                   NAT Port Mapping Protocol (NAT-PMP)

Abstract

   This document describes a protocol for automating the process of
   creating Network Address Translation (NAT) port mappings. Included
   in the protocol is a method for retrieving the external IPv4 address
   of a NAT gateway, thus allowing a client to make its external IPv4
   address and port known to peers that may wish to communicate with it.
   From 2005 onwards, this protocol was implemented in Apple products
   including Mac OS X, Bonjour for Windows, and AirPort wireless base
   stations. In 2013, NAT Port Mapping Protocol (NAT-PMP) was
   superseded by the IETF Standards Track RFC "Port Control Protocol
   (PCP)", which builds on NAT-PMP and uses a compatible packet format,
   but adds a number of significant enhancements.

Status of This Memo

   This document is not an Internet Standards Track specification; it is
   published for informational purposes.

   This is a contribution to the RFC Series, independently of any other
   RFC stream. The RFC Editor has chosen to publish this document at
   its discretion and makes no statement about its value for
   implementation or deployment. Documents approved for publication by
   the RFC Editor are not a candidate for any level of Internet
   Standard; see Section 2 of RFC 5741.

   Information about the current status of this document, any errata,
   and how to provide feedback on it may be obtained at
   http://www.rfc-editor.org/info/rfc6886.

Table of Contents

1. Introduction

   Network Address Translation (NAT) is a method of sharing one public
   Internet address with a number of devices. This document is focused
   on devices that are formally classified as "NAPTs" (Network
   Address/Port Translators) [RFC2663]. A full description of NAT is
   beyond the scope of this document. The following brief overview will
   cover the aspects relevant to this port mapping protocol. For more
   information on NAT, see "Traditional IP Network Address Translator
   (Traditional NAT)" [RFC3022].

   NATs have one or more external IP addresses. A private network is
   set up behind the NAT. Client devices on that private network behind
   the NAT are assigned private addresses, and those client devices use
   the private address of the NAT device as their default gateway.

   When a packet from any device behind the NAT is sent to an address on
   the public Internet, the packet first passes through the NAT box.
   The NAT box looks at the source port and address. In some cases, a
   NAT will also keep track of the destination port and address. The
   NAT then creates a mapping from the internal address and internal
   port to an external address and external port if a mapping does not
   already exist.

   The NAT box replaces the internal address and port in the packet with
   the external entries from the mapping and sends the packet on to the
   next gateway.

   When a packet from any address on the Internet is received on the
   NAT's external side, the NAT will look up the destination address and
   port (external address and port) in the list of mappings. If an
   entry is found, it will contain the internal address and port to
   which the packet should be sent. The NAT gateway will then rewrite
   the destination address and port with those from the mapping. The
   packet will then be forwarded to the new destination addresses. If
   the packet did not match any mapping, the packet will most likely be
   dropped. Various NATs implement different strategies to handle this.
   The important thing to note is that if there is no mapping, the NAT
   does not know to which internal address the packet should be sent.

   Mappings are usually created automatically as a result of observing
   outbound packets. There are a few exceptions. Some NATs may allow
   manually created permanent mappings that map an external port to a
   specific internal IP address and port. Such a mapping allows
   incoming connections to the device with that internal address. Some
   NATs also implement a default mapping where any inbound packet that

does not match any other more specific mapping will be forwarded to a
specified internal address. Both types of mappings are usually set
up manually through some configuration tool. Such manual
configuration of port mappings is unreasonably onerous for most
residential NAT users.

Without these manually created inbound port mappings, clients behind
the NAT would be unable to receive inbound connections, which
represents a loss of connectivity when compared to the original
Internet architecture [ETEAISD]. For those who view this loss of
connectivity as a bad thing, NAT-PMP allows clients to operate more
like a host directly connected to the unrestricted public Internet,
with an unrestricted public IPv4 address. NAT-PMP allows client
hosts to communicate with the NAT gateway to request the creation of
inbound mappings on demand. Having created a NAT mapping to allow
inbound connections, the client can then record its external IPv4
address and external port in a public registry (e.g., the worldwide
Domain Name System) or otherwise make it accessible to peers that
wish to communicate with it.

1.1. Transition to Port Control Protocol

NAT-PMP enjoyed almost a decade of useful service, and operational
experience with NAT-PMP informed the design of its IETF Standards
Track successor, Port Control Protocol (PCP) [RFC6887]. PCP builds
on NAT-PMP, using the same UDP ports 5350 and 5351, and a compatible
packet format. PCP also adds significant enhancements, including
IPv6 support, management of outbound mappings, management of firewall
rules, full compatibility with large-scale NATs with a pool of
external addresses, error lifetimes, and an extension mechanism to
enable future enhancements.

Because of the significant enhancements in PCP, all existing NAT-PMP
implementations are encouraged to migrate to PCP. The version number
in the packet header is 0 for NAT-PMP and 2 for PCP, so the packets
are easily distinguished. (Version number 1 was used by a vendor
that shipped products that use a protocol that is incompatible with
the IETF Standard. PCP implementations MUST NOT use version
number 1.)

For NAT-PMP servers, adding PCP support is simple. When packets are
received, if the version number is 2, then the packet is interpreted
as a PCP request, the request is handled, and replies and updates
pertaining to that mapping are sent using PCP format. If the version
number is 0, then the existing code handles the request exactly as it
already does, and replies and updates pertaining to that mapping are

sent using NAT-PMP format. If the version number is 1 or any other
unsupported version, then result code 1 (Unsupported Version) is
returned.

NAT-PMP clients should add PCP support, and should default to sending
requests using PCP format, which will cause clients to prefer the
newer format when possible. If a PCP request is sent to an old
NAT-PMP server that doesn't understand the new PCP format, then it
will return result code 1 (Unsupported Version), and the client
should then immediately retry the same request using NAT-PMP format.
The presence of the Unsupported Version reply allows fast fail-over
to NAT-PMP format, without waiting for timeouts, retransmissions, or
other arbitrary delays. It is recommended that clients always try
PCP first for every new request, retransmission, and renewal, and
only try NAT-PMP in response to an "Unsupported Version" error.
Clients SHOULD NOT record that a given server only speaks NAT-PMP and
subsequently default to NAT-PMP for that server, since NAT firmware
gets updated, and even a NAT gateway that speaks only NAT-PMP today,
may be updated to speak PCP tomorrow.

2. Conventions and Terminology Used in This Document

   The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
   "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this
   document are to be interpreted as described in "Key words for use in
   RFCs to Indicate Requirement Levels" [RFC2119].

3. Protocol and Packet Format

   The NAT Port Mapping Protocol runs over UDP. Every packet starts
   with an 8-bit version followed by an 8-bit operation code.

   All numeric quantities in NAT-PMP larger than a single byte (e.g.,
   error values, Seconds Since Start of Epoch, and mapping lifetime) are
   transmitted in the traditional IETF network byte order (i.e., most
   significant byte first).

   Non-numeric quantities in NAT-PMP larger than a single byte (e.g.,
   the NAT gateway's external IP address) are transmitted in the natural
   byte order, with no byte swapping.

   This document specifies version 0 of the protocol. Any NAT-PMP
   gateway implementing this version of the protocol, receiving a
   request with a version number other than 0, MUST return result code 1
   (Unsupported Version), indicating the highest version number it does
   support (i.e., 0) in the version field of the response.

Opcodes between 0 and 127 are client requests. Opcodes from 128 to
255 are corresponding server responses. Responses always contain a
16-bit result code in network byte order. A result code of zero
indicates success. Responses also contain a 32-bit unsigned integer
corresponding to the number of seconds since the NAT gateway was
rebooted or since its port mapping state was otherwise reset.

This protocol SHOULD only be used when the client determines that its
primary IPv4 address is in one of the private IPv4 address ranges
defined in "Address Allocation for Private Internets" [RFC1918].
This includes the address ranges 10/8, 172.16/12, and 192.168/16.

