Multiple Provisioning Domain Architecture
draft-ietf-mif-mpvd-arch-09

Abstract

   This document is a product of the work of the MIF Architecture Design
   team. It outlines a solution framework for some of the issues
   experienced by nodes that can be attached to multiple networks
   simultaneously. The framework defines the concept of a Provisioning
   Domain (PvD) which is a a consistent set of network configuration
   information. PvD aware nodes learn PvD specific information from the
   networks they are attached to and / or other sources. PvDs are used
   to enable separation and configuration consistency in presence of
   multiple concurrent connections.

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

Nodes attached to multiple networks may encounter problems from
conflicting configuration between the networks, or attempts to
simultaneously use more than one network. While various techniques
are currently used to tackle these problems ([RFC6419]), in many
cases issues may still appear. The MIF problem statement document
[RFC6418] describes the general landscape and discusses many of the
specific issues and scenario details.

Problems, enumerated in [RFC6418], can be grouped into 3 categories:

1. Lack of consistent and distinctive management of configuration
    elements associated with different networks.

2. Inappropriate mixed use of configuration elements associated with
    different networks during a particular network activity or
    connection.

3. Use of a particular network that is not consistent with the
    intended use of the network, or the intent of the communicating
    parties, leading to connectivity failure and / or other undesired
    consequences.

An example of (1) is a single, node-scoped list of DNS server IP
addresses learned from different networks leading to failures or
delays in resolution of names from particular namespaces; an example
of (2) is an attempt to resolve the name of an HTTP proxy server
learned from network A using a DNS server learned from network B; an
example of (3) is the use of an employer-provided VPN connection for
peer-to-peer connectivity unrelated to employment activities.

This architecture provides solutions to these categories of problems,
respectively, by:

1. Introducing the formal notion of PvDs, including identity for
    PvDs, and describing mechanisms for nodes to learn the intended
    associations between acquired network configuration information
    elements.

2. Introducing a reference model for PvD-aware nodes that prevents
    the inadvertent mixed use of configuration information which may
    belong to different PvDs.

3. Providing recommendations on PvD selection based on PvD identity
    and connectivity tests for common scenarios.

1.1. Requirements Language

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 RFC 2119 [RFC2119].

2. Definitions and Types of PvDs

Provisioning Domain:
   A consistent set of network configuration information.
   Classically, all of the configuration information available on a
   single interface is provided by a single source (such as a network
   administrator) and can therefore be treated as a single
   provisioning domain. In modern IPv6 networks, multihoming can
   result in more than one provisioning domain being present on a
   single link. In some scenarios, it is also possible for elements
   of the same PvD to be present on multiple links.

   Typical examples of information in a provisioning domain learned
   from the network are:

   * Source address prefixes for use by connections within the
      provisioning domain

   * IP address(es) of DNS server(s)

   * Name of HTTP proxy server (if available)

   * DNS suffixes associated with the network

   * Default gateway address

PvD-aware node:
   A node that supports the association of network configuration
   information into PvDs and the use of these PvDs to serve requests
   for network connections in ways consistent with the
   recommendations of this architecture.

2.1. Explicit PvDs

A node may receive explicit information from the network and / or
other sources conveying the presence of PvDs and the association of
particular network information with a particular PvD. PvDs that are
constructed based on such information are referred to as "explicit"
in this document.

Protocol changes or extensions will likely be required to support
explicit PvDs through IETF-defined mechanisms. As an example, one
could think of one or more DHCP options carrying PvD identity and /

or its elements.

A different approach could be the introduction of a DHCP option which
only carries the identity of a PvD. Here, the associations between
network information elements with the identity is implemented by the
respective protocols, for example with a Router Discovery [RFC4861]
option associating an address range with a PvD.

Another example of a delivery mechanism for PvDs are key exchange or
tunneling protocols, such as IKEv2 [RFC5996] that allow the
transport of host configuration information.

Specific, existing or new features of networking protocols that
enable the delivery of PvD identity and association with various
network information elements will be defined in companion design
documents.

Link-specific and / or vendor-proprietary mechanisms for the
discovery of PvD information (differing from IETF-defined mechanisms)
can be used by nodes either separate from, or in conjunction with,
IETF-defined mechanisms; providing they allow the discovery of the
necessary elements of the PvD(s).

In all cases, nodes must by default ensure that the lifetime of all
dynamically discovered PvD configuration is appropriately limited by
relevant events. For example, if an interface media state change is
indicated, previously discovered information relevant to that
interface may no longer be valid and so need to be confirmed or re-
discovered.

It is expected that the way a node makes use of PvD information is
generally independent of the specific mechanism / protocol that the
information was received by.

In some network topologies, network infrastructure elements may need
to advertise multiple PvDs. Generally, the details of how this is
performed will be defined in companion design documents.

2.2. Implicit PvDs and Incremental Adoption of Explicit PvDs

For the foreseeable future, there will be networks which do not
advertise explicit PvD information, because deployment of new
features in networking protocols is a relatively slow process.

When connected to networks which don't advertise explicit PvD
information, a PvD-aware node shall automatically create separate

PvDs for received configuration. Such PvDs are referred to in this
document as "implicit".

Through the use of implicit PvDs, PvD-aware nodes may still provide
benefits to their users (when compared to non-PvD aware nodes) by
following the best practices described in Section 5.

PvD-aware nodes shall treat network information from different
interfaces, which is not identified as belonging explicitly to some
PvD, as belonging to separate PvDs, one per interface.

