Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: February 28, 2014
Z. Cao
China Mobile
A. Ding
Cambridge University / Helsinki University
August 27, 2013

Service Discovery in a Multiple Connection Environment: Problem Statement
draft-cao-mif-srv-dis-ps-03

Abstract

This document analyzes the problems of service discovery in a multiple connection environment. A multiple connection environment consists of multiple-interfaced nodes connecting to multiple networks or multiple provisioning domains. Given a type of service a multiple-interfaced client is looking for, the discovery progress ought to return a correct pointer to the service instance that the client is able to access without trying every available channel.

Status of This Memo

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

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

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

This Internet-Draft will expire on February 28, 2014.

Copyright Notice

Copyright (c) 2013 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 2
2. Requirements and Terminology 3
2.1. Requirements 3
2.2. Terminology 3
3. Scenarios 3
3.1. Mif Scenario 3
3.2. Homenet Scenario 5
4. Problem Analysis 5
5. IANA Considerations 9
6. Security Considerations 9
7. References 9
7.1. Normative References 9
7.2. Informative References 9
Authors' Addresses 10

1. Introduction

A multihomed host has multiple provisioning domains via physical and/or virtual interfaces. A multihomed host receives node configuration information from each of its access networks, through various mechanisms such as DHCP, PPP and IPv6 Router Advertisements. When the received node-scoped configuration objects have different values from each administration domains, such as different DNS servers IP addresses, different default gateways or different address selection policies, the node has to decide which it will use or how it will merge them.

Issues regarding how the multi-homed host uses the configuration objects have been addresses in [RFC6418]. Current practices of how the various implementations handle these problems are introduced in [RFC6419]. [RFC6731] extends DHCPv6 to inform the host which DNS server it ought to select to send the query request, and DNS based Service Discovery (DNS-SD) has been specified in [RFC6763].

This document analyzes the problem of service discovery in a multiple connection environment. A multiple connection environment consists of multiple-interfaces nodes connecting to multiple networks or multiple provisioning domains. Given a type of service a multiple-interfaced client is looking for, the discovery progress ought to return a correct pointer to the service instance that the a client is able to access.

2. Requirements and Terminology

2.1. Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2.2. Terminology

Service Domain

A set of services that can be accessed by users. Besides providing services, a service domain is responsible for delivering configuration and pointers that ensure a guaranteed service access.

Service Discovery

Procedure to acquire information that is necessary to access service.

Multiple Connection Environment

Consists of multiple-interfaced nodes that connect to multiple networks or multiple provisioning domains.

3. Scenarios

We describe two scenarios in this section, one related to Multiple Interfaces, and the other one related to Home Networks (homenet).

3.1. Mif Scenario

The service discovery process can be summarized as the following five steps.

1. Service Discovery Preparation: the host determines which interface it should send a query request based on the configuration information.

2. Service Query Request: the host sends a query request to find a service. The query should include a description of the service, for example, a full-qualified domain name, a URI, or an application-specific naming of the service.

3. Service Request Handling: any entity that receives the query request should handle the request. The entity should understand

the meaning of the request, and check the semantics of the request language before giving an answer back.

4. Service Query Response: the entity that receives the query request should reply with an answer to the query. The answer should include a pointer to the service.

5. Service Access: the host accesses the service via the pointer provided in the query response. The host is supposed to be able to get the service instance via the pointer under a successful and efficient service discovery mechanism, unless the servers in such service domain encounter problems e.g. a web server is down.

Figure 1 shows a typical scenario for service discovery in a multiple connection environment. It is common in today's mobile Internet that a host is equipped with multiple network interfaces. On the service domain, different services are deployed and some services may not be accessible to a certain interface on the host due to security concern or access policy. The connectivity each interface provides may not be restricted to Internet access. For instance, WLAN and bluetooth can offer direct access to potential services e.g. printers via local ad-hoc connectivity. In such multiple connection environment, the service discovery process should return a correct point to the host and ensure that the host can access the service via this pointer. This situation makes the multiple interface service discovery different from the typical one-interface Internet access scenario. Furthermore, the growing usage of IPv6 in the homenet environment has made service discovery more challenging that requires thorough investigation.

```
                 +-------------------------------------------+
                 | Host with multiple interfaces             |
                 |                                           |
                 |                                           |
                 |   +---+    +---+    +---+    ...  +---+   |
                 |   |   |    |   |    |   |    ...  |   |   |
                 +-----+--------+--------+-------------+-----+
                       |        |        |             |
                      3G       WLAN__  Bluetooth ... Wired
                      / \________    \__     \
                     /           \____  \     \
      ---------------------------------------------------------------------
             +-+-+-+-+-+-+-+-+-+-       +-+-+-+-+-+-+-+-+-+     Service Domain
            (      Service     )     (     Service     )
             \_________________/       \_________________/
```

Figure 1: Multiple Interface Host with Multiple Available Services

3.2. Homenet Scenario

We also describe the issues related to the homenet architecture [I-D.ietf-homenet-arch], as depicted in Figure 2.

Suppose one MIF host is connected to three domains: homenet domain, 3gpp domain and a WiFi or enterprise domain. There is one service that is named with the private domain name, say 'temperature.ietf', which is only resolvable via the domain name service residing inside the homenet and is supported by the multicast dns service [RFC6762].

There are several problems in this scenario. First of all, since the host has two unicast dns domains configured over the 3GPP and WiFi, and as well as a multicast service discovery domain within the homenet, the host does not know which domain it should send a dns resolution request. Secondly, even if coupled with the split dns solution [RFC6731], the configuration information obtained from DHCP supports only those two unicast dns domains, but not the homenet domain which is normally considered as 'zero-configuration'. Third, the service discovery problem will become more complicated if the host is connecting to more than one home networks, i.e., multiple multicast dns domains.

```
                                        +--------------+
                                        |  3GPP Domain |
                                        +--------------+
                                               |
 +--------------+      +---+                  |
 |Homenet Domain|------| H |----------------+
 +--------------+      +---+                  |
                                               |
                                        +--------------+
                                        |     Wi-Fi    |
                                        |    Domain    |
                                        +--------------+
```

Figure 2: Homenet Scenario

4. Problem Analysis

The problems that a multiple-interfaced host may meet during the service discovery include:

1. How the query requests are sent? Because there are multiple interfaces available and multiple service rendezvous existing,

the host should decide which destination it ought to send the query to. And if there is a round-robin mechanism, the host should determine the order of the query request.

2. How to handle the query and reply? Some pointers to the service are restricted to a local scope or a certain interface, e.g., an office printer may not be accessible to the 3G-interface. The service discovery process is supposed to return a pointer that is accessible to the host.

3. How to access the service? Given the pointer to the service, the host should be able to determine from which interface it can access the service.

The existing work of [RFC6418] and [RFC6419] have covered the general problems encountered by hosts accessing multiple provisioning domains, but the focus is on connectivity and configuration. Proposal of Happy Eyeball in [I-D.ietf-mif-happy-eyeballs-extension] allows a host with multiple interfaces to pick a suitable one for access and enables automatic fallback. In a DNS based service discovery [RFC6763], the problem of domain split is analyzed in the [RFC6731]. The document defines an extension to the DHCPv4 and DHCPv6 to inform the MIF host which domain scope the Recursive DNS Server(RDNSS) is serving for, so that the "service query request" can be sent to the correct RDNSS to get an answer.

