

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: January 12, 2012

G. Chen
China Mobile
C. Williams
Consultant
July 11, 2011

Happy Eyeballs Extension for Multiple Interfaces
draft-chen-mif-happy-eyeballs-extension-02

Abstract

The memo has been proposed to extend happy eyeballs algorithm to fit into multiple interfaces environment. Based on this extended heuristic algorithm, a client with multiple interface could determine the optimal flow path in which specific interface has been chosen. Furthermore, an appropriate IP address family for each interface can be also identified to guarantee user experiences during IPv6 transition period.

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 January 12, 2012.

Copyright Notice

Copyright (c) 2011 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.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	3
2. Heuristic Happy Eyeballs Extension Algorithm	3
2.1. The Framework for Extended Algorithm	3
2.2. Interface Weighting Consideration	4
2.3. Interface Selection Process	5
2.4. IPv4/IPv6 Selection Algorithm for Individual Interface	6
3. Additional Considerations	6
3.1. Usage Scope	6
3.2. Flow Continuity	6
3.3. Default Address Selection	6
4. IANA Considerations	6
5. Security Considerations	6
6. Normative References	6
Authors' Addresses	7

1. Introduction

In multiple interface context, the problems raised by hosts with multiple interfaces have been discussed. The MIF problem statement[MIF-PS] has described the various issues when using a MIF node on which multiple interfaces are used and results in wrong domain selection. Happy Eyeballs [HAPPY-EYEBALLS] has described how a dual-stack client can determine the functioning path to a dual-stack server. It's using heuristic algorithm help applications to quickly determine if IPv6 or IPv4 is the most optimal to connect to a server. That is a good method to achieve intelligent path selection. However, the assumption here is single-homed host. The interaction with multiple interfaces is still waiting for further study.

This memo has been proposed to extend happy eyeballs algorithm to fit into multiple interfaces environment. That could achieve win-win situation. Based on this extended heuristic algorithm, a client with multiple interface could determine the optimal flow path in which specific interface has been chosen. Furthermore, an appropriate IP address family for each interface can be also identified to guarantee user experiences during IPv6 transition period.

2. Heuristic Happy Eyeballs Extension Algorithm

The section details extended Happy Eyeballs algorithm, including defined framework, interface weighting consideration and and computation process.

2.1. The Framework for Extended Algorithm

The Figure 1 shows the proposed framework for extended algorithm.

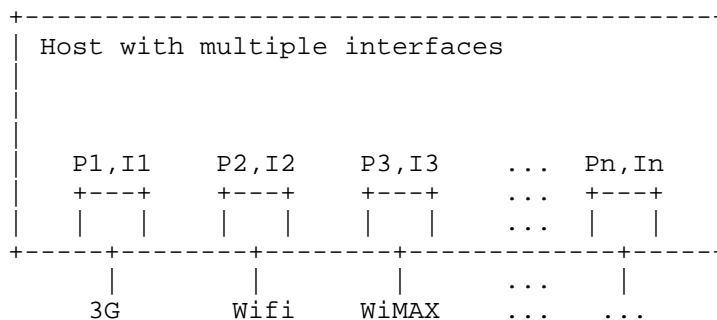


Figure 1: Multiple Interface Mode for Extended Algorithm

Each interface will be configured with weighting coefficient, which is composed of pair values. Apart from value P, which is following current definition in [HAPPY-EYEBALLS], value I is defined to indicate preference of interfaces selection. In general, value I is responsible for interface selection; value P is a indication to identify IPv4 or IPv6 family has been preferred.

2.2. Interface Weighting Consideration

According to the definition, applications will take account of value I to identify which interface has been chosen before sending out data packages .

Each interface is configured with one value, I. I is served as an indication to identify which interface is preferred for a specific destination or hostname. A positive value indicates preference of specific interface compared to others. The value is justified by taking various factors into account. The impact factors could be categorized in two groups.

- o Hard preconditions: It's mandatory indications that interface behaviour should comply with such preconditions guidance. The following factors belong to hard preconditions.
 - * Operator policies: operators would deliver the customized policies in particular network environments due to charging or area regulation considerations.
 - * User preferences: Users might configure to enable a specific interface to access network. for example, user may choose wifi interface to surf Internet considering low cost.
- o Soft preconditions: It's optimal choice for transmitting data packages through a specific interface compared to others. The following factors would contribute soft preconditions justification.
 - * Routing policies: DHCPv6 Route Option[draft-ietf-mif-dhcpv6-route-option-01] and RFC4191[RFC4191] allow configuration of specific routes and influence a nodes' ability to pick an appropriate route to a destination. A weighting for an interface headed to

destination address that matches a specific route would be increased.