Clients always send their NAT-PMP requests to their default gateway,
as learned via DHCP [RFC2131], or similar means. This protocol is
designed for small home networks, with a single logical link (subnet)
where the client's default gateway is also the NAT for that network.
For more complicated networks where the NAT is some device other than
the client's default gateway, this protocol is not appropriate.

3.1. Requests and Responses

NAT gateways are often low-cost devices, with limited memory and CPU
speed. For this reason, to avoid making excessive demands on the NAT
gateway, clients SHOULD NOT issue multiple concurrent requests. If a
client needs to perform multiple requests (e.g., on boot, wake from
sleep, network connection, etc.), it SHOULD queue them and issue them
serially, one at a time. Once the NAT gateway responds to one
request the client machine may issue the next. In the case of a fast
NAT gateway, the client may be able to complete requests at a rate of
hundreds per second. In the case of a slow NAT gateway that takes
perhaps half a second to respond to a NAT-PMP request, the client
SHOULD respect this and allow the NAT gateway to operate at the pace
it can manage, and not overload it by issuing requests faster than
the rate it's answering them.

To determine the external IPv4 address, or to request a port mapping,
a NAT-PMP client sends its request packet to port 5351 of its
configured gateway address, and waits 250 ms for a response. If no
NAT-PMP response is received from the gateway after 250 ms, the
client retransmits its request and waits 500 ms. The client SHOULD
repeat this process with the interval between attempts doubling each
time. If, after sending its ninth attempt (and then waiting for 64
seconds), the client has still received no response, then it SHOULD
conclude that this gateway does not support NAT Port Mapping Protocol
and MAY log an error message indicating this fact. In addition, if
the NAT-PMP client receives an "ICMP Port Unreachable" message from

the gateway for port 5351, then it can skip any remaining
retransmissions and conclude immediately that the gateway does not
support NAT-PMP.

As a performance optimization the client MAY record this information
and use it to suppress further attempts to use NAT-PMP, but the
client should not retain this information for too long. In
particular, any event that may indicate a potential change of gateway
or a change in gateway configuration (hardware link change
indication, change of gateway MAC address, acquisition of new DHCP
lease, receipt of NAT-PMP announcement packet from gateway, etc.)
should cause the client to discard its previous information regarding
the gateway's lack of NAT-PMP support, and send its next NAT-PMP
request packet normally.

When deleting a port mapping, the client uses the same initial 250 ms
timeout, doubling on each successive interval, except that clients
may choose not to try the full nine times before giving up. This is
because mapping deletion requests are in some sense advisory. They
are useful for efficiency, but not required for correctness; it is
always possible for client software to crash, or for power to fail,
or for a client device to be physically unplugged from the network
before it gets a chance to send its mapping deletion request(s), so
NAT gateways already need to cope with this case. Because of this,
it may be acceptable for a client to retry only once or twice before
giving up on deleting its port mapping(s), but a client SHOULD always
send at least one deletion request whenever possible, to reduce the
amount of stale state that accumulates on NAT gateways.

A client need not continue trying to delete a port mapping after the
time when that mapping would naturally have expired anyway.