The existing proposals resolve the partial problem in the service discovery process mentioned above. To highlight the missing blocks, Figure 3 provides a ’gap’ analysis. In the figure, we compare three existing solutions on service discovery, DNS-SD[RFC6763], DNS-Server-Selection [RFC6731], and MIF Happy Eyeball [I-D.ietf-mif-happy-eyeballs-extension], from three aspects as mentioned above. The DNS-Srv-Sel solution uses the defined DHCP option for the MIF host to select the corresponding DNS Server, and MIF-HE inherits this method in its most updated version. The MIF-HE can help host failover to the workable interface during service access while DNS-Srv-Sel does not handle this particular issue. The DNS-SD is not designed for a multiple interfaces environment and DNS server selection and request handling are based on standard DNS behaviors.

Aspects \Sol	DNS-SD	DNS-Srv-Sel	MIF-HE
How to Send Query	Std. DNS behavior	DHCP Option informed	Same as DNS-Srv-Sel

How to Handle Queries	Std. DNS server behavior	selection based on option	Same as DNS-Srv-Sel
How to Access service	no guarantee of connectivity	not possible if ports rejected	Failover to the Happiest one

Figure 3: Gap Analysis of Existing Service Discovery Methods

In a complicated network as shown in Figure 4 , the host connects to the enterprise network via the wired interface, a WLAN network with the 802.11 interface, and an operator's network via the cellular interface. The three intranets have their own Firewall policies to the global Internet. On the enterprise network, many outgoing ports are restricted, and on the WLAN and operator's public network, there is more freedom. If the MIF host makes a DNS-SRV query to a service in a global domain, all the RDNS servers have the corresponding records. But say the service port number has been blocked by the enterprise network administrator, the DNS has no such information. Even if the DNS returns a pointer to the MIF host, the MIF host cannot access this service via the wired interface.

```
                          +---------------+
                          | RDNSS with    |    |   Enterprise
   +------+               | public +      |----|   Intranet
   |      |               | enterprise's  |    |
   |      |------Wired -----| private names |    |
   |      |               +---------------+    +----------+----+
   | MIF  |                                              | FW |
   | node |                                              +----+
   |      |               +---------------+  +----+         |
   |      |----- WLAN ------| RDNSS with    |--| FW |-----| Public
   |      |               | public names  |  +----+     | Internet
   |      |               +---------------+                |
   |      |                                              +----+
   |      |                                              | FW |
   |      |               +---------------+    +----------+----+
   |      |---- Cellular ---| RDNSS with    |    |
   +------+               | public +      |    |   Operator
                          | operator's    |----|   Intranet
                          | private names |    |
                          +---------------+
```

Figure 4: Scenario of Multiple Interface DNS Service Discovery

CoAP [I-D.ietf-core-coap] is an IETF designed RESTful protocol for constrianed environment. CoAP defines a link-format for service discovery of the particular CoAP server, i.e., "/.well-known/core". If the CoAP client has multiple access networks as shown in Figure 5, the situation turns to be more complex. For instance, if the MIF client wants to find a humidity sensing resource, but does not know which domain contains the information, it basically needs to send multiple CoAP GET requests with the well-known URL. Once it gets a response for the required resource, it can send the corresponding request to get the information. However this way is sub-optimal especially for constrained devices. MIF service discovery SHOULD consider the efficiency of the service discovery process.

```
+---------------+        +--------+          +----------+
|    Internal   |________|   MIF  |__________| External |
|     Domain    |        |  Host  |          |  Domain  |
+---------------+        +--------+          +----------+
   |  GET /.well-known/core  |                      |
   |<------------------------|  GET /.well-known/core |
   |                         |----------------------->|
   |     ACK CON </light>    | ACK CON </temp> </hum> |
   |------------------------>|<-----------------------|
   |                         |       GET ~/hum        |
   |                         |----------------------->|
```

Figure 5: CoAP Service Discovery for a MIF Host

As a summary of the above analysis, the general problems and requirements of service discovery in a MIF environment can be summarized as follows:

Service Directory Service Configuration: Service directory is the entity that stores or can get the stored relationship between service names and service pointers. Different interfaces or provisioning domains have their different service directories. How to configure them on the MIF host and how the MIF host utilizes the configured information are important for the service discovery process to behave correctly.

Service Directory Selection: After the service directory information is configured on the host, the host is able to select the correct directory to send the query. The host can utilize auxiliary information available or send the query to all the directories that have been configured. The behavior of MIF host to select a correct directory is also important for a stable system.

Service Pointer/Address Resolution: The same service may have different available addresses and pointers, and some service has limited connectivity. So the resolution process should be able to return to the MIF host a record that is accessible from at least one of the interfaces. Efficiency SHOULD be taken into consideration in this phase.

Service Route Selection: With the pointers returned, the host should route the service level data to the service instance identified by the returned pointers.

5. IANA Considerations

This document has no IANA requests.

6. Security Considerations

The query response exchanges should be protected by security mechanisms. If the response contains invalid information, e.g. a pointer to a worm website, it harms. As a consequence, the service discovery should protect bogus information injected by attackers or intruders. The security consideration ought to be made by the underlining protocols, and it is out the scope of this problem statement document.

7. References

7.1. Normative References

[I-D.ietf-mif-happy-eyeballs-extension]
Chen, G., Williams, C., Wing, D., and A. Yourtchenko, "Happy Eyeballs Extension for Multiple Interfaces", draft-ietf-mif-happy-eyeballs-extension-02 (work in progress), February 2013.

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

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, February 2013.

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, February 2013.

7.2. Informative References

[I-D.ietf-core-coap]

Shelby, Z., Hartke, K., and C. Bormann, "Constrained Application Protocol (CoAP)", draft-ietf-core-coap-18 (work in progress), June 2013.

[I-D.ietf-homenet-arch]
Chown, T., Arkko, J., Brandt, A., Troan, O., and J. Weil, "Home Networking Architecture for IPv6", draft-ietf-homenet-arch-10 (work in progress), August 2013.

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

[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.

Authors' Addresses

Zhen Cao
China Mobile
Xuanwumenxi Ave. No. 32
Beijing 100871
China

Phone: +86-10-52686688
Email: zehn.cao@gmail.com, caozhen@chinamobile.com

Aaron Yi Ding
Cambridge University / Helsinki University
William Gates Building, 15 JJ Thomson Ave
CB3 0FD Cambridge
United Kingdom

Phone: +44-7934034801; +358-9-19151296
Email: Aaron.Ding@cl.cam.ac.uk

MIF
Internet-Draft
Intended status: Informational
Expires: August 19, 2014

D. Liu
China Mobile
Ted. Lemon
Nominum
Yuri. Ismailov
Ericsson
Z. Cao
China Mobile

February 15, 2014

MIF API consideration
draft-ietf-mif-api-extension-05

Abstract

Hosts may connect to the internet using more than one network API at a time, or to a single network on which service is provided by more than one provider. Existing APIs are inadequate to allow applications to successfully use the network in this environment. This document presents a new abstract API that provides the minimal set of messages required to enable an application to communicate successfully in this environment.

