Optimizing NAT and Firewall Keepalives Using Port Control Protocol (PCP)
                    draft-ietf-pcp-optimize-keepalives-04

Abstract

   This document describes how Port Control Protocol is useful in
   reducing NAT and firewall keepalive messages for a variety of
   applications.

Status of This Memo

   This Internet-Draft is submitted in full conformance with the
   provisions of BCP 78 and BCP 79.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF). Note that other groups may also distribute
   working documents as Internet-Drafts. The list of current Internet-
   Drafts is at http://datatracker.ietf.org/drafts/current/.

   Internet-Drafts are draft documents valid for a maximum of six months
   and may be updated, replaced, or obsoleted by other documents at any
   time. It is inappropriate to use Internet-Drafts as reference
   material or to cite them other than as "work in progress."

   This Internet-Draft will expire on March 23, 2015.

the Trust Legal Provisions and are provided without warranty as
described in the Simplified BSD License.

Table of Contents

1. Introduction

Many types of applications need to keep their Network Address
Translator (NAT) and Firewall (FW) mappings alive for long periods of
time, even when they are otherwise not sending or receiving any
traffic. This is typically done by sending periodic keep-alive
messages just to prevent the mappings from expiring. As NAT/FW
mapping timers may be short and unknown to the endpoint, the
frequency of these keepalives may be high. An IPv4 or IPv6 host can
use the Port Control Protocol (PCP)[RFC6887] to flexibly manage the
IP address and port mapping information on NATs and FWs to facilitate
communications with remote hosts. This document describes how PCP
can be used to reduce keepalive messages for both client-server and
peer-to-peer type of communication.

The mechanism described in this document is especially useful in cellular mobile networks, where frequent keepalive messages make the radio transition between active and power-save states causing congestion in the signaling path. The excessive time spent on the active state due to keepalives also greatly reduces the battery life of the cellular connected devices such as smartphones or tablets. According to requirement #14 in [I-D.binet-v6ops-cellular-host-reqs-rfc3316update] a cellular host SHOULD support PCP in order to save battery consumption exacerbated by keepalive messages.

2. Notational Conventions

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 [RFC2119].

This note uses terminology defined in [RFC5245] and [RFC6887].

3. Overview of Operation

3.1. Application Scenarios

PCP can help both client-server and peer-to-peer applications to reduce their keepalive rate. The relevant applications are the ones that need to keep their NAT/FW mappings alive for long periods of time, for instance to be able to send or receive application messages in both directions at any time.

A typical client-server scenario is depicted in Figure 1. A client, who may reside behind one or multiple layers of NATs/FWs, opens a connection to a globally reachable server, and keeps it open to be able to receive messages from the server at any time. The connection may be a connection-oriented transport protocol such as TCP or SCTP or connection-less transport protocol such as UDP. Protocols operating in this manner include Session Initiation Protocol (SIP) [RFC3261], Extensible Messaging and Presence Protocol (XMPP) [RFC3921], Internet Mail Application Protocol (IMAP) [RFC2177] with its IDLE command, the WebSocket protocol [RFC6455] and the various HTTP long-polling protocols. There are also a number of proprietary instant messaging, Voice over IP, e-mail and notification delivery protocols that belong in this category. All of these protocols aim to keep the client-server connection alive for as long as the application is running. When the application has otherwise no traffic to send, specific keepalive messages are sent periodically to ensure that the NAT/FW state in the middle does not expire. The client can use PCP to keep the required mapping at the NAT/FW and use

application keepalives to keep the state on the Application Server/
Peer as mentioned in Section 3.4.

```
         PCP             PCP
       Client          Server         _____
   +-----------+    +------+     /          \    +-----------+
   |Application|___| NAT/ |____| Internet |___|Application|
   | Client    |   | FW   |    |          |   |  Server   |
   +-----------+    +------+     _____/    +-----------+
                   (multiple
                    layers)

        ------------> PCP

        ---------------------------------------->
                  Application keepalive
```
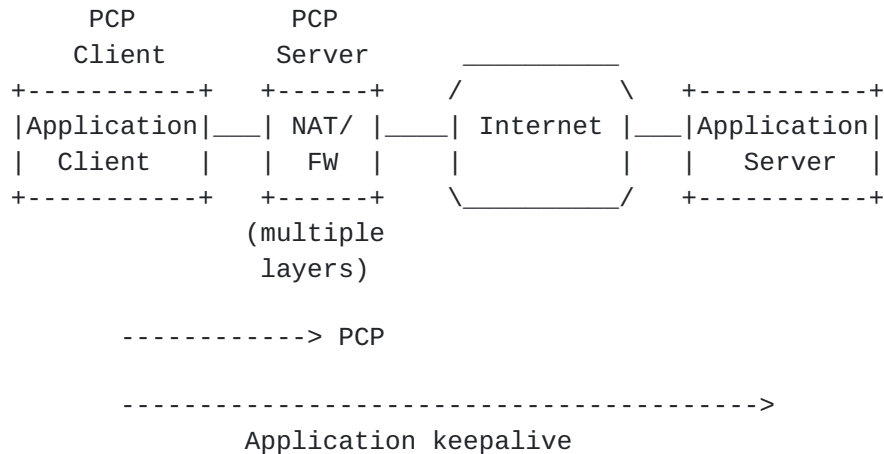
              Figure 1: PCP with Client-Server applications

