P. Srisuresh
Lucent Technologies
Der-hwa Gan
Juniper Networks, Inc.

INTERNET-DRAFT
Category: Informational
Expire in six months

March 1998

Load Sharing using IP Network Address Translation (LSNAT)

<draft-srisuresh-lsnat-02.txt>

Status of this Memo

Preface

This document combines the idea of address translation described in Ref[1] with real-time load share algorithms to introduce Load Share Network Address Translators(or, simply LSNATs). LSNATs would transparently offload network load on a single server and distribute the load across a pool of servers.

Abstract

Network Address Translators (NATs) translate IP addresses in a datagram, transparent to end nodes, while routing the datagram. NATs have traditionally been been used to allow private network domains to connect to Global networks using as few as one

globally unique IP address. In this document, we extend the use of NATs to offer Load share feature, where session load can be distributed across a pool of servers, instead of directing to a single server. Load sharing is beneficial to service providers and system administrators alike in grappling with scalability of servers with increasing session load.

1. Introduction

Traditionally, Network Address Translators, or simply NATs were used to connect private network domains to globally unique public domain IP networks. Applications originate in private domain and NATs would transparently translate datagrams belonging to these applications in either direction. This document combines the characteristic of transparent address translation with real-time load share algorithms to introduce Load Share Network Address Translators.

The problem of Load sharing or load balancing is not new and goes back to many years. A variety of techniques were applied to address the problem. Some very ad-hoc and platform specific and some employing clever schemes to reorder DNS resource records. Ref[11] identifies DNS zone transfer program in name servers to periodically shuffle the order of resource records for server nodes based on a pre-determined load balancing algorithm. The problem with this approach is that reordering time periods can be very large in the order of minutes and does not reflect real-time load variations on the servers. Secondly, all hosts in the server pool are assumed to have equal capability to offer all services. This may not often be the case and there may be requirements to support load balancing for a few specific services only. The load share approach outlined in this document addresses both these concerns and offers a solution that does not require changes to clients or servers and one that can be tailored to individual services or for all services.

For the reminder of this document, we will refer NAT routers that provide load sharing support as LSNATs. Unlike traditional NATs, LSNATs are not required to operate between private and public domain routing realms alone. LSNATs also operate in a single routing realm and provide load sharing functionality.

The need for Load sharing arises when a single server is not able to cope with increasing demand for multiple sessions simultaneously. Clearly, load sharing across multiple servers

would enhance responsiveness and scale well with session load. Popular applications inundating servers would include Web

browsers, remote login, file transfer and mail applications.

When a client attempts to access a server through an LSNAT router, the router would select a node in server pool, based on a load share algorithm and redirect the request to that node. LSNATs pose no restriction on the organization and rearrangement of nodes in server pool. Nodes in a pool may be replaced, new nodes may be added and others may be in transition. Any changes to server pool configuration can be shielded from users by centralizing server pool change management to LSNAT router.

There are limitations to using LSNATs. Firstly, it is mandatory that all requests and responses pertaining to a session between a client and server be routed via the same LSNAT router. For this reason, we recommend LSNATs to be operated on a single border router to a stub domain in which the server pool would be confined. This would ensure that all traffic directed to servers from clients outside the domain and vice versa would necessarily traverse through the LSNAT border router. Later in the document, we will examine a special case of LSNAT setup, which gets around the topological constraint on server pool. Another limitation of LSNATs is the inability to switch loads between hosts, in the midst of sessions. This is because, LSNATs measure load in granularity of sessions. Once a session is assigned to a host, the session cannot be moved to a different host till the end of that session. Other limitations, inherent to NATs, outlined in ref[1] are also applicable to LSNATs.

As with traditional NATs, LSNATs have the disadvantage of taking away the end-to-end significance of an IP address. The major advantage, however, is that it can be installed without changes to clients or servers.

2. Terminology and concepts used

The terminology used in Ref[1] is borrowed almost verbatim here, with a few additions introduced here.

2.1. TU ports, Server ports, Client ports

For the reminder of this document, we will refer TCP/UDP ports associated with an IP address simply as "TU ports".

For most TCP/IP hosts, TU port range 0-1023 is used by servers listening for incoming connections. Clients trying to initiate a connection typically select a TU port in the range of 1024-65535.

However, this convention is not universal and not always followed. It is possible for client nodes to initiate connections using a TU port number in the range of 0-1023, and there are applications listening on TU port numbers in the range of 1024-65535.