Status of This Memo

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

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

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

This Internet-Draft will expire on August 19, 2014.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents

(http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 2
2. Conventions used in this document 3
3. MIF API Concept 3
3.1. Provisioning Domains 4
3.2. MIF API Elements 4
3.2.1. Application Element 5
3.2.2. High Level API 5
3.2.3. MIF API 6
3.2.4. Communications API 6
3.2.5. Network Link API 6
3.2.6. MIF API communication model 7
3.2.7. MIF Messages 7
3.3. Example Usage 14
4. Security Considerations 16
5. IANA Considerations 16
6. Acknowledgments 16
7. References 16
7.1. Normative References 16
7.2. Informative References 16
Authors' Addresses 17

1. Introduction

Traditionally, applications that communicate on the network have done so over a single network link, which is provided by a single service provider. However, this operating environment is now the exception rather than the rule. Most devices now have multiple wireless interfaces that are, in practice, connected to networks operated by different providers. These networks may or may not have different reachability characteristics with respect to any given service an application may wish to connect to.

For example, consider a typical modern host with two wireless interfaces: a wireless interface connected to a broadband network, and another connected to some kind of cellular network. The same host may also have a wired interface which is sometimes connected to a third broadband link. It is also quite common for hosts to have

VPN links that are configured, for example, for access to corporate networks, or for access to network privacy services.

As a result, it is now quite typical that a program attempting to communicate in such an environment will be presented with conflicting configuration information from more than one provider. In addition, the cost of bandwidth on different links and the power required ny those links may require consideration.

The API specified in this document is intended to describe the minimal complete set of API calls required to implement higher level APIs that solve these problems. It is not expected that applications will be implemented to this API, although it should be possible to do so. Rather, we expect this API to be used as a basis for building higher-level APIs that provide domain-specific solutions to these problems. The reason for specifying a lower-level API is to enable any arbitrary domain- specific API to be implemented, since no single higher-level API is likely to satisfy the needs of every application.

The API specified here is an abstract API. This means that we specify the functionality that is required to implement the API, but we do not provide specific bindings for any programming language: these are left up to the implementation. The API is described in terms of messages sent and messages received, rather than in terms of procedure calls, because it is necessary to be able to interleave these messages; a procedure call API necessarily precludes interleaving.

This document is intended to be read and used as a checklist by operating system vendors who are interested in providing adequate functionality to applications that must run on hosts in environments like the ones described here. It should also be useful to purchasers of devices that must operate in such environments, so that they can tell if they are getting a device that can actually succeed in these environments.

2. Conventions used in this document

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].

3. MIF API Concept

The MIF API is intended to deal with situations where more than one interface may be active at a time. It must also deal with situations where a single interface is connected to a link that provides more

than one type of network service. The most common example of this that we expect is a dual-stack network configuration.

3.1. Provisioning Domains

Document [I-D.ietf-mif-mpvd-arch] defines Provisioning Domain (PvD) architecture and its associated mechanism, such as PvD identity/ naming concept, conveying mechanism etc. According to [I-D.ietf-mif-mpvd-arch], a provisioning domain is a consistent set of network configuration information. Classically, the entire set available on a single interface is provided by a single source, such as 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.

To properly handle these multiple-service interfaces, we specify the API not in terms of interfaces, but in terms of provisioning domains. From the perspective of the MIF API, a provisioning domain consists of a link, plus all the configuration information received on that link for that provisioning domain. So for an IPv4 provisioning domain, that would be whatever information is received from the DHCP server. For an IPv6 provisioning domain, the information received through router advertisements would be combined with the information recieved via DHCPv6.

3.2. MIF API Elements

There are a number of different, essentially independent, pieces of software that need to be connected together in order to fully support a successful MIF communication strategy. These elements are shown in figure 3.1.

```
                        +-------------------------------------------+
                        |               Application                 |
                        +-------------------------------------------+
                               /\   ||          /\  ||    /\  ||
                               ||   \/          ||  ||    ||  ||
                        +--------------------+  ||  ||    ||  ||
                        | High Level API     |  ||  ||    ||  ||
                        +--------------------+  ||  ||    ||  ||
                               /\   ||          ||  ||    ||  ||
                               ||   \/          ||  \/    ||  ||
                        +------------------------------+  ||  ||
                        |           MIF API            |  ||  ||
                        +------------------------------+  ||  ||
                           /\  ||                          ||  \/
                           ||  ||   +------------------------------+
                           ||  ||   +      Communications API      +
                           ||  ||   +------------------------------+
                           ||  ||                /\  ||
                           ||  \/                ||  \/
                        +-------------------------------------------+
                        |            Network Link API               |
                        +-------------------------------------------+
                                /\  ||                /\  ||
                                ||  \/                ||  \/
                        +-------------------+  +--------------------+
                        | Network Interface |  | Network Interface  |
                        |         1         |  |          2         |
                        +-------------------+  +--------------------+
```

Figure 1: MIF API Elements

3.2.1. Application Element

This is an actual application. Applications fall into a variety of broad categories, including network servers, web browsers, peer-to-peer programs, and so on. Although we are focusing here on the mechanisms required to allow these applications to originate connections to remote nodes, it is worth noting that applications must also be able to receive connections from remote nodes.

3.2.2. High Level API

Applications are generally expected to originate connections using some general-purpose high-level API suited to their particular function. It is likely that different applications may use different high-level APIs to communicate, depending on their particular needs. We do not describe the functioning of such high-level APIs; however,

one such API under current consideration is the Happy Eyeballs for MIF [reference]. These APIs are expected to be able to be implemented using functionality like that described in the MIF API.

3.2.3. MIF API

This is the API being described in this document. Generally speaking, this API is used by higher-level APIs. However, it is permissible for applications to use the MIF API when it is deemed necessary. Currently, several modern web browsers take this approach to establishing network connections, rather than relying on vendor-provided connection mechanisms.

3.2.4. Communications API

Once an application has originated a connection with a remote node using either a high-level API or the MIF API, it must communicate. Similarly, when an application receives a connection from a remote node, it must communicate with that remote node. The communications API is used for this communication. Popular examples of such APIs include the POSIX socket API and a variety of other related APIs.

It is likely that in some instances, implementations of the MIF API will be done as extensions to the Communications API provided by a particular operating system; the functional separation we show here is intended to allow us to illustrate only those features required in a MIF environment, while relying on existing communications APIs to provide the rest.

3.2.5. Network Link API

This is the software that is responsible for actually managing whatever network links are present on a node, whether these are physical links or tunnels. What precisely this functional box contains may vary greatly from device to device. On a typical modern computer workstation, this functionality would almost certainly reside entirely in the system kernel; however, on an embedded device everything from the Application down to the Network Link API could easily be running together on the bare metal as a single program.

The Network Link API can completely concealed from the Application, so we don’t show a connection between them on the functional diagram, and indeed we do not talk about the functionality provided by this API. The reason for showing it on the functional diagram is simply to show that there likely is an API in common between MIF and the Communications API.

3.2.6. MIF API communication model

MIF API requests are made in the form of messages posted to the MIF API, and messages received from it. To accomplish this, several API calls are available. These calls mediate communication between the MIF API and the High Level API, or between the MIF API and the Application. In addition, the CHECK MESSAGE call allows the application to probe for or wait for messages from any of the APIs.

3.2.6.1. POST MESSAGE call

This call causes a message to be posted to the MIF API. The call posts the message, and then returns.