There are also scenarios where the long-term communication
association is between two peers, both of whom may reside behind one
or more layers of NAT/FW. This is depicted in Figure 2. The
initiation of the association may have happened using mechanisms such
as Interactive Communications Establishment (ICE), perhaps first
triggered by a "signaling" protocol such as SIP or XMPP or WebRTC
[I-D.ietf-rtcweb-overview]. Examples of the peer-to-peer protocols
include RTP and WebRTC data channel. A number of proprietary VoIP or
video call or streaming or file transfer protocols also exist in this
category. Typically the communication is based on UDP, but TCP or
SCTP may be used. If there is no traffic flowing, the peers have to
inject periodic keepalive packets to keep the NAT/FW mappings on both
sides of the communication active. Instead of application
keepalives, both peers can use PCP to control the mappings on the
NAT/FWs to reduce the keepalive frequency as explained in
Section 3.4.

```
      PCP            PCP                        PCP           PCP
     Client         Server        _____   Server        Client
   +-----------+   +------+     /           \  +------+    +-----------+
   |Application|___| NAT/ |____| Internet  |___| NAT/ |___|Application|
   |  Peer     |   |  FW  |    |           |   |  FW  |   |   Peer    |
   +-----------+   +------+     _____/   +------+    +-----------+
                  (multiple                    (multiple
                   layers)                      layers)

         ------------> PCP                 PCP <------------

         <----------------------------------------------------->
                         Application keepalive
```
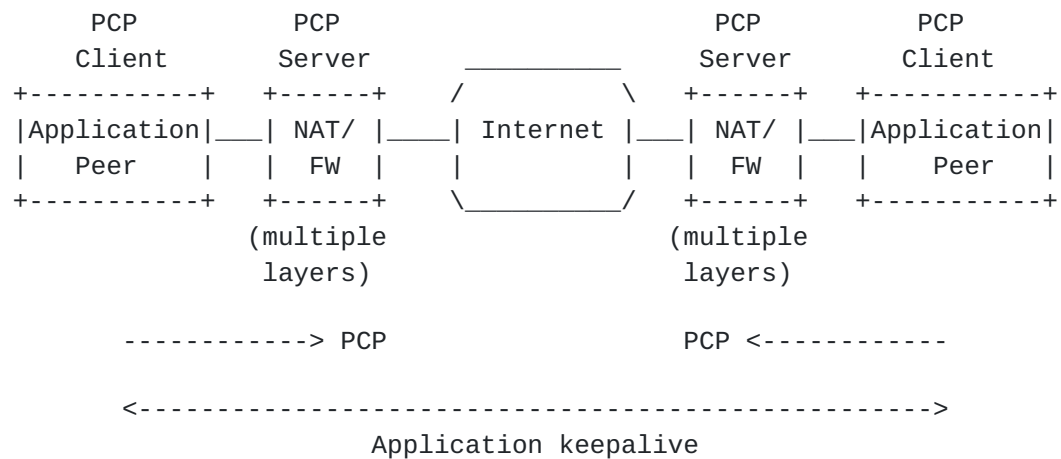
Figure 2: PCP with Peer-to-Peer applications

3.2. NAT Topologies and Detection

Before an application can reduce its keepalive rate, it has to make
sure it has all of the NATs and firewalls on its path under control.
This means it has to detect the presence of any PCP-unaware NATs and
firewalls on its path to the Internet.