A complete list of TU port services may be found in Ref[2]. The TU ports used by servers to listen for incoming connections are called "Server Ports" and the TU ports used by clients to initiate a connection to server are called "Client Ports".

2.2. Session flow vs. Packet flow

Connection or session flows are different from packet flows. A session flow indicates the direction in which the session was initiated with reference to a network port. Packet flow is the direction in which the packet has traversed with reference to a network port. A session flow is uniquely identified by the direction in which the first packet of that session traversed.

Take for example, a telnet session. The telnet session consists of packet flows in both inbound and outbound directions. Outbound telnet packets carry terminal keystrokes from the client and inbound telnet packets carry screen displays from the telnet server. Performing address translation for a telnet session would involve translation of incoming as well as outgoing packets belonging to that session.

Packets belonging to a TCP/UDP session are uniquely identified by the tuple of (source IP address, source TU port, target IP address, target TU port). ICMP query sessions are uniquely identified by the tuple of (source IP address, ICMP Query

Identifier, target IP address). For lack of well-known ways to distinguish, all other types of sessions are lumped together and distinguished by the tuple of (source IP address, IP protocol, target IP address).

2.3. Start of session for TCP, UDP and others

The first packet of every TCP session tries to establish a session and contains connection startup information. The first packet of a TCP session may be recognized by the presence of SYN bit and absence of ACK bit in the TCP flags. All TCP packets, with the exception of the first packet must have the ACK bit set.

The first packet of every session, be it a TCP session, UDP session, ICMP query session or any other session, tries to establish a

session. However, there is no deterministic way of recognizing the start of a UDP session or any other non-TCP session.

Start of session is significant with NATs, as a state describing translation parameters for the session is established at the start of session. Packets pertaining to the session cannot undergo translation, unless a state is established by NAT at the start of session.

2.4. End of session for TCP, UDP and others

The end of a TCP session is detected when FIN is acknowledged by both halves of the session or when either half sets RST bit in TCP flags field. Within a couple seconds after this, the session can be safely assumed to have been terminated.

For all other types of session, there is no deterministic way of determining the end of session. Many heuristic approaches are used to terminate sessions. TCP sessions that have not been used for say, 24 hours, should be safe to assume to have been terminated. Non-TCP sessions that have not been used for say, 1 minute, should also be safe to assume to have been terminated. However, these idle period Session timeouts may vary considerably across the board and should optionally be made user configurable. Another

way to handle session terminations is to timestamp sessions and keep them as long as possible and retire the longest idle session when it becomes necessary.

2.5. Load share

Load sharing for the purpose of this document is defined as the spread of session load amongst a cluster of servers which are functionally similar or the same. In other words, each of the nodes in cluster can support a client session equally well with no discernible difference in functionality. Once a node is assigned to service a session, that session is bound to that node till termination. Sessions are not allowed to swap between nodes in the midst of session.

Load sharing may be applicable for all services, if all hosts in server cluster carry the capability to carry out all services. Alternately, load sharing may be limited to one or more specific services alone and not to others.

Note, the term "Session load" used in the context of load share is different from the term "system load" attributed to hosts by

way of CPU, memory and other resource usage on the system.

3. Overview of Load sharing

While both traditional NATs and LSNATs perform address translations, and provide transparent connectivity between end nodes, there are distinctions between the two. Traditional NATs initiate translations on outbound sessions, by binding a private address to a global address (basic NAT) or by binding a tuple of (private address, local TU port) to a tuple of (global address, assigned TU port). LSNATs, on the other hand, initiate translations on inbound sessions, by binding each session represented by a tuple such as (client address, client TU port, virtual server address, server TU port) to one of server pool nodes, selected based on a real-time load-share algorithm. A virtual server address is a globally unique IP address that identifies a physical server or a group of servers that can provide similar or same functionality.

For the reminder of this document, we will refer traditional NATs simply as NATs and refer LSNATs exclusively in the context of load share, without implying traditional NAT functionality.

LSNATs are not limited to operate between private and public domain routing realms. LSNATs may operate within a single routing realm with globally unique IP addresses, just as well as between private and public network domains. The only requirement is that server pool be confined to a stub domain, accessible to clients outside the domain through a single LSNAT border router. However, as you will notice later, this topology limitation on server pool can be overcome under certain configurations.

Load Share NAT operates as follows. A client attempts to access a server by using the server virtual address. The LSNAT router transparently redirects the request to one of the hosts in server pool, selected using a real-time load sharing algorithm. Multiple sessions may be initiated from the same client, and each session could be directed to a different host based on load balance across server pool hosts at the time. If load share is desired for just a few specific services, the configuration on LSNAT could be defined to restrict load share for just the services desired.