Implicit PvDs can also occur in a mixed mode, i.e., where of multiple
networks that are available on an attached link, only some advertise
PvD information. In this case, the PvD-aware node shall create
explicit PvDs from information explicitly labeled as beloinging to
PvDs.   It shall associate configuration information not labeled with
an explicit PvD with implicit PvD(s) created for that interface.

2.3. Relationship Between PvDs and Interfaces

By default, implicit PvDs are limited to the network configuration
information received on a single interface and by default one such
PvD is formed for each interface. If additional information is
available to the host (through mechanisms out of scope of this
document), the host may form implicit PvDs with different
granularity. For example, PvDs spanning multiple interfaces such a
home network with a router that has multiple internal interfaces, or
multiple PvDs on a single interface such as a network that has
multiple uplink connections.

In the simplest case, explicit PvDs will be scoped for configuration
related only to a specific interface. However, there is no
requirement in this architecture for such a limitation. Explicit
PvDs may include information related to more than one interface if
the node learns the presence of the same PvD on those interfaces and
the authentication of the PvD ID meets the level required by the node
policy (authentication of a PvD ID may be also required in scenarios
involving only one connected interface and / or PvD).

This architecture supports such scenarios. Hence, no hierarchical
relationship exists between interfaces and PvDs: it is possible for
multiple PvDs to be simultaneously accessible over one interface, as
well as a single PvD to be simultaneously accessible over multiple
interfaces.

2.4. PvD Identity / Naming

For explicit PvDs, the PvD ID is a value that is, or has a high
probability of being globally unique, and is received as part of PvD
information. It shall be possible to generate a human-readable form

of the PvD ID to present to the end-user, either based on the PvD ID itself, or using meta-data associated with the ID. For implicit PvDs, the node assigns a locally generated ID with a high probability of being globally unique to each implicit PvD.

A PvD-aware node may use these IDs to select a PvD with a matching ID
for special-purpose connection requests in accordance with node
policy, as chosen by advanced applications, or to present a human-
readable representation of the IDs to the end-user for selection of
PvDs.

A single network provider may operate multiple networks, including
networks at different locations. In such cases, the provider may
chose whether to advertise single or multiple PvD identities at all
or some of those networks as it suits their business needs. This
architecture does not impose any specific requirements in this
regard.

When multiple nodes are connected to the same link with one or more
explicit PvDs available, this architecture assumes that the
information about all available PvDs is made available by the
networks to all the connected nodes. At the same time, connected
nodes may have different heuristics, policies and / or other
settings, including their configured sets of trusted PvDs. This may
lead to different PvDs actually being used by different nodes for
their connections.

Possible extensions, whereby networks advertize different sets of
PvDs to different connected nodes are out of scope of this document.

2.5. The Relationship to Dual-Stack Networks

When applied to dual-stack networks, the PvD definition allows for
multiple PvDs to be created whereby each PvD contains information
relevant to only one address family, or for a single PvD containing
information for multiple address families. This architecture
requires that accompanying design documents describing PvD-related
protocol changes must support PvDs containing information from
multiple address families. PvD-aware nodes must be capable of
creating and using both single-family and multi-family PvDs.

For explicit PvDs, the choice of either of these approaches is a
policy decision for the network administrator and / or the node user/
administrator. Since some of the IP configuration information that
can be learned from the network can be applicable to multiple address
families (for instance DHCP Address Selection Policy Opt [RFC7078]),
it is likely that dual-stack networks will deploy single PvDs for
both address families.

By default for implicit PvDs, PvD-aware nodes shall include multiple
IP families into a single implicit PvD created for an interface. At
the time of writing, in dual-stack networks it appears to be common
practice for the configuration of both address families to be

provided by a single source.

A PvD-aware node that provides an API to use, enumerate and inspect
PvDs and / or their properties shall provide the ability to filter

PvDs and / or their properties by address family.

3. Conveying PvD information using DHCPv6 and Router Advertisements

DHCPv6 and Router Advertisements are the two most common methods of
configuring hosts. To support the architecture described in this
document, these protocols would need to be extended to convey
explicit PvD information. The following sections describe topic
which must be considered before finalizing a mechanism to augment
DHCPv6 and RAs with PvD information.

3.1. Separate Messages or One Message?

When information related to several PvDs is available from the same
configuration source, there are two possible ways of distributing
this information: One way is to send information from each different
provisioning domain in separate messages. The second method is
combining the information from multiple PvDs into a single message.
The latter method has the advantage of being more efficient but could
have problems with to authentication and authorization, as well as
potential issues with accommodating information not tagged with any
PvD information.

3.2. Securing PvD Information

DHCPv6 and RAs both provide some form of authentication to ensure the
identity of the source as well as the integrity of the secured
message content. While this is useful, determining authenticity does
not tell a node whether the configuration source is actually allowed
to provide information from a given PvD. To resolve this, there must
be a mechanism for the PvD owner to attach some form of authorization
token or signature to the configuration information that is
delivered.

3.3. Backward Compatibility

The extensions to RAs and DHCPv6 should be defined in such a manner
than unmodified hosts (i.e. hosts not aware of PvDs) will continue
to function as well as they did prior to PvD information being added.
This could imply that some information may need to be duplicated in
order to be conveyed to legacy hosts. Similarly, PvD aware hosts
need to be able to correctly utiilize legacy configuration sources
which do not provide PvD information. There are also several
initiatives that are aimed at adding some form of additional
information to prefixes [I-D.bhandari-dhc-class-based-prefix] and
[I-D.korhonen-dmm-prefix-properties] and any new mechanism should try
to consider co-existence with such deployed mechanisms.