3.2.1. PCP based detection

PCP itself is able to detect unexpected NATs between the PCP client
and PCP server as depicted in Figure 3. The PCP client includes its
own IP address and UDP port within the PCP request. The PCP server
compares them to the source IP address and UDP port it sees on the
packet. If they differ, there are one or more additional NATs
between the PCP client and PCP server, and the server will return an
error. Unless the application has some other means (like UPnP) to
control these PCP unaware NATs, it has to fall back to its default
keepalive mechanism.

```
        PCP              PCP         PCP
      Client           Unaware      Aware        _____
   +-----------+     +------+    +------+    /           \    +-----------+
   |Application|___| NAT  |___| NAT/ |____| Internet |___|Application|
   | Client    |    |      |    |  FW  |    |           |    |  Server   |
   +-----------+     +------+    +------+    _____/    +-----------+

        <-----------///---------->
              PCP based detection
```
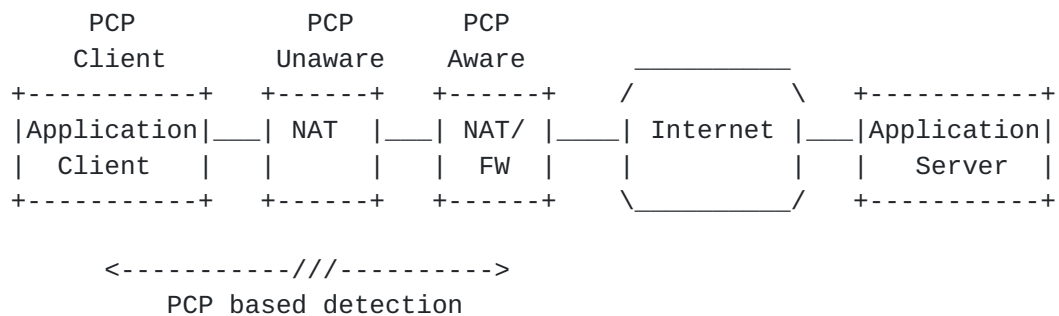
Figure 3: PCP unaware NAT between PCP client and PCP server

3.2.2. Application based detection

Figure 4 shows a topology where one or more PCP unaware NATs are
deployed on the exterior of the PCP capable NAT/FWs. To detect this,
the application client must have the capability to request from its
application server or peer what IP and transport address it sees. If
those differ from the IP and transport address given by the PCP aware
NAT/FW then the application client can determine that there is at
least one PCP unaware NAT on the path. In this case, the application
client has to fall back to its default keepalive mechanism.

```
        PCP              PCP         PCP
      Client           Aware      Unaware        _____
   +-----------+     +------+    +------+    /           \    +-----------+
   |Application|___| NAT/ |___| NAT  |____| Internet |___|Application|
   | Client    |    |  FW  |    |      |    |           |    |  Server   |
   +-----------+     +------+    +------+    _____/    +-----------+

        <------------>
              PCP

        <---------------------///--------------------------->
                     Application based detection
```
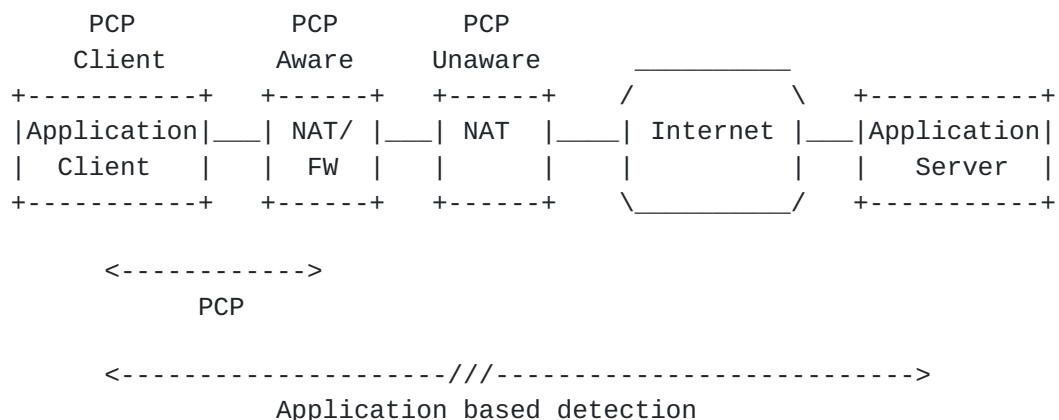
Figure 4: PCP unaware NAT external to the last PCP aware NAT