In the case where virtual server address is same as the interface address of an LSNAT router, server applications (such as telnet) on LSNAT router must be disabled for external access on that address. This is the limitation to using address owned by LSNAT router as the virtual server address.

Load share NAT operation is also applicable during individual server upgrades as follows. Say, a server, that needs to be upgraded is statically mapped to a backup server on the inbound. Subsequent to this mapping, new session requests to the original server would be redirected by LSNAT to the backup server. As an extension, it is also possible to statically map a specific TU port service on a server to that of backup sever.

We illustrate the operation of LSNAT in the following subsections, where (a) servers are confined to a stub domain, and belong to globally unique address space as shared by clients, (b) servers

are confined to private address space stub domain, and (c) servers are not restrained by any topological limitations.

3.1 Operation of LSNAT in a globally unique address space

In this section, we will illustrate the operation of LSNAT in a globally unique address space. The border router with LSNAT enabled on WAN link would perform load sharing and address translations for inbound sessions. However, sessions outbound from the hosts in server pool will not be subject to any type of translation, as all nodes have globally unique IP addresses.

In the example below, servers S1 (172.85.0.1), S2(172.85.0.2) and S3(172.85.0.3) form a server pool, confined to a stub domain. LSNAT on the border router is enabled on the WAN link, such that the virtual server address S(172.87.0.100) is mapped to the server pool consisting of hosts S1, S2 and S3. When a client 198.76.29.7 initiates a HTTP session to the virtual server S, the LSNAT router examines the load on hosts in server pool and selects a host, say S1 to service the request. The transparent address and TU port translations performed by the LSNAT router become apparent as you follow the down arrow line. IP packets on the return path go through similar address translation. Suppose, we have another client 198.23.47.2 initiating telnet session to the same virtual server S. The LSNAT would determine that host S3 is a better choice to service this session as S1 is busy with a session and redirect the session to S3. The second session redirection path is delineated with colons. The procedure continues for any number of sessions the same way.

Notice that this requires no changes to clients or servers. All the configuration and mapping necessary would be limited just to the LSNAT router.