3.2.6.2. CHECK MESSAGE call

This call checks to see if there is a message waiting either from the High Level API, the MIF API, or the Communications API. Ideally it should be able to report the availability of any message or event that the application might anticipate receiving, so that the application can simply block waiting for such an event using this call. The application should be able to do a non-blocking probe, wait for some limited period of time, or wait indefinitely.

An example of a function of this type in existing practice is the POSIX poll() system call.

3.2.6.3. GET MESSAGE call

This call checks to see if there is a message waiting. If there is no message, it returns a status code indicating that there is no message waiting. If there is a message, it returns the message.

3.2.7. MIF Messages

MIF messages always go in one direction or the other: from the subscriber to the MIF API, or to the subscriber from the MIF API. We use the term "subscriber" here to mean either the Application or the High Level API, since either is permitted to communicate with the MIF API.

Messages described here are grouped according to function.

3.2.7.1. Announce Interfaces

This message is sent to the MIF API to ask it to send a message announcing the existence of any interface. When the MIF API receives this message from a subscriber, it iterates across the list of all

known interfaces; for each known interface, it sends an Interface Announcement message to the subscriber.

In addition, the MIF API sets a flag indicating that the subscriber is interested in learning about new interfaces. When the MIF API detects the presence of a new interface, it sends an Interface Announcement message for that interface to the subscriber. This would happen, for instance, when a new tunnel is configured, or when a USB device that is a network interface is discovered by the Network API.

Also, if a network interface goes away, either because the physical network device is disconnected, or because a tunnel is disabled, the MIF API will send a No Interface Announcement message to the subscriber.

3.2.7.2. Stop Announcing Interfaces

This message is sent to the MIF API when a subscriber is no longer interested in receiving announcements about new interfaces. Subsequently, the MIF API will no longer send Interface Announcement or No Interface Announcement messages to the subscriber.

3.2.7.3. Interface Announcement

This message announces the existence of an interface. The announcement includes an interface display name and interface identifier.

3.2.7.4. No Interface Announcement

This message announces that an interface that had been previously announced is no longer present. The announcement includes the interface identifier.

3.2.7.5. Announce Provisioning Domain

This message requests the MIF API to announce the availability of any provisioning domains configured on a particular interface. The interface identifier must be specified.

Upon receipt, the MIF API will iterate across the list of Provisioning Domains present for a particular interface, and will send a Provisioning Domain Announcement for each such Provisioning Domain.

In addition, the MIF API will set a flag indicating that the subscriber wishes to know about new provisioning domains as they

appear. Subsequently, when a new Provisioning Domain appears, the MIF API will send a Provisioning Domain Announcement message to the subscriber.

Finally, if a Provisioning Domain expires or is invalidated, the MIF API will send the subscriber a No Provisioning Domain Announcement message for that Provisioning Domain.

In the event that an interface on which provisioning domains has been announced goes away, a No Provisioning Domain Announcement message will be sent for each provisioning domain that had previously been announced on that interface before the No Interface Announcement message is sent.

Once a No Interface Announcement message has been sent, any subscriber that had subscribed to Provisioning Domain announcements for that interface will be automatically unsubscribed.

3.2.7.6. Stop Announcing Provisioning Domains

This message requests that the MIF API stop sending the subscriber Provisioning Domain Announcement and No Provisioning Domain Announcement messages. The subscriber must indicate the interface for which it no longer wishes to receive Provisioning Domain announcements.

3.2.7.7. Provisioning Domain Announcement

This message is sent by the MIF API to the subscriber to indicate that a new Provisioning Domain has successfully been configured on an interface. The announcement includes the interface identifier and the provisioning domain identifier.

3.2.7.8. No Provisioning Domain Announcement

This message is sent by the MIF API to the subscriber to indicate that an existing, previously announced provisioning domain has expired or otherwise become invalid, and can no longer be used.

3.2.7.9. Announce Configuration Element

This message is sent by the subscriber to request a specific configuration element from a specific provisioning domain. A provisioning domain identifier must be specified.

The MIF API will respond by iterating across the complete list of configuration elements for a provisioning domain, sending a

Configuration Element Announcement message to the subscriber for each one.

Additionally, if any Configuration Elements subsequently complete for a particular provisioning domain, the MIF API will send a Configuration Element Announcement message to the subscriber for each such element. If a Configuration Element becomes invalidated after it has been announced, the MIF API will send a No Configuration Element message.

If a provisioning domain expires or becomes invalid, the MIF API will iterate across the list of remaining configuration elements for that provisioning domain amd send a No Configuration Element Announcement message for each such configuration element.

3.2.7.10. Configuration Element Announcement

The Configuration Element Announcement message includes a Provisioning Domain ID and a Configuration Element Type, which can be one of the following: Config Element RA Config Element DHCPv6 Config Element DHCPv4 etc.

3.2.7.11. No Configuration Element Announcement

The No Configuration Element Announcement message indicates that a previously valid configuration element for a provisioning domain is no longer valid. The message includes a provisioning domain identifier and a configuration element type.

3.2.7.12. Stop Announce Configuration Element

The Stop Announce Configuration Element message requests that MIF API stop announce configuration element.

3.2.7.13. Announce Address

This message is sent by the subscriber to request announcements of valid IP addresses for a specific provisioning domain. A provisioning domain identifier must be specified.

The MIF API will respond by iterating across the complete list of configuration elements for a provisioning domain, sending a Address Announcement message to the subscriber.

Additionally, if any new Address is subsequently configured on a particular provisioning domain, the MIF API will send an Address Announcement message to the subscriber for each such element. If an

address becomes invalidated after it has been announced, the MIF API will send a No Address Announcement message.

If a provisioning domain expires or becomes invalid, the MIF API will iterate across the list of remaining configuration elements for that provisioning domain amd send a No Address Announcement message for each such address.

3.2.7.14. Address Announcement

The Address Announcement message includes single IPv4 or IPV6 address and a Provisioning Domain identifier, as well as the valid and preferred lifetimes for that IP address (IPv6 only).

3.2.7.15. Stop Announcing Address

The Stop Announcing Address message requests the MIF API to stop announcing address.

3.2.7.16. No Address Announcement

The No Address Announcement message indicates that a previously valid address for a provisioning domain is no longer valid. The message includes a provisioning domain identifier and an IPv4 or IPv6 address.

3.2.7.17. Get Configuration Data

The Get Configuration Data message is sent to the MIF API, and includes a Provisioning Domain ID, a Configuration Element Type, and a Configuration Information Identifier.

Configuration Information Identifiers: DNS Server List etc.

The MIF API searches the configuration database for the specific type of Configuration Element on the specified Provisioning Domain to see if there is any configuration data of the specified type. If so, the MIF API sends a Configuration Data message to the subscriber; otherwise it sends a No Configuration Data message to the subscriber.

3.2.7.18. Translate Name

The Translate Name message is sent to the MIF API. It includes a provisioning domain and a name, which is a UTF8 string naming a network node. The message also includes a Translation Identifier, which the subscriber must ensure is unique across all outstanding name service requests.

The MIF API begins a name resolution process. As results come in from the name resolution process, the MIF API sends Name Translation messages to the subscriber for each such result.

Name resolution can be handled by one or more translations systems such as local host table lookup, Domain Name System, NIS, LLMNR, and is implementation-dependent. **need to think about this

3.2.7.19. Stop Translating Name

This message is sent to the MIF API to indicate that the subscriber is no longer interested in additional results from a particular name translation process. The message includes the Translation Identifier.

