PCP Working Group
Internet-Draft
Intended status: Informational
Expires: November 17, 2013

M. Boucadair
France Telecom
R. Penno
Cisco

May 16, 2013

Analysis of Port Control Protocol (PCP) Failure Scenarios
draft-boucadair-pcp-failure-06

Abstract

This document identifies and analyzes several PCP failure scenarios. Identifying these failure scenarios is useful to assess the efficiency of the protocol and also to decide whether new PCP extensions are needed.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document discusses several failure scenarios that may occur when deploying PCP [RFC6887].

2. PCP Client Failure Scenarios

2.1. Change of the IP Address of The PCP Server

When a new IP address is used to reach its PCP Server, the PCP Client must re-create all of its explicit dynamic mappings using the newly discovered IP address.

The PCP Client must undertake the same process as per refreshing an existing explicit dynamic mapping (see [RFC6887]); the only difference is the PCP requests are sent to a distinct IP address. No specific behavior is required from the PCP Server for handling these requests.

Proposed Action: No particular extension is required to be added to the base specification to mitigate this failure scenario.

2.2. Application Crash

When a fatal error is encountered by an application relying on PCP to open explicit dynamic mappings on an upstream device, and upon the restart of that application, the PCP Client should issue appropriate requests to refresh the explicit dynamic mappings of that application (e.g., clear old mappings and install new ones using the new port number used by the application).

If the same port number is used but a distinct Mapping Nonce is generated, the request will be rejected with a NOT_AUTHORIZED error with the Lifetime of the error indicating duration of that existing mapping (see Section 2.7 of [I-D.boucadair-pcp-flow-examples]).

Proposed Action: A solution to recover the Mapping Nonce used when instantiating the mapping may be envisaged; this solution may not be viable if PCP authentication is not in use. Mapping Nonce recovery in the simple PCP threat model (especially when Mapping Check validation is enabled) induces the same security threated as those discussed in [RFC6887].

If a distinct port number is used by the application to bound its service (i.e., a new internal port number is to be signaled in PCP), the PCP Server may honor the refresh requests if the per-subscriber quota is not exceeded. A distinct external port number would be assigned by the PCP Server due to the presence of "stale" explicit dynamic mapping(s) associated with the "old" port number.

Proposed Action: To avoid this inconvenience induced by stale explicit dynamic mappings, the PCP Client may clear the "old" mappings before issuing the refresh requests; but this would require the PCP Client to store the information about the "old" port number. This can be easy to solve if the PCP Client is embedded in the application. In some scenarios, this is not so easy because the PCP Client may handle PCP requests on behalf of several applications and no means to identify the requesting application may be supported. Means to identify the application may be envisaged.

> [RFC6887] does not allow a PCP Client to issue a request to delete all the explicit dynamic mappings associated with an internal IP address. If a PCP Client is allowed to clear all mappings bound to the same IP address, this would have negative impact on other applications and PCP Client(s) which may use the same internal IP address to instruct their explicit dynamic mappings in the PCP Server.

2.3. PCP Client Crash

The PCP Client may encounter a fatal error leading to its restart. In such case, the internal IP address and port numbers used by requesting applications are not impacted. Therefore, the explicit dynamic mappings as maintained by the PCP Server are accurate and there is no need to refresh them.

On the PCP Client side, a new UDP port should be assigned to issue PCP requests. As a consequence, if outstanding requests have been sent to the PCP Server, the responses are likely to be lost.

If the PCP Client stores its explicit dynamic mappings in a persistent memory, there is no need to retrieve the list of active mappings from the PCP Server.

> Proposed Action: If several PCP Clients are co-located on the same host, related PCP mapping tables should be uniquely distinguished (e.g., a PCP Client does not delete explicit dynamic mappings instructed by another PCP Client).

If the PCP Client does not store the explicit dynamic mappings and new Mapping Nonces are assigned, the PCP Server will reject to refresh these mappings.

> Proposed Action: This issue can be solved if the PCP Client uses GET OpCode (Appendix B) to recover the mapping nonces used when instantiating the mappings if PCP authentication is used or Mapping Nonce validation check is disabled.