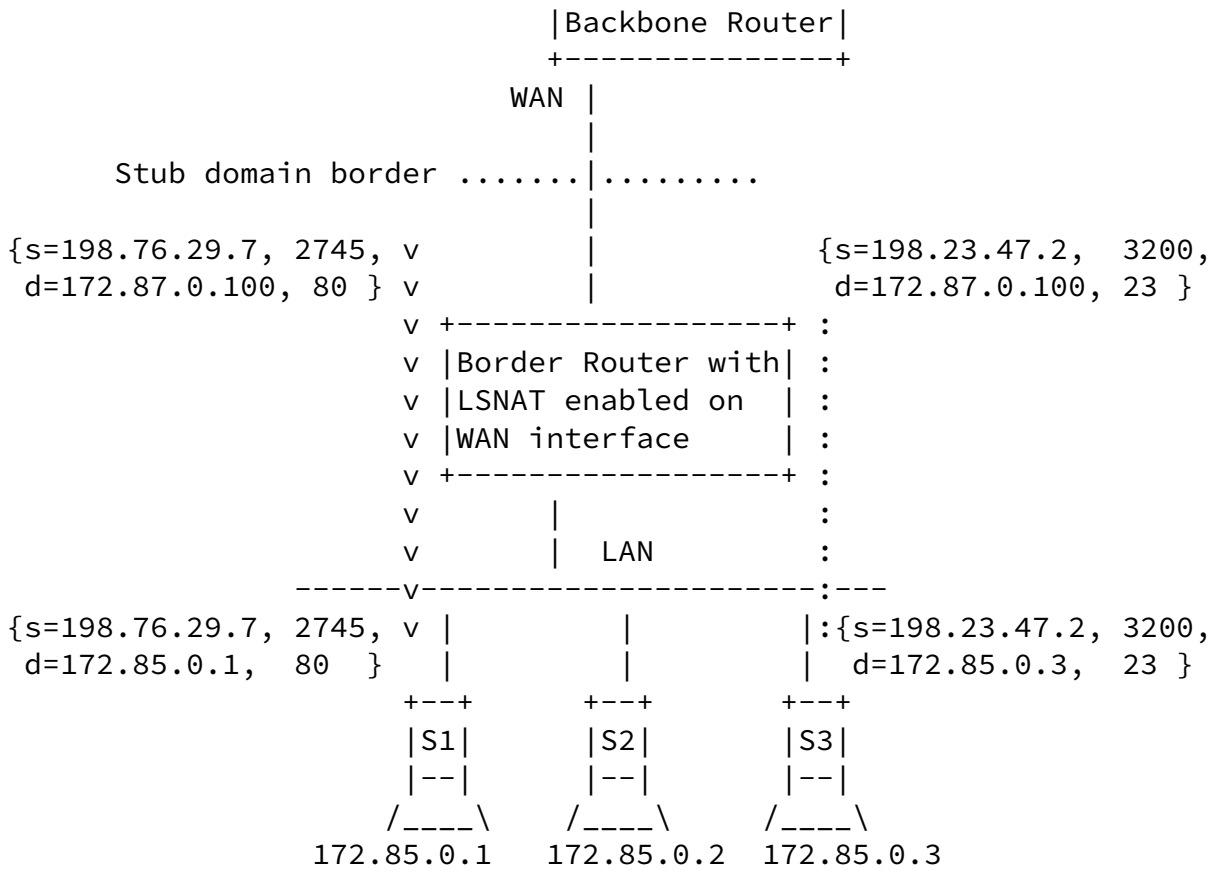


Figure 1: Operation of LSNAT in Globally unique address space

3.2. Operation of LSNAT in conjunction with a private network

In this section, we will illustrate the operation of LSNAT in conjunction with NAT on the same router. The NAT configuration is required for translation of outbound sessions and could be either Basic NAT or NAPT. The illustration below will assume NAPT on the outbound and LSNAT on the inbound on WAN link.

Say, an organization has a private IP network and a WAN link to backbone router. The private network's stub router is assigned a globally valid address on the WAN link and the remaining nodes in the organization have IP addresses that have only local significance. The border router is NAPT configured on the outbound allowing access to external hosts, using the single registered IP address.

In addition, say the organization has servers S1 (10.0.0.1), S2(10.0.0.2) and S3 (10.0.0.3) that form a pool to provide inbound access to external clients. This is made possible by enabling LSNAT on the WAN link of the border router, such that virtual server address S(198.76.28.4) is mapped to the server pool consisting of hosts S1, S2 and S3. When an external client 198.76.29.7 initiates a HTTP session to the virtual server S, the LSNAT router examines load on hosts in server pool and selects a host, say S1 to service the request. The transparent address and TU port translations performed by the LSNAT router are apparent as you follow the down arrow line. IP packets on the return path go through similar address translation. Suppose, we have another client 198.23.47.2 initiating telnet session to the same address. The LSNAT would determine that host S3 is a better choice to service this session as S1 is busy with a session and redirect the session to S3. The second session redirection path is delineated with colons. The procedure continues for any number of sessions the same way.