- * DNS selection: if improved DNS server selection [draft-ietf-mif-dns-server-selection-03] takes effects, the weighting for those interfaces over which DNS suffix matching the requested name should also be increased.
- * Other factors: There are many other factors could contribute optimal interface selection. This documents would like to focus on the main ones and treat others in a best effort manner. The key factors are expected to be added in future discussion.

2.3. Interface Selection Process

The selection of a particular interface from the viable set implies a selection of one particular network path in preference to other viable paths. Interface weighting must be computed in advance but also be recomputed during session. The whole process for interface selection could be divided into two stages.

At stage I, upon the connection attempt, interface set should be filtered through the hard preconditions, and then aggregate the results within that kind of "policy group".

At stage II, the soft preconditions should be applied to the resulted interface set. According to particular soft preconditions, the preferred interface would be chosen by increasing I and delaying the connection attempts on the "undesirable" interfaces. This would allow to dial the preference between the different interfaces. The less desirable interface would get penalised a-priori.

To be specific, when one interface defeats others, the corresponding value I will be set to positive value. Other interfaces will be set negative value. A value of 0 indicates equal weight for multiple interfaces.

When interface values I have been configured, the traffic flow targeted to specific destination address or hostname will follow this guidance to choose proper interface. Hence, initial connection attempt would be sent over the interface that has matching particular rules and other interfaces would be tried only if no reply on the preferred one. Network condition may change during the session, interface reselection should be triggered. When connection problems are occurred to preferred connection, the value I need to be adjusted. The adjustment of value I will do polling-based scheme. The value I corresponding to suboptimal interface will be configured

as positive. And previously optimal value I will be set to most-negative.

2.4. IPv4/IPv6 Selection Algorithm for Individual Interface

For a specific interface in a dual-stack single interface node, the choice of IP address family relies on Happy Eyeballs algorithm, which is defined in [HAPPY-EYEBALLS].

3. Additional Considerations

3.1. Usage Scope

Happy Eyeballs is targeting to HTTP context, but it is useful and applicable to other time-sensitive applications.

3.2. Flow Continuity

Usually, interface changing happens at the beginning of new session. So, there is no flow continuity issues for ongoing TCP session. Dynamic movement of traffic flows are addressed by other IETF protocols as well.

3.3. Default Address Selection

If more than one IPv6 address is assigned to the interface, the native IPv6 address is given preference.

4. IANA Considerations

This memo includes no request to IANA.

5. Security Considerations

TBD

6. Normative References

[HAPPY-EYEBALLS]

Wing, D., "Happy Eyeballs: Success with Dual-Stack Hosts", draft-ietf-v6ops-happy-eyeballs-03.txt (work in progress), July 2011.

[MIF-PS] Blanchet, M., "Multiple Interfaces and Provisioning

Domains Problem Statement",
draft-ietf-mif-problem-statement-15.txt (work in
progress), May 2011.

[RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and
More-Specific Routes", RFC 4191, November 2005.

[draft-ietf-mif-dhcpv6-route-option-01]
Dec , W., "DHCPv6 Route Option",
draft-ietf-mif-dhcpv6-route-option-01 (work in progress),
March 2011.

[draft-ietf-mif-dns-server-selection-03]
Savolainen, T., "Improved DNS Server Selection for Multi-
Homed Nodes", draft-ietf-mif-dns-server-selection-03.txt
(work in progress), June 2011.

Authors' Addresses

Gang Chen
China Mobile
53A,Xibianmennei Ave.,
Xuanwu District,
Beijing 100053
China

Email: chengang@chinamobile.com

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

Email: carlw@mcsr-labs.org

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: January 29, 2012

M. Wasserman
Painless Security, LLC
P. Seite
France Telecom - Orange
July 28, 2011

Current Practices for Multiple Interface Hosts
draft-ietf-mif-current-practices-12

Abstract

An increasing number of hosts are operating in multiple-interface environments. This document summarizes current practices in this area, and describes in detail how some common operating systems cope with challenges ensue from this context.

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 January 29, 2012.