2.4. Change of the Internal IP Address

When a new IP address is assigned to a host embedding a PCP Client, the PCP Client must install on the PCP Server all the explicit dynamic mappings it manages, using the new assigned IP address as the internal IP address. The hinted external port number won't be assigned by the PCP Server since a "stale" mapping is already instantiated by the PCP Server (but it is associated with a distinct internal IP address).

For a host configured with several addresses, the PCP Client must maintain a record about the target IP address it used when issuing its PCP requests. If no record is maintained and upon a change of the IP address or de-activation of an interface, the PCP-instructed explicit dynamic mappings are broken and inbound communications will fail to be delivered.

Depending on the configured policies, the PCP Server may honor all or part of the requests received from the PCP Client. Upon receipt of the response from the PCP Server, the PCP Client must update its local PCP state with the new assigned port numbers and external IP address.

> Proposed Action: Because of the possible negative impact if the quota is exceed due to the presence of stale mappings (see the example in Section 2.14 of [I-D.boucadair-pcp-flow-examples]), a procedure to clear stale mappings may have some benefits.

A PCP Client may be used to manage explicit dynamic mappings on behalf of a third party (i.e., the PCP Client and the third party are not co-located on the same host). If a new internal IP address is assigned to that third party (e.g., webcam), the PCP Client should be instructed to delete the old mapping(s) and create new one(s) using the new assigned internal IP address. When the PCP Client is co-located with the DHCP server (e.g., PCP Proxy [I-D.ietf-pcp-proxy], IWF in the CP router [I-D.ietf-pcp-upnp-igd-interworking]), the state can be updated using the state of the local DHCP server. Otherwise, it is safe to recommend the use of static internal IP addresses if PCP is used to configure third-party explicit dynamic mappings.

> Proposed Action: No particular extension is required to be added to the base specification to mitigate this failure scenario.

2.5. Change of the CPE WAN IP Address

The change of the IP address of the WAN interface of the CPE would have an impact on the accuracy of the explicit dynamic mappings instantiated in the PCP Server:

- For the DS-Lite case [RFC6333]: if a new IPv6 address is used by the B4 element when encapsulating IPv4 packets in IPv6 ones, the explicit dynamic mappings should be refreshed: If the PCP Client is embedded in the B4, the refresh operation is triggered by the change of the B4 IPv6 address. This would be more complicated when the PCP Client is located in a device behind the B4. If a PCP Proxy is embedded in the CPE, the proxy can use ANNOUNCE OpCode towards internal IPv4 hosts behind the DS-Lite CPE.

- For the NAT64 case [RFC6146], any change of the assigned IPv6 prefix delegated to the CPE will be detected by the PCP Client (because this leads to the allocation of a new IPv6 address). The PCP Client has to undertake the operation described in Section 2.4.

- For the NAT444 case, similar problems are encountered because the PCP Client has no reasonable way to detect the CPE's WAN address changed.

  Proposed Action: Means to help detecting the CPE's WAN address change would help in mitigating this failure scenario.

2.6. UPnP IGD/PCP IWF

In the event an UPnP IGD/PCP IWF [I-D.ietf-pcp-upnp-igd-interworking] fails to renew a mapping, there is no mechanism to inform the UPnP Control Point about this failure.

> Proposed Action: This issue can not be solved.

On the reboot of the IWF, if no mapping table is maintained in a permanent storage, "stale" mappings will be maintained by the PCP Server and per-user quota will be consumed. This is even exacerbated if new mapping nonces are assigned by the IWF.

> Proposed Action: This issue can be soften by synchronizing the mapping table owing to the invocation of the GET OpCode defined in Appendix B. This procedure is supported only if Mapping Nonce validation checks are disabled.

3. Restart or Failure of the PCP Server

This section covers failure scenarios encountered by the PCP Server.

3.1. Basic Rule

In any situation the PCP Server loses all or part of its PCP state, the Epoch value must be reset when replying to received requests. Doing so would allow PCP Client to audit its explicit dynamic mapping table.

If the state is not lost, the PCP Server must not reset the Epoch value returned to requesting PCP Clients.

> Proposed Action: No action is required to update the base PCP specification for this failure scenario.

