Network Working Group
Request for Comments: 3235
Category: Informational

D. Senie
Amaranth Networks Inc.

January 2002

Network Address Translator (NAT)-Friendly Application Design Guidelines

Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2002). All Rights Reserved.

Abstract

This document discusses those things that application designers might wish to consider when designing new protocols. While many common Internet applications will operate cleanly in the presence of Network Address Translators, others suffer from a variety of problems when crossing these devices. Guidelines are presented herein to help ensure new protocols and applications will, to the extent possible, be compatible with NAT (Network Address Translation).

1. Introduction

Other documents that describe Network Address Translation (NAT) discuss the Terminology and Considerations [RFC2663] and Protocol Issues [RFC3022], [RFC3027] or discuss the implications of NAT [RFC2993]. All of those relate to various issues with the NAT mechanism, effects on protocols and effects upon general Internet architecture.

It is the focus of this document to provide recommendations to authors of new protocols about the effects to consider when designing new protocols such that special handling is not required at NAT gateway points.

When a protocol is unable to pass cleanly through a NAT, the use of an Application Level Gateway (ALG) may still permit operation of the protocol. Depending on the encoding used in a protocol, an ALG may be difficult or easy to construct, though in some cases it may not be possible at all. While adjunct to NAT, the formulation of protocols that cannot directly operate through NAT should be considered such

that the ALG design may be simple and automated. ALGs typically operate inside small routers along with the NAT component. Ideally, the ALG should be simple and not require excessive computation or state storage.

Many of the same issues in application design that create issues for NAT (and thus can require ALG support) are also issues for firewalls. An application designer would do well to keep this in mind, as any protocol that does require special handling by NAT or firewall products will be more difficult to deploy than those that require no special handling.

2. Discussion

Network Address Translation presents a challenge to some existing applications. In many cases, it should be possible for developers of new applications to avoid problems if they understand the issues. This document aims to provide the application designer with information on what things they can do and what to avoid when trying to build applications that are able to function across NAT.

The proliferation of NAT, especially in homes and small offices cannot be dismissed. The marketing of these technologies to homes and small businesses is often focused on a single-computer environment, and thus providers only give out a single IP address to each user. NAT has become a popular choice for connecting more than a single system per location.

Clearly the most common problem associated with NAT implementations is the passing of addressing data between stations. Where possible, applications should find alternatives to such schemes. Studying a few existing protocols will serve to highlight the different approaches possible.

Two common forms of Traditional NAT exist. With Basic NAT, only the IP addresses of packets are altered by the NAT implementation. Many applications will operate correctly with Basic NAT. The other common form is Network Address Port Translation. With NAPT, both the IP addresses and the source and destination ports (for TCP and UDP) are potentially altered by the gateway. As such, applications passing only port number information will work with Basic NAT, but not with NAPT.

Application designers should strive for compatibility with NAPT, as this form of NAT is the most widely deployed. This is also the form of NAT that will likely see the greatest penetration in homes and small offices. Not all applications lend themselves to the architectural model imposed by NAPT.

3. Recommendations and Examples

Application designers who work within the constraints of NAT, and who do not rely on the presence of ALGs will generally find the easier acceptance in user communities where NAT is common. When designing a new application or service, the requirement for an ALG will limit deployment until the required additional code is incorporated into the many devices which implement NAT.

Each of the areas called out below are examples of issues to consider when building an application. This list is likely not comprehensive, but does cover a number of important issues and considerations.

3.1 Issues and Recommendations affecting all types of Network Address Translators

3.1.1. Peer-to-Peer Applications Limitations

Peer to peer applications are problematic in a NAT world. Client-server applications are more generally workable. Peer-to-peer applications rely on each peer being reachable as a server (i.e., bound to a listening port, and able to accept connections) for the other to connect to. With NAPT, there are likely many machines behind one address. With other types of NAT such as Basic NAT with Static Address Assignment (providing one-to-one mappings), there is a greater chance of making such applications work.

Some implementations of NAT can be made to function for UDP-based peer-to-peer applications. This capability is dependent on the methodology used to implement the UDP sessions in the NAT device. If the NAT device tracks the tuple (private address, private port, public port) then it is possible for an outbound UDP packet to establish a channel by which incoming traffic can flow from a source other than that originally contacted by the system. The source IP address is NOT used in this case to match incoming packets to UDP sessions, allowing any source address using the UDP port number to be translated.

NAT devices which track source and destination IP addresses, in addition to port numbers, will not permit third-party packets. NAT is often implemented in conjunction along with stateful-inspection firewall functionality. As such the latter implementation of UDP association tracking would be considered more secure.