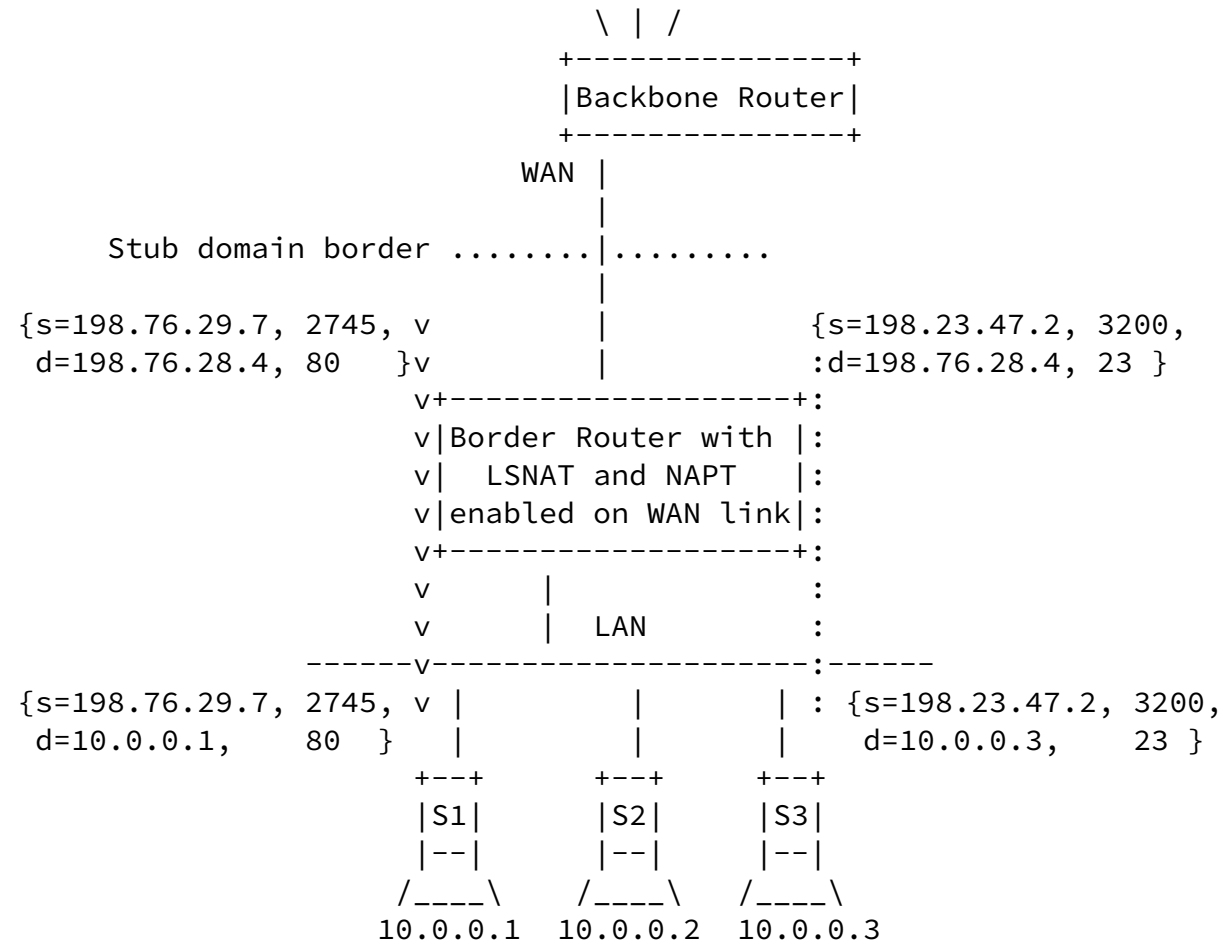


Figure 2: Operation of LSNAT, in coexistence with NAPT

Once again, notice that this requires no changes to clients or servers. The translation is completely transparent to end nodes. Address mapping on the LSNAT performs load sharing and address translations for inbound sessions. Sessions outbound from hosts in server pool are subject to NAPT. Both NAT and LSNAT co-exist with each other in the same router.

3.3. Load Sharing with no topological restraints on servers

In this section, we will illustrate a configuration in which load sharing can be accomplished on a router without enforcing topological limitations on servers. In this configuration, virtual server address will be owned by the router that supports load sharing. I.e., virtual server address will be same as address of one of the interfaces of load share router. We will distinguish this configuration from LSNAT by referring this as "Load Share Network Address Port Translation" (LS-NAPT). Routers that support the LS-NAPT configuration will be termed "LS-NAPT routers", or simply LS-NAPTs.

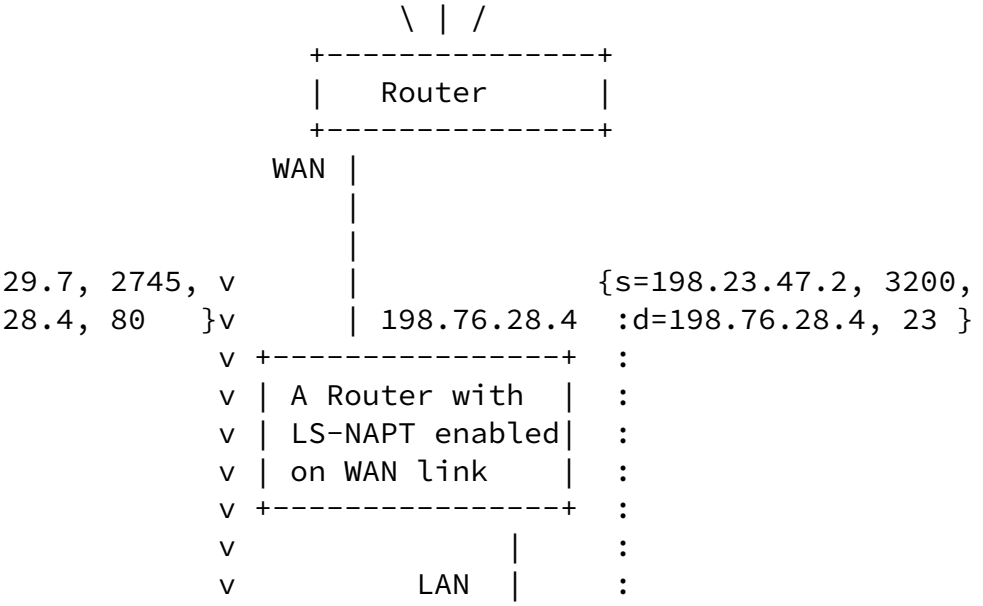
In an LSNAT router, inbound TCP/UDP sessions, represented by the tuple of (client address, client TU port, virtual server address, service port) are translated into a tuple of (client address, client TU port, selected server address, service port). Translation is carried out on all datagrams pertaining to the same session, in either direction. Whereas, LS-NAPT router would translate the same session into a tuple of (virtual server address, virtual server TU port, selected server, service port). Notice that LS-NAPT router translates the client address and TU port with the address and TU port of virtual server, which is same as the address of one of its interfaces. By doing this, datagrams from clients as well as servers are forced to bear the address of LS-NAPT router as the destination address, thereby guaranteeing that the datagrams would necessarily traverse the LS-NAPT router. As a result, there is no need to require servers to be under topological constraints.