3.2. Clear PCP Mappings

When a command line or a configuration change is enforced to clear all or a subset of PCP explicit dynamic mappings maintained by the PCP Server, the PCP Server must reset its Epoch to zero value.

In order to avoid all PCP Clients to update their explicit dynamic mappings, the PCP Server should reset the Epoch to zero value only for impacted users.

Proposed Action: No action is required to update the base PCP specification for this failure scenario.

3.3. State Redundancy is Enabled

When state redundancy is enabled, the state is not lost during failure events. Failures are therefore transparent to requesting PCP Clients. When a backup device takes over, Epoch must not be reset to zero.

Proposed Action: No action is required to update the base PCP specification for this failure scenario.

3.4. Cold-Standby without State Redundancy

In this section we assume that a redundancy mechanisms is configured between a primary PCP-controlled device and a backup one but without activating any state synchronization for the PCP-instructed explicit dynamic mappings between the backup and the primary devices.

If the primary PCP-controlled device fails and the backup one takes over, the PCP Server must reset the Epoch to zero value. Doing so would allow PCP Clients to detect the loss of states in the PCP Server and proceed to state synchronization.

Proposed Action: No action is required to update the base PCP specification for this failure scenario.

3.5. Anycast Redundancy Mode

When an anycast-based mode is deployed (i.e., the same IP address is used to reach several PCP Servers) for redundancy reasons, the change of the PCP Server which handles the requests of a given PCP Client won't be detected by that PCP Client.

Tweaking the Epoch (Section 8.5 of [RFC6887]) may help to detect the loss of state and therefore to re-create missing explicit dynamic mappings.

Proposed Action: No action is required to update the base PCP specification for this failure scenario.

4. Security Considerations

PCP-related security consideratiosn are discussed in [RFC6887].

5. IANA Considerations

No action is required from IANA.

6. Acknowledgements

Francis Dupont contributed text to this document. Many thanks to him.

7. References

7.1. Normative References

[I-D.ietf-pcp-proxy]
Boucadair, M., Penno, R., and D. Wing, "Port Control Protocol (PCP) Proxy Function", draft-ietf-pcp-proxy-02 (work in progress), February 2013.

[I-D.ietf-pcp-upnp-igd-interworking]
Boucadair, M., Penno, R., and D. Wing, "Universal Plug and Play (UPnP) Internet Gateway Device (IGD)-Port Control Protocol (PCP) Interworking Function", draft-ietf-pcp-upnp-igd-interworking-10 (work in progress), April 2013.

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

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

7.2. Informative References

[I-D.boucadair-pcp-flow-examples]
Boucadair, M., "PCP Flow Examples", draft-boucadair-pcp-flow-examples-00 (work in progress), February 2013.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, April 2011.

[RFC6333] Durand, A., Droms, R., Woodyatt, J., and Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion", RFC 6333, August 2011.

Appendix A. PCP State Synchronization: Overview

The following sketches the state synchronization logic:

- One element (i.e., PCP Client/host/application, PCP Server, PCP Proxy, PCP IWF) of the chain is REQUIRED to use stable storage

- If the PCP Client (resp., the PCP Server) crashes and restarts it just have to synchronize with the PCP Server (resp., the PCP Client);

- If both crash then one has to use stable storage and we fall back in the previous case as soon as we know which one (the Epoch value gives this information);

- PCP Server -> PCP Client not-disruptive synchronization requires a GET/NEXT mechanism to retrieve the state from the PCP Server; without this mechanism the only way to put the PCP Server in a known state is for the PCP Client to send a delete all request, a clearly disruptive operation.

- PCP Client -> PCP Server synchronization is done by a re-create or refresh of the state. The PCP Client MAY retrieve the PCP Server state in order to prevent stale explicit dynamic mappings.

Appendix B. GET/NEXT Operation

This section defines a new PCP OpCode called GET and its associated Option NEXT.

These PCP Opcode and Option are used by the PCP Client to retrieve an explicit mapping or to walk through the explicit dynamic mapping table maintained by the PCP Server for this subscriber and retrieves a list of explicit dynamic mapping entries it instantiated.

GET can also be used by a NoC to retrieve the list of mappings for a given subscriber.