Copyright Notice

Copyright (c) 2011 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. Summary of Current Approaches	3
2.1. Centralized Connection Management	3
2.2. Per Application Connection Settings	4
2.3. Stack-Level Solutions to Specific Problems	4
2.3.1. DNS Resolution Issues	5
2.3.2. First hop selection	5
2.3.3. Address Selection Policy	5
3. Current Practices in Some Operating Systems	6
3.1. Mobile Handset Operating Systems	6
3.1.1. Nokia S60 3rd Edition, Feature Pack 2	7
3.1.2. Microsoft Windows Mobile and Windows Phone 7	9
3.1.3. RIM BlackBerry	10
3.1.4. Google Android	11
3.1.5. Qualcomm Brew	12
3.1.6. Leadcore Tech. Arena	14
3.2. Desktop Operating Systems	14
3.2.1. Microsoft Windows	14
3.2.1.1. First hop selection	14
3.2.1.2. Outbound and Inbound Addresses	14
3.2.1.3. DNS Configuration	15
3.2.2. Linux and BSD-based Operating Systems	16
3.2.2.1. First hop selection	16
3.2.2.2. Outbound and Inbound Addresses	17
3.2.2.3. DNS Configuration	17
4. Acknowledgements	18
5. IANA Considerations	19
6. Security Considerations	19
7. Contributors	19
8. References	20
8.1. Normative References	20
8.2. Informative References	20
Authors' Addresses	22

1. Introduction

Multiple-interface hosts face several challenges not faced by single-interface hosts, some of which are described in the MIF problem statement, [I-D.ietf-mif-problem-statement]. This document summarizes how current implementations deal with the problems identified in the MIF problem statement.

Publicly-available information about the multiple-interface solutions implemented in some widely used operating systems, including both mobile handset and desktop operating systems, is collected in this document, including: Nokia S60 [S60], Microsoft Windows Mobile [WINDOWSMOBILE], Blackberry [BLACKBERRY], Google Android [ANDROID], Microsoft Windows, Linux and BSD-based operating systems.

2. Summary of Current Approaches

This section summarizes current approaches that are used to resolve the multi-interface issues described in the Multiple Interface Problem Statement [I-D.ietf-mif-problem-statement]. These approaches can be broken down into three major categories:

- o Centralized connection management
- o Per-application connection settings
- o Stack-level solutions to specific problems

2.1. Centralized Connection Management

It is a common practice for mobile handset operating systems to use a centralized connection manager that performs network interface selection based on application or user input. However, connection managers usually restrict the problem to the selection of the interface and do not cope with selection of the provisioning domain, as defined in [I-D.ietf-mif-problem-statement]. The information used by the connection manager may be programmed into an application or provisioned on a handset-wide basis. When information is not available to make an interface selection, the connection manager will query the user to choose between available choices.

Routing tables are not typically used for network interface selection when a connection manager is in use, as the criteria for network selection is not strictly IP-based but is also dependent on other properties of the interface (cost, type, etc.). Furthermore, multiple overlapping private IPv4 address spaces are often exposed to a multiple-interface host, making it difficult to make interface

selection decisions based on prefix matching.

2.2. Per Application Connection Settings

In mobile handsets, applications are often involved in choosing what interface and related configuration information should be used. In some cases, the application selects the interface directly, and in other cases the application provides more abstract information to a connection manager that makes the final interface choice.

2.3. Stack-Level Solutions to Specific Problems

In most desktop operating systems, multiple interface problems are dealt with in the stack and related components, based on system-level configuration information, without the benefit of input from applications or users. These solutions tend to map well to the problems listed in the problem statement:

- o DNS resolution issues
- o Routing
- o Address selection policy

The configuration information for desktop systems comes from one of the following sources: DHCP, router advertisements, proprietary configuration systems or manual configuration. While these systems universally accept IP address assignment on a per-interface basis, they differ in what set of information can be assigned on a per-interface basis and what can be configured only on a per-system basis.

When choosing between multiple sets of information provided, these systems will typically give preference to information received on the "primary" interface. The mechanism for designating the "primary" interface differs by system.

There is very little commonality in how desktop operating systems handle multiple sets of configuration information, with notable variations between different versions of the same operating system and/or within different software packages built for the same operating system. Although these systems differ widely, it is not clear that any of them provide a completely satisfactory user experience in multiple-interface environments.

The following sections discuss some of the solutions used in each of the areas raised in the MIF problem statement.

2.3.1. DNS Resolution Issues

There is very little commonality in how desktop operating systems handle the DNS server list. Some systems support per-interface DNS server lists, while others only support a single system-wide list.

On hosts with per-interface DNS server lists, different mechanisms are used to determine which DNS server is contacted for a given query. In most cases, the first DNS server listed on the "primary" interface is queried first, with back off to other servers if an answer is not received.

Systems that support a single system-wide list differ in how they select which DNS server to use in cases where they receive more than one DNS server list to configure (e.g. from DHCP on multiple interfaces). Some accept the information received on the "primary" interface, while others use either the first or last set DNS server list configured.

2.3.2. First hop selection

Routing information is also handled differently on different desktop operating systems. While all systems maintain some sort of routing cache, to handle redirects and/or statically configured routes, most packets are routed based on configured default gateway information.