3.4. Retracting / Updating PvD Information

After PvD information is provisioned to a host, it may become
outdated or superseded by updated information before the hosts would
normally request updates. To resolve this requires that the
mechanism be able to update and / or withdraw all (or some subset) of
the information related to a given PvD. For efficiency reasons, there
should be a way to specify that all information from the PvD needs to
be reconfigured instead of individually updating each item associated
with the PvD.

3.5. Conveying Configuration Information using IKEv2

Internet Key Exchange protocol version 2 (IKEv2) [RFC5996] [RFC5739]
is another widely used method of configuring host IP information.
For IKEv2, the provisioning domain could be implicitly learned from
the Identification - Responder (IDr) payloads that the IKEv2
initiator and responder inject during their IKEv2 exchange. The IP
configuration may depend on the named IDr. Another possibility could
be adding a specific provisioning domain identifying payload
extensions to IKEv2. All of the considerations for DHCPv6 and RAs
listed above potentially apply to IKEv2 as well.

4. Example Network Configurations

4.1. A Mobile Node

Consider a mobile node with two network interfaces: one to the mobile
network, the other to the Wi-Fi network. When the mobile node is
only connected to the mobile network, it will typically have one PvD,
implicit or explicit. When the mobile node discovers and connects to
a Wi-Fi network, it will have zero or more (typically one) additional
PvD(s).

Some existing OS implementations only allow one active network
connection. In this case, only the PvD(s) associated with the active
interface can be used at any given time.

As an example, the mobile network can explicitly deliver PvD
information through the PDP context activation process. Then, the
PvD aware mobile node will treat the mobile network as an explicit
PvD. Conversely, the legacy Wi-Fi network may not explicitly
communicate PvD information to the mobile node. The PvD aware mobile
node will associate network configuration for the Wi-Fi network with
an implicit PvD in this case.

The following diagram illustrates the use of different PvDs in this
scenario:

```
            <----------- Wi-Fi 'Internet' PvD ------->
       +--------+
       | +----+ |     +----+        _   __             _  _
       | |WiFi| |     |    |       ( `    )           ( `   )_
       | |-IF +-|----+     |-------------------------(         `)
       | |    | |    |WiFi|     (           )        (_ Internet _)
       | +----+ |    | AP |     (             )       `- __ _) -
       |        |    |    |    (    Service    )
       |        |    +----+    (   Provider's   )
       |        |              (    Networks     -
       | +----+ |               `_              )        _  _
       | |CELL| |              (           )          ( `    )_
       | |-IF +-|-------------------------------------(         `)
       | |    | |                 (_      __)        (_ Internet _)
       | +----+ |                  `- --                `- __ _) -
       +--------+
            <------- Mobile 'Internet' PvD ----------->
```
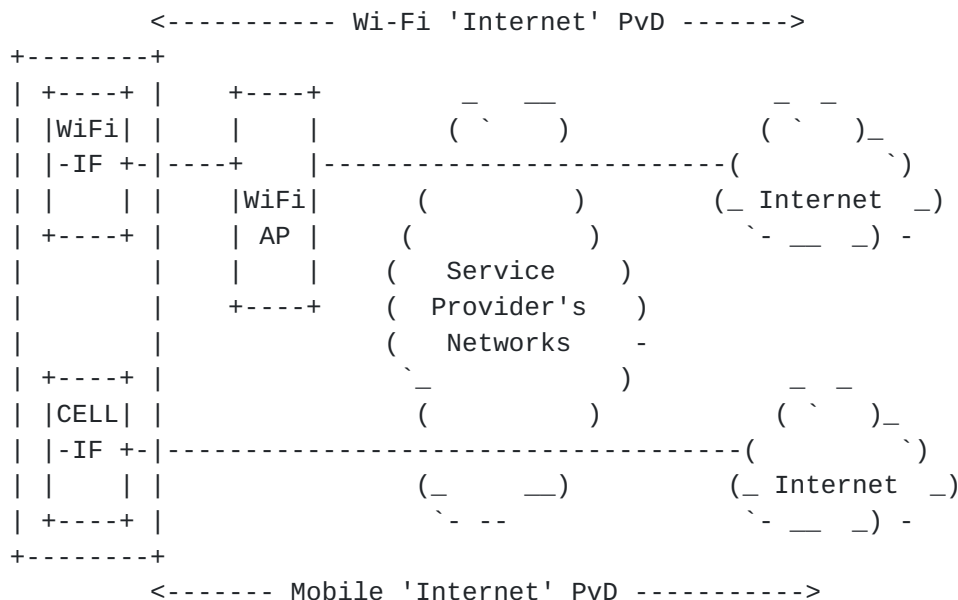
        An example of PvD use with Wi-Fi and mobile interfaces.

4.2. A Node with a VPN Connection

   If the node has established a VPN connection, zero or more (typically
   one) additional PvD(s) will be created. These may be implicit or
   explicit. The routing to IP addresses reachable within this PvD will
   be set up via the VPN connection, and the routing of packets to
   addresses outside the scope of this PvD will remain unaffected. If a
   node already has N connected PvDs, after the VPN session has been
   established typically there will be N+1 connected PvDs.