3.3. Detection of PCP unaware firewalls

PCP and application based detection mechanisms explained in
Section 3.2.1 and Section 3.2.2 are based on change in the address
and will not detect PCP unaware firewalls. In order to detect a PCP

unaware firewall, the application client sends a STUN [RFC5389]
Binding request to the STUN server. If STUN server supports the STUN
extensions defined in [RFC5780] then it returns its alternate IP
address and alternate port in OTHER-ADDRESS attribute in the STUN
Binding response. The client then uses PCP to send MAP request with
FILTER option to PCP server to permit STUN server to reach the client
using the STUN servers alternate IP address and alternate port. The
client then sends a Binding request to the primary address of the
STUN server with the CHANGE-REQUEST attribute set to change-port and
change-IP. This will cause the server to send its response from its
alternate IP address and alternate port. If the client receives a
response then the client is aware that on path firewall devices are
PCP aware. If the client does not receive a response then the client
is aware that there could be one or more on path PCP unaware firewall
devices. Application client will perform the tests separately for
each transport protocol. If no response is received, the client will
then repeat the test at most three times for connectionless transport
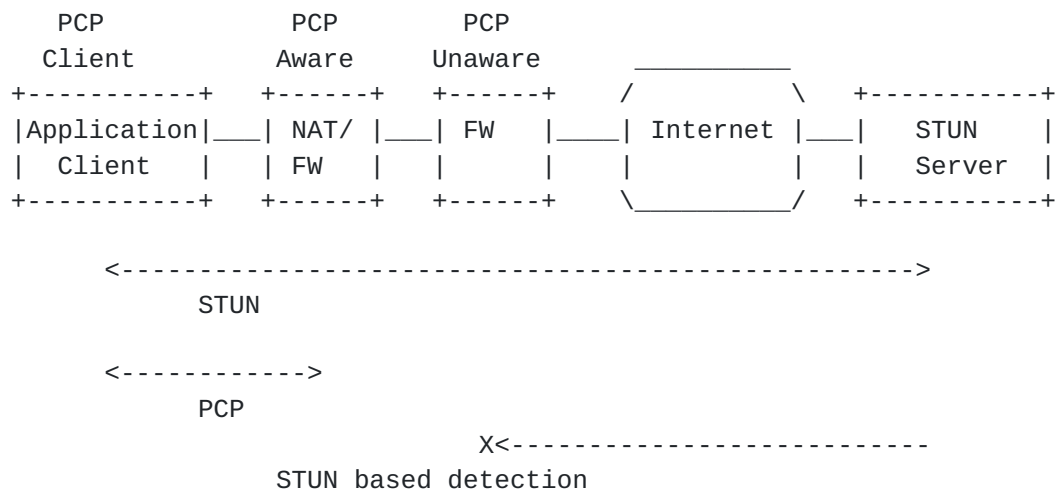protocols.

```
   PCP              PCP          PCP
  Client           Aware       Unaware         _____
+-----------+    +------+     +------+     /            \    +-----------+
|Application|___| NAT/ |___| FW    |____| Internet |___|   STUN    |
|  Client   |   | FW   |  |      |    |          |   |  Server   |
+-----------+    +------+     +------+     _____/    +-----------+

      <--------------------------------------------------->
            STUN

      <------------>
          PCP
                              X<---------------------------
                STUN based detection
```

                 Figure 5: PCP unaware firewall

   This procedure can be adopted by other protocols to detect PCP
   unaware firewalls.

3.4. Keepalive Optimization

   If the application determines that all NATs and firewalls on its path
   to the Internet support PCP, it can start using PCP instead of its
   default keepalives to maintain the NAT/FW state. It can use PCP PEER
   Request with the Requested Lifetime set to an appropriate value. The

application may still send some application-specific heartbeat
messages end-to-end.

Processing the lifetime value of the PEER Opcode is described in
Sections 10.3 and 15 of [RFC6887]. Sending a PEER request with a
very short Requested Lifetime can be used to query the lifetime of an
existing mapping. PCP recommends that lifetimes of mapping created
or lengthened with PEER be longer than the lifetimes of implicitly-
created NAT and firewall mappings. Thus PCP can be used to reduce
power consumption by making PCP PEER message interval longer than
what the application would normally use to the keep the middle box
state alive, and strictly shorter than the server state refresh
interval.

An example of savings with PCP is described in Appendix B.

**4. Keepalive Interval Determination Procedure when PCP unaware Firewall
     or NAT is detected**

If PCP unaware NAT/firewall is detected then a client can use the
following heuristics method to determine the keepalive interval:

1. The client sends a STUN Binding request to the STUN server. This
    connection is called the Primary Channel. STUN server will
    return its alternate IP address and alternate port in OTHER-
    ADDRESS in the Binding response [RFC5780].

2. The client then sends STUN Binding request to the STUN server
    using alternate IP address and alternate port. This connection
    is called the Secondary Channel.