Some systems do allow the configuration of different default router lists for different interfaces. These systems will always choose the default gateway on the interface with the lowest routing metric, with different behavior when two or more interfaces have the same routing metric.

Most systems do not allow the configuration of more than one default router list, choosing instead to use the first or last default router list configured and/or the router list configured on the "primary" interface.

2.3.3. Address Selection Policy

There is somewhat more commonality in how desktop hosts handle address selection. Applications typically provide the destination address for an outgoing packet, and the IP stack is responsible for picking the source address.

IPv6 specifies a specific source address selection mechanism in [RFC3484], and several systems implement this mechanism with similar support for IPv4. However, many systems do not provide any mechanism to update this default policy, and there is no standard way to do so.

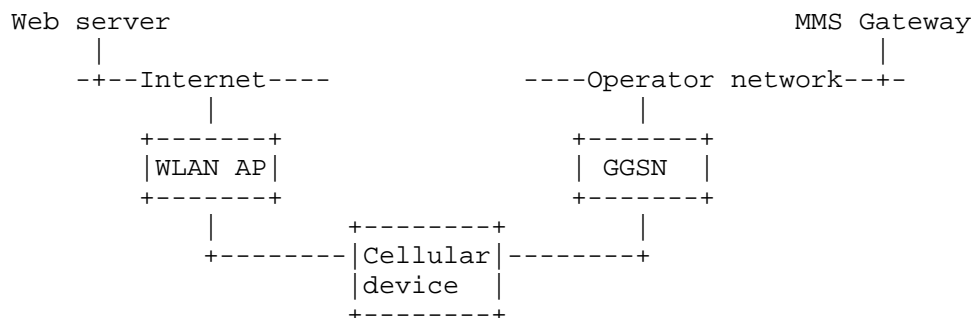
In some cases, the routing decision (including which interface to use) is made before source address selection is performed, and a source address is chosen from the outbound interface. In other cases, source address selection is performed before, or independently from outbound interface selection.

3. Current Practices in Some Operating Systems

The material presented in this section is derived from contributions from people familiar with the Operating Systems described, and those people are listed in Section 7. The authors and the IETF take no position about the Operating Systems described, and understand that other Operating Systems also exist. Furthermore, it should be understood that Section 3 describes particular behaviors that were believed to be current at the time of documentation: earlier and later versions of the Operating Systems described may exhibit different behaviors. Please refer to the References section for pointers to original documentation, including further details.

3.1. Mobile Handset Operating Systems

Cellular devices typically run a variety of applications in parallel, each with different requirements for IP connectivity. A typical scenario is shown in figure 1, where a cellular device is utilizing WLAN access for web browsing and GPRS access for transferring multimedia messages (MMS). Another typical scenario would be a real-time VoIP session over one network interface in parallel with best effort web browsing on another network interface. Yet another typical scenario would be global Internet access through one network interface and local (e.g. corporate VPN) network access through another.



A cellular device with two network interfaces

Figure 1

Different network access technologies require different settings. For example, WLAN requires Service Set Identifier (SSID) and the GPRS network requires the Access Point Name (APN) of the Gateway GPRS Support Node (GGSN), among other parameters. It is common that different accesses lead to different destination networks (e.g. to "Internet", "intranet", cellular network services, etc.).

3.1.1.1. Nokia S60 3rd Edition, Feature Pack 2

S60 is a software platform for mobile devices running on the Symbian OS. S60 uses the concept of an Internet Access Point (IAP) [S60] that contains all information required for opening a network connection using a specific access technology. A device may have several IAPs configured for different network technologies and settings (multiple WLAN SSIDs, GPRS APNs, dial-up numbers, and so forth). There may also be 'virtual' IAPs that define parameters needed for tunnel establishment (e.g. for VPN).

For each application, a correct IAP needs to be selected at the point when the application requires network connectivity. This is essential, as the wrong IAP may not be able to support the application or reach the desired destination. For example, MMS application must use the correct IAP in order to reach the MMS Gateway, which typically is not accessible from the public Internet. As another example, an application might need to use the IAP associated with its corporate VPN in order to reach internal corporate servers. Binding applications to IAPs avoids several problems, such as choosing the correct DNS server in the presence of split DNS (as an application will use the DNS server list from its bound IAP), and overlapping private IPv4 address spaces used for different interfaces (as each application will use the default routes

from its bound IAP).

If multiple applications utilize the same IAP, the underlying network connection can typically be shared. This is often the case when multiple Internet-using applications are running in parallel.

The IAP for an application can be selected in multiple ways:

- o Statically: e.g. from a configuration interface, via client provisioning/device management system, or at build-time.
- o Manually by the user: e.g. each time an application starts the user may be asked to select the IAP to use. This may be needed, for example, if a user sometimes wishes to access his corporate intranet and other times would prefer to access the Internet directly.
- o Automatically by the system: after the destination network has been selected statically or dynamically.