   The following diagram illustrates the use of different PvDs in this
   scenario:

```
           <----------- 'Internet' PvD ------>
    +--------+
    | +----+ |    +----+       _   __        _  _
    | |Phy | |    |    |      ( `    )      ( `   )_
    | |-IF +-|----+    |-------------------(         `)
    | |    | |    |    |     (           ) (_ Internet _)
    | +----+ |    |    |    (             )  `- __ _) -
    |        |    |Home|    (    Service    )      ||
    |        |    |gate|    (   Provider's   )     ||
    |        |    |-way|    (    Network      -    ||
    | +----+ |    |    |     `_              ) +---------+  +------------+
    | |VPN | |    |    |    (           )     |   VPN   |  |            |
    | |-IF +-|----+    |--------------------+ Gateway |--+  Private   |
    | |    | |    |    |       (_      __)    |         |  | Services  |
```
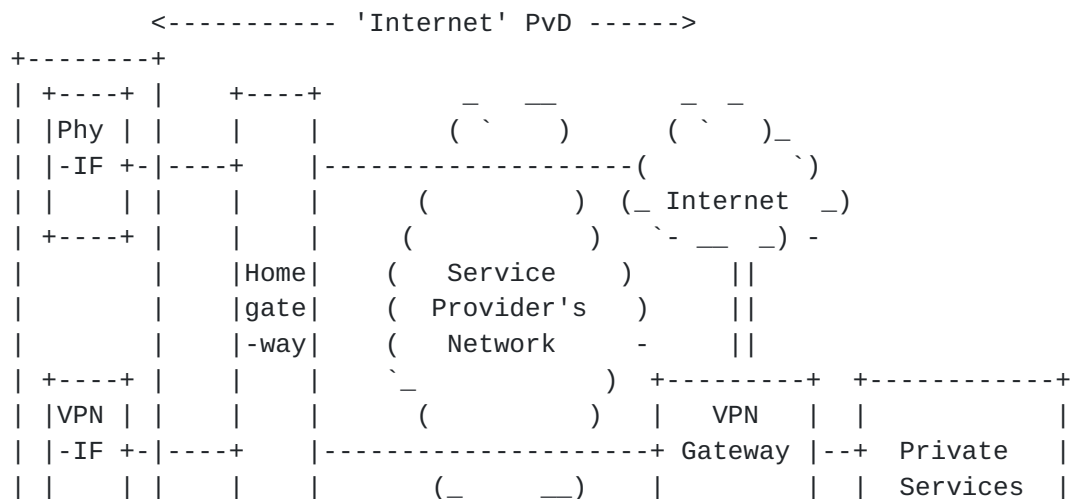
```
   | +----+ |     +----+          `- --        +---------+  +------------+
  +--------+
            <-------------- Explicit 'VPN' PvD ----->

                   An example of PvD use with VPN.
```

4.3. **A Home Network and a Network Operator with Multiple PvDs**

   An operator may use separate PvDs for individual services which they
   offer to their customers. These may be used so that services can be
   designed and provisioned to be completely independent of each other,
   allowing for complete flexibility in combinations of services which
   are offered to customers.

   From the perspective of the home network and the node, this model is
   functionally very similar to being multihomed to multiple upstream
   operators: Each of the different services offered by the service
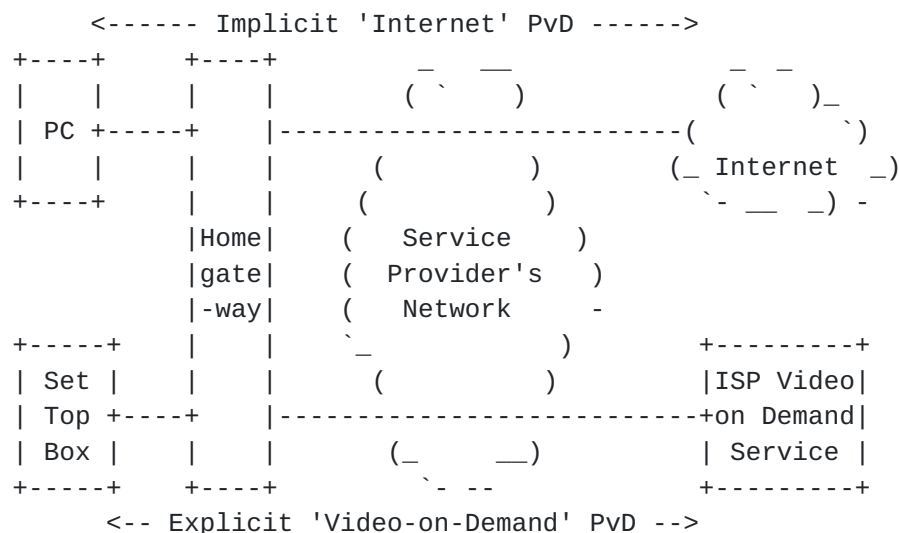   provider is its own PvD with associated PvD information. In this
   case, the operator may provide a generic / default PvD (explicit or
   implicit), which provides Internet access to the customer.
   Additional services would then be provisioned as explicit PvDs for
   subscribing customers.