3. The Client will initially set the default keepalive interval for
    NAT/FW mappings to 60 seconds (FWa).

4. After FWa seconds the Client will send a Binding request to the
    STUN server using the Primary Channel with the CHANGE-REQUEST
    attribute set to change-port and change-IP. This will cause the
    STUN server to send its response from the Secondary channel.

5. If the client receives response from the server then it will
    increase the keepalive interval value FWa = (old FWa) + (old
    FWa)/2. This indicates that NAT/FW mappings are alive.

6. Steps 4 and 5 will be repeated until there is no response from
    the STUN server. If there is no response from the STUN server
    then the client will use the old FWa value as Keepalive interval
    to refresh FW/NAT mappings.

The above procedure will be done separately for each transport
protocol. For connectionless transport protocols like UDP if timer
of 2 seconds elapses without response from the STUN server then the
client will repeat step 4 at most three times to handle packet loss.

This procedure can be adopted by other protocols to use Primary and
Secondary channels, so that the client can determine the keepalive
interval to refresh FW/NAT mapping. This procedure only serves as a
guideline and if applications already use some other heuristics
method to determine keepalive, they can continue with the existing
logic. For example Teredo determines Refresh interval using the
procedure in "Optional Refresh Interval Determination Procedure"
(Section 5.2.7 of [RFC4380]).

Note: The keepalive interval learnt using the above described
heuristics method can be inaccurate if a firewall is configured with
application specific inactivity timeout.

To improve reliability, applications SHOULD continue to use PCP to
lengthen the FW/NAT mappings even if the above described mechanism is
used to detect PCP unaware NAT/firewall. This ensures that PCP aware
FW/NAT do not close old mappings with no packet exchange when there
is a resource-crunch situation.

5. Application-Specific Operation

This section describes how PCP is used with specific application
protocols.

5.1. SIP

For connection-less transports the User Agent (UA) sends a STUN
Binding request over the SIP flow as described in section 4.4.2 of
[RFC5626]. The UA then learns the External IP Address and Port using
a PEER request/response. If the XOR-MAPPED-ADDRESS in the STUN
Binding response matches the external address and port provided by
PCP PEER response then the UA optimizes the keepalive traffic as
described in Section 3.4. There is no further need to send STUN
Binding requests over the SIP flow to keep the NAT Binding alive.

If the XOR-MAPPED-ADDRESS in the STUN Binding response does not match
the external address and port provided by the PCP PEER response then
PCP will not be used to keep the NAT bindings alive for the flow that
is being used for the SIP traffic. This means that multiple layers
of NAT are involved and intermediate NATs are not PCP aware. In this
case the UA will continue to use the technique in section 4.4.2 of
[RFC5626].

For connection-oriented transports, the UA sends a STUN Binding
request multiplexed with SIP over the TCP connection. STUN
multiplexed with other data over a TCP or TLS-over-TCP connection is
explained in section 7.2.2 of [RFC5389]. The UA then learns the
External IP address and port using a PEER request/response. If the
XOR-MAPPED-ADDRESS in the STUN Binding response matches the external
address and port provided by PCP PEER response then the UA optimizes
the keepalive traffic as described in Section 3.4.

If the XOR-MAPPED-ADDRESS in the STUN Binding response does not match
the external address and port provided by PCP PEER response then PCP
will not be used to keep the NAT bindings alive. In this case the UA
performs a keepalive check by sending a double-CRLF (the "ping") then
waits to receive a single CRLF (the "pong") using the technique in
section 4.4.1 of [RFC5626].

5.2. HTTP

Web Applications that require persistent connections use techniques
such as HTTP long polling and Websockets for session keep alive as
explained in section 3.1 of [I-D.isomaki-rtcweb-mobile]. In such
scenarios, after the client establishes a connection with the HTTP
server, it can execute server side scripts such as PHP residing on
the server to provide the transport address and port of the HTTP
client seen at the HTTP server. In addition, the HTTP client also
learns the external IP Address and port using the PCP PEER request/
response.

If the IP address and port learned from the server matches the
external address and port provided by PCP PEER response then the HTTP
client optimizes keepalive traffic as described in Section 3.4.

If the IP address and port do not match then PCP will not be used to
keep the NAT bindings alive for the flow that is being used for the
HTTP traffic. This means that there are NATs or HTTP proxies between
the PCP server and the HTTP server. The HTTP client will have to
resort to use existing techniques for keep alive. Please see
Appendix A for an example server side PHP script to obtain the client
source IP address.

HTTP protocol allows intermediaries like transparent proxies to be
involved and there is no way for the client to know that request/
response is relayed through a proxy.