NAT/Firewall device implementations could be constructed to have a software switch within them, permitting the consumer the ability to select whether they want the greater security, or greater ability to run peer-to-peer applications.

3.1.2. Applications Requiring End-to-End IPSec Will Fail

Use of IPSec for end-to-end security will not function in the presence of a NAT implementation. Application designers may want to explore the use of Transport Layer Security (TLS) [RFC2246] as a transport mode that will traverse NAT cleanly. See [RFC2709] for additional discussion on combining NAT with Tunnel-mode IPSec security on the same device.

3.1.3. Use DNS Names, Not IP Addresses In Payload

Applications should, where possible, use fully qualified domain names rather than IP addresses when referring to IP endpoints. When endpoints are across a NAT gateway, private addresses must not be allowed to leak to the other endpoint. An example of where this can happen today is with the HTTP and HTML protocols. It is possible for web pages to be specified with numeric IP addresses, rather than with names, for example http://192.168.1.10/index.html could be used as a URL, but would likely create a problem if this address is on a server located behind a NAT gateway. Users outside the gateway would not be able to reach the address 192.168.1.10, and so would not see the page.

Further exacerbating the problem is the possibility of duplicate addresses between realms. If a server offers a link with a private address space IP address embedded within it, such as 192.168.1.10, the page referenced may resolve to a system on the local network the browser is on, but would be a completely different server. The resulting confusion to end-users would be significant. Sessions involving multiple NAT implementations would be exceptionally vulnerable to address reuse issues of this sort.

3.1.4. Multicast Considerations

Not all NAT devices implement multicast routing protocols. Application designers should verify whether the devices in the networks where their applications will be deployed are able to process multicast traffic if their applications rely on that capability.

3.1.5. Retention Of Address Mapping

With the exception of statically configured NAT bindings, applications should not assume address mapping will be maintained from one session (association between machines, for whatever protocol for a period of time) to another. An example of this is RSVP, which forms one connection to reserve the resources, then the actual session for which resources were reserved is started. The sessions

do not necessarily overlap. There is no guarantee that the NAT implementation will keep the binding association. As such, applications that rely on subsequent sessions being mapped to the same host IP address may not function without an ALG.

Another consideration is the number of addressing realms. It is entirely possible to have multiple levels of NAT implementations between the two end points involved. As such, one must think about the lifetime of such mappings at all such levels.

Load balancers and other devices may use a single IP address and port to map to multiple actual end points. Many products implement variations on this theme, sometimes using NAT, sometimes using other technologies. The lack of guarantee of mapping is important to understand, since the mapping to one actual system to another may not survive across such intermediate boxes.

Don’t assume systems know their own IP addresses. A system behind a NAT may be reachable via a particular IP address, but that address may not be recognized by the system itself. Consider the case of Static, one-to-one mapping using Basic NAT. A server in this context will have an IP address from the private realm, and may not know the public address which maps to it. Similarly, some such systems may not know their own DNS names, while others may. This is largely dependent on the configuration of the servers and the network within the private realm.

3.2 Recommendations for NAPT

As many of the issues specifically address NAPT issues, this section will group these issues. NAPT is the most common form of NAT in actual deployment in routers, especially in smaller offices and home offices.

3.2.1 IP Addresses Specific To A Realm

Avoid the use of IP address and port number information within the payload of packets. While in some cases ALGs will permit such protocols to function, this presupposes every NAT device can be updated in a timely fashion to support a new protocol. Since this is unlikely, application writers are urged to avoid placing addressing information in payloads all together.

In addition to avoiding addresses and port numbers within packet payloads, it is important to avoid assumptions of (address, port) tuples are unique beyond the scope of the present session. Load balancing devices implementing NAT may, for example, map subsequent sessions to other systems in the private realm.

3.2.2 Avoid Session Bundles

Independent sessions, such as used by POP or SMTP, are preferred to protocols that attempt to manage a bundle of related sessions, such as FTP. The term "session" here is used to refer to any association between end systems, and may be using any transport protocol or combination of protocols (UDP, TCP, SCTP).

In the FTP protocol, port information is passed over one TCP connection and is used to construct a second TCP connection for passing the actual data. Use of a separate connection to transfer the file data makes determination of file end quite simple, however other schemes could be envisioned which could use a single connection.

The HTTP protocol, for example, uses a header and content length approach to passing data. In this model, all data is transferred over the single TCP connection, with the header portion indicating the length of the data to follow. HTTP has evolved to allow multiple objects to be passed on a single connection (thereby cutting the connection establishment overhead). Clearly a new file transfer function could be built that would perform most of the functions of FTP without the need for additional TCP connections.