B.1. OpCode Format

The GET OpCode payload contains a Filter used for explicit dynamic mapping matching: only the explicit dynamic mappings of the subscriber which match the Filter in a request are considered so could be returned in response.

Implementation Note: Some existing implementations use 98 (0x62) codepoint for GET OpCode, 131 for AMBIGUOUS error code, and 131 (0x83) for NEXT Option.

The layout of GET OpCode is shown in Figure 1.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Protocol    |                 Reserved                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:   Filter internal IP address (always 128 bits)                :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:   Filter external IP address (always 128 bits)                :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Filter internal port        |   Filter external port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: GET: OpCode format

For all fields, the value 0 in a request means wildcard filter/any value matches. Of course this has to be sound: no defined port with protocol set to any.

These fields are described below:

Protocol: Same than for MAP [RFC6887].

Reserved: MUST be sent as 0 and MUST be ignored when received.

Filter internal IP address: Conveys the internal IP address (including an unspecified IPv4IPv6 address). The encoding of this field follows Section 5 of [RFC6887].

Filter external IP address: Conveys the external IP address (including an unspecified IPv4IPv6 address). The encoding of this field follows Section 5 of [RFC6887].

Filter internal port: The internal port (including 0).

Filter external port: The external port (including 0).

Responses include a bit-to-bit copy of the OpCode found in the request.

B.2. OpCode-Specific Result Code

This OpCode defines two new specific Result Code

TBD: NONEXIST_MAP, e.g., no explicit dynamic mapping matching the Filter was found.

TBD: AMBIGUOUS. This code is returned when the PCP Server is not able to decide which mapping to return. Existing implementations use 131 as codepoint.

B.3. Ordering and Equality

The PCP server is required to implement an order between matching explicit dynamic mappings. The only property of this order is to be stable: it doesn't change (*) between two GET requests with the same Filter.

(*) "change" means two mappings are not gratuitously swapped: expiration, renewal or creation are authorized to change the order but they are at least expected by the PCP client.

> [Ed. Note: We have two proposals for the order: lexicographical order and lifetime order. Both work, this should be left to the implementor.]

Equality is defined by:

- same protocol and;
- same internal address and;
- same external address and;
- same internal port and;
- same external port.

B.4. NEXT Option

Formal definition:

Name: NEXT

Number: at most one in requests, any in responses.

Purpose: carries a Locator in requests, matching explicit dynamic mappings greater than the Locator in responses.

Is valid for OpCodes: GET OpCode.

Length: variable, the minimum is 11.

May appear in: both requests and responses.

Maximum occurrences: one for requests, bounded by maximum message size for PCP responses [RFC6887].

The layout of the NEXT Option is shown in Figure 2.

Version=1

```
     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
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |   Protocol    |          Reserved              |  MORE/END     |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    :   Mapping internal IP address (always 128 bits)               :
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    :   Mapping external IP address (always 128 bits)               :
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |   Mapping remaining lifetime                                  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |   Mapping internal port       |   Mapping external port       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    |                       Mapping Options                         :
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Version=2

```
      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
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     :                 Mapping Nonce (96 bits)                       :
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |   Protocol    |          Reserved              |  MORE/END     |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                                                               |
     :   Mapping internal IP address (always 128 bits)               :
     |                                                               |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                                                               |
     :   Mapping external IP address (always 128 bits)               :
```

```
     |                                                               |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |   Mapping remaining lifetime                                  |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |   Mapping internal port       |   Mapping external port       |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |                                                               |
     |                       Mapping Options                         :
     |                                                               |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: NEXT: Option format

In requests the NEXT Option carries a Locator: a position in the list of explicit dynamic mappings which match the Filter. The following two useful forms of Locators are considered:

- the "Undefined" form where the Protocol, Addresses, Ports fields are set to zero.

- the "Defined" form where none of the Protocol, Addresses and Ports is set to zero.

The new fields in a Locator (a.k.a., the NEXT Option in a GET request) are described below:

MORE/END: The value 0 denotes "MORE" and means the response MAY include multiple NEXT Options; a value other than 0 (1 is RECOMMENDED) denotes "END" and means the response SHALL include at most one NEXT Option.

Mapping remaining lifetime: MUST be sent as 0 and MUST be ignored when received.

