapsulation within IP*". RFC 2003, Oct 1996

[8] A. Cohen et al. "On the performance of TCP splicing for URL-aware redirection" USENIX, Oct 1999

[9] V.S. Pai et al. "Locality aware request distribution in cluster-based network servers". ASPLOS, Oct 1998

[10] Resonate Inc. http://www.resonate.com

[11] S. Deering, et al. "Internet Protocol, Version 6 (IPv6) Specification". RFC 2460, December 1998

[12] V. Cardellini, E. Casalicchio, M. Colajanni, P.S. Yu, ``*The state of the art in locally distributed Web-server systems*", *ACM Computing Surveys*, Vol. 34, No. 2, pp. 263-311, June 2002.

[13] Cisco Systems http://www.cisco.com

[14] Linux Virtual Server Project http://www.linuxvirtualserver.org

[15] F5 Networks BIG/IP http://www.f5labs.com/products/bigip/

[16] Foundry Networks ServerIron http://www.foundrynet.com/products/webswitches/serveriron/docs.html

[17] IBM Network Dispatcher http://www.ibm.com/software/webservers/edgeserver/

[18] Coyote Point Systems Equalizer http://www.coyotepoint.com

[19] Allot Communications NetEnforcer http://www.allot.com

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

[21] Radware Inc. http://www.radware.com

[22] Lucent Web Switch: http://www.bell-labs.com/project/webswitch

[23] Zeus ZXTM-LB http://www.zeus.com

[24] TCPHA for Linux Virtual Server http://dragon.linux-vs.org/~dragonfly/htm/tcpha.htm

[25] TCPSP for the Linux Kernel: http://www.linuxvirtualserver.org/software/tcpsp/index.html

[26] Tony Bourke: *Server Load Balancing*, O'Reilly, ISBN 0-596-00050-2

[27] T. Clauser et al. "*NEMO Route Optimisation Problem Statement*" (Internet Draft), October 2004

[28] W. Haddad et al. "*Applying Cryptographically Generated Addresses to Optimize MIPv6 (CGA-OMIPv6)"* (Internet Draft) March 2006

[29] V. Devarapalli, R. Wakikawa, A. Petrescu, P. Thubert. *Network Mobility (NEMO) Basic Support Protocol*. RFC 3963, January 2005

[30] C. Perkins, Securing Mobile IPv6 Route Optimization Using a Static Shared Key. RFC 4449, June 2006

[31] R. Vadali et al. "Agent-Based Route Optimization for Mobile IP," *IEEE 54th Vehicular Technology Conference*, 2001.

[32] Chun-Hsin Wu et al. "*Bi-directional Route Optimization in Mobile IP over Wireless LAN*", IEEE Vehicular Technology Conf, 2002