Using PCP to trigger dynamic DNS updates
draft-deng-pcp-ddns-01

Xiaohong Deng, Mohamed Boucadair and Xu Wang

IETF 84, July 2012
Contents

• Focuses on problems encountered when using dynamic DNS in address sharing context (e.g., DS-Lite, NAT64, A+P)
• Particular focus on the DS-Lite case
• Issues, possible solutions
• Preliminary implementation results
• Validation of one of the solutions
Current use of DDNS

• Dynamic DNS (DDNS) is a widely deployed service to facilitate hosting servers at home premises

• Works in a client and server mode

• Update IP address changes to DDNS Server when the DDNS client detects any change of the assigned IP address
Challenges for DDNS in IP Address Sharing

Context

• The DDNS service MUST be able to maintain an alternative port number instead of the default port number

• Appropriate means to instantiate port mapping in the address sharing device MUST be supported to allow for incoming sessions

• DDNS client MUST be triggered by the change of the external IP address and the port number assigned by the address sharing device
Locate a Service Port

• Use service URIs (e.g., FTP, SIP, HTTP) which embed an explicit port number
  – Uniform Resource Identifier (URI) defined in [RFC3986] allows to carry port number in the syntax (e.g., mydomain.example:15687)

• Use SRV records
  – The majority of browsers however do not support this record type
Detect the changes

1. PCP MAP request
2. PCP MAP response
3. DDNS updates (if any)

DDNS Client
PCP Client/IWF ON
CPE or the host itself

CGN/PCP Server
(PCP mapping for 80:8080 port forwarding)

Client
Means of detecting the change

• Two alternatives have been considered:
  – If PCP mapping repair is not supported, the DDNS client issues periodically PCP requests to check whether any change has occurred
  – If PCP mapping repair is supported, the PCP Server will notify the client when there is any change. The DDNS client does only check a local mapping table
Means of detecting the change (1)

• The host issues periodically (e.g., 1h) PCP MAP requests with a lifetime (e.g., 30mn) for the purpose of enquiring external IP address
  – Companion mappings are needed to reach the internal servers. The short-lived mapping will trigger refreshing these mappings
  – The server will assign the minimum lifetime it is configured

• Upon change of the external IP address, the DDNS client updates the records
  – The mappings are needed to be re-installed
Means of detecting the change (2)

• It repeatedly checks the local mapping table periodically (e.g., 5mn, 30mn, 1h) if there is any change of the state

• Upon change of a mapping at the server side, mapping repair mechanism is executed

• Upon change of the external IP address, the DDNS client updates the records in the DDNS server
An implementation: Topology

- Web Visitor
- DDNS
- AFTR
- B4 (CPE)
- Web Server
An implementation: Typology
Implementation overview

- Implemented nodes: DDNS Server (gnudip), DDNS Client (gdipc)
- the new function introduced: PCP support (gdipc could fetch External IP address and port number from PCP Server)
- Program language: Perl
- Platform (OS): Openwrt
- Open source ddns system
  - name: GnuDIP
  - version: 2.3
An implementation : functions

• DDNS Client
  – PCP support (could get PCP records)
  – Update DDNS records dynamically
    • Now the application has supported IP address dynamically update (not for port number so far)
    • Some commercial DDNS server supports port redirect
  – Two communication methods (HTTP and TCP)
    • HTTP used to update external IP change
An implementation: Configuration and running

• gdipc configuration:

```bash
#!/gdipc.pl -c
Using Update Configuration Mode
Configuration file name: /root/.GnuDIP2
Username: gesaint
Domain: dyn.borgmann.tv
Connect by direct TCP (d) or web server (w) [d]: w
GnuDIP Server - host[:port]: 170.56.59.197
Server URL [gdudip/cgi-bin/gdipupdt.cgi]: /cgi-bin/gdipupdt.cgi
Password: gesaint
Cache File [root/.GnuDIP2.cache.gesaint.dyn.borgmann.tv]:
Minimum Seconds Between Updates [0]: 10
Maximum Seconds Between Updates [2073600]: 30
```