   The following diagram illustrates this, using video-on-demand as a
   service-specific PvD:

```
                <------ Implicit 'Internet' PvD ------>
         +----+     +----+           _   __               _   _
         |    |     |    |         ( `    )             ( `    )_
         | PC +-----+    |    |-------------------------(          `)
         |    |     |    |        (          )           (_ Internet  _)
         +----+     |    |       (             )           `- __  _) -
                    |Home|    (     Service      )
                    |gate|      (   Provider's    )
                    |-way|      (    Network        -
         +-----+    |    |      `_                 )        +---------+
         | Set |    |    |        (             )          |ISP Video|
         | Top +----+    |    |-------------------------+on Demand|
         | Box |    |    |        (_       __)           | Service |
         +-----+     +----+        `- --                 +---------+
             <-- Explicit 'Video-on-Demand' PvD -->
```

             An example of PvD use within a home network.

   In this case, the number of PvDs that a single operator could
   provision is based on the number of independently provisioned
   services which they offer. Some examples may include:

   o Real-time packet voice

   o Streaming video

   o Interactive video (n-way video conferencing)

   o Interactive gaming

o Best effort / Internet access

5. **Reference Model for the PvD-aware Node**

5.1. **Constructions and Maintenance of Separate PvDs**

   It is assumed that normally, the configuration information contained
   in a single PvD shall be sufficient for a node to fulfill a network
   connection request by an application, and hence there should be no
   need to attempt to merge information across different PvDs.

   Nevertheless, even when a PvD lacks some necessary configuration
   information, merging of information associated with different PvD(s)
   shall not be done automatically as this will typically lead to the
   issues described in [RFC6418].

   A node may use other sources, for example: node local policy, user
   input or other mechanisms not defined by the IETF for any of the
   following:

   o Construction of a PvD in its entirety (analogous to statically
      configuring IP on an interface)

   o Supplementing some, or all learned PvDs with particular
      configuration elements

   o Merging of information from different PvDs (if this is explicitly
      allowed by policy)

   As an example, a node administrator could inject a DNS server which
   is not ISP-specific into PvDs for use on any of the networks that the
   node could attach to. Such creation / augmentation of PvD(s) could
   be static or dynamic. The specific mechanism(s) for implementing
   this are outside of scope of this document.

5.2. **Consistent use of PvDs for Network Connections**

   PvDs enable PvD-aware nodes to consistently use the correct set of
   configuration elements to serve specific network requests from
   beginning to end. This section provides examples of such use.

5.2.1. **Name Resolution**

   When a PvD-aware node needs to resolve the name of the destination
   for use by a connection request, the node could use one, or multiple
   PvDs for a given name lookup.

   The node shall chose a single PvD if, for example, the node policy
   required the use of a particular PvD for a specific purpose (e.g. to
   download an MMS message using a specific APN over a cellular
   connection). To make this selection, the node could use a match
   between the PvD DNS suffix and an FQDN which is being resolved or
   match of PvD ID, as determined by the node policy.

The node may pick multiple PvDs, if for example, the PvDs are for
general purpose Internet connectivity, and the node is attempting to
maximize the probability of connectivity similar to the Happy
Eyeballs [RFC6555] approach. In this case, the node could perform
DNS lookups in parallel, or in sequence. Alternatively, the node may
use only one PvD for the lookup, based on the PvD connectivity
properties, user configuration of preferred Internet PvD, etc.

If an application implements an API that provides a way of explicitly
specifying the desired interface or PvD, that interface or PvD should
be used for name resolution (and the subsequent connection attempt),
provided that the host's configuration permits this.

In either case, by default a node uses information obtained via a
name service lookup to establish connections only within the same PvD
as the lookup results were obtained.

For clarification, when it is written that the name service lookup
results were obtained "from a PvD", it should be understood to mean
that the name service query was issued against a name service which
is configured for use in a particular PvD. In that sense, the
results are "from" that particular PvD.

Some nodes may support transports and / or APIs which provide an
abstraction of a single connection, aggregating multiple underlying
connections. MPTCP [RFC6182] is an example of such a transport
protocol. For connections provided by such transports/APIs, a PvD-
aware node may use different PvDs for servicing that logical
connection, provided that all operations on the underlying
connections are performed consistently within their corresponding
PvD(s).

5.2.2. Next-hop and Source Address Selection

For the purpose of this example, let us assume that the preceding
name lookup succeeded in a particular PvD. For each obtained
destination address, the node shall perform a next-hop lookup among
routers associated with that PvD. As an example, the node could
determine such associations via matching the source address prefixes/
specific routes advertised by the router against known PvDs, or
receiving an explicit PvD affiliation advertised through a new Router
Discovery [RFC4861] option.

For each destination, once the best next-hop is found, the node
selects the best source address according to rules defined in
[RFC6724], but with the constraint that the source address must

belong to a range associated with the used PvD. If needed, the node
would use prefix policy from the same PvD for selecting the best
source address from multiple candidates.

When destination / source pairs are identified, they are sorted using
the [RFC6724] destination sorting rules and prefix policy table from
the used PvD.

5.2.3. Listening Applications

Consider a host connected to several PvDs, running an application
that opens a listening socket / transport API object. The
application is authorized by the host policy to use a subset of
connected PvDs that may or may not be equal to the complete set of
the connected PvDs. As an example, in the case where there are
different PvDs on the Wi-Fi and cellular interfaces, for general
Internet traffic the host could use only one, preferred PvD at a time
(and accordingly, advertise to remote peers the host name and
addresses associated with that PvD), or it could use one PvD as the
default for outgoing connections, while still allowing use of the
other PvDs simultaneously.

Another example is a host with an established VPN connection. Here,
security policy could be used to permit or deny application's access
to the VPN (and other) PvD(s).