Take for example, figure 1 in section 3.1. Let us say the router on which load sharing is enabled is not just a border router, but

can be any kind of router. Let us also say that the virtual server address S (172.87.0.100) is same as the address of WAN link and LS-NAPT is enabled on the WAN interface. Figure 3 summarizes the new router configuration.

When a client 198.76.29.7 initiates a HTTP session to the virtual server address S (i.e., address of the WAN interface), the LS-NAPT router examines load on hosts in server pool and selects a host,

say S1 to service the request. Appropriately, the destination address is translated to be S1 (172.85.0.1). Further, original client address and TU port are replaced with the address and TU port of the WAN link. As a result, destination addresses as well as source address and source TU port are translated when the packet reaches S1, as can be noticed from the down-arrow path. IP packets on the return path go through similar translation. The second client 198.23.47.2 initiating telnet session to the same virtual server address S is load share directed to S3. This packet once again undergoes LS-NAPT translation, just as with the first client. The data path and translations can be noticed following the colon line. The procedure continues for any number of sessions the same way. The translations made to datagrams in either direction are completely transparent to end nodes.



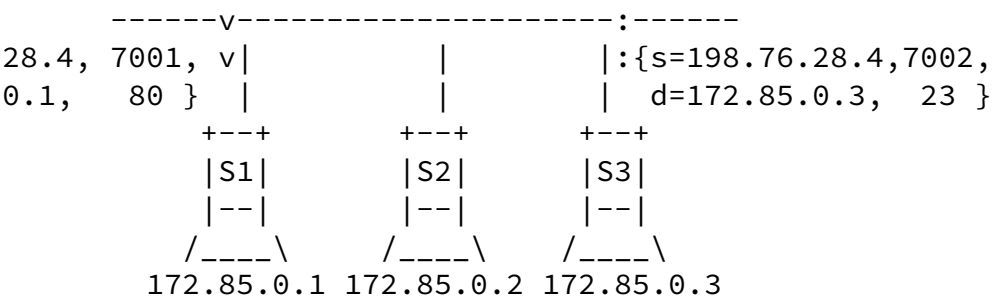


Figure 3: LS-NAPT configuration on a router

As you will notice, datagrams from clients as well as servers are forced to be directed to the router, because they use WAN interface address of router as the destination address in their datagrams.

With the assurance that all packets from clients and servers would traverse the router, there is no longer a requirement for servers to be confined to a stub domain and for LSNAT to be enabled only on border router to the stub domain.

The LS-NAPT configuration described in this section involves more translations and hence is more complex compared to LSNAT configurations described in the previous sections. While the processing is complex, there are benefits to this configuration. Firstly, it breaks down restraints on server topology. Secondly, it scales with bandwidth expansion for client access. Even if Service providers have one link today for client access, the LS-NAPT configuration allows them to expand to more links in the future guaranteeing the same LS-NAPT load share service on newer links.

The configuration is not without its limitations. Server applications (such as telnet) on the router box would have to be disabled for the interface address assigned to be virtual server address. Load sharing would be limited to TCP and UDP applications only. Maximum concurrently allowed sessions would be limited by the maximum allowed TCP/UDP client ports on the same address. Assuming that ports 0-1023 must be set aside as well-known service ports, that would leave a maximum of 63K TCP client ports and 63K of UDP client ports on the LS-NAPT router to communicate with each load-share server. As a result, LS-NAPT routers will not be able

to concurrently support more than a maximum of (63K * count of Load-share servers) TCP sessions and (63K * count of Load-share servers) UDP sessions.