• gdipc -p option: PCP support

```
[root@localhost bin]# ./gdipc.pl -p
==== gdipc.pl running: Fri Apr 27 18:17:46 2012 ====
Configuration file name: /root/.GnuDIP2
Cache file name: /root/.GnuDIP2.cache.gesaint.dyn.borgmann.tv
resultcode:0
PCP External IP Address:198.18.200.1
Cache IP:123.127.237.248
Attempting update at dyn.borgmann.tv ...
Update to address 198.18.200.1 from 123.127.237.248 successful for gesaint.dyn.borgmann.tv
```
Implementation: running code & packets on wires (1)

Connection data and socket initialization:

```perl
if($opt_p){
    my $pcpserveripaddr = "2001:da8:202:108::1";
    my $pcpserverport = 5351;
    my $sinpcpserveraddr = sockaddr_in6($pcpserverport, $pcpserveripaddr);
    socket(SERVER, PF_INET6, SOCK_STREAM, getprotobyname('udp'));
}
```

Issuing packet:

```perl
$pcpmapreq = pack('b8', $version);
........
$pcpmapreq .= inet_pton(AF_INET6, $sipipaddr);
```

Communicate with PCP Server

```perl
send(SERVER, $pcpmapreq, 0, $sinpcpserveraddr);
recv(SERVER, $pcpmapres, 256, 0);
```

Parse result code and PCP external IP address

```perl
my $resultcode = substr $pcpmapres, 3, 1;
my $pcpextipaddr = substr $pcpmapres, (32 + 12), 4;
```

Update process

```perl
if ($opt_p) {
    $ip = $pcpextipaddr;
}
```
# Implementation: running code & packets on wires (2)

1. **PCP request from DDNS Client**

<table>
<thead>
<tr>
<th>Time</th>
<th>Source IP</th>
<th>Destination IP</th>
<th>Protocol</th>
<th>Source Port</th>
<th>Destination Port</th>
<th>Requested Lifetime</th>
<th>Result Code</th>
<th>Epoch Time</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:000 000</td>
<td>2001:ffff:ffff:300::1</td>
<td>2001:ffff:ffff:300::2</td>
<td>PCP</td>
<td>132</td>
<td>34453</td>
<td>3600</td>
<td>0</td>
<td>47</td>
<td></td>
</tr>
</tbody>
</table>

Capture packets file: `capture_packets.pcap`

2. **PCP response from PCP server**

<table>
<thead>
<tr>
<th>Time</th>
<th>Source IP</th>
<th>Destination IP</th>
<th>Protocol</th>
<th>Source Port</th>
<th>Destination Port</th>
<th>Requested Lifetime</th>
<th>Result Code</th>
<th>Epoch Time</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>2:000 007</td>
<td>2001:ffff:ffff:300::2</td>
<td>2001:ffff:ffff:300::1</td>
<td>PCP</td>
<td>132</td>
<td>34453</td>
<td>3600</td>
<td>0</td>
<td>47</td>
<td></td>
</tr>
</tbody>
</table>

3. **External IP Update from DDNS Client to DDNS Server:**

<table>
<thead>
<tr>
<th>Time</th>
<th>Source IP</th>
<th>Destination IP</th>
<th>Protocol</th>
<th>Source Port</th>
<th>Destination Port</th>
<th>Requested Lifetime</th>
<th>Result Code</th>
<th>Epoch Time</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>3:016 888</td>
<td>192.168.1.1</td>
<td>170.56.59.197</td>
<td>TCP</td>
<td>76</td>
<td>59440</td>
<td>5080 Len=0 MSS=14</td>
<td>1</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>4:016 888</td>
<td>192.168.1.1</td>
<td>170.56.59.197</td>
<td>TCP</td>
<td>108</td>
<td>59440</td>
<td>701 win=6912 Len=0</td>
<td>1</td>
<td>277</td>
<td></td>
</tr>
<tr>
<td>5:016 888</td>
<td>192.168.1.1</td>
<td>170.56.59.197</td>
<td>TCP</td>
<td>108</td>
<td>59440</td>
<td>701 win=6912 Len=0</td>
<td>1</td>
<td>277</td>
<td></td>
</tr>
</tbody>
</table>

Capture packets file: `capture_packets.pcap`
Next Step

• For Mapping domain name to IP address + port number

• DDNS Client
  – support maintaining port number from client
    (now client has supported IP updating)

• DDNS Server
  – Mappings between domain names and IP address + port number maintaining
Q & A

Thanks!