3.2.7.20. Name Translation

The MIF API sends a Name Translation message to subscribers whenever results come in from a name translation process being performed on behalf of the subscriber. The Name Translation message includes the Translation ID generated by the subscriber, and an IP address returned by the translation process. If a single translation result contains more than one IP address, or IP addresses of different types, the MIF API sends a single Name Translation message for each such IP address.

3.2.7.21. Connect to PvD

The Connect to PvD message is used for the advanced application to select the PvD. Advanced application can use this message to select a specific PvD by providing the PvD identifier as parameter. This is the advanced case that discussed in section 6.3 of [I-D.ietf-mif-mpvd-arch].

3.2.7.22. Connect to Address

The Connect to Address message contains an IP address, a provisioning domain identifier, and a connection identifier which the subscriber must ensure is unique. The MIF API attempts to initiate a TCP connection to the specified IP address using one or more source addresses that are valid for the specified provisioning domain, according to the source address selection policy for that provisioning domain.

If the connection subsequently succeeds, the MIF API will send a Connected message to the subscriber. If it subsequently fails, the MIF API will send a Not Connected message to the subscriber.

3.2.7.23. Connect to Address From Address

The Connect to Address From Address message contains a source IP address, a destination IP address, a provisioning domain identifier, and a connection identifier which the subscriber must ensure is unique. The MIF API attempts to initiate a TCP connection to the specified IP address using the specified source address.

If the connection subsequently succeeds, the MIF API will send a Connected message to the subscriber. If it subsequently fails, the MIF API will send a Connection Failed message to the subscriber.

3.2.7.24. Connected

The Connected message contains the connection identifier that was provided in a previous Connect to Address or Connect to Address From Address message sent by the subscriber. It also contains an token, suitable for use with the connection API, for communicating with the end node to which the connection was established.

3.2.7.25. Not Connected

The Not Connected message contains the connection identifier that was provided in a previous Connect to Address or Connect to Address From Address message sent by the subscriber. It also contains an indication as to what went wrong with the connection.

3.2.7.26. Application Connectivity Management

The following APIs are used for application connectivity management.

3.2.7.26.1. Application: Wants to connect

This message is sent by the application to the MIF API that indicates the application wants to connect to the network. The purpose of this call is to trigger the MIF API to engage in any work that is required to configure the network. If all interfaces are already operational, this message is a no-op. An application would typically send this message either because it has no provisioning domains on which it can attempt to connect, or because it has failed to connect on any existing provisioning domain.

3.2.7.26.2. Application: Connection is idle

This message is sent by the applicaiton to the MIF API to indicate that the application is not expecting to receive any data or send any data. This is a signal to the MIF API that, for example a radio that consumes a lot of power can be put into a temporary idle state, but

that the application expects to resume communication in the future using the existing connection.

3.2.7.26.3. Application: Connection can be broken

This message is sent by the application to the MIF API to indicate that the application can tolerate the connection being broken. This is a signal that the application could use the connection in the future if it were not broken, but can re-establish the connection if it is broken without any loss of functionality. A MIF API implementation on a power-conservative device might take this as a signal to shut down radios to conserve power.

3.2.7.26.4. Interface is going away

This message is sent by the MIF API to the application to indicate that an interface is going away. This can happen when the interface is still up but the system intends to take it down.

3.2.7.26.5. Interface is going up

This message is sent by the MIF API to the application to indicate that an interface is going up. This can happen when the interface is still down but the system intends to take it up.

3.3. Example Usage

```
                +-------+                                  +-------+
                |  APP  |                                  |  API  |
                +-------+                                  +-------+
                    |          Announce Interfaces                |
                    |--------------------------------------------->|
                    |          Interface 1, eth0                  |
                    |<---------------------------------------------|
                    |          Announce PDs on Interface 1        |
                    |--------------------------------------------->|
                    |          PD 1                               |
                    |<---------------------------------------------|
                    |          Interface 2, wa0                   |
                    |<---------------------------------------------|
                    |          PD 2                               |
                    |<---------------------------------------------|
                    |          Announce PDs on Interface 2        |
                    |--------------------------------------------->|
                    |          PD 3                               |
                    |DNS query 2001::1, host.example.com A,AAAA   |
                    |DNS query 192.168.1.1,host.example.com A,AAAA|
                    |DNS query 2001::1, host.example.com A,AAAA   |
                    |--------------------------------------------->|
                    |14. 2001::1 DNS response:                    |
                    |    host.example.com                         |
                    |    IN A 14.15.16.17                         |
                    |    IN AAAA 2001:192:321::1                  |
                    |                                             |
                    |    2002::1 DNS response:...                 |
                    |    192.168.1.1 DNS response:                |
                    |    IN A 192.168.1.1                         |
                    |<---------------------------------------------|
                    | 15. SYN: 14.15.16.17 @ IF1                  |
                    |     SYN: 2001:192:321::1 @ IF1              |
                    |     SYN: 2001:192:321::1 @ IF2              |
                    |     SYN: 192.168.1.1 @ IF1                  |
                    |--------------------------------------------->|
                    | 16. SYN+ACK @ 192.168.1.1  IF1              |
                    |     SYN+ACK @ 2001:192:321::1  IF2          |
                    |     SYN+ACK @ 2001:192:321::1  IF1          |
                    |<---------------------------------------------|
                    |                                             |
```

MIF API communication model

As shown in the preceding example, the application first invokes the MIF API to get a list of all the network interfaces in the host. As

soon as each interface has been identified, the application invokes the MIF API to get a list of provisioning domains that are attached to that interface.

The application then invokes the MIF API to look up a name in the context of each provisioning domain. The name lookup may return more than one IP address for each queried host name.

The The application then tries to connect to each such IP addresses by sending tcp SYN packet to each destination IP addresses through the provisioning domain on which it received that name. Some of the destination IP addresses may return an ACK packet; others may not.

The application then chooses a connection based on its preferred criteria. For example, the criteria may based on the quality of the link, who answered first, or whether, for example, a TLS authentication succeeds on that connection.

4. Security Considerations

This document specifies an abstract API and will not affect any existing protocols. It does not introduce any new security risk.

5. IANA Considerations

None

6. Acknowledgments

The authors want to thank Teemu Savolainen from Nokia, Dayi Zhao from Bitway, Dave Thaler from Microsoft and others for their useful suggestions and discussions. We would also like to acknowledge Yuri Ismailov’s work as the author of the initial version of this document, but was drawn away by other work and let us continue.

7. References

7.1. Normative References

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

7.2. Informative References

[I-D.ietf-mif-mpvd-arch] Anipko, D., "Multiple Provisioning Domain Architecture", draft-ietf-mif-mpvd-arch-00 (work in progress), February 2014.

[I-D.scharf-mptcp-api]
Scharf, M. and A. Ford, "MPTCP Application Interface Considerations", draft-scharf-mptcp-api-02 (work in progress), July 2010.

[RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, February 2003.