5.3. **Media and data channels with ICE**

   The ICE agent learns the External IP Addresses and Ports using the
   MAP opcode. If server reflexive candidates learnt using STUN
   [RFC5389] and external IP addresses learnt using PCP are different
   then candidates learnt through both STUN and PCP are encoded in the
   ICE offer and answer . When using the Recommended Formula explained
   in section 4.1.2.1 of [RFC5245] to compute priority for the candidate
   learnt through PCP, the ICE agent should use a preference value
   greater than the server reflexive candidate and hence tested before
   the server reflexive candidate. The recommended type preference
   value is 105 for candidates discovered using PCP and is explained in
   section 4.2 of [RFC6544].

   The ICE agent, in addition to the ICE connectivity checks, performs
   the following:

   The ICE agent checks if the XOR-MAPPED-ADDRESS from the STUN Binding
   response received as part of ICE connectivity check matches the
   External IP address and Port provided by PCP MAP response.

   1. If the match is successful then PCP will be used to keep the NAT
       bindings alive. The ICE agent optimizes keepalive traffic by
       refreshing the mapping via a new PCP MAP request containing
       information from the earlier PCP response.

   2. If the match is not successful then PCP will not be used for keep
       NAT binding alive. The ICE agent will use the technique in
       section 4.4 of [RFC6263] to keep NAT bindings alive. This means
       that multiple layers of NAT are involved and intermediate NATs
       are not PCP aware.

   Some network operators deploying a PCP Server may allow PEER but not
   MAP. In such cases the ICE agent learns the external IP address and
   port using a STUN Binding request/response during ICE connectivity
   checks. The ICE agent also learns the external IP Address and port
   using a PCP PEER request/response. If the IP address and port
   learned from the STUN Binding response matches the external address
   and port provided by the PCP PEER response then the ICE agent
   optimizes keepalive traffic as described in Section 3.4.

5.4. **Detecting Flow Failure**

   Using the Rapid Recovery technique in section 14 of [RFC6887] upon
   receiving a PCP ANNOUNCE from a PCP server, a PCP client becomes
   aware that the PCP server has rebooted or lost its mapping state.
   The PCP client issues new PCP requests to recreate any lost mapping
   state and thus reconstructs lost mappings fast enough that existing

   media, HTTP and SIP flows do not break. If the NAT state cannot be
   recovered the endpoint will find the new external address and port as
   part of the Rapid Recovery technique in PCP itself and reestablish a
   connection with the peer.

5.5. Firewalls

   PCP allows applications to communicate with firewall devices with PCP
   functionality to create mappings for incoming connections. In such
   cases PCP can be used by the endpoint to create an explicit mapping
   on firewall in order to permit inbound traffic. The endpoint can
   further use PCP to send keepalives to keep the firewall mappings
   alive.

5.5.1. IPv6 Network with Firewalls

   For scenarios where the client uses the ICE Lite implementation
   explained in section 2.7 of [RFC5245], the ICE Lite endpoint will not
   generate its own ICE connectivity checks, by definition. As part of
   the call setup, the ICE Lite endpoint would gather its host
   candidates and relayed candidate from a TURN server and send the
   candidates in the offer to the peer endpoint. On receiving the
   answer from the peer endpoint, the ICE Lite endpoint sends a PCP MAP
   request with FILTER opcode to create a dynamic mapping in firewall to
   permit ICE connectivity checks and subsequent media traffic from the
   remote peer. This way, the ICE Lite endpoint and its network are
   protected from unsolicited incoming UDP traffic, and can still
   operate using ICE Lite (rather than full ICE).

5.5.2. Mobile Network with Firewalls

   Mobile Networks are also making use of a firewall to protect their
   customers from various attacks like downloading malicious content.
   The firewall is usually configured to block all unknown inbound
   connections as explained in section 2.1 of
   [I-D.chen-pcp-mobile-deployment]. As described in Section 3.4, in
   such cases PCP can be used by Mobile devices to create an explicit
   mapping on the firewall to permit inbound traffic and optimize the
   keepalive traffic. This would result in saving of radio and power
   consumption of the Mobile device while protecting it from attacks.

6. IANA Considerations

   None

7. Security Considerations

The security considerations in [RFC5245] and [RFC6887] apply to this use.

8. Acknowledgements

Authors would like to thank Dave Thaler, Basavaraj Patil, Anca Zamfir, Reinaldo Penno, Suresh Kumar and Dilipan Janarthanan for valuable inputs to the document.

9. References

9.1. Normative References

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

[RFC5245]   Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

[RFC5389]   Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.