The goal is to keep to single connections where possible. This keeps us from needing to pass addressing information of any sort across the network. However, multiplexing traffic over a single connection can create problems as well.

3.2.3. Session Bundles Originate From Same End

Origination of connections is an important consideration. Where possible, the client should originate all connections. The FTP protocol is the most obvious example, where by default the server opens the data connection to a port on the client (the client having specified the port number via a PORT command over the control TCP session).

As pointed out in [RFC1579], the use of the passive open option in FTP (PASV) remedies this situation as the client is responsible for opening the connection in this case. With client-opened connections, the standard functions of NAPT will process the request as it would any other simple TCP connection, and so an ALG is not required.

In cases where session bundles are unavoidable, each session in the bundle should originate from the same end station.

3.2.4. Choice of Transport Protocol

NAPT gateways must track which sessions are alive, and flush old sessions. TCP has clear advantages in this area, since there are specific beginning and end of session indicators in the packets (SYN and FIN packets). While UDP works for some types of applications with NAT, there can be issues when that data is infrequent. Since there is no clean way to know when an end station has finished using a UDP session, NAT implementations use timeouts to guess when a UDP session completes. If an application doesn’t send data for a long period of time, the NAT translation may time out.

NAT implementations also use timers to guess when TCP sessions have disappeared. While TCP sessions should disappear only after FIN packets are exchanged, it is possible that such packets may never come, for example if both end stations die. As such, the NAT implementation must use a timer for cleaning up its resources.

NAT implementers in many cases provide several timeouts, one for live TCP sessions, one for TCP sessions on which a FIN has been seen, and one for UDP sessions. It is best when such flexibility is provided, but some implementations appear to apply a single timer to all traffic.

Protocols other than TCP and UDP can work with Traditional NAT in many cases, provided they are not carrying addressing information. For NAPT implementations use of any protocols other than TCP and UDP will be problematic unless or until such protocols are programmed into the implementations.

It’s important to note that NAPT deployments are based on the assumption of a client-server application model, with the clients in the private realm.

3.2.5. IP Fragmentation

Applications should attempt to avoid fragmentation when packets pass over NAPT devices. While not always practical or possible, there are failures that can occur with NAPT. Specifically, if two stations in the private realm pick matching fragmentation identifiers, and talk to the same remote host, it may be impossible to determine which fragments belong to which session. A clever NAPT implementation could track fragmentation identifiers and map those into a unique space, though it is not clear how many do so.

Ideally, applications should limit packet size, use Path MTU Discovery or both. Unfortunately, at least some firewall/NAT devices block Path MTU Discovery, apparently believing all ICMP packets are evil.

Some implementations of NAT may implement fragment reassembly prior to Forwarding, however many do not. Application designers are advised to design assuming the devices do not reassemble fragments.

3.3 Issues and recommendations for Basic NAT

If only Basic NAT implementations are involved, not NAPT, then many of the issues above do not apply. This is not to say that this form of NAT is better or worse than NAPT. Application designers may think they could just specify users must use Basic NAT, and many application issues would go away. This is unrealistic, however, as many users have no real alternative to NAPT due to the way their providers sell service.

Many of the issues raised earlier still apply to Basic NAT, and many protocols will not function correctly without assistance.

3.3.1. Use IP and TCP/UDP Headers Alone

Applications that use only the information in the IP and TCP or UDP headers for communication (in other words, do not pass any additional addressing information in the payload of the packets), are clearly easier to support in a NAT environment. Where possible, applications designers should try to limit themselves in this area.

This comes back to the same recommendation made for NAPT, that being to use a single connection whenever possible.

The X windowing system, for example, uses fixed port numbers to address X servers. With X, the server (display) is addressed via ports 6000 through 6000 + n. These map to hostname:0 through hostname:n server displays. Since only the address and port are used, the NAT administrator could map these ports to one or more private addresses, yielding a functioning solution.

The X example, in the case of NAPT, requires configuration of the NAT implementation. This results in the ability for no more than one station inside the NAT gateway to use such a protocol. This approach to the problem is thus OK for NAT but not recommended for NAPT environments.

3.3.2. Avoid Addressing In Payload

As with NAPT, transporting IP address and/or port number information in the payload is likely to cause trouble. As stated earlier, load balancers and similar platforms may well map the same IP address and port number to a completely different system. Thus it is problematic to assume an address or port number which is valid in the realm on one side of a NAT is valid on the other side.

3.4 Bi-directional NAT

Bi-directional NAT makes use of DNS mapping of names to point sessions originating outside the private realm to servers in the private realm. Through use of a DNS-ALG [RFC2694], lookups are performed to find the proper host and packets are sent to that host.