The static approach is fine for certain applications, like MMS, for which configuration can be provisioned by the network operator and does not change often. Manual selection works, but may be seen as troublesome by the user. An automatic selection mechanism needs to have some way of knowing which destination network the user, or an application, is trying access.

S60 3rd Edition, Feature Pack 2, introduces a concept of Service Network Access Points (SNAPs) that group together IAPs that lead to the same destination. This enables static or manual selection of the destination network for an application and leaves the problem of selecting the best of the available IAPs within a SNAP to the operating system.

When SNAPs are used, the operating system can notify applications when a preferred IAP, leading to the same destination, becomes available (for example, when a user comes within range of his home WLAN access point), or when the currently used IAP is no longer available. If so, applications have to reconnect via another IAP (for example, when a user goes out of range of his home WLAN and must move to the cellular network).

In S60 3.2 does not support RFC 3484 for source address selection mechanisms. Applications are tightly bound the network interface selected for them or by them. E.g. an application may be connected to IPv6 3G connection, IPv4 3G connection, WLAN connection, or VPN connection. The application can change between the connections, but uses only one at a time. If the interface happens to be dual-stack,

then IPv4 is preferred over IPv6.

DNS configuration is per-interface; an application bound to an interface will always use the DNS settings for that interface. Hence the device itself remembers these pieces of information for each interface separately.

The S60 3.2 manages with totally overlapping addresses spaces. Each interface can even have same IPv4 address configured on it without issues. This is so because interfaces are kept totally separate from each other. This also implies that the interface selection has to be done at application layer, as from network layer point of view device is not multihomed in the IP-sense.

Please see the source documentation for more details and screenshots: [S60].

3.1.2. Microsoft Windows Mobile and Windows Phone 7

Microsoft Windows Mobile leverages on a Connection Manager [WINDOWSMOBILE] to handle multiple network connections. This architecture centralizes and automates network connection establishment and management, and makes it possible to automatically select a connection, to dial-in automatically or by user initiation, and to optimize connection and shared resource usage. Connection Manager periodically re-evaluates the validity of the connection selection. The Connection Manager uses various attributes such as cost, security, bandwidth, error rate, and latency in its decision making.

The Connection Manager selects the best possible connection for the application based on the destination network the application wishes to reach. The selection is made between available physical and virtual connections (e.g. VPN, GPRS, WLAN, and wired Ethernet) that are known to provide connectivity to the destination network, and the selection is based on the costs associated with each connection. Different applications are bundled to use the same network connection when possible, but in conflict situations when a connection cannot be shared, higher priority applications take precedence, and the lower priority applications lose connectivity until the conflict situation clears.

During operation, Connection Manager opens new connections as needed, and also disconnects unused or idle connections.

To optimize resource use, such as battery power and bandwidth, Connection Manager enables applications to synchronize network connection usage by allowing applications to register their

requirements for periodic connectivity. An application is notified when a suitable connection becomes available for its use.

In comparison to Windows Mobile connection management, Windows phone 7 updates the routing functionality in the case where the terminal can be attached simultaneously to several interfaces. Windows Phone 7 selects the first hop corresponding to the interface which has a lower metric. When there are multiple interfaces, the applications system will, by default, choose from an ordered list of available interfaces. The default connection policy will prefer wired over wireless and WLAN over cellular. Hence, if an application wants to use cellular 3G as the active interface when WLAN is available, the application needs to override the default connection mapping policy. An application specific mapping policy can be set via a microsoft API or provisioned by the Mobile Operator. The application, in compliance with the security model, can request connection type by interface (WLAN, cellular), by minimum interface speed (x kbps, y mbps), or by name (Access Point Name).

In dual-stack systems, Windows mobile and Windows phone 7 implement address selection rules as per [WNDS-RFC3484]. An administrator can configure a policy table that can override the default behavior of the selection algorithms. It is reminded that the policy table specifies precedence values and preferred source prefixes for destination prefixes (see [RFC3484], section 2.1, for details). If the system has not been configured, then the default policy table specified in [RFC3484] is used.