For non-PvD aware applications, the operating system has policies
that determine the authorized set of PvDs and the preferred outgoing
PvD. For PvD-aware applications, both the authorized set of PvDs and
the default outgoing PvD can be determined as the common subset
produced between the OS policies and the set of PvD IDs or
characteristics provided by the application.

Application input could be provided on per-application, per-
transport-API-object or per-transport-API-call basis. The API for
application input may have an option for specifying whether the input
should be treated as a preference instead of a requirement.

5.2.3.1. Processing of Incoming Traffic

Unicast IP packets are received on a specific IP address associated
with a PvD. For multicast packets, the host can derive the PvD
association from other configuration information, such as an explicit
PvD property or local policy.

The node OS or middleware may apply more advanced techniques for
determining the resultant PvD and / or authorization of the incoming
traffic. Those techniques are outside of scope of this document.

If the determined receiving PvD of a packet is not in the allowed
subset of PvDs for the particular application / transport API object,
the packet should be handled in the same way as if there were no
listener.

5.2.3.1.1. Connection-oriented APIs

For connection-oriented APIs, when the initial incoming packet is
received, the packet PvD is remembered for the established connection
and used for handling of outgoing traffic for that connection. While
typically, connection-oriented APIs use a connection-oriented
transport protocol, such as TCP, it is possible to have a connection-
oriented API that uses a generally connectionless transport protocol,
such as UDP.

For APIs/protocols that support multiple IP traffic flows associated
with a single transport API connection object (for example, multi
path TCP), the processing rules may be adjusted accordingly.

5.2.3.1.2. Connectionless APIs

For connectionless APIs, the host should provide an API that PvD-
aware applications can use to query the PvD associated with the
packet. For outgoing traffic on this transport API object, the OS
should use the selected outgoing PvDs, determined as described above.

5.2.4. Enforcement of Security Policies

By themselves, PvDs do not define, and cannot be used for
communication of, security policies. When implemented in a network,
this architecture provides the host with information about connected
networks. The actual behavior of the host then depends on the host's
policies (provisioned through mechanisms out of scope of this
document), applied taking received PvD information into account. In
some scenarios, e.g. a VPN, such policies could require the host to
use only a particular VPN PvD for some / all of the application's
traffic (VPN 'disable split tunneling' also known as 'force
tunneling' behavior), or apply such restrictions only to selected
applications and allow the simultaneous use of the VPN PvD together
with the other connected PvDs by the other or all applications (VPN
'split tunneling' behavior).

5.3. Connectivity Tests

Although some PvDs may appear as valid candidates for PvD selection
(e.g. good link quality, consistent connection parameters, etc.),
they may provide limited or no connectivity to the desired network or
the Internet. For example, some PvDs provide limited IP connectivity
(e.g., scoped to the link or to the access network), but require the
node to authenticate through a web portal to get full access to the

Internet. This may be more likely to happen for PvDs which are not

trusted by a given PvD-aware node.

An attempt to use such a PvD may lead to limited network connectivity or application connection failures. To prevent the latter, a PvD-aware node may perform a connectivity test for the PvD before using it to serve application network connection requests. In current implementations, some nodes already implement this e.g., by trying to reach a dedicated web server (see [RFC6419]).

Section 5.2 describes how a PvD-aware node shall maintain and use multiple PvDs separately. The PvD-aware node shall perform a connectivity test and, only after validation of the PvD, consider using it to serve application connections requests. Ongoing connectivity tests are also required, since during the IP session, the end-to-end connectivity could be disrupted for various reasons (e.g. L2 problems, IP QoS issues); hence, a connectivity monitoring function is needed to check the connectivity status and remove the PvD from the set of usable PvDs if necessary.

There may be cases where a connectivity test for PvD selection may not be appropriate and should be complemented, or replaced, by PvD selection based on other factors. For example, this could be realized by leveraging some 3GPP and IEEE mechanisms, which would allow the exposure of some PvD characteristics to the node (e.g. 3GPP Access Network Discovery and Selection Function (ANDSF) [TS23402], IEEE 802.11u [IEEE802.11u]/ANQP).

5.4. Relationship to Interface Management and Connection Managers

Current devices, such as mobile handsets make use of proprietary mechanisms and custom applications to manage connectivity in environments with multiple interfaces and multiple sets of network configuration. These mechanisms or applications are commonly known as connection managers [RFC6419].

Connection managers sometimes rely on policy servers to allow a node that is connected to multiple networks to perform network selection. They can also make use of routing guidance from the network (e.g. 3GPP ANDSF [TS23402]). Although connection managers solve some connectivity problems, they rarely address network selection problems in a comprehensive manner. With proprietary solutions, it is challenging to present coherent behavior to the end user of the device, as different platforms present different behaviors even when connected to the same network, with the same type of interface, and for the same purpose. The architecture described in this document should improve the hosts behavior by providing the hosts with tools and guidance to make informed network selection decisions.

6. PvD support in APIs

For all levels of PvD support in APIs described in this chapter, it
is expected that the notifications about changes in the set of
available PvDs are exposed as part of the API surface.

6.1. **Basic**

   Applications are not PvD-aware in any manner and only submit
   connection requests. The node performs PvD selection implicitly,
   without any application participation, based purely on node-specific
   administrative policies and / or choices made by the user from a user
   interface provided by the operating environment, not by the
   application.