Requirements for applications are the same as for Basic NAT. Addresses are mapped one-to-one to servers. Unlike Traditional NAT devices, Bi-directional NAT devices (in conjunction with DNS-ALG) are amenable to peer-to-peer applications.

3.5 Twice NAT

Twice NAT is address translation where both source and destination IP addresses are modified due to addressing conflicts between two private realms. Two bi-directional NAT boxes connected together would essentially perform the same task, though a common address space that is not otherwise used by either private realm would be required.

Requirements for applications to work in the Twice NAT environment are the same as for Basic NAT. Addresses are mapped one to one.

3.6 Multi-homed NAT

Multi-homed NAT is the use of multiple NAT implementations to provide redundancy. The multiple implementations share configuration information so that sessions might continue in the event of a fail-over. Unless the multiple implementations share the same external addresses, sessions will have to restart regardless.

Requirements for multi-homed NAT are the same as for Basic NAT or NAPT, depending on how the multi-homed NAT is implemented and configured.

3.7 Realm Specific IP (RSIP)

Realm Specific IP is described in [RFC2663] and defined in [RSIP] and related documents. Clients within a private realm using RSIP are aware of the delineation between private and public, and access a server to allocate address (and optionally port) information for use in conversing with hosts in the public realm. By doing this, clients create packets that need not be altered by the RSIP server on their way to the remote host. This technique can permit IPSec to function, and potentially makes any application function as if there were no special processing involved at all.

RSIP uses a view of the world in which there are only two realms, the private and public. This isn’t always the case. Situations with multiple levels of NAT implementations are growing. For example, some ISPs are handing out [RFC1918] addresses to their dialup users, rather than obtaining real addresses. Any user of such an ISP who also uses a NAT implementation will see two levels of NAT, and the advantages of RSIP will have been wasted.

3.8 Performance Implications of Address Translation Implementations

Resource utilization on the NAT gateway should be considered. An application that opens and closes many TCP connections, for example, will use up more resources on the NAT router than an application performing all transfers over a single TCP connection. HTTP 1.0 opened a connection for each object on a web page, whereas HTTP 1.1 permits the TCP session to be held open for additional objects that may need to be transferred. Clearly the latter imposes a lower overhead on the NAT gateway, as it is only maintaining state on a single connection instead of multiple connections.

New session establishment will typically remain a software function even in implementations where the packet-by-packet translation work is handled by hardware forwarding engines. While high-performance NAT boxes may be built, protocols that open many sessions instead of multiplexing will be slower than those that do not.

Applications with different types of data, such as interactive conferencing, require separate streams for the different types of data. In such cases the protocol needs of each stream must be optimized. While the goal of multiplexing over a single session is preferred, clearly there are cases where this is impractical.

The latency of NAT translation overhead is implementation dependent. On a per-packet basis, for established sessions only the source or destination IP address is replaced, the source or destination port (for NAPT) and the checksums for IP, and TCP or UDP are recalculated.

The functionality can be efficiently implemented in hardware or software.

4. Security Considerations

Network Address Translators have implications for IPSec, as noted above. When application developers are considering whether their applications function with NAT implementations, care should be given to selection of security methodology. Transport Layer Security (TLS) [RFC2246] operates across translation boundaries. End-to-end IPSec will prove problematic in many cases.

5. References

[RFC1579] Bellovin, S., "Firewall Friendly FTP", RFC 1579, February 1994.

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[RFC2993] Hain, T., "Architectural Implications of NAT", RFC 2993, November 2000.

[RFC3027] Holdrege, M. and P. Srisuresh, "Protocol Complications with the IP Network Address Translator (NAT)", RFC 3027, January 2001.

[RFC2663] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.

[RFC2709] Srisuresh, P., "Security Model with Tunnel-mode IPsec for NAT Domains", RFC 2709, October 1999.

[RFC3102] Borella, M., Lo, J., Grabelsky, D. and G. Montenegro, "Realm Specific IP: Framework", RFC 3102, October 2001.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.

[RFC2694] Srisuresh, P., Tsirtsis, G., Akkiraju, P. and A. Heffernan, "DNS extensions to Network Address Translators (DNS_ALG)", RFC 2694, September 1999.

6. Acknowledgements

I'd like to thank Pyda Srisuresh for his invaluable input and feedback, and Keith Moore for his extensive comments.

7. Author's Address

Daniel Senie
Amaranth Networks Inc.
324 Still River Road
Bolton, MA 01740

Phone: (978) 779-6813
EMail: dts@senie.com

8. Full Copyright Statement

Copyright (C) The Internet Society (2002). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.