3.2. Determining the External Address

   To determine the external address, the client behind the NAT sends
   the following UDP payload to port 5351 of the configured gateway
   address:

```
   0                   1
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Vers = 0      | OP = 0        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

A compatible NAT gateway MUST generate a response with the following
format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Vers = 0      | OP = 128 + 0  | Result Code (net byte order)  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Seconds Since Start of Epoch (in network byte order)          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| External IPv4 Address (a.b.c.d)                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

This response indicates that the NAT gateway implements this version
of the protocol, and returns the external IPv4 address of the NAT
gateway. If the result code is non-zero, the value of the External
IPv4 Address field is undefined (MUST be set to zero on transmission,
and MUST be ignored on reception).

The NAT gateway MUST fill in the Seconds Since Start of Epoch field
with the time elapsed since its port mapping table was initialized on
startup, or reset for any other reason (see Section 3.6, "Seconds
Since Start of Epoch").

Upon receiving a response packet, the client MUST check the source IP
address, and silently discard the packet if the address is not the
address of the gateway to which the request was sent.

3.2.1. Announcing Address Changes

Upon boot, acquisition of an external IPv4 address, subsequent change
of the external IPv4 address, reboot, or any other event that may
indicate possible loss or change of NAT mapping state, the NAT
gateway MUST send a gratuitous response to the link-local multicast
address 224.0.0.1, port 5350, with the packet format above, to notify
clients of the external IPv4 address and Seconds Since Start of
Epoch.

To accommodate packet loss, the NAT gateway SHOULD multicast 10
address notifications. The interval between the first two
notifications SHOULD be 250 ms, and the interval between each
subsequent notification SHOULD double. The Seconds Since Start of
Epoch field in each transmission MUST be updated appropriately to
reflect the passage of time, so as not to trigger unnecessary
additional mapping renewals (see Section 3.7, "Recreating Mappings on
NAT Gateway Reboot").

Upon receiving a gratuitous address announcement packet, the client
MUST check the source IP address, and silently discard the packet if
the address is not the address of the client's current configured
gateway. This is to guard against inadvertent misconfigurations
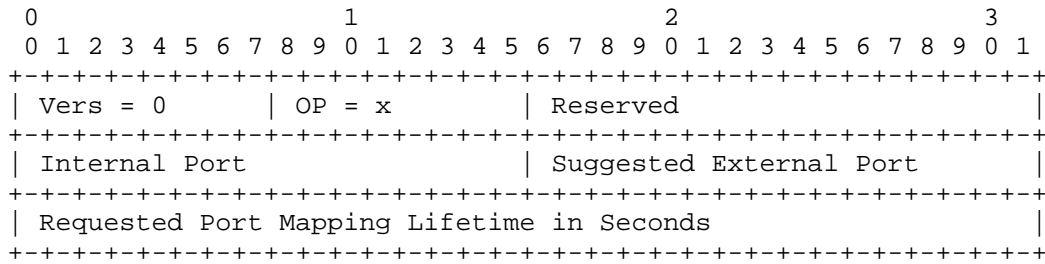where there may be more than one NAT gateway active on the network.

If the source IP address is correct, then the Seconds Since Start of
Epoch field is checked as described in Section 3.6, and if the value
is outside the expected plausible range, indicating that a NAT
gateway state loss has occurred, then the NAT-PMP client promptly
recreates all its active port mapping leases, as described in Section
3.7, "Recreating Mappings on NAT Gateway Reboot".

IMPLEMENTATION NOTE: Earlier implementations of NAT-PMP used UDP port
5351 as the destination both for client requests (address and port
mapping) and for address announcements. NAT-PMP servers would listen
on UDP 5351 for client requests, and NAT-PMP clients would listen on
UDP 5351 for server announcements. However, implementers encountered
difficulties when a single device is acting in both roles, for
example, a home computer with Internet Sharing enabled. This
computer is acting in the role of NAT-PMP server to its DHCP clients,
yet, at the same time, it has to act in the role of NAT-PMP client in
order to determine whether it is, itself, behind another NAT gateway.
While in principle it might be possible on some operating systems for
two processes to coordinate sharing of a single UDP port, on many
platforms this is difficult or even impossible, so, for pragmatic
engineering reasons, it is convenient to have clients listen on UDP
5350 and servers listen on UDP 5351.

IMPLEMENTATION NOTE: A given host may have more than one independent
NAT-PMP client running at the same time, and address announcements
need to be available to all of them. Clients should therefore set
the SO_REUSEPORT option or equivalent in order to allow other
processes to also listen on port 5350. Additionally, implementers
have encountered issues when one or more processes on the same device
listen to port 5350 on *all* addresses. Clients should therefore
bind specifically to 224.0.0.1:5350, not to 0.0.0.0:5350.

3.3. Requesting a Mapping

   To create a mapping, the client sends a UDP packet to port 5351 of
   the gateway's internal IP address with the following format:

    0   1   2   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+
   | Vers = 0 | OP = x | Reserved |
   +-+
   | Internal Port | Suggested External Port |
   +-+
   | Requested Port Mapping Lifetime in Seconds |
   +-+

   Opcodes supported:
   1 - Map UDP
   2 - Map TCP

   The Reserved field MUST be set to zero on transmission and MUST be
   ignored on reception.

   The Ports and Lifetime are transmitted in the traditional network
   byte order (i.e., most significant byte first).

   The Internal Port is set to the local port on which the client is
   listening.

   If the client would prefer to have a high-numbered "anonymous"
   external port assigned, then it should set the Suggested External
   Port to zero, which indicates to the gateway that it should allocate
   a high-numbered port of its choosing. If the client would prefer
   instead to have the mapped external port be the same as its local
   internal port if possible (e.g., a web server listening on port 80
   that would ideally like to have external port 80), then it should set
   the Suggested External Port to the desired value. However, the
   gateway is not obliged to assign the port suggested, and may choose
   not to, either for policy reasons (e.g., port 80 is reserved and
   clients may not request it) or because that port has already been
   assigned to some other client. Because of this, some product
   developers have questioned the value of having the Suggested External
   Port field at all. The reason is for failure recovery. Most low-
   cost home NAT gateways do not record temporary port mappings in
   persistent storage, so if the gateway crashes or is rebooted, all the
   mappings are lost. A renewal packet is formatted identically to an
   initial mapping request packet, except that for renewals the client
   sets the Suggested External Port field to the port the gateway
   actually assigned, rather than the port the client originally wanted.

When a freshly rebooted NAT gateway receives a renewal packet from a
client, it appears to the gateway just like an ordinary initial
request for a port mapping, except that in this case the Suggested
External Port is likely to be one that the NAT gateway *is* willing
to allocate (it allocated it to this client right before the reboot,
so it should presumably be willing to allocate it again). This
improves the stability of external ports across NAT gateway restarts.

The RECOMMENDED Port Mapping Lifetime is 7200 seconds (two hours).

After sending the port mapping request, the client then waits for the
NAT gateway to respond. If after 250 ms the client hasn't received a
response from the gateway, the client SHOULD reissue its request as
described above in Section 3.1, "Requests and Responses".

The NAT gateway responds with the following packet format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Vers = 0      | OP = 128 + x  | Result Code                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Seconds Since Start of Epoch                                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Internal Port                 | Mapped External Port          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Port Mapping Lifetime in Seconds                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
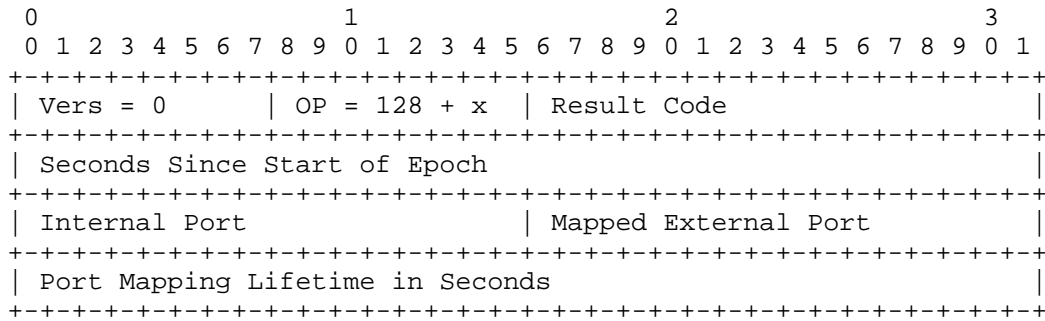
The epoch time, ports, and lifetime are transmitted in the
traditional network byte order (i.e., most significant byte first).

The 'x' in the OP field MUST match what the client requested. Some
NAT gateways are incapable of creating a UDP port mapping without
also creating a corresponding TCP port mapping, and vice versa, and
these gateways MUST NOT implement NAT Port Mapping Protocol until
this deficiency is fixed. A NAT gateway that implements this
protocol MUST be able to create TCP-only and UDP-only port mappings.
If a NAT gateway silently creates a pair of mappings for a client
that only requested one mapping, then it may expose that client to
receiving inbound UDP packets or inbound TCP connection requests that
it did not ask for and does not want.

While a NAT gateway MUST NOT automatically create mappings for TCP
when the client requests UDP, and vice versa, the NAT gateway MUST
reserve the companion port so the same client can choose to map it in
the future. For example, if a client requests to map TCP port 80,

as long as the client maintains the lease for that TCP port mapping,
another client with a different internal IP address MUST NOT be able
to successfully acquire the mapping for UDP port 80.

The client normally requests the external port matching the internal
port. If that external port is not available, the NAT gateway MUST
return an available external port if possible, or return an error
code if no external ports are available.

The source address of the packet MUST be used for the internal
address in the mapping. This protocol is not intended to facilitate
one device behind a NAT creating mappings for other devices. If
there are legacy devices that require inbound mappings, permanent
mappings can be created manually by the user through an
administrative interface, just as they are today.

If a mapping already exists for a given internal address and port
(whether that mapping was created explicitly using NAT-PMP,
implicitly as a result of an outgoing TCP SYN packet, or manually by
a human administrator) and that client requests another mapping for