3.1.3. RIM BlackBerry

Depending on the network configuration, applications in Research In Motion (RIM) BlackBerry devices [BLACKBERRY] can use direct TCP/IP connectivity or different application proxys to establish connections over the wireless network. For instance, some wireless service providers provide an Internet gateway to offer direct TCP/IP connectivity to the Internet while some others can provide a WAP gateway that allows HTTP connections to occur over the WAP (Wireless Application Protocol) protocol. It is also possible to use the BlackBerry Enterprise Server [BLACKBERRY] as a network gateway, The BlackBerry Enterprise Server provides an HTTP and TCP/IP proxy service to allow the application to use it as a secure gateway for managing HTTP and TCP/IP connections to the intranet or the Internet. An application connecting to the Internet, can use either the BlackBerry Internet Service or the Internet gateway of the wireless server provider or direct Internet connectivity over WLAN to manage connections. The problem of gateway selection is supposed to be managed independently by each application. For instance, an application can be designed to always use the default Internet

gateway, while another application can be designed to use a preferred proxy when available.

A BlackBerry device [BLACKBERRY] can be attached to multiple networks simultaneously (wireless/wired). In this case, Multiple network interfaces can be associated to a single IP stack or multiple IP stacks. The device, or the application, can select the network interface to be used in various ways. For instance, the device can always map the applications to the default network interface (or the default access network). When multiple IP stacks are associated to multiple interfaces, the application can select the source address corresponding to the preferred network interface. Per-interface IP stacks also allow to manage overlapping addresses spaces. When multiple network interfaces are aggregated into a single IP stack, the device associates each application to the more appropriate network interface. The selection can be based on cost, type-of-service and/or user preference.

The BlackBerry uses per-interface DNS configuration; applications bound to a specific interface will use the DNS settings for that interface.

3.1.4. Google Android

Android is based on a Linux kernel and, in many situations, behaves like a Linux device as described in Section 3.2.2. As per Linux, Android can manage multiple routing tables and rely on policy based routing associated with packet filtering capabilities (see Section 3.2.2.1 for details). Such a framework can be used to solve complex routing issue brought by multiple interfaces terminals, e.g. address space overlapping.

For incoming packets, Android implements the weak host model [RFC1122] on both IPv4 and IPv6. However, Android can also be configured to support the strong host model.

Regarding DNS configuration, Android does not list the DNS servers in the file /etc/resolv.conf, used by Linux. However, as per Linux, DNS configuration is node-scoped, even if DNS configuration can rely on the DHCP client. For instance, the udhcp client [UDHCP], which is also available for Linux, can be used on Android. Each time new configuration data is received by the host from a DHCP server, regardless of which interface it is received on, the DHCP client rewrites the global configuration data with the most recent information received.

Actually, the main difference between Linux and Android is on the address selection mechanism. Android version prior to 2.2 simply

prefers IPv6 connectivity over IPv4. However, it should be noted that, at the time of writing, IPv6 is available only on WiFi and virtual interfaces, but not on the cellular interface (without IPv6 in IPv4 encapsulation). Android 2.2 has been updated with [ANDROID-RFC3484], which implements some of the address selection rules defined in [RFC3484]. All RFC3484 rules are supported, except rule 3 (avoid deprecated addresses), 4 (prefer home addresses) and 7 (prefer native transport). Also, rule 9 (use longest matching prefix) has been modified so it does not sort IPv4 addresses.

The Android reference documentation describes the `android.net` package [ANDROID] and the `ConnectivityManager` class that applications can use to request the first hop to a specified destination address via a specified network interface (3GPP or WLAN). Applications also ask Connection Manager for permission to start using a network feature. The Connectivity Manager monitors changes in network connectivity and attempts to failover to another network if connectivity to an active network is lost. When there are changes in network connectivity, applications are notified. Applications are also able to ask for information about all network interfaces, including their availability, type and other information.

3.1.5. Qualcomm Brew

This section describes how multi-interface support is handled by Advanced Mobile Station Software (AMSS) that comes with Brew OS for all Qualcomm chipsets (e.g., MSM, Snapdragon etc). AMSS is a low level connectivity platform, on top of which manufacturers can build to provide the necessary connectivity to applications. The interaction model between AMSS, the Operating System, and the applications is not unique and depend on the design chosen by the manufacturer. The Mobile OS can let an application invoke the AMSS directly (via API), or provide its own connection manager that will request connectivity to the AMSS based on applications needs. The interaction between the OS connection manager and the applications is OS dependent.

AMSS supports a concept of netpolicy which allows each application to specify the type of network connectivity desired. The netpolicy contains parameters such as access technology, IP version type and network profile. Access technology could be a specific technology type such as CDMA or WLAN or could be a group of technologies, such as ANY_Cellular or ANY_Wireless. IP version could be one of IPv4, IPv6 or Default. The network profile identifies a type of network domain or service within a certain network technology, such as 3GPP APN or Mobile IP Home Agent. It also specifies all the mandatory parameters required to connect to the domain such authentication credentials and other optional parameters such as QoS attributes.