Mapping Options: The Option Codes of the PCP Client wants to get in the response (e.g., THIRD_PARTY). The format is the same than for the UNPROCESSED Option (see rev 17 of[RFC6887]).

In responses the NEXT Options carry the returned explicit dynamic mappings, one per NEXT Option. The fields are described below:

Protocol: The protocol of the returned mapping.

MORE/END: The value 0 when there are explicit dynamic mapping matching the Filter and greater than this returned mapping; a value other than 0 (1 is RECOMMENDED) when the return mapping is the greatest explicit dynamic mapping matching the Filter.

Mapping internal IP address: the internal address of the returned mapping. The encoding of this field follows Section 5 of [RFC6887].

Mapping external IP address: the external address of the returned mapping. The encoding of this field follows Section 5 of [RFC6887].

Mapping remaining lifetime: The remaining lifetime in seconds of the returned mapping.

Mapping internal port: the internal port of the returned mapping.

Mapping external port: the external port of the returned mapping.

Mapping Options: An embedded list of option values. Each corresponding Option Code MUST be present in the request NEXT Option, each option MUST be related to the returned mapping or not related to any mapping.

B.5. GET/NEXT PCP Client Theory of Operation

GET requests without a NEXT Option have low usage but with a full wildcard Filter they ask the PCP Server to know if it has at least one explicit dynamic mapping for this subscriber.

GET requests with an END NEXT Option are "pure" GET: they asks for the status and/or the remaining lifetime or options of a specific explicit dynamic mapping. It is recommended to use an Undefined Locator and to use the Filter to identify the mapping.

GET requests with a MORE NEXT Option are for the whole explicit dynamic mapping table retrieval from the PCP Server. The initial request contains an Undefined Locator, other requests a Defined Locator filled by a copy of the last returned mapping with the Lifetime and Option fields reseted to the original values. An END NEXT Option marks the end of the retrieval.

B.6. GET/NEXT PCP Server Theory of Operation

The PCP Server behavior is described below:

- on the reception of a valid GET request the ordered list of explicit dynamic mapping of the subscriber matching the given Filter is (conceptually) built.

- if the list is empty a NONEXIST_MAP error response is returned. It includes no NEXT Option.

- o the list is scanned to find the Locator using the Equality defined in Appendix B.3. If it is found the mappings less than the Locator are removed from the list, so the result is a list which begins by the mapping equals to the Locator followed by greater mappings.

- o if the NEXT Option in the request is an END one, the first mapping of the list is returned in an only NEXT option, marked END if the list contains only this mapping, marked MORE otherwise.

- o if the NEXT option in the request is a MORE one, as many as can fit mappings are returned in order in the response, marked as MORE but if the whole list can be returned the last is marked END.

"Returned" means to include required options when they are defined for a mapping: if the mapping M has 3 REMOTE_PEER_FILTERs and the REMOTE_PEER_FILTER code was in the request NEXT, the NEXT carrying M will get the 3 REMOTE_PEER_FILTER options embedded.

B.7. Flow Examples

As an illustration example, let's consider the following explicit dynamic mapping table is maintained by the PCP Server:

Pro	Internal IP Address	Internal Port	External IP Address	External Port	Remaining Lifetime
UDP	198.51.100.1	25655	192.0.2.1	15659	1659
TCP	198.51.100.2	12354	192.0.2.1	32654	3600
TCP	198.51.100.2	8596	192.0.2.1	25659	6000
UDP	198.51.100.1	19856	192.0.2.1	42654	7200
TCP	198.51.100.1	15775	192.0.2.1	32652	9000

Table 1: Excerpt of a mapping table

As shown in Table 1, the PCP Server sorts the explicit dynamic mapping table using the internal IP address and the remaining lifetime.