[RFC5626]   Jennings, C., Mahy, R., and F. Audet, "Managing Client-Initiated Connections in the Session Initiation Protocol (SIP)", RFC 5626, October 2009.

[RFC5780]   MacDonald, D. and B. Lowekamp, "NAT Behavior Discovery Using Session Traversal Utilities for NAT (STUN)", RFC 5780, May 2010.

[RFC6263]   Marjou, X. and A. Sollaud, "Application Mechanism for Keeping Alive the NAT Mappings Associated with RTP / RTP Control Protocol (RTCP) Flows", RFC 6263, June 2011.

[RFC6544]   Rosenberg, J., Keranen, A., Lowekamp, B., and A. Roach, "TCP Candidates with Interactive Connectivity Establishment (ICE)", RFC 6544, March 2012.

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

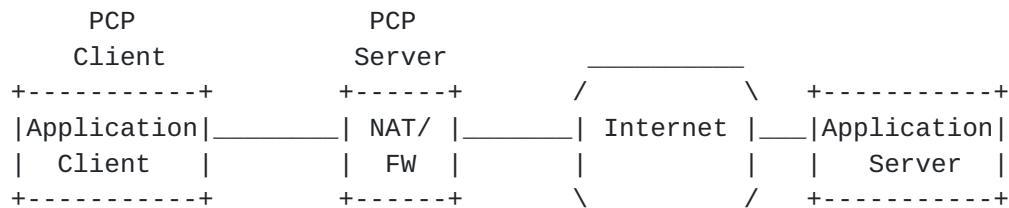9.2. Informative References

   [I-D.binet-v6ops-cellular-host-reqs-rfc3316update]
              Binet, D., Boucadair, M., Ales, V., Byrne, C., and G.
              Chen, "Internet Protocol Version 6 (IPv6) for Cellular
              Hosts", draft-binet-v6ops-cellular-host-reqs-
              rfc3316update-03 (work in progress), October 2012.

   [I-D.chen-pcp-mobile-deployment]
              Chen, G., Cao, Z., Boucadair, M., Ales, V., and L.
              Thiebaut, "Analysis of Port Control Protocol in Mobile
              Network", draft-chen-pcp-mobile-deployment-04 (work in
              progress), July 2013.

   [I-D.ietf-rtcweb-overview]
              Alvestrand, H., "Overview: Real Time Protocols for
              Browser-based Applications", draft-ietf-rtcweb-overview-11
              (work in progress), August 2014.

   [I-D.isomaki-rtcweb-mobile]
              Isomaki, M., "RTCweb Considerations for Mobile Devices",
              draft-isomaki-rtcweb-mobile-00 (work in progress), July
              2012.

   [RFC2177] Leiba, B., "IMAP4 IDLE command", RFC 2177, June 1997.

   [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston,
              A., Peterson, J., Sparks, R., Handley, M., and E.
              Schooler, "SIP: Session Initiation Protocol", RFC 3261,
              June 2002.

   [RFC3921] Saint-Andre, P., Ed., "Extensible Messaging and Presence
              Protocol (XMPP): Instant Messaging and Presence", RFC
              3921, October 2004.

   [RFC4380] Huitema, C., "Teredo: Tunneling IPv6 over UDP through
              Network Address Translations (NATs)", RFC 4380, February
              2006.

   [RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC
              6455, December 2011.

Appendix A. Example PHP script

```
<html>
Connected to <?PHP echo gethostname(); ?> on port <?PHP echo
getenv(SERVER_PORT)?> on <?PHP echo date("d-M-Y H:i:s");?>
Pacific Time
<p>
Your IP address is: <?PHP echo getenv(REMOTE_ADDR); ?>,
port <?PHP echo getenv(REMOTE_PORT); ?>
</p>;
</html>
```

Appendix B.   **Savings with PCP**

The following example illustrates the savings in keepalive messages
with PCP.

```
        PCP                 PCP
      Client              Server            _____
   +-----------+         +------+          /          \    +-----------+
   |Application|_____| NAT/ |_____| Internet |___|Application|
   |  Client   |         | FW   |         |          |   |  Server   |
   +-----------+         +------+          _____/    +-----------+
```

With Application Heartbeat (without PCP):

```
       <--------------------///-------------------------->
         Application heartbeat (Max Interval = 120 seconds)
       <--------------------///-------------------------->
         Application heartbeat (Max Interval = 120 seconds)
       <--------------------///-------------------------->
         Application heartbeat (Max Interval = 120 seconds)
       <--------------------///-------------------------->
         Application heartbeat (Max Interval = 120 seconds)
               ....
               ....
               ....
               ....
```