Network Profile is technology specific and set of parameters contained in the profile could vary for different technologies.

Two models of network usage are supported:

- o Applications requiring network connectivity specify an appropriate netpolicy in order to select the desired network. The netpolicy may match one or more network interfaces. AMSS system selection module selects the best interface out of the ones that match the netpolicy based on various criteria such as cost, speed or other provisioned rules. Application explicitly starts the selected network interface and, as a result, the application also gets bound to the corresponding network interface. All outbound packets from this application are always routed over this bound interface using the source address of the interface.
- o Applications may rely on a separate connection manager to control (e.g. start/stop) the network interface. In this model, applications are not necessarily bound to any one interface. All outbound packets from such applications are routed on one of the interfaces that match its netpolicy. The routing decision is made individually for each packet and selects the best interface based on the criteria described above and the destination address. Source address is always that assigned to the interface used to transmit the packet.

All of the routing/interface selection decisions are based on the netpolicy and not just on the destination address to avoid overlapping private IPv4 address issue. This also allows multiple interfaces to be configured with the same IP address, for example, to handle certain tunnelling scenarios. Applications that do not specify a netpolicy are routed by AMSS to the best possible interface using the default netpolicy. Default netpolicy could be pre-defined or provisioned by the administrator or operator. Hence default interface could vary from device to device and also depends upon the available networks at any given time.

AMSS allows each interface to be configured with its own set of DNS configuration parameters (e.g. list of DNS servers, domain names etc.). Interface selected to make a DNS resolution is the one to which application making the DNS query is bound. Applications can also specify a different netpolicy as part of DNS request to select another interface for DNS resolution. Regardless, all the DNS queries are sent only over this selected interface using the DNS configuration from the interface. DNS resolution is first attempted with the primary server configured in the interface. If a response is not received, the queries are sent to all the other servers configured in the interface in a sequential manner using a backoff

mechanism.

3.1.6. Leadcore Tech. Arena

Arena, a mobile OS based on Linux, provides a Connection Manager, which is described in [I-D.zhang-mif-connection-manager-arena] and [I-D.yang-mif-connection-manager-impl-req]. The arena connection manager provides a means for applications to register their connectivity requirement. The Connection Manager can then choose an interface that matches the application's needs while considering other factors such as availability, cost and stability. Also, the Connection Manager can handle multiple-interfaces issues such as connection sharing.

3.2. Desktop Operating Systems

Multi-interface issues also occur in desktop environments in those cases where a desktop host has multiple (logical or physical) interfaces connected to networks with different reachability properties, such as one interface connected to the global Internet, while another interface is connected to a corporate VPN.

3.2.1. Microsoft Windows

The multi-interface functionality currently implemented in Microsoft Windows operation systems is described in more detail in [I-D.montenegro-mif-multihoming].

3.2.1.1. First hop selection

It is possible, although not often desirable, to configure default routers on more than one Windows interface. In this configuration, Windows will use the default route on the interface with the lowest routing metric (i.e. the fastest interface). If multiple interfaces share the same metric, the behavior will differ based on the version of Windows in use. Prior to Windows Vista, the packet would be routed out of the first interface that was bound to the TCP/IP stack, the preferred interface. In Windows vista, host-to-router load sharing [RFC4311] is used for both IPv4 and IPv6.

3.2.1.2. Outbound and Inbound Addresses

If the source address of the outgoing packet has not been determined by the application, Windows will choose from the addresses assigned to its interfaces. Windows implements [RFC3484] for source address selection in IPv6 and, in Windows Vista, for IPv4. Prior to Windows Vista, IPv4 simply chose the first address on the outgoing interface.

For incoming packets, Windows will check if the destination address matches one of the addresses assigned to its interfaces. Windows has implemented the weak host model [RFC1122] on IPv4 in Windows 2000, Windows XP and Windows Server 2003. The strong host model became the default for IPv4 in Windows Vista and Windows server 2008, however the weak host model is available via per-interface configuration. IPv6 has always implemented the strong host model.

3.2.1.3. DNS Configuration

Windows largely relies on suffixes to solve DNS resolution issues. Suffixes are used for four different purposes that are reminded hereafter:

1. DNS Suffix Search List (aka domain search list): suffix is added to non-FQDN names.
2. Interface-specific suffix list, which allows sending different DNS queries to different DNS servers.
3. Suffix to control Dynamic DNS Updates: determine which DNS server will receive a dynamic update for a name with a certain suffix.
4. Suffix in the Name Resolution Policy Table [NRPT] to aid in identifying a Namespace that requires special handling (feature available only after Windows 7 and its server counterpart, Windows Server 2008 R2).

However, this section focuses on the interface-specific suffix list since it is the only suffix usage in the scope of this document.