   As an example, PvD selection can be done at the name service lookup
   step by using the relevant configuration elements, such as those
   described in [RFC6731]. As another example, PvD selection could be
   made based on application identity or type (i.e., a node could always
   use a particular PvD for a VOIP application).

6.2. **Intermediate**

   Applications indirectly participate in PvD selection by specifying
   hard requirements and soft preferences. As an example, a real time
   communication application intending to use the connection for the
   exchange of real time audio / video data may indicate a preference or
   a requirement for connection quality, which could affect PvD
   selection (different PvDs could correspond to Internet connections
   with different loss rates and latencies).

   Another example is the connection of an infrequently executed
   background activity, which checks for application updates and
   performs large downloads when updates are available. For such
   connections, a cheaper or zero cost PvD may be preferable, even if
   such a connection has a higher relative loss rate or lower bandwidth.
   The node performs PvD selection based on applications' inputs and
   policies and / or user preferences. Some / all properties of the
   resultant PvD may be exposed to applications.

6.3. **Advanced**

   PvDs are directly exposed to applications for enumeration and
   selection. Node polices and / or user choices may still override the
   applications' preferences and limit which PvD(s) can be enumerated
   and / or used by the application, irrespective of any preferences
   which the application may have specified. Depending on the
   implementation, such restrictions (imposed by node policy and / or
   user choice) may or may not be visible to the application.

7. **PvD Trust for PvD-Aware Node**

7.1. **Untrusted PvDs**

   Implicit and explicit PvDs for which no trust relationship exists are

considered untrusted. Only PvDs which meet the requirements in
Section 7.2 are trusted; any other PvD is untrusted.

In order to avoid the various forms of misinformation that could
occur when PvDs are untrusted, nodes that implement PvD separation
cannot assume that two explicit PvDs with the same identifier are
actually the same PvD. A node that makes this assumption will be
vulnerable to attacks where, for example, an open Wifi hotspot might
assert that it was part of another PvD and thereby attempt to draw
traffic intended for that PvD onto its own network.

Since implicit PvD identifiers are synthesized by the node, this
issue cannot arise with implicit PvDs.

Mechanisms exist (for example, [RFC6731]) whereby a PvD can provide
configuration information that asserts special knowledge about the
reachability of resources through that PvD. Such assertions cannot be
validated unless the node has a trust relationship with the PvD;
therefore, assertions of this type must be ignored by nodes that
receive them from untrusted PvDs. Failure to ignore such assertions
could result in traffic being diverted from legitimate destinations
to spoofed destinations.

7.2. Trusted PvDs

Trusted PvDs are PvDs for which two conditions apply: First, a trust
relationship must exist between the node that is using the PvD
configuration and the source that provided that configuration; this
is the authorization portion of the trust relationship. Second,
there must be some way to validate the trust relationship. This is
the authentication portion of the trust relationship. Two mechanisms
for validating the trust relationship are defined.

It shall be possible to validate the trust relationship for all
advertised elements of a trusted PvD, irrespective of whether the PvD
elements are communicated as a whole, e.g., in a single DHCP option,
or separately, e.g., in supplementary RA options. The feasibility of
mechanisms to implement a trust relationship for all PvD elements
will be determined in the respective companion design documents.

7.2.1. Authenticated PvDs

One way to validate the trust relationship between a node and the
source of a PvD is through the combination of cryptographic
authentication and an identifier configured on the node.

If authentication is done using a public key mechanism such as a TLS
certificate or DANE, authentication by itself is not enough since
theoretically any PvD could be authenticated in this way. In
addition to authentication, the node would need configuration to
trust the identifier being authenticated. Validating the
authenticated PvD name against a list of PvD names configured as

trusted on the node would constitute the authorization step in this
case.

7.2.2. PvDs Trusted by Attachment

In some cases, a trust relationship may be validated by some means
other than those described in Section 7.2.1 simply by virtue of the
connection through which the PvD was obtained. For instance, a
handset connected to a mobile network may know through the mobile
network infrastructure that it is connected to a trusted PvD.
Whatever mechanism was used to validate that connection constitutes
the authentication portion of the PvD trust relationship.
Presumably, such a handset would be configured from the factory (or
else through mobile operator or user preference settings) to trust
the PvD, and this would constitute the authorization portion of this
type of trust relationship.

8. Contributors

The following individuals contributed to this document (listed in no
specific order): Alper Yegin (alper.yegin@yegin.org), Aaron Yi Ding
(yding@cs.helsinki.fi), Zhen Cao (caozhenpku@gmail.com), Dapeng Liu
(liudapeng@chinamobile.com), Dave Thaler (dthaler@microsoft.com),
Dmitry Anipko (dmitry.anipko@microsoft.com), Hui Deng
(denghui@chinamobile.com), Jouni Korhonen (jouni.nospam@gmail.com),
Juan Carlos Zuniga (JuanCarlos.Zuniga@InterDigital.com), Konstantinos
Pentikousis (k.pentikousis@huawei.com), Marc Blanchet
(marc.blanchet@viagenie.ca), Margaret Wasserman
(margaretw42@gmail.com), Pierrick Seite (pierrick.seite@orange.com),
Suresh Krishnan (suresh.krishnan@ericsson.com), Teemu Savolainen
(teemu.savolainen@nokia.com), Ted Lemon (ted.lemon@nominum.com) and
Tim Chown (tjc@ecs.soton.ac.uk).