the same internal port (possibly requesting a different external
port), then the mapping request should succeed, returning the
already-assigned external port. This is necessary to handle the case
where a client requests a mapping with suggested external port X, and
is granted a mapping with actual external port Y, but the
acknowledgment packet gets lost. When the client retransmits its
mapping request, it should get back the same positive acknowledgment
as was sent (and lost) the first time.

The NAT gateway MUST NOT accept mapping requests destined to the NAT
gateway's external IP address or received on its external network
interface. Only packets received on the internal interface(s) with a
destination address matching the internal address(es) of the NAT
gateway should be allowed.

The NAT gateway MUST fill in the Seconds Since Start of Epoch field
with the time elapsed since its port mapping table was initialized on
startup or reset for any other reason (see Section 3.6, "Seconds
Since Start of Epoch").

The Port Mapping Lifetime is an unsigned integer in seconds. The NAT
gateway MAY reduce the lifetime from what the client requested. The
NAT gateway SHOULD NOT offer a lease lifetime greater than that
requested by the client.

Upon receiving the response packet, the client MUST check the source
IP address, and silently discard the packet if the address is not the
address of the gateway to which the request was sent.

The client SHOULD begin trying to renew the mapping halfway to expiry
time, like DHCP. The renewal packet should look exactly the same as
a request packet, except that the client SHOULD set the Suggested
External Port to what the NAT gateway previously mapped, not what the
client originally suggested. As described above, this enables the
gateway to automatically recover its mapping state after a crash or
reboot.

3.4. Destroying a Mapping

A mapping may be destroyed in a variety of ways. If a client fails
to renew a mapping, then at the time its lifetime expires, the
mapping MUST be automatically deleted. In the common case where the
gateway device is a combined DHCP server and NAT gateway, when a
client's DHCP address lease expires, the gateway device MAY
automatically delete any mappings belonging to that client.
Otherwise, a new client being assigned the same IP address could
receive unexpected inbound UDP packets or inbound TCP connection
requests that it did not ask for and does not want.

A client MAY also send an explicit packet to request deletion of a
mapping that is no longer needed. A client requests explicit
deletion of a mapping by sending a message to the NAT gateway
requesting the mapping, with the Requested Lifetime in Seconds set to
zero. The Suggested External Port MUST be set to zero by the client
on sending, and MUST be ignored by the gateway on reception.

When a mapping is destroyed successfully as a result of the client
explicitly requesting the deletion, the NAT gateway MUST send a
response packet that is formatted as defined in Section 3.3,
"Requesting a Mapping". The response MUST contain a result code of
0, the internal port as indicated in the deletion request, an
external port of 0, and a lifetime of 0. The NAT gateway MUST
respond to a request to destroy a mapping that does not exist as if
the request were successful. This is because of the case where the
acknowledgment is lost, and the client retransmits its request to
delete the mapping. In this case, the second request to delete the
mapping MUST return the same response packet as the first request.

If the deletion request was unsuccessful, the response MUST contain a
non-zero result code and the requested mapping; the lifetime is
undefined (MUST be set to zero on transmission, and MUST be ignored
on reception). If the client attempts to delete a port mapping that
was manually assigned by some kind of configuration tool, the NAT
gateway MUST respond with a "Not Authorized" error, result code 2.

When a mapping is destroyed as a result of its lifetime expiring or
for any other reason, if the NAT gateway's internal state indicates
that there are still active TCP connections traversing that now-
defunct mapping, then the NAT gateway SHOULD send appropriately
constructed TCP RST (reset) packets both to the local client and to
the remote peer on the Internet to terminate that TCP connection.

A client can request the explicit deletion of all its UDP or TCP
mappings by sending the same deletion request to the NAT gateway with
the external port, internal port, and lifetime set to zero. A client
MAY choose to do this when it first acquires a new IP address in
order to protect itself from port mappings that were performed by a
previous owner of the IP address. After receiving such a deletion
request, the gateway MUST delete all its UDP or TCP port mappings
(depending on the opcode). The gateway responds to such a deletion
request with a response as described above, with the internal port
set to zero. If the gateway is unable to delete a port mapping, for
example, because the mapping was manually configured by the
administrator, the gateway MUST still delete as many port mappings as
possible, but respond with a non-zero result code. The exact result
code to return depends on the cause of the failure. If the gateway
is able to successfully delete all port mappings as requested, it
MUST respond with a result code of zero.

3.5. Result Codes

Currently defined result codes:

0 - Success
1 - Unsupported Version
2 - Not Authorized/Refused
    (e.g., box supports mapping, but user has turned feature off)
3 - Network Failure
    (e.g., NAT box itself has not obtained a DHCP lease)
4 - Out of resources
    (NAT box cannot create any more mappings at this time)
5 - Unsupported opcode

If the version in the request is not zero, then the NAT-PMP server
MUST return the following "Unsupported Version" error response to the
client:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Vers = 0      | OP = 0        | Result Code = 1               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Seconds Since Start of Epoch (in network byte order)          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

If the opcode in the request is 128 or greater, then this is not a
request; it's a response, and the NAT-PMP server MUST silently ignore
it. Otherwise, if the opcode in the request is less than 128, but is
not a supported opcode (currently 0, 1, or 2), then the entire
request MUST be returned to the sender, with the top bit of the
opcode set (to indicate that this is a response) and the result code
set to 5 (Unsupported opcode).

For version 0 and a supported opcode (0, 1, or 2), if the operation
fails for some reason (Not Authorized, Network Failure, or Out of
resources), then a valid response MUST be sent to the client, with
the top bit of the opcode set (to indicate that this is a response)
and the result code set appropriately. Other fields in the response
MUST be set appropriately. Specifically:

o Seconds Since Start of Epoch MUST be set correctly

o The External IPv4 Address should be set to the correct address, or
  to 0.0.0.0, as appropriate.

o The Internal Port MUST be set to the client's requested Internal
  Port. This is particularly important, because the client needs
  this information to identify which request suffered the failure.

o The Mapped External Port and Port Mapping Lifetime MUST be set
  appropriately -- i.e., zero if no successful port mapping was
  created.

Should future NAT-PMP opcodes be defined, their error responses MUST
similarly be specified to include sufficient information to identify
which request suffered the failure. By design, NAT-PMP messages do
not contain any transaction identifiers. All NAT-PMP messages are
idempotent and self-describing; therefore, the specifications of
future NAT-PMP messages need to include enough information for their
responses to be self-describing.

Clients MUST be able to properly handle result codes not defined in
this document. Undefined results codes MUST be treated as fatal
errors of the request.

3.6. Seconds Since Start of Epoch

Every packet sent by the NAT gateway includes a Seconds Since Start
of Epoch (SSSoE) field. If the NAT gateway resets or loses the state
of its port mapping table, due to reboot, power failure, or any other
reason, it MUST reset its epoch time and begin counting SSSoE from
zero again. Whenever a client receives any packet from the NAT
gateway, either unsolicited or in response to a client request, the
client computes its own conservative estimate of the expected SSSoE
value by taking the SSSoE value in the last packet it received from
the gateway and adding 7/8 (87.5%) of the time elapsed according to
the client's local clock since that packet was received. If the
SSSoE in the newly received packet is less than the client's
conservative estimate by more than 2 seconds, then the client
concludes that the NAT gateway has undergone a reboot or other loss
of port mapping state, and the client MUST immediately renew all its
active port mapping leases as described in Section 3.7, "Recreating
Mappings on NAT Gateway Reboot".

3.7. Recreating Mappings on NAT Gateway Reboot

The NAT gateway MAY store mappings in persistent storage so that,
when it is powered off or rebooted, it remembers the port mapping
state of the network.

However, maintaining this state is not essential for correct
operation. When the NAT gateway powers on or clears its port mapping
state as the result of a configuration change, it MUST reset the
epoch time and re-announce its IPv4 address as described in Section
3.2.1, "Announcing Address Changes". Reception of this packet where
time has apparently gone backwards serves as a hint to clients on the
network that they SHOULD immediately send renewal packets (to
immediately recreate their mappings) instead of waiting until the
originally scheduled time for those renewals. Clients who miss
receiving those gateway announcement packets for any reason will
still renew their mappings at the originally scheduled time and cause
their mappings to be recreated; it will just take a little longer for
these clients.

A mapping renewal packet is formatted identically to an original
mapping request; from the point of view of the client, it is a
