Discovering Provisioning Domain Names and Data

draft-ietf-intarea-provisioning-domains-00

P. Pfister, E. Vyncke, T. Pauly, D. Schinazi, M. Keane
Hosts and networks are multi-homed

Just a few examples...
Multi-Homing, the legacy way...

- Corp. ISP1
  - Public Address 1
- Corp. ISP2
  - Public Address 2
- Mobile SP
- Phone Connection Sharing
- NAT
  - Private Addresses
- Routing and/or DPI

Phone Connection Sharing

intarea WG IETF 99
Bundling IP address & DNS resolver

Multihoming and CDNs

- Name lookups for resources stored on CDNs give different answers depending on the network connection
- Host on homenet may look up name using resolver from provider A, then connect to CDN using provider B
- This will generate support requests
- What to do?

Ted Lemon, Homenet WG, IETF-99
Service Selection

Two prefixes from router
1. For service Red
2. For service Blue
At least two global addresses
1. From prefix Blue
2. From prefix Red

Traffic engineering
Different QoS
Different routing (€€€ or security)
The purpose of this draft is to:

1. Identify Provisioning Domains (PvDs).
   
   [RFC7556] Provisioning Domains (PvDs) are consistent sets of network properties that can be implicit, or advertised explicitly.
   
   Differentiate provisioning domains by using FQDN identifiers.

2. Give PvD Additional Information.
   
   Name, characteristics, captive portal, etc...
Step 1: Identify PvDs

With the PvD ID Router Advertisement Option

- At most **one occurrence in each RA**.
- **PvD ID** is an FQDN associated with options in the RA.
- **H bit** to indicate **Additional Information** is available with HTTPS.
- **L bit** to indicate the **PvD has DHCPv4 on the link**.
- **Seq. number** used for **push-based refresh**.

intarea WG IETF 100
Step 2: Get the PvD Additional Data

When the H bit is set:

GET https://<pvd-id>/.well-known/pvd

Using network configuration (source address, default route, DNS, etc...) associated with the received PvD.
Step 2: Get the PvD Additional Data

```json
{
  "name": "Foo Wireless",
  "expires": "2017-07-23T06:00:00Z",
  "prefixes": ["2001:db8:1::/48", "2001:db8:4::/48"],
  "localizedName": "Foo-Hôtel à Paris Wifi",
  "dnsZones": ["example.com", "sub.example.com"],
  "characteristics": {
    "maxThroughput": { "down": 200000, "up": 50000 },
    "minLatency": { "down": 0.1, "up": 1 }
  }
}
```

Some other examples (see also [https://smart.mpvd.io/.well-known/pvd](https://smart.mpvd.io/.well-known/pvd)):

- `noInternet` : true,
- `metered` : true,
- `captivePortalURL` : "https://captive.org/foo.html"
Step 2: Additional Data Describing the Network

- Cost of the network access
- Performance of the first uplink (ADSL, FTTH, ...)
- Captive portal
- Walled garden
- ...

![Diagram of network access](image-url)
Implementation status

Linux - [https://github.com/IPv6-mPvD](https://github.com/IPv6-mPvD)

- pvdd: A Daemon to manage PvD IDs and Additional Data
- Linux Kernel patch for RA processing
- iproute tool patch to display PvD IDs
- Wireshark dissector
- RADVD and ODHCPD sending PvD ID

Implemented in one commercial vendor router
A New, Evolutive API and Transport-Layer Architecture for the Internet:  [https://www.neat-project.org/](https://www.neat-project.org/)

European H-2020 project
10 partners (Cisco, Mozilla, EMC, Celerway...)

Integration to NEAT code: [https://github.com/NEAT-project/neat/pull/80](https://github.com/NEAT-project/neat/pull/80)

---

**Wednesday, June 29 2016: plenary session in Oslo**
IPv6 Multiprefix NEAT Integration

<table>
<thead>
<tr>
<th>Plan</th>
<th>Duration</th>
<th>Data</th>
<th>Cost</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LTE (Orange)</strong></td>
<td>🗒️ 2 mins</td>
<td>$0.5 GB</td>
<td>$0</td>
<td>🗒️ 3%</td>
</tr>
<tr>
<td><strong>VPN over LTE (Orange)</strong></td>
<td>🗒️ 6 mins</td>
<td>$0.5 GB</td>
<td>$0</td>
<td>🗒️ 4%</td>
</tr>
<tr>
<td><strong>Wi-Fi (Oslo Hostel Wi-Fi)</strong></td>
<td>🗒️ 11 mins</td>
<td></td>
<td></td>
<td>🗒️ 1%</td>
</tr>
</tbody>
</table>