Figure 3 illustrates the exchange that occurs when a PCP Client tries to retrieve the information related to a non-existing explicit dynamic mapping.

```
                 +------+                            +------+
                 | PCP  |                            | PCP  |
```

```
                    |Client|                           |Server|
                    +------+                           +------+
                       |       (1) PCP GET Request         |
                       |           protocol= TCP           |
                       | internal-ip-address= 198.51.100.1 |
                       |        internal-port= 59864       |
                       |         Undefined Locator         |
                       |---------------------------------->|
                       |                                   |
                       |        (2) PCP GET Response       |
                       |       error= NONEXIST_MAP         |
                       |<----------------------------------|
                       |                                   |
```

Figure 3: Example of a failed GET operation

Figure 4 shows an example of a PCP Client which retrieves successfully an existing mapping from the PCP Server.

```
                    +------+                           +------+
                    | PCP  |                           | PCP  |
                    |Client|                           |Server|
                    +------+                           +------+
                       |       (1) PCP GET Request         |
                       |           protocol= TCP           |
                       | internal-ip-address= 198.51.100.1 |
                       |        internal-port= 25655       |
                       |         Undefined Locator         |
                       |---------------------------------->|
                       |                                   |
                       |      (2) PCP GET Response         |
                       |             END                   |
                       |           protocol= TCP           |
                       | internal-ip-address= 198.51.100.1 |
                       |        internal-port= 25655       |
                       |   external-ip-address= 192.0.2.1  |
                       |        external-port= 15659       |
                       |       remaining-lifetime= 1659    |
                       |<----------------------------------|
                       |                                   |
                       |      (3) PCP MAP4 Request         |
                       |           protocol= TCP           |
                       | internal-ip-address= 198.51.100.1 |
                       |        internal-port= 25655       |
                       |   external-ip-address= 192.0.2.1  |
                       |        external-port= 15659       |
                       |       requested-lifetime= 0       |
```

```
                       |---------------------------------->|
                       |                                   |
```

Figure 4: Example of a successful GET operation

In reference to Figure 5, the PCP Server returns the explicit dynamic mappings having the internal address equal to 192.0.2.1 ordered by increasing remaining lifetime.

```
                    +------+                           +------+
                    | PCP  |                           | PCP  |
                    |Client|                           |Server|
                    +------+                           +------+
                       |       (1) PCP GET Request         |
                       | internal-ip-address= 198.51.100.2 |
                       |         Undefined Locator         |
                       |---------------------------------->|
                       |                                   |
                       |       (2) PCP GET Response        |
                       |               MORE                |
                       |           protocol= TCP           |
                       | internal-ip-address= 198.51.100.2 |
                       |        internal-port= 12354       |
                       |   external-ip-address= 192.0.2.1  |
                       |        external-port= 32654       |
                       |       remaining-lifetime= 3600    |
                       |               END                 |
                       |           protocol= TCP           |
                       | internal-ip-address= 198.51.100.2 |
                       |        internal-port= 8596        |
                       |   external-ip-address= 192.0.2.1  |
                       |        external-port= 25659       |
                       |       remaining-lifetime= 6000    |
                       |<----------------------------------|
                       |                                   |
```

Figure 5: Flow example of GET/NEXT

In reference to Figure 6, the PCP Server returns the explicit dynamic mappings having the internal address equal to 192.0.2.2 ordered by increasing remaining lifetime. In this example, the same internal port is used for TCP and UDP.

```
                    +------+                           +------+
                    | PCP  |                           | PCP  |
                    |Client|                           |Server|
                    +------+                           +------+
                       |       (1) PCP GET Request         |
```

```
                       | internal-ip-address= 198.51.100.1 |
                       |        internal-port= 25655       |
                       |         Undefined Locator         |
                       |---------------------------------->|
                       |                                   |
                       |       (2) PCP GET Response        |
                       |               MORE                |
                       |           protocol= UDP           |
                       | internal-ip-address= 198.51.100.1 |
                       |        internal-port= 25655       |
                       |   external-ip-address= 192.0.2.1  |
                       |        external-port= 15659       |
                       |       remaining-lifetime= 1659    |
                       |                END                |
                       |           protocol= TCP           |
                       | internal-ip-address= 198.51.100.1 |
                       |        internal-port= 25655       |
                       |   external-ip-address= 192.0.2.1  |
                       |        external-port= 32652       |
                       |       remaining-lifetime= 9000    |
                       |<----------------------------------|
                       |                                   |
```

Figure 6: Flow example of GET/NEXT: same internal port number

Authors' Addresses

Mohamed Boucadair
France Telecom
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Reinaldo Penno
Cisco
USA

Email: repenno@cisco.com