renewal of an existing mapping, but from the point of view of the
freshly rebooted NAT gateway, it appears as a new mapping request.

This self-healing property of the protocol is very important.

The remarkable reliability of the Internet as a whole derives in
large part from the fact that important state is held in the
endpoints, not in the network itself [ETEAISD]. Power-cycling an
Ethernet switch results only in a brief interruption in the flow of
packets; established TCP connections through that switch are not
broken, merely delayed for a few seconds. Indeed, a failing Ethernet
switch can even be replaced with a new one, and as long as the cables
are transferred over reasonably quickly, after the upgrade all the
TCP connections that were previously going through the old switch
will be unbroken and now going through the new one. The same is true
of IP routers, wireless base stations, etc. The one exception is NAT
gateways. Because the port mapping state is required for the NAT
gateway to know where to forward inbound packets, loss of that state
breaks connectivity through the NAT gateway. By allowing clients to
detect when this loss of NAT gateway state has occurred, and recreate
it on demand, we turn hard state in the network into soft state, and
allow it to be recovered automatically when needed.

Without this automatic recreation of soft state in the NAT gateway,
reliable long-term networking would not be achieved. As mentioned
above, the reliability of the Internet does not come from trying to
build a perfect network in which errors never happen, but from
accepting that in any sufficiently large system there will always be
some component somewhere that's failing, and designing mechanisms
that can handle those failures and recover. To illustrate this point
with an example, consider the following scenario: Imagine a network
security camera that has a web interface and accepts incoming
connections from web browser clients. Imagine this network security
camera uses NAT-PMP or a similar protocol to set up an inbound port
mapping in the NAT gateway so that it can receive incoming
connections from clients on the other side of the NAT gateway. Now,
this camera may well operate for weeks, months, or even years.
During that time, it's possible that the NAT gateway could experience
a power failure or be rebooted. The user could upgrade the NAT
gateway's firmware, or even replace the entire NAT gateway device
with a newer model. The general point is that if the camera operates
for a long enough period of time, some kind of disruption to the NAT
gateway becomes inevitable. The question is not whether the NAT
gateway will lose its port mappings, but when, and how often. If the
network camera and devices like it on the network can detect when the
NAT gateway has lost its port mappings, and recreate them

automatically, then these disruptions are self-correcting and largely
invisible to the end user. If, on the other hand, the disruptions
are not self-correcting, and after a NAT gateway reboot the user has
to manually reset or reboot all the other devices on the network too,
then these disruptions are *very* visible to the end user. This
aspect of the design is part of what makes the difference between a
protocol that keeps on working indefinitely over a time scale of
months or years, and a protocol that works in brief testing, but in
the real world is continually failing and requiring manual
intervention to get it going again.

When a client renews its port mappings as the result of receiving a
packet where the Seconds Since Start of Epoch (SSSoE) field indicates
that a reboot or similar loss of state has occurred, the client MUST
first delay by a random amount of time selected with uniform random
distribution in the range 0 to 5 seconds, and then send its first
port mapping request. After that request is acknowledged by the
gateway, the client may then send its second request, and so on, as
rapidly as the gateway allows. The requests SHOULD be issued
serially, one at a time; the client SHOULD NOT issue multiple
concurrent requests.

The discussion in this section focuses on recreating inbound port
mappings after loss of NAT gateway state, because that is the more
serious problem. Losing port mappings for outgoing connections
destroys those currently active connections, but does not prevent
clients from establishing new outgoing connections. In contrast,
losing inbound port mappings not only destroys all existing inbound
connections, but also prevents the reception of any new inbound
connections until the port mapping is recreated. Accordingly, we
consider recovery of inbound port mappings more important. However,
clients that want outgoing connections to survive a NAT gateway
reboot can also achieve that using NAT-PMP, in the common case of a
residential NAT gateway with a single, relatively stable, external IP
address. After initiating an outbound TCP connection (which will
cause the NAT gateway to establish an implicit port mapping), the
client should send the NAT gateway a port mapping request for the
source port of its TCP connection, which will cause the NAT gateway
to send a response giving the external port it allocated for that
mapping. The client can then store this information, and use it
later to recreate the mapping if it determines that the NAT gateway
has lost its mapping state.

3.8. NAT Gateways with NAT Function Disabled

Note that only devices that are *currently* acting in the role of NAT
gateway should participate in NAT-PMP protocol exchanges with
clients. A network device that is capable of NAT (and NAT-PMP) but

is currently configured not to perform that function (e.g., it is
acting as a traditional IP router, forwarding packets without
modifying them) MUST NOT respond to NAT-PMP requests from clients nor
send spontaneous NAT-PMP address-change announcements.

In particular, a network device not currently acting in the role of
NAT gateway should not even respond to NAT-PMP requests by returning
an error code such as 2, "Not Authorized/Refused", since to do so is
misleading to clients -- it suggests that NAT port mapping is
necessary on this network for the client to successfully receive
inbound connections, but is not available because the administrator
has chosen to disable that functionality.

Clients should also be careful to avoid making unfounded assumptions,
such as the assumption that if the client has an IPv4 address in one
of the private IPv4 address ranges [RFC1918], then that means NAT
necessarily must be in use. Net 10/8 has enough addresses to build a
private network with millions of hosts and thousands of
interconnected subnets, all without any use of NAT. Many
organizations have built such private networks that benefit from
using standard TCP/IP technology, but by choice do not connect to the
public Internet. The purpose of NAT-PMP is to mitigate some of the
damage caused by NAT. It would be an ironic and unwanted side effect
of this protocol if it were to lead well-meaning but misguided
developers to create products that refuse to work on a private
network *unless* they can find a NAT gateway to talk to.
Consequently, a client finding that NAT-PMP is not available on its
network should not give up, but should proceed on the assumption that
the network may be a traditional routed IP network, with no address
translation being used. This assumption may not always be true, but
it is better than the alternative of falsely assuming the worst and
not even trying to use normal (non-NAT) IP networking.

If a network device not currently acting in the role of NAT gateway
receives UDP packets addressed to port 5351, it SHOULD respond
immediately with an "ICMP Port Unreachable" message to tell the
client that it needn't continue with timeouts and retransmissions,
and it should assume that NAT-PMP is not available and not needed on
this network. Typically, this behavior can be achieved merely by not
having an open socket listening on UDP port 5351.

3.9. All Mappings Are Bidirectional

   All NAT mappings, whether created implicitly by an outbound packet,
   created explicitly using NAT-PMP, or configured statically, are
   bidirectional. This means that when an outbound packet from a
   particular internal address and port is translated to an external

address and port, replies addressed to that external address and port
need to be translated back to the corresponding internal address and
port.

The converse is also true. When an inbound packet is received that
is addressed to an external address and port that matches an existing
mapping (implicit, explicit, or static), it is translated to the
corresponding internal address and port and forwarded. Outbound
replies from that internal address and port need to be translated to
the correct external address and port so that they are correctly
recognized by the remote peer.

In particular, if an outbound UDP reply that matches an existing
explicit or static mapping is instead treated like a "new" outbound
UDP packet, and a new dynamic mapping is created (with a different
external address and port), then at the time that packet arrives at
the remote peer it will not be recognized as a valid reply. For TCP
this bug is quickly spotted because all TCP implementations will
ignore replies with the wrong apparent source address and port. For
UDP this bug can more easily go unnoticed because some UDP clients
neglect to check the source address and port of replies; thus, they
will appear to work some of the time with NAT gateways that put the
wrong source address and port in outbound packets. All NAT gateways
MUST ensure that mappings, however created, are bidirectional.

4. UNSAF Considerations

The document "IAB Considerations for UNilateral Self-Address Fixing
(UNSAF) Across Network Address Translation (NAT)" [RFC3424] covers a
number of issues when working with NATs. It outlines some
requirements for any document that attempts to work around problems
associated with NATs. This section addresses those requirements.

4.1. Scope

This protocol addresses the needs of TCP and UDP transport peers that
are separated from the public Internet by exactly one IPv4 NAT. Such
peers must have access to some form of directory server for
registering the public IPv4 address and port at which they can be