9. Acknowledgments

The authors would like to thank (in no specific order) Ian Farrer,
Marcus Stenberg and Mikael Abrahamsson for their review and comments.

10. IANA Considerations

This memo does not include any IANA requests.

11. Security Considerations

There are at least three different forms of attacks that can be
performed using configuration sources that support multiple
provisioning domains.

Tampering with provided configuration information: An attacker may
   attempt to modify information provided inside the PvD container
   option. These attacks can easily be prevented by using message
   integrity features provided by the underlying protocol used to
   carry the configuration information. E.g. SEND [RFC3971] would
   detect any form of tampering with the RA contents and the DHCPv6
   [RFC3315] AUTH option that would detect any form of tampering with
   the DHCPv6 message contents. This attack can also be performed by
   a compromised configuration source by modifying information inside
   a specific PvD, in which case the mitigations proposed in the next
   subsection may be helpful.

Rogue configuration source: A compromised configuration source, such
   as a router or a DHCPv6 server, may advertise information about
   PvDs that it is not authorized to advertise. E.g., a coffee shop
   WLAN may advertise configuration information purporting to be from
   an enterprise and may try to attract enterprise related traffic.
   This may also occur accidentally if two sites choose the same
   identifier (e.g., "linsky").

   In order to detect and prevent this, the client must be able to
   authenticate the identifier provided by the network.   This means
   that the client must have configuration information that maps the
   PvD identifier to an authenticatable identity, and must be able to
   authenticate that identity.

   In addition, the network must provide information the client can
   use to authenticate the identity.   This could take the form of a
   PKI-based or DNSSEC-based trust anchor, or a key remembered from a
   previous leap-of-faith authentication of the identifier.

   Because the PvD-specific information may come to the network
   infrastructure with which the client is actually communicating
   from some upstream provider, it is necessary in this case that the
   PvD container and its contents be relayed to the client unchanged,
   leaving the upstream provider's signature intact.

Replay attacks: A compromised configuration source or an on-link
   attacker may try to capture advertised configuration information
   and replay it on a different link, or at a future point in time.
   This can be avoided by including a replay protection mechanism
   such as a timestamp or a nonce inside the PvD container to ensure
   the validity of the provided information.

12. References

12.1. Normative References

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate

Requirement Levels", BCP 14, RFC 2119, March 1997.

12.2. Informative References

   [I-D.bhandari-dhc-class-based-prefix]
              Systems, C., Halwasia, G., Gundavelli, S., Deng, H.,
              Thiebaut, L., Korhonen, J. and I. Farrer, "DHCPv6 class
              based prefix", Internet-Draft draft-bhandari-dhc-class-
              based-prefix-05, July 2013.

   [I-D.korhonen-dmm-prefix-properties]
              Korhonen, J., Patil, B., Gundavelli, S., Seite, P. and D.
              Liu, "IPv6 Prefix Mobility Management Properties",
              Internet-Draft draft-korhonen-dmm-prefix-properties-03,
              October 2012.

   [IEEE802.11u]
              IEEE, "IEEE Standard 802.11u-2011 (Amendment 9:
              Interworking with External Networks)", 2011.

   [RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C. and
              M. Carney, "Dynamic Host Configuration Protocol for IPv6
              (DHCPv6)", RFC 3315, July 2003.

   [RFC3971] Arkko, J., Kempf, J., Zill, B. and P. Nikander, "SEcure
              Neighbor Discovery (SEND)", RFC 3971, March 2005.

   [RFC4861] Narten, T., Nordmark, E., Simpson, W. and H. Soliman,
              "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861,
              September 2007.

   [RFC5739] Eronen, P., Laganier, J. and C. Madson, "IPv6
              Configuration in Internet Key Exchange Protocol Version 2
              (IKEv2)", RFC 5739, February 2010.

   [RFC5996] Kaufman, C., Hoffman, P., Nir, Y. and P. Eronen, "Internet
              Key Exchange Protocol Version 2 (IKEv2)", RFC 5996,
              September 2010.

   [RFC6182] Ford, A., Raiciu, C., Handley, M., Barre, S. and J.
              Iyengar, "Architectural Guidelines for Multipath TCP
              Development", RFC 6182, March 2011.

   [RFC6418] Blanchet, M. and P. Seite, "Multiple Interfaces and
              Provisioning Domains Problem Statement", RFC 6418,
              November 2011.

   [RFC6419] Wasserman, M. and P. Seite, "Current Practices for
              Multiple-Interface Hosts", RFC 6419, November 2011.

   [RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with
              Dual-Stack Hosts", RFC 6555, April 2012.

   [RFC6724] Thaler, D., Draves, R., Matsumoto, A. and T. Chown,
              "Default Address Selection for Internet Protocol Version 6
              (IPv6)", RFC 6724, September 2012.

   [RFC6731]   Savolainen, T., Kato, J. and T. Lemon, "Improved Recursive
               DNS Server Selection for Multi-Interfaced Nodes", RFC
               6731, December 2012.

   [RFC7078]   Matsumoto, A., Fujisaki, T. and T. Chown, "Distributing
               Address Selection Policy Using DHCPv6", RFC 7078, January
               2014.

   [TS23402]   3GPP, "3GPP TS 23.402; Architecture enhancements for non-
               3GPP accesses; release 12", 2014.

Author's Address

   Dmitry Anipko, editor
   Unaffiliated

   Phone: +1 425 442 6356
   Email: dmitry.anipko@gmail.com