Authors' Addresses

Dapeng Liu
China Mobile
Unit2, 28 Xuanwumenxi Ave,Xuanwu District
Beijing 100053
China

Email: liudapeng@chinamobile.com

Ted Lemon
Nominum
Redwood City
CA 94063
USA

Email: Ted.Lemon@nominum.com

Yuri Ismailov
Ericsson
Stockholm
Sweden
USA

Email: yuri@ismailov.eu

Zhen Cao
China Mobile
Unit2, 28 Xuanwumenxi Ave,Xuanwu District
Beijing 100053
China

Email: caozhen@chinamobile.com

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: May 17, 2017

G. Chen
China Mobile
C. Williams
Consultant
D. Wing
A. Yourtchenko
Cisco Systems, Inc.
November 13, 2016

Happy Eyeballs Extension for Multiple Interfaces
draft-ietf-mif-happy-eyeballs-extension-11

Abstract

This memo proposes extensions to the Happy Eyeball's algorithm requirements defined in RFC6555 for use with the multiple provisioning domain architecture. The Happy Eyeballs in MIF would make the selection process smoother by using connectivity tests over pre-filtered interfaces according to defined policy. This would choose the best interface with an automatic fallback mechanism.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 17, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 3
2. Terminology 3
3. Use Cases 3
3.1. WiFi is broken 3
3.2. Policy Conflict 4
4. Happiness Parameters 4
4.1. Hard Set 5
4.1.1. Operator Policy 5
4.1.2. User Preference 5
4.2. Soft Set 6
4.2.1. Provisioning Domain Identity 6
4.2.2. DNS Selection 6
4.2.3. Next Hop 6
4.2.4. Source Address Selection 6
4.2.5. Common Practice 6
5. HE-MIF Process Requirements 7
5.1. First Step, Filter 7
5.2. Second Step, Sort 8
5.2.1. Interface Validation 8
5.2.2. Name Resolution 8
5.2.3. Connection Establishment 8
6. Implementation Framework 9
7. Additional Considerations 9
7.1. Usage Scope 9
7.2. Fallback Timeout 9
7.3. DNS Selections 10
7.4. Flow Continuity 11
7.5. Interworking with Happy Eyeball 11
7.6. Multipath Applicability 11
8. IANA Considerations 11
9. Security Considerations 12
10. Acknowledgements 12
11. References 12
11.1. Normative References 12
11.2. Informative References 13
Authors' Addresses 14

1. Introduction

The MIF problem statement [RFC6418] describes problems specific for nodes attached to multiple provisioning domains. Specifically, there is a issue description that a node has selected an interface and obtained a valid IP address from the network, but Internet connectivity is not available. This memo intends to address the issue and elaborate more in Section 3.1.

[RFC7556] describes the multiple provisioning domain architecture. It refers to using connectivity tests to validate a Provisioning Domain (PvD). Given a number of implicit/explicit PvDs, plus preferences/policy, what is the process to follow to select the best PvD to use for any given connection. In the event that two or more are deemed to be best, how are the Happy Eyeballs (HE) techniques applied to find the best and deal with resilience. This memo also proposes process requirements using Happy Eyeballs (HE) extensions.

There are a variety of algorithms that can be envisioned. This document describes additional parameters and processes that need to be considered in addition to the HE algorithm requirements defined in [RFC6555] necessary to support multiple interfaces, so that a node with multiple interfaces can select the best path for a particular connection-oriented flow (e.g., TCP, SCTP).

2. Terminology

This document makes use of following terms:

- o Happy Eyeballs (HE): specifies requirements for an algorithm that reduces the user-visible connection delay for dual-stack hosts with a single interface per-protocol.

- o Happy Eyeballs - Multi-Interface (HE-MIF): Extends the Happy Eyeballs concept to the multiple provisioning domain architecture. It describes additional requirements for algorithms that offer connectivity tests on PVD-aware or non-PVD-aware nodes [RFC7556] to select the best interface for a specific connection request.

3. Use Cases

The section describes scenarios the HE-MIF targeted to use.

3.1. WiFi is broken

Assuming a MIF node has both a 3GPP mobile network interface and a WiFi interface, a common practice would be to always prefer the WiFi connection when the node enters an area with WiFi available. In this

situation, a node might assume that because a valid IP address has been allocated, the WiFi link provides connectivity to destinations through the Internet. However, this might not be the case for several reasons:

- o WiFi access-point authentication requirements
- o WiFi has no global Internet connectivity
- o Instability at layer 2

In order to resolve this problem, the user would need to disable the device’s interface preferences, e.g. by disabling the WiFi interface. HE-MIF offers users the possibility of configuring their preferences for the choice of the most suitable network interface to use, such as via setting on their mobile phone.

In this case, users may prefer to wait an appropriate time period for connections to be established over a WiFi path. If no connection can be made it will fall back to attempting the connection over a 3GPP mobile network path.

3.2. Policy Conflict

A node has network access via both WiFi and 3GPP networks. In a mobile network, IPv6-only may be preferable since IPv6 has the potential to be simpler than dual-stack. The WiFi access offers IPv4 only. In this scenario, the combination of source address selection [RFC6724] and preferring the WiFi interface may cause a problem. The transition to IPv6 may mean that IPv6 is the preferred protocol, so the 3GPP interface should be chosen even though it could be considered a suboptimal selection e.g. the WiFi interface likely is less expensive.

4. Happiness Parameters

This section provides input parameter proposal that HE-MIF should catch. Two sets of "Happiness" parameters have been defined. It serves applications and initiates HE-MIF connection tests subsequently. By following the process described below, MIF nodes can select an appropriate interface that best meets the configuration parameters defined by the user. The two sets of "Happiness" parameters are called Hard Set and Soft Set respectively.

4.1. Hard Set

Hard set contains parameters which should be complied with. It helps to select candidate interfaces through which a particular flow should be directed. These should be seen as constraints on the choice, such as provider policies, support for IPv4 or IPv6, and other parameters which would prevent a particular interface and transport from being used by a particular flow. Parameters in the hard set should be easy to use and understand. When several parameters in the hard set are in conflict, the user’s preference should be prioritized.

4.1.1. Operator Policy

Operators may deliver the customized policies for a particular network environment because of geo-location or service regulation considerations. One example relevant for 3GPP networks is an operator delivering policies from an Access Network Discovery and Selection function (ANDSF) [TS23.402].

The ANDSF provides a node with policies and network selection information to influence the selection between different access technologies, such as 3GPP mobile networks, WiFi access. The ANDSF can provide the node with three types of information[TS24.302].

- o Access network discovery and selection information: it includes a list of access networks available in the vicinity of the node. The information may include the access technology types (e.g. WiFi), network identifiers (e.g. SSID in the case of WiFi) as well as validity conditions (e.g. where and when).

- o Inter-System Mobility Policies (ISMPs): they are a set of operator-defined rules and preferences that affect the inter-system mobility decisions, e.g. decisions about whether to use 3GPP mobile network or a WiFi network.

- o Inter-System Routing Policies (ISRPs): the node uses ISRPs when it can route IP traffic simultaneously over multiple radio access networks. It could provide routing policies in an IP flow granularity.

4.1.2. User Preference

User’s preference: users may express preferences which likely not have a formally technical language , like "No 3/4G while roaming", "Only download applications larger than 20Mb over WiFi", etc. Those information are normally input from User Interface (UI).

4.2. Soft Set

Soft set contains factors which impact the selection of the path across which a particular flow should be transmitted among the available interfaces and transports which meet the hard set requirements described above.