4.0. Translation phases of a session in LSNAT router.

As with NATs, LSNATs must monitor the following three phases in relation to Address translation.

4.1. Session binding:

Session binding is the phase in which an incoming session is associated with the address of a host in server pool. This association essentially sets the translation parameters for all subsequent datagrams pertaining to the session. For addresses that have static mapping, the binding happens at startup time. Otherwise, each incoming session is dynamically bound to a different host based on a load sharing algorithm.

4.2. Address lookup and translation:

Once session binding is established for a connection setup, all subsequent packets belonging to the same connection will be subject to session lookup for translation purposes.

For outbound packets of a session, the source IP address (and source TU port, in case of TCP/UDP sessions) and related fields (such as IP, TCP, UDP and ICMP header checksums) will undergo translation. For inbound packets of a session, the destination IP address (and destination TU port, in case of TCP/UDP sessions) and related fields such as IP, TCP, UDP and ICMP header checksums) will undergo translation.

The header and payload modifications made to IP datagrams subject to LSNAT will be exactly same as those subject to traditional NATs, described in section 5.0 of Ref[1]. Hence, the reader is urged to refer ref[1] document for packet translation process.

4.3. Session unbinding:

Session unbinding is the phase in which a server node is no longer responsible for the session. Usually, session unbinding happens when the end of session is detected. As described in the terminology section, it is not always easy to determine end of session.

5. Load share algorithms

Many algorithms are available to select a host from a pool of servers to service a new session. The load distribution is based primarily on (a) cost of accessing the network on which a server resides and load on the network interface used to access the server, and (b)resource availability and system load on the server. A variety of policies can be adapted to distribute sessions across the servers in a server pool.

For simplicity, we will consider two types algorithms, based on proximity between server nodes and LSNAT router. The higher the cost of access to a sever, the farther the proximity of server is assumed to be. The first kind of algorithms will assume that all server pool members are at equal or nearly equal proximity to LSNAT router and hence the load distribution can be based solely on resource availability or system load on remote servers. Cost of network access will be considered

irrelevant. The second kind would assume that all server pool members have equal resource availability and the criteria for selection would be proximity to servers. In other words, we consider algorithms which take into account the cost of network access.

5.1. Local Load share algorithms

Ideally speaking, the selection process would have precise knowledge of real-time resource availability and system load for each host in server pool, so that the selection of host with maximum unutilized capacity would be the obvious choice.

However, this is not so easy to achieve.

We consider here two kinds of heuristic approaches to monitor session load on server pool members. The first kind is where the load share selector tracks system load on individual servers in non-intrusive way. The second kind is where the individual members actively participate in communicating with the load share selector, notifying the selector of their load capacity.

Listed below are the most common selection algorithms adapted in the non-intrusive category.

1. Round-Robin algorithm
   This is the simplest scheme, where a host is selected simply on a round robin basis, without regard to load on the host.

2. Least Load first algorithm
   This is an improvement over round-robin approach, in that, the host with least number of sessions bound to it is selected to service a new session. This approach is not without its caveats. Each session is assumed to be as resource consuming as any other session, independent of the type of service the session represents and all hosts in server pool are assumed to be equally resourceful.

3. Least traffic first algorithm
   A further improvement over the previous algorithm would be to measure system load by tracking packet count or byte count directed from or to each of the member hosts over a period of time. Although packet count is not the same as system load, it is a reasonable approximation.

4. Least Weighted Load first approach
   This would be an enhancement to the first two. This would

allow administrators to assign (a) weights to sessions, based on likely resource consumption estimates of session types and (b) weights to hosts based on resource availability.

The sum of all session loads by weight assigned to a server,

divided by weight of server would be evaluated to select the server with least weighted load to assign for each new session. Say, FTP sessions are assigned 5 times the weight(5x) as a telnet session(x), and server S3 is assumed to be 3 times as resourceful as server S1. Let us also say that S1 is assigned 1 FTP session and 1 telnet session, whereas S3 is assigned 2 FTP sessions and 5 telnet sessions. When a new telnet session need assignment, the weighted load on S3 is evaluated to be (2*5x+5*x)/3 = 5x, and the load on S1 is evaluated to be (1*5x+1*x) = 6x. Server S3 is selected to bind the new telnet session, as the weighted load on S3 is smaller than that of S1.