DNS configuration information can be host-wide or interface specific. Host-wide DNS configuration is input via static configuration or, in sites that use Active Directory, Microsoft's Group Policy. Interface specific DNS configuration can be input via static configuration or via DHCP.

The host-wide configuration consists of a primary DNS suffix to be used for the local host, as well as a list of suffix that can be appended to names being queried. Before Windows Vista and Windows Server 2008, there was also a host-wide DNS server list that took precedent over per-interface DNS configuration.

The interface-specific DNS configuration comprises an interface-specific suffix list and a list of DNS server IP addresses.

Windows uses a host-wide "effective" server list for an actual query, where the effective server list may be different for different names.

In the list of DNS server addresses, the first server is considered the "primary" server, with all other servers being secondary.

When a DNS query is performed in Windows, the query is first sent to the primary DNS server on the preferred interface. If no response is received in one second, the query is sent to the primary DNS servers on all interfaces under consideration. If no response is received for 2 more seconds, the DNS server sends the query to all of the DNS servers on the DNS server lists for all interfaces under consideration. If the host still doesn't receive a response after 4 seconds, it will send to all of the servers again and wait 8 seconds for a response.

3.2.2. Linux and BSD-based Operating Systems

3.2.2.1. First hop selection

In addition to the two commonly used routing tables (the local and main routing tables), the kernel can support up to 252 additional routing tables which can be added in the file `/etc/iproute2/rt_tables`. A routing table can contain an arbitrary number of routes, the selection of route is classically made according to the destination address of the packet. Linux also provides more flexible routing selection based on the Type of Service, scope, output interface. In addition, since kernel version 2.2, Linux supports policy based routing using the multiple routing tables capability and a routing policy database. This database contains routing rules used by the kernel. Using policy based routing, the source address, the ToS flags, the interface name and an "fwmark" (a mark carried through added in the data structure representing the packet) can be used as route selectors.

Policy based routing can be used in addition to Linux packet filtering capabilities, e.g provided by the "iptables" tool. In a multiple interfaces context, this tool can be used to mark the packets, i.e assign a number to fwmark, in order to select the routing rule according to the type of traffic. This mark can be assigned according to parameters like protocol, source and/or destination addresses, port number and so on.

Such a routing management framework allows to deal with complex situation such as address space overlapping. In this situation, the administrator can use packet marking and policy based routing to select the correct interface.

3.2.2.2. Outbound and Inbound Addresses

By default, source address selection follows the following basics rules: the initial source address for an outbound packet can be chosen by the application using the `bind()` call. Without information from the application, the kernel chooses the first address configured on the interface which belongs to the same subnet than the destination address or the nexthop router.

Linux also implements [RFC3484] for source address selection for IPv6 and dual-stack configurations. However, the address sorting rules from [RFC3484] are not always adequate. For this reason, Linux allows the system administrator to dynamically change the sorting. This can be achieved with the `/etc/gai.conf` file.

For incoming packets, Linux checks if the destination address matches one of the addresses assigned to its interfaces then, processes the packet according the configured host model. By default, Linux implements the weak host model [RFC1122] on both IPv4 and IPv6. However, Linux can also be configured to support the strong host model.

3.2.2.3. DNS Configuration

Most BSD and Linux distributions rely on their DHCP client to handle the configuration of interface-specific information (such as an IP address and netmask), and a set of system-wide configuration information, (such a DNS server list, an NTP server list and default routes). Users of these operating systems have the choice of using any DHCP client available for their platform, with an operating system default. This section discusses the behavior of several DHCP clients that may be used with Linux and BSD distributions.

The Internet Systems Consortium (ISC) DHCP Client [ISCDHCP] and its derivative for OpenBSD [OPENBSDDHCLIENT] can be configured with specific instructions for each interface. However, each time new configuration data is received by the host from a DHCP server, regardless of which interface it is received on, the DHCP client rewrites the global configuration data, such as the default routes and the DNS server list (in `/etc/resolv.conf`) with the most recent information received. Therefore, the last configured interface always become the primary one. The ISC DHCPv6 client behaves similarly. However, OpenBSD provides two mechanisms allowing to not overwrite the configuration that the user made manually:

- o `OPTION MODIFIERS` (default, supersede, prepend, and append): this mechanism allows the user to override the DHCP options. For example, the `supersede` statement defines, for some options, the

values the client should always use rather than any value supplied by the server.

- o `resolv.conf.tail`: it allows the user to append anything to the `resolv.conf` file created by the DHCP client.

The Phystech `dhcpcd` client [`PHYSTECHDHCPD`] behaves similarly to the ISC client. It replaces the DNS server list in `/etc/