4.2.1. Provisioning Domain Identity

A PVD-aware node uses PvD Identity(PvD-ID) to select a PvD with a matching ID for special-purpose connection requests. The PvD-ID may be generated by the node implicitly or received from the network explicitly. for explicit PvDs, the node could take the parameter from PvD ID Option [I-D.ietf-mif-mpvd-id] via the configuration protocols ([I-D.ietf-mif-mpvd-dhcp-support] or [I-D.ietf-mif-mpvd-ndp-support]). A PVD-aware node may decide to use one preferred PVD or allow the use of multiple PVDs simultaneously for applications. The node behavior should be consistent with MPVD architecture [RFC7556].

4.2.2. DNS Selection

At the name service lookup step, the node has to choose a recursive DNS server to use. A HE-MIF node should take the parameter of RDNSS Selection DHCP Option [RFC6731] to select an interface for a particular namespace.

4.2.3. Next Hop

[RFC4191] allows the configuration of specific routes to a destination. A HE-MIF node should take the parameters of router preference and route information to identify the next hop.

4.2.4. Source Address Selection

For each destination, once the best next hop is found, the node should consider IP prefix and precedence parameter in policy table to select the best source address according to the rule defined in [RFC6724].

4.2.5. Common Practice

There is relevant common practice related to interface selection, e.g. Prefer WiFi over a 3GPP interface, if available. Such conventions should also be considered.

5. HE-MIF Process Requirements

An HE-MIF node may use the two sets of parameters as two steps in the interface selection process. The first step is to use the Hard Set to synthesize policies from different actors (e.g., users or network operators). These hard set parameters will provide a filter which will exclude not qualifying interfaces from any further consideration.

The second step is to influence how a node makes a connection when multiple interfaces still remain in the candidate list after first step. This is essentially sorting behavior. In the multiple provisioning domain architecture, a PVD aware node makes connectivity tests as described in Section 5.3 of [RFC7556]. A PVD agnostic node take other parameters apart from PVD-ID in the Soft Set to proceed the sort process.

The two steps are described in more details in the following sub-sections. It should be noted that HE-MIF does not prescribe such two-step model. It will be very specific to particular cases and implementations. The two step model mainly describes requirements for how to use the hard/soft set.

5.1. First Step, Filter

One goal of the filter is to reconcile multiple selection policies from users or operators. Afterwards, merged demands would be mapped to a set of candidate interfaces, which are judged as qualified.

Decision on the reconciliation of different policies will depend very much on the deployment scenario. An implementation may not be able to determine priority for each policies without explicit configuration provided by users or administrator. For example, an implementation may by default always prefer the WiFi because of cost saving consideration. Whereas, other users may turn off a device’s WiFi interface to guarantee use of a 3GPP network interface to assure higher reliability or security.

The decision on mergence of policies may be made by implementations, or by node administrators. However, it’s worth to note that a demand from users should be normally considered higher priority than from other actors.

The merged policies serve as a filter which is iterated across the list of available interfaces. Qualified interfaces are selected and the proceed to the second step.

5.2. Second Step, Sort

5.2.1. Interface Validation

The Sort process aims to select the best interface and provide fallback capacities. As stated in [RFC7556], a PVD-aware node shall perform connectivity tests and, only after validation of the PVD, consider using it to serve application connections requests. In current implementations, some nodes already implement this, e.g., by trying to reach a dedicated web server (see Section 3.1.2 [RFC6419]). If anything is abnormal, it assumes there is a proxy on the path. This status detection is recommended to be used in HE-MIF to detect DNS interception or an HTTP proxy that forces a login or a click-through. Unexamined PVDs or interfaces should be accounted as "unconnected". It should not join the sort process.

5.2.2. Name Resolution

Name resolution is executed on the validated interfaces. Before the requests are initiated, it should check if there is a matching PVD ID for the destination name. A PVD agnostic node may request DNS server selection DHCP option [RFC6731] for interface selection guidance. Those information may weight a particular interface to be preferred to others sending resolving requests. If the node can’t find useful information in the Soft Set, DNS queries would be sent out on multiple interfaces in parallel to maximize chances for connectivity. Some additional discussions of DNS selection consideration of HE-MIF are described in Section 7.3.

5.2.3. Connection Establishment

Once a destination address was resolved, a connection is to be setup. For the given destination address, a PVD-aware node selects a next-hop and source address associated with that PVD in the name resolution process. A PVD agnostic node may receive certain next hop in a RA message [RFC4191], the node selects best source address according to the rules [RFC6724].

The interface identified by the source address should be treated to initiate the connection prior to others. This could avoid thrashing the network, by not making simultaneous connection attempts on multiple interfaces. After making a connection attempt on the preferred pairs and failing to establish a connection within a certain time period (see Section 7.2), a HE-MIF implementation will decide to initiate connection attempt using rest of interfaces in parallel. This fallback consideration will make subsequent connection attempts successful on non-preferable interfaces.

The node would cache information regarding the outcome of each connection attempt. Cache entries would be flushed periodically. A system-defined timeout may take place to age the state. Maximum on the order of 10 minutes defined in [RFC6555] is recommended to keep the interface state changes synchronizing with IP family states.

If there is no specific Soft Set provided, all selected interfaces should be treated equally. for a node implementing multipath transports (for example, Multipath TCP (MPTCP) [RFC6182]), the interfaces could be treated as valid to perform subsequent multipath process, such as starting subflow. A node only supporting single physical transport would initiate on several interface simultaneously. The goal here is to provide the most fast connection for users, by quickly attempting to connect using each candidate interface. Afterwards, the node would do the same caching and flushing process as described above.

6. Implementation Framework

The simplest way to implement the processes described in this document is within the application itself. This would not require any specific support from the operating system beyond the commonly available APIs that provide transport service. It could also be implemented using a high-level API approach, linking to the MIF-API [I-D.ietf-mif-api-extension].

7. Additional Considerations

7.1. Usage Scope

Connection-oriented transports (e.g., TCP, SCTP) are directly applied as scoped in [RFC6555]. For connectionless transport protocols (e.g., UDP), a similar mechanism can be used if the application has request/response semantics. Further investigations are out of the document scope.

7.2. Fallback Timeout

When the preferred interface was failed, HE-MIF would trigger a fallback process to start connection initiation on several candidate interfaces. A period of time should be set to invalidate the interface and fallback to others. Aggressive timeouts may achieve quick interface handover, but at the cost of traffic that may be chargeable on certain networks, e.g. the handover from WiFi to 3GPP networks brings a charge to customers. Considering the reasons, it is recommended to prioritize the input from users (e.g., real customers or applications) through user interface. For default-setting on a system, a hard error [RFC1122] in replied ICMP could

serve as a trigger for the fallback process. When the ICMP soft error is present or non-response was received, it’s recommended that the timeout should be large enough to allow connection retransmission. [RFC1122] states that such timer must be at least 3 minutes to provide TCP retransmission. However, several minutes delay may not inappropriate for user experiences. A widespread practice [RFC5461] sets 75 seconds to optimize connection process.

More optimal timer may be expected. The particular setting will be very specific to implementations and cases. The memo didn’t try to provide a concrete value because of following concerns.

- o RTT (Round-Trip Time) on different interfaces may vary quite a lot. A particular value of timeout may not accurately help to make a decision that this interface doesn’t work at all. On the contrary, it may cause a misjudgment on a interface, which is not very fast. In order to compensate the issues, the timeout setting based on past experiences of a particular interface may help to make a fair decision. Whereas, it’s going beyond the capability of Happy Eyeballs [RFC6555]. Therefore, it leaves a particular implementation.

- o In some cases, fast interface may not be treated as "best". For example, a interface could be evaluated in the principle of bandwidth-delay, termed "Bandwidth-Delay-Product ". Happy Eyeballs measures only connection speed. That is, how quickly a TCP connection is established . It does not measure bandwidth. If the fallback has to take various factors into account and make balanced decision, it’s better to resort to a specific context and implementation.

7.3. DNS Selections

During the Sort process, HE-MIF prioritizes PVD-ID match or [RFC6731] inputs to select a proper server. It could help to address following two cases.

- o A DNS answer may be only valid for a specific provisioning domain, but the DNS resolver may not be aware of that because the DNS reply is not kept with the provisioning from which the answer comes. The situation may become worse if asking internal name with public address response or asking public name with private address answers.

- o Some FQDNs can be resolvable only by sending queries to the right server (e.g., intranet services). Otherwise, a response with NXDOMAIN is replied. Fast response is treated as optimal only if

the record is valid. That may cause messy for data connections, since NXDOMAIN doesn’t provide useful information.

HE-MIF can help to solve the issues of DNS interception with captive portal. The DNS server modified and replied the answer with the IP address of captive portal rather than the intended destination address. In those cases, TCP connection may succeed, but Internet connectivity is not available. It results in lack of service unless user has authenticated. HE-MIF recommended using network connectivity status probes to examine a pre-configured URL for detecting DNS interception on the path (see more in Section 5.2). The node will be able to automatically rely upon other interfaces to select right DNS servers by excluding the unexamined interfaces.

7.4. Flow Continuity

[I-D.deng-mif-api-session-continuity-guide] describes session continuity guidance for application developers. The flow continuity topic is beyond this document scope.

7.5. Interworking with Happy Eyeball

HE-MIF process could cooperate with HE [RFC6555]. HE is executed on an interface which is selected to make connection establishment (see Section 5.2.3). for example, a node following PvD policy to pick a interface and make both IPv4/IPv6 connection attempts in consistent with HE requirements. The interface state management in HE-MIF is designed to synchronize with IP family states. It could facilitate the HE executions.

7.6. Multipath Applicability

Some nodes may support transports that provide an abstraction of a single connection, aggregating multiple underlying connections. Multipath TCP (MPTCP) [RFC6182] is an example of such a transport protocol. For connections provided by such transports, a node may leverage the "happiness" parameters and process on the underlying connections. Following the HE-MIF requirements, each connection could be performed consistently with user/operator’s preference and corresponding provisioning domain information.

8. IANA Considerations

This memo does not include any IANA requests.

9. Security Considerations

The security consideration is following the statement in [RFC6555] and [RFC6418].

10. Acknowledgements

The authors would like to thank Margaret Wasserman, Hui Deng, Erik Kline, Stuart Cheshire, Teemu Savolainen, Jonne Soininen, Simon Perreault, Zhen Cao, Dmitry Anipko, Ted Lemon, Daniel Migault, Russ White and Bing Liu for their helpful comments.

Many thanks to Ralph Droms, Ian Farrer, Jouni Korhonen, Mirja Khlewind and Suresh Krishnan for their detailed reviews.

11. References

11.1. Normative References

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <http://www.rfc-editor.org/info/rfc1122>.

[RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, DOI 10.17487/RFC4191, November 2005, <http://www.rfc-editor.org/info/rfc4191>.

[RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", RFC 6555, DOI 10.17487/RFC6555, April 2012, <http://www.rfc-editor.org/info/rfc6555>.

[RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <http://www.rfc-editor.org/info/rfc6724>.

[RFC6731] Savolainen, T., Kato, J., and T. Lemon, "Improved Recursive DNS Server Selection for Multi-Interfaced Nodes", RFC 6731, DOI 10.17487/RFC6731, December 2012, <http://www.rfc-editor.org/info/rfc6731>.

[TS23.402] 3rd Generation Partnership Project, 3GPP., "Architecture enhancements for non-3GPP accesses v8.8.0", December 2009.

[TS24.302]
3rd Generation Partnership Project, 3GPP., "Access to the 3GPP Evolved Packet Core (EPC) via non-3GPP access networks v14.0.0", June 2016.

11.2. Informative References

[I-D.deng-mif-api-session-continuity-guide]
Deng, H., Krishnan, S., Lemon, T., and M. Wasserman, "Guide for application developers on session continuity by using MIF API", draft-deng-mif-api-session-continuity-guide-04 (work in progress), July 2014.

[I-D.ietf-mif-api-extension]
Liu, D., Lemon, T., Ismailov, Y., and Z. Cao, "MIF API consideration", draft-ietf-mif-api-extension-05 (work in progress), February 2014.

[I-D.ietf-mif-mpvd-dhcp-support]
Krishnan, S., Korhonen, J., and S. Bhandari, "Support for multiple provisioning domains in DHCPv6", draft-ietf-mif-mpvd-dhcp-support-02 (work in progress), October 2015.

[I-D.ietf-mif-mpvd-id]
Krishnan, S., Korhonen, J., Bhandari, S., and S. Gundavelli, "Identification of provisioning domains", draft-ietf-mif-mpvd-id-02 (work in progress), October 2015.

[I-D.ietf-mif-mpvd-ndp-support]
Korhonen, J., Krishnan, S., and S. Gundavelli, "Support for multiple provisioning domains in IPv6 Neighbor Discovery Protocol", draft-ietf-mif-mpvd-ndp-support-03 (work in progress), February 2016.

[RFC5461] Gont, F., "TCP’s Reaction to Soft Errors", RFC 5461, DOI 10.17487/RFC5461, February 2009, <http://www.rfc-editor.org/info/rfc5461>.

[RFC6182] Ford, A., Raiciu, C., Handley, M., Barre, S., and J. Iyengar, "Architectural Guidelines for Multipath TCP Development", RFC 6182, DOI 10.17487/RFC6182, March 2011, <http://www.rfc-editor.org/info/rfc6182>.

[RFC6418] Blanchet, M. and P. Seite, "Multiple Interfaces and Provisioning Domains Problem Statement", RFC 6418, DOI 10.17487/RFC6418, November 2011, <http://www.rfc-editor.org/info/rfc6418>.

[RFC6419] Wasserman, M. and P. Seite, "Current Practices for Multiple-Interface Hosts", RFC 6419, DOI 10.17487/RFC6419, November 2011, <http://www.rfc-editor.org/info/rfc6419>.

[RFC7556] Anipko, D., Ed., "Multiple Provisioning Domain Architecture", RFC 7556, DOI 10.17487/RFC7556, June 2015, <http://www.rfc-editor.org/info/rfc7556>.

Authors' Addresses

Gang Chen
China Mobile
29, Jinrong Avenue
Xicheng District,
Beijing 100033
China

Email: phdgang@gmail.com, chengang@chinamobile.com

Carl Williams
Consultant
El Camino Real
Palo Alto, CA 94306
USA

Email: carlw@mcsr-labs.org

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

Email: dwing@cisco.com

Andrew Yourtchenko
Cisco Systems, Inc.
De Kleetlaan, 7
Diegem B-1831
Belgium

Email: ayourtch@cisco.com