reached.

4.2. Transition Plan

Any client making use of this protocol SHOULD implement IPv6 support.
If a client supports IPv6 and is running on a device with a global
IPv6 address, that IPv6 address SHOULD be preferred to the IPv4
external address learned via this NAT mapping protocol. In case
other clients do not have IPv6 connectivity, both the IPv4 and IPv6

addresses SHOULD be registered with whatever form of directory server
is used. Preference SHOULD be given to IPv6 addresses when
available. By implementing support for IPv6 and using this protocol
for IPv4, vendors can ship products today that will work under both
scenarios. As IPv6 becomes more widely deployed, clients of this
protocol following these recommendations will transparently make use
of IPv6.

4.3. Failure Cases

Aside from NATs that do not implement this protocol, there are a
number of situations where this protocol may not work.

4.3.1. NAT behind NAT

Some people's primary IPv4 address, assigned by their ISP, may itself
be a NAT address. In addition, some people may have an external IPv4
address, but may then double NAT themselves, perhaps by choice or
perhaps by accident. Although it might be possible in principle for
one NAT gateway to recursively request a mapping from the next one,
this document does not advocate that and does not try to prescribe
how it would be done.

It would be a lot of work to implement nested NAT port mapping
correctly, and there are a number of reasons why the end result might
not be as useful as we might hope. Consider the case of an ISP that
offers each of its customers only a single NAT address. This ISP
could instead have chosen to provide each customer with a single
public IPv4 address, or, if the ISP insists on running NAT, it could
have chosen to allow each customer a reasonable number of addresses,
enough for each customer device to have its own NAT address directly
from the ISP. If, instead, this ISP chooses to allow each customer
just one and only one NAT address, forcing said customer to run
nested NAT in order to use more than one device, it seems unlikely
that such an ISP would be so obliging as to provide a NAT service
that supports NAT-PMP. Supposing that such an ISP did wish to offer
its customers NAT service with NAT-PMP so as to give them the ability
to receive inbound connections, this ISP could easily choose to allow
each client to request a reasonable number of DHCP addresses from
that gateway. Remember that Net 10/8 [RFC1918] allows for over 16
million addresses, so NAT addresses are not in any way in short
supply. A single NAT gateway with 16 million available addresses is
likely to run out of packet forwarding capacity before it runs out of
internal addresses to hand out. In this way, the ISP could offer
single-level NAT with NAT-PMP, obviating the need to support nested
NAT-PMP. In addition, an ISP that is motivated to provide their
customers with unhindered access to the Internet by allowing incoming
as well as outgoing connections has better ways to offer this

service. Such an ISP could offer its customers real public IPv4
addresses instead of NAT addresses, or could choose to offer its
customers full IPv6 connectivity, where no mapping or translation is
required at all.

Note: In the nine years since NAT-PMP was designed, the pool of
available IPv4 addresses has been exhausted, and many ISPs now offer
translated IPv4 addresses out of necessity. Such ISPs have indicated
a willingness to offer PCP service to their customers.

4.3.2. NATs with Multiple External IPv4 Addresses

If a NAT maps internal addresses to multiple external addresses, then
it SHOULD pick one of those external addresses as the one it will
support for inbound connections, and for the purposes of this
protocol it SHOULD act as if that address were its only address.

4.3.3. NATs and Routed Private Networks

In some cases, a large network may be subnetted. Some sites may
install a NAT gateway and subnet the private network. Such
subnetting breaks this protocol because the router address is not
necessarily the address of the device performing NAT.

Addressing this problem is not a high priority. Any site with the
resources to set up such a configuration should have the resources to
add manual mappings or attain a range of globally unique addresses.

Not all NATs will support this protocol. In the case where a client
is run behind a NAT that does not support this protocol, the software
relying on the functionality of this protocol may be unusable.

4.3.4. Communication between Hosts behind the Same NAT

NAT gateways supporting NAT-PMP should also implement "hairpin
translation". Hairpin translation means supporting communication
between two local clients being served by the same NAT gateway.

Suppose device A is listening on internal address and port
10.0.0.2:80 for incoming connections. Using NAT-PMP, device A has
obtained a mapping to external address and port x.x.x.x:80, and has
recorded this external address and port in a public directory of some
kind. For example, it could have created a DNS SRV record containing
this information, and recorded it, using DNS Dynamic Update
[RFC3007], in a publicly accessible DNS server. Suppose then that
device B, behind the same NAT gateway as device A, but unknowing or
uncaring of this fact, retrieves device A's DNS SRV record and
attempts to open a TCP connection to x.x.x.x:80. The outgoing

packets addressed to this public Internet address will be sent to the
NAT gateway for translation and forwarding. Having translated the
source address and port number on the outgoing packet, the NAT
gateway needs to be smart enough to recognize that the destination
address is in fact itself, and then feed this packet back into its
packet reception engine, to perform the destination port mapping
lookup to translate and forward this packet to device A at address
and port 10.0.0.2:80.

4.3.5. Non-UDP/TCP Transport Traffic

Any communication over transport protocols other than TCP and UDP
will not be served by this protocol. Examples are Generic Routing
Encapsulation (GRE), Authentication Header (AH), and Encapsulating
Security Payload (ESP).

4.4. Long-Term Solution

As IPv6 is deployed, clients of this protocol supporting IPv6 will be
able to bypass this protocol and the NAT when communicating with
other IPv6 devices. In order to ensure this transition, any client
implementing this protocol SHOULD also implement IPv6 and use this
solution only when IPv6 is not available to both peers.

4.5. Existing Deployed NATs

Existing deployed NATs will not support this protocol. This protocol
will only work with NATs that are upgraded to support it.

5. Security Considerations

As discussed in Section 3.2, "Determining the External Address", only
a client on the internal side of the NAT may create port mappings,
and it may do so only on its own behalf. By using IP address
spoofing, it's possible for one client to delete the port mappings of
another client. It's also possible for one client to create port
mappings on behalf of another client. In cases where this is a
concern, it can be dealt with using IPsec [RFC4301].

The multicast announcements described in Section 3.2.1, "Announcing
Address Changes", could be spoofed, facilitating a denial-of-service
attack. This makes NAT-PMP unsuitable for use on LANs with large
numbers of hosts where one or more of the hosts may be untrustworthy.

Another concern is that rogue software running on a local host could
create port mappings for unsuspecting hosts, thereby rendering them
vulnerable to external attack. However, it's not clear how realistic
this threat model is, since rogue software on a local host could

attack such unsuspecting hosts directly itself, without resorting to
such a convoluted indirect technique. This concern is also a little
misguided because it is based on the assumption that a NAT gateway
and a firewall are the same thing, which they are not.

Some people view the property of NATs blocking inbound connections as
a security benefit that is undermined by this protocol. The authors
of this document have a different point of view. In the days before
NAT became prevalent, all hosts had unique public IP addresses, and
had unhindered ability to communicate with any other host on the
Internet (a configuration that is still surprisingly common). Using
NAT breaks this unhindered connectivity, relegating hosts to second-
class status, unable to receive inbound connections. This protocol
goes some way to partially reverse that damage. The purpose of a NAT
gateway should be to allow several hosts to share a single address,
not to simultaneously impede those host's ability to communicate
freely. Security is most properly provided by end-to-end
cryptographic security, and/or by explicit firewall functionality, as
appropriate. Blocking of certain connections should occur only as a
result of explicit and intentional firewall policy, not as an
accidental side effect of some other technology.

However, since many users do have an expectation that their NAT
gateways can function as a kind of firewall, any NAT gateway
implementing this protocol SHOULD have an administrative mechanism to
disable it, thereby restoring the pre-NAT-PMP behavior.

6. IANA Considerations

   UDP ports 5350 and 5351 have been assigned for use by NAT-PMP, and
   subsequently by its successor, Port Control Protocol [RFC6887].

   No further IANA services are required by this document.

7. Acknowledgments

   The concepts described in this document have been explored,
   developed, and implemented with help from Mark Baugher, Bob Bradley,
   Josh Graessley, Rory McGuire, Rob Newberry, Roger Pantos, John
   Saxton, Kiren Sekar, Jessica Vazquez, and James Woodyatt.

   Special credit goes to Mike Bell, the Apple Vice President who
   recognized the need for a clean, elegant, reliable Port Mapping
   Protocol, and made the decision early on that Apple's AirPort base
   stations would support NAT-PMP.

8. Deployment History

   In August 2004, NAT-PMP client software first became available to the
   public through Apple's Darwin Open Source code. In April 2005,
   NAT-PMP implementations began shipping to end users with the launch
   of Mac OS X 10.4 Tiger and Bonjour for Windows 1.0, and in June 2005
   the protocol was first publicly documented in the original draft
   version of this document.

   The NAT-PMP client in Mac OS X 10.4 Tiger and Bonjour for Windows
   exists as part of the mDNSResponder/mdnsd system service. When a
   client advertises a service using Wide Area Bonjour [RFC6763], and
   the machine is behind a NAT-PMP-capable NAT gateway, and the machine
   is so configured, the mDNSResponder system service automatically uses
   NAT-PMP to set up an inbound port mapping, and then records the
   external IPv4 address and port in the global DNS. Existing client
   software using the Bonjour programming APIs [Bonjour] got this new
   NAT traversal functionality automatically. The logic behind this
   decision was that if client software publishes its information into
   the global DNS via Wide Area Bonjour service advertising, then it's
   reasonable to infer an expectation that this information should
   actually be usable by the peers retrieving it. Generally speaking,
   recording a private IPv4 address like 10.0.0.2 in the public DNS is
   likely to be pointless because that address is not reachable from
   clients on the other side of the NAT gateway. In the case of a home
   user with a single computer directly connected to their Cable or DSL
   modem, with a single global IPv4 address and no NAT gateway (a common
   configuration at that time), publishing the machine's global IPv4
   address into the global DNS is useful, because that IPv4 address is
   globally reachable. In contrast, a home user using a NAT gateway to
   share a single global IPv4 address between several computers loses
   this ability to receive inbound connections. This breaks many peer-
   to-peer collaborative applications, like the multi-user text editor
   SubEthaEdit [SEE]. For many users, moving from one computer with a
   global IPv4 address, to two computers using NAT to share a single
   global IPv4 address, loss of inbound reachability was an unwanted
   side effect of using NAT for address sharing. Automatically creating
   the necessary inbound port mappings helped remedy this unwanted side
   effect of NAT.

   The server side of the NAT-PMP protocol is implemented in Apple's
   AirPort Extreme, AirPort Express, and Time Capsule wireless base
   stations, and in the Internet Sharing feature of Mac OS X 10.4 and
   later. Some third-party NAT vendors, such as Peplink, also offer
   NAT-PMP in their products.

   In Mac OS X 10.4 Tiger, the NAT-PMP client was invoked automatically
   as a side effect of clients requesting Wide Area Bonjour service
   registrations. Using NAT-PMP without an associated Wide Area Bonjour
   service registration required use of a third-party client library.

   In October 2007, Mac OS X 10.5 Leopard added the "DNSServiceNATPort-
   MappingCreate" API, which made NAT-PMP client functionality directly
   available, so software could use it with other directory and
   rendezvous mechanisms in addition to Wide Area Bonjour DNS Updates.

   In 2013, NAT-PMP was superseded by the IETF Standards Track Port
   Control Protocol [RFC6887]. PCP builds on NAT-PMP and uses a
   compatible packet format, and adds a number of significant
   enhancements, including IPv6 support, management of outbound
   mappings, management of firewall rules, full compatibility with
   large-scale NATs with a pool of external addresses, error lifetimes,
   and an extension mechanism to enable future enhancements.

9. Noteworthy Features of NAT Port Mapping Protocol and PCP

   Some readers have asked how NAT-PMP and PCP compare to other similar
   solutions, particularly the UPnP Forum's Internet Gateway Device
   (IGD) Device Control Protocol [IGD].

   The answer is that although the Universal Plug and Play (UPnP) IGD
   protocol is often used as a way for client devices to create port
   mappings programmatically, it's not ideal for that task. Whereas
   NAT-PMP was explicitly designed to be used primarily by software
   entities managing their own port mappings, UPnP IGD is more tailored
   towards being used by humans configuring all the settings of their
   gateway using some GUI tool. This difference in emphasis leads to
   protocol differences. For example, while it is reasonable and
   sensible to require software entities to renew their mappings
   periodically to prove that they are still there (like a device
   renewing its DHCP address lease), it would be unreasonable to require
   the same thing of a human user. When a human user configures their
   gateway, they expect it to stay configured that way until they decide
   to change it. If they configure a port mapping, they expect it to
   stay configured until they decide to delete it.

   Because of this focus on being a general administration protocol for
   all aspects of home gateway configuration, UPnP IGD is a large and
   complicated collection of protocols (360 pages of specification
   spread over 13 separate documents, not counting supporting protocol
   specifications like Simple Service Discovery Protocol (SSDP) and
   Extensible Markup Language (XML)). While it may be a fine way for

human users to configure their home gateways, it is not especially
suited to the task of programmatically creating dynamic port
mappings.

The requirements for a good port mapping protocol, requirements that
are met by NAT-PMP, are outlined below.

9.1. Simplicity

Many home gateways, and many of the devices that connect to them, are
small, low-cost devices, with limited RAM, flash memory, and CPU
resources. Protocols they use should be considerate of this,
supporting a small number of simple operations that can be
implemented easily with a small amount of code. A quick comparison,
based on page count of the respective documents alone, suggests that
NAT-PMP is at least ten times simpler than UPnP IGD.

9.2. Focused Scope

The more things a protocol can do, the more chance there is that
something it does could be exploited for malicious purposes. NAT-PMP
is tightly focused on the specific task of creating port mappings.
Were the protocol to be misused in some way, this helps limit the
scope of what mischief could be performed using the protocol.

Because UPnP IGD allows control over all home gateway configuration
settings, the potential for mischief is far greater. For example, a
UPnP IGD home gateway allows messages that tell it to change the DNS
server addresses that it sends to clients in its DHCP packets. Using
this mechanism, a single item of malicious web content (e.g., a rogue
Flash banner advert on a web page) can make a persistent change to
the home gateway's configuration without the user's knowledge, such
that all future DNS requests by all local clients will be sent to a
rogue DNS server. This allows criminals to perform a variety of
mischief, such as hijacking connections to bank web sites and
redirecting them to the criminals' web servers instead [VU347812].

9.3. Efficiency

In addition to low-cost home gateways, many of the clients will also
be similarly constrained low-cost devices with limited RAM resources.

When implementing a NAT-PMP client on a constrained device, it's
beneficial to have well-defined bounds on RAM requirements that are
fixed and known in advance. For example, when requesting the
gateway's external IPv4 address, a NAT-PMP client on Ethernet knows

that to receive the reply it will require 14 bytes for the Ethernet
header, 20 bytes for the IPv4 header, 8 bytes for the UDP header, and
12 bytes for the NAT-PMP payload, making a total of 54 bytes.

In contrast, UPnP IGD uses an XML reply of unbounded size. It is not
uncommon for a UPnP IGD device to return an XML document 4000 to 8000
bytes in size to communicate its 4-byte external IPv4 address, and
the protocol specification places no upper bound on how large the XML
response may be, so there's nothing to stop the reply being even
larger. This means that developers of UPnP client devices can only
guess at how much memory they may need to receive the XML reply.
Operational experience suggests that 10,000 bytes is usually enough
for most UPnP IGD home gateways today, but that's no guarantee that
some future UPnP IGD home gateway might not return a perfectly legal
XML reply much larger than that.

In addition, because the XML reply is too large to fit in a single
UDP packet, UPnP IGD has to use a TCP connection, thereby adding the
overhead of TCP connection setup and teardown.

The process of discovering a UPnP IGD home gateway's external IPv4
address consists of:

o SSDP transaction to discover the TCP port to use, and the "URL" of
  the XML document to fetch from the gateway. Following the SSDP
  specification, this is 3 multicast requests, eliciting 9 unicast
  responses.

o HTTP "GET" request to get the device description. Typically, 16
  packets: 3 for TCP connection setup, 9 packets of data exchange,
  and a 4-packet FIN-ACK-FIN-ACK sequence to close the connection.

o HTTP "POST" to request the external IPv4 address. Typically, 14
  packets: 3 for TCP connection setup, 7 packets of data exchange,
  and a 4-packet FIN-ACK-FIN-ACK sequence to close the connection.

To retrieve the external IPv4 address NAT-PMP takes a 2-packet UDP
exchange (44-byte request, 54-byte response); the same thing using
UPnP IGD takes 42 packets and thousands of bytes.

Similarly, UPnP IGD's HTTP "POST" request for a port mapping is
typically a 14-packet exchange, compared with NAT-PMP's 2-packet UDP
exchange.

9.4. Atomic Allocation Operations

   Some of the useful properties of NAT-PMP were inspired by DHCP, a
   reliable and successful protocol. For example, DHCP allows a client
   to request a desired IP address, but if that address is already in
   use the DHCP server will instead assign some other available address.

   Correspondingly, NAT-PMP allows a client to request a desired
   external port, and if that external port is already in use by some
   other client, the NAT-PMP server will instead assign some other
   available external port.

   UPnP IGD does not do this. If a UPnP IGD client requests an external
   port that has already been allocated, then one of two things happens.

   Some UPnP IGD home gateways just silently overwrite the old mapping
   with the new one, causing the previous client to lose connectivity.
   If the previous client renews its port mapping, then it in turn
   overwrites the new mapping, and the two clients fight over the same
   external port indefinitely, neither achieving reliable connectivity.

   Other IGD home gateways return a "Conflict" error if the port is
   already in use, which does at least tell the client what happened,
   but doesn't tell the client what to do. Instead of the NAT gateway
   (which does know which ports are available) assigning one to the
   client, the NAT gateway makes the client (which doesn't know) keep
   guessing until it gets lucky. This problem remains mild as long as
   not many clients are using UPnP IGD, but gets progressively worse as
   the number of clients on the network requesting port mappings goes
   up. In addition, UPnP IGD works particularly badly in conjunction
   with the emerging policy of allocating pre-assigned port ranges to
   each client. If a client is assigned TCP port range 63488-64511, and
   the UPnP IGD client requests TCP port 80, trying successive
   incrementing ports until it succeeds, then the UPnP IGD client will
   have to issue 63,409 requests before it succeeds.

9.5. Garbage Collection

   In any system that operates for a long period of time (as a home
   gateway should), it is important that garbage data does not
   accumulate indefinitely until the system runs out of memory and
   fails.

   Similar to how DHCP leases an IP address to a client for a finite
   length of time, NAT-PMP leases an external port to a client for a
   finite length of time. The NAT-PMP client must renew the port
   mapping before it expires, or, like an unrenewed DHCP address, it
   will be reclaimed. If a laptop computer is abruptly disconnected

from the network without the opportunity to delete its port mappings, the NAT gateway will reclaim those mappings when they are not renewed.

In principle, UPnP IGD should allow clients to specify a lifetime on port mappings. However, a Google search for "UPnP NewLeaseDuration" shows that in practice pretty much every client uses "<NewLeaseDuration>0</NewLeaseDuration>" to request an infinite lease, and the protocol has no way for the NAT gateway to decline that infinite lease request and require the client to renew it at reasonable intervals. Furthermore, anecdotal evidence is that if the client requests a lease other than zero, there are IGD home gateways that will ignore the request, fail in other ways, or even crash completely. As a client implementer then, you would be well advised not to attempt to request a lease other than zero, unless you want to suffer the support costs and bad publicity of lots of people complaining that your device brought down their entire network.

Because none of the early UPnP IGD clients requested port mapping leases, many UPnP IGD home gateway vendors never tested that functionality, and got away with shipping home gateways where that functionality was buggy or nonexistent. Because there are so many buggy UPnP IGD home gateways already deployed, client writers wisely stick to the well-trodden path of only requesting infinite leases. Because there are now few (if any) clients attempting to request non-zero leases, home gateway vendors have little incentive to expend resources implementing a feature no one uses.

This unfortunate consequence of the way UPnP IGD was developed and deployed means that in practice it has no usable port mapping lease facility today, and therefore when run for a long period of time UPnP IGD home gateways have no good way to avoid accumulating an unbounded number of stale port mappings.

9.6. State Change Announcements

When using DHCP on the external interface, as is the norm for home gateways, there is no guarantee that a UPnP IGD home gateway's external IPv4 address will remain unchanged. Indeed, some ISPs change their customer's IPv4 address every 24 hours (possibly in an effort to make it harder for their customers to "run a server" at home). What this means is that if the home gateway's external IPv4 address changes, it needs to inform its clients, so that they can make any necessary updates to global directory information (e.g., performing a Dynamic DNS update to update their address record).

When a NAT-PMP gateway's external IPv4 address changes, it broadcasts announcement packets to inform clients of this. UPnP IGD does not.

9.7. Soft State Recovery

   When run for a long enough period of time, any network will have
   devices that fail, get rebooted, suffer power outages, or lose state
   for other reasons. A home gateway that runs for long enough is
   likely to suffer some such incident eventually. After losing state,
   it has no record of the port mappings it created, and clients suffer
   a consequent loss of connectivity.

   To handle this case, NAT-PMP has the "Seconds Since Start of Epoch"
   mechanism. After a reboot or other loss of state, a NAT-PMP gateway
   broadcasts announcement packets giving its external IPv4 address,
   with the Seconds Since Start of Epoch field reset to begin counting
   from zero again. When a NAT-PMP client observes packets from its
   NAT-PMP gateway where the gateway's notion of time has apparently
   gone backwards compared to the client's, the client knows the gateway
   has probably lost state, and immediately recreates its mappings to
   restore connectivity.

   UPnP IGD has no equivalent mechanism.

9.8. On-Path NAT Discovery

   For any given host, it is only useful to request NAT port mappings in
   the NAT gateway through which that host's packets are flowing. A NAT
   port mapping is a request for packets to be translated in a certain
   way; the NAT gateway can only perform that translation if it's
   actually forwarding inbound and outbound packets for that host.

   This is why NAT-PMP sends its requests to the host's default router,
   since this is the device that is forwarding (and possibly
   translating) inbound and outbound packets for that host. (In a
   larger network with multiple hops between a host and its NAT gateway,
   some other mechanism would need to be used to discover the correct
   on-path NAT for a host; this is possible, but outside the scope of
   this document.)

   In contrast, UPnP IGD does not limit itself to using only on-path
   NATs. UPnP IGD uses a multicast SSDP query, and uses any device it
   finds on the local network claiming UPnP IGD capability, regardless
   of whether any inbound or outbound traffic is actually flowing
   through that device. Over the past few years this led to many bug
   reports being sent to Apple with the general form: "Port Mapping
   doesn't work on my Mac and that's a bug because everything else on my
   network says UPnP IGD is working fine." Upon investigation it always
   turned out that: (i) these people had NAT gateways that either didn't
   support port mapping requests, or had that capability disabled, and
   (ii) for some reason they also had some other old NAT device still

connected to their network, and those other NAT devices were
advertising UPnP IGD capability, even though they were not the active
NAT gateway for the network. This led to UPnP IGD clients falsely
reporting that they were "working fine", and only the Mac correctly
reporting that it was unable to make any useful port mappings. In
many cases the people reporting this "bug" had devices like game
consoles on their home network that for many years had been reporting
that UPnP IGD was "working fine", yet during those years they had
never once successfully received any inbound network packet or
connection. The irony is that, for these people who were reporting
bugs to Apple, UPnP IGD "working fine" had been indistinguishable
from UPnP IGD doing nothing useful at all. It was only when Back to
My Mac [RFC6281] started reporting that it was unable to make any
functional port mappings that these people discovered they'd never
had any working port mappings on their NAT gateway.

10. References

10.1. Normative References

   [RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G.,
              and E. Lear, "Address Allocation for Private Internets",
              BCP 5, RFC 1918, February 1996.

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
              Requirement Levels", BCP 14, RFC 2119, March 1997.

10.2. Informative References

   [Bonjour] Apple "Bonjour" <http://developer.apple.com/bonjour/>.

   [ETEAISD] J. Saltzer, D. Reed and D. Clark: "End-to-end arguments in
              system design", ACM Trans. Comp. Sys., 2(4):277-88,
              November 1984.

   [IGD]   UPnP Standards "Internet Gateway Device (IGD) Standardized
              Device Control Protocol V 1.0", November 2001,
              <http://www.upnp.org/standardizeddcps/igd.asp>.

   [RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC
              2131, March 1997.

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

   [RFC3007] Wellington, B., "Secure Domain Name System (DNS) Dynamic
              Update", RFC 3007, November 2000.

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

   [RFC3424] Daigle, L., Ed., and IAB, "IAB Considerations for
              UNilateral Self-Address Fixing (UNSAF) Across Network
              Address Translation", RFC 3424, November 2002.

   [RFC4301] Kent, S. and K. Seo, "Security Architecture for the
              Internet Protocol", RFC 4301, December 2005.

   [RFC6281] Cheshire, S., Zhu, Z., Wakikawa, R., and L. Zhang,
              "Understanding Apple's Back to My Mac (BTMM) Service", RFC
              6281, June 2011.

   [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service
              Discovery", RFC 6763, February 2013.

   [RFC6887] Wing, D., Ed., Cheshire, S., Boucadair, M., Penno, R., and
              P. Selkirk, "Port Control Protocol (PCP)", RFC 6887, April
              2013.

   [SEE]   SubEthaEdit, <http://www.codingmonkeys.de/subethaedit/>.

   [VU347812] United States Computer Emergency Readiness Team
              Vulnerability Note VU#347812,
              <http://www.kb.cert.org/vuls/id/347812>.

Authors' Addresses

   Stuart Cheshire
   Apple Inc.
   1 Infinite Loop
   Cupertino, CA 95014
   USA

   EMail: cheshire@apple.com

   Marc Krochmal
   Apple Inc.
   1 Infinite Loop
   Cupertino, CA 95014
   USA

   EMail: marc@apple.com