5. Ping to find the most responsive host.
   Till now, capacity of a member host is determined exclusively by the LSNAT using heuristic approaches. In reality, it is impossible to predict system capacity from remote, without interaction with member hosts. A prudent approach would be to periodically ping member hosts and measure the response time to determine how busy the hosts really are. Use the response time in conjunction with the heuristics to select the host most appropriate for the new session.

In the active category, we involve individual member hosts in resource utilization monitoring process. An agent software on each node would notify the monitoring agent on resource availability. Clearly, this would imply having an application program (one that does not consume significant resources, by itself) to run on each member node. This strategy of involving member hosts in system load monitoring is likely to yield the most optimal results in the selection process.

5.2. Distributed Load share algorithms

When server nodes are distributed geographically across different areas and cost to access them vary widely, the load share selector could use that information in selecting a server to service a new session. In order to do this, the load share selector would need to consult the routing tables maintained by routing protocols such as RIP and OSPF to find the cost of accessing a server.

All algorithms listed below would be non-intrusive kind where

the server nodes do not actively participate in notifying the load share selector of their load capacity.

1. Weighted Least Load first algorithm
   The selection criteria would be based on (a) cost of access to server, and (b) the number of sessions assigned to server. The product of cost and session load for each server would be evaluated to select the server with least weighted load for each new session. Say, cost of accessing server S1 is twice as much as that of server S2. In that case, S1 will be assigned twice as much load as that of S2 during the distribution process. When a server is not accessible due to network failure, the cost of access is set to infinity and hence no further load can be assigned to that server.

2. Weighted Least traffic first algorithm
   An improvement over the previous algorithm would be to measure network load by tracking packet count or byte count directed from or to each of the member hosts over a period of time. Although packet count is not the same as system load, it is a reasonable approximation. So, the product of cost and traffic load (over a fixed duration) for each server would be evaluated to select the server with least weighted traffic load for each new session.

6. Dead host detection

As sessions are assigned to hosts, it is important to detect the live-ness of the hosts. Otherwise, sessions could simply be black-holed into a dead host. Many heuristic approaches are adopted. Sending pings periodically would be one way to determine the live-ness. Another approach would be to track datagrams originating from a member host in response to new session assignments. If no response is detected in a few seconds, declare the server dead and do not assign new sessions to this host. The server can be monitored later again after a long pause (say, in the order of a few minutes) by periodically reassigning new sessions and monitoring response times and so on.

7. Current Implementations

Many commercial implementations are available in the industry that perform load sharing based on address translation. However, the authors are not aware of any publicly available software.

8. Miscellaneous

The IETF has been notified of potential intellectual Property Rights (IPR) issues with the technology described in this document. Interested people are requested to look in the IETF web page (http://www.ietf.org) under the Intellectual property Rights Notices section for the current information.

9. Security Considerations

All security considerations associated with NAT routers, described in ref[1] are applicable to LSNAT routers as well.

REFERENCES

[1] P. Srisuresh, and K. Egevang "The IP Network Address Translator (NAT)", <draft-rfced-info-srisuresh-03.txt> or its successor.

[2] J. Reynolds and J. Postel, "Assigned Numbers", RFC 1700 or its successor.

[3] R. Braden, "Requirements for Internet Hosts -- Communication Layers", RFC 1122 or its successor.

[4] R. Braden, "Requirements for Internet Hosts -- Application and Support", RFC 1123 or its successor.

[5] F. Baker, "Requirements for IP Version 4 Routers", RFC 1812 or its successor.

[6] J. Postel, J. Reynolds, "FILE TRANSFER PROTOCOL (FTP)", RFC 959 or its successor.

[7] "TRANSMISSION CONTROL PROTOCOL (TCP) SPECIFICATION", RFC 793 or its successor.

[8] J. Postel, "INTERNET CONTROl MESSAGE (ICMP) SPECIFICATION", RFC 793 or its successor.

[9] J. Postel, "User Datagram Protocol (UDP)", RFC 768 or its successor.

[10] J. Mogul, J. Postel, "Internet Standard Subnetting Procedure", RFC 950 or its successor.

[11] T. Brisco, "DNS Support for Load Balancing", RFC 1794 or its successor.

Authors' Addresses

Pyda Srisuresh
Lucent Technologies
Pleasanton, CA 94588-8519
U.S.A.

Voice: (510) 737-2153
Fax: (510) 737-2110
EMail: suresh@livingston.com

Der-hwa Gan
Juniper Networks, Inc.
385 Ravensdale Drive.
Mountain View, CA 94043
U.S.A.

Voice: (650) 526-8074
Fax: (650) 526-8001
EMail: dhg@juniper.net