With PCP:
```
       <----------------->
          PCP PEER request
       (Max Lifetime = 3600 seconds)
               ....
               ....
       <----------------->
          PCP PEER request
       (Max Lifetime = 3600 seconds)
```
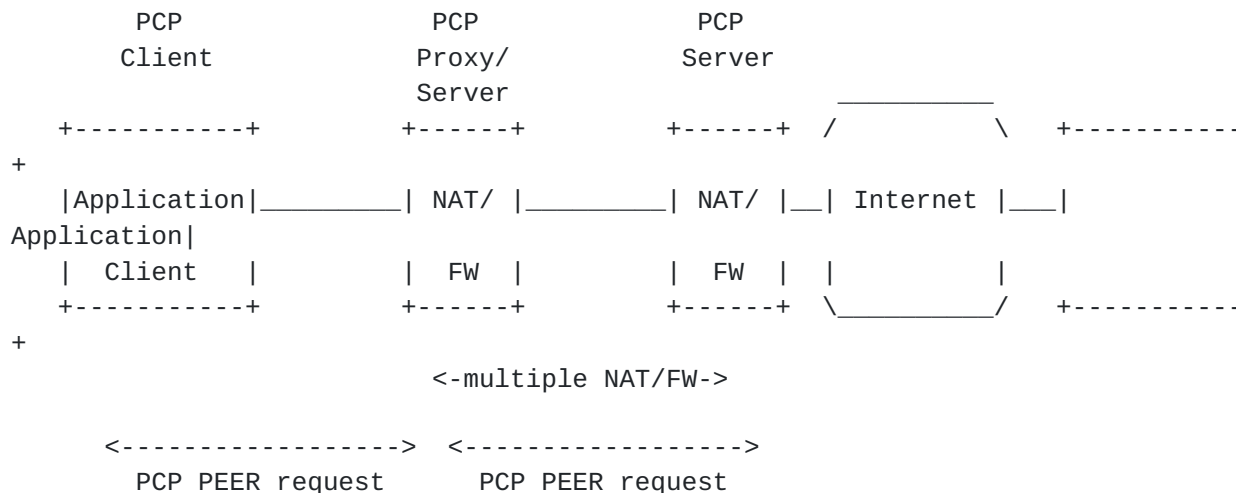
                       Figure 6: Savings with PCP

   In the example above, let's suppose normally an application would
   need to send a heartbeat every 120s to keep mappings active on the
   NAT/firewall device. In 24 hours, in the absence of PCP, the number
   of packets sent by the application to keep those mappings active
   would be (86400/120) = 720 packets.

   If the same application uses PCP PEER to create a mapping, with a
   lifetime of 3600 seconds, on a PCP controlled NAT/firewall device,
   the number of packets sent by the application to keep those mappings
   active would be (86400/3600) = 24 packets.

   With the above assumptions, using PCP saves 96.67% of keepalive
   traffic. As the number of applications running on a host increase,

savings in cost of sending application heartbeats are significant
with the use of PCP.

```
        PCP                 PCP              PCP
      Client              Proxy/           Server
                          Server                      _____
   +-----------+         +------+         +------+  /          \   +-----------
+
   |Application|_____| NAT/ |_____| NAT/ |__| Internet |___|
Application|
   | Client    |         | FW   |         | FW   | |           |   |
   +-----------+         +------+         +------+  _____/    +-----------
+

                      <-multiple NAT/FW->


     <------------------>  <------------------>
       PCP PEER request     PCP PEER request
```

If there are multiple PCP aware NAT/firewall devices on a clients
path to the internet eg., PCP server at home CPE and also at CGN,
savings with PCP are the same. The PCP server at the home CPE is a
PCP proxy that can create associated mappings on the PCP server at
CGN. The client will only have to communicate with the PCP proxy and
receives a single mapping lifetime that needs to be refreshed.

Authors' Addresses

   Tirumaleswar Reddy
   Cisco Systems, Inc.
   Cessna Business Park, Varthur Hobli
   Sarjapur Marathalli Outer Ring Road
   Bangalore, Karnataka 560103
   India

   Email: tireddy@cisco.com

   Markus Isomaki
   Nokia
   Keilalahdentie 2-4
   FI-02150 Espoo
   Finland

   Email: markus.isomaki@nokia.com

Dan Wing
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, California 95134
USA

Email: dwing@cisco.com

Prashanth Patil
Cisco Systems, Inc
Bangalore
India

Email: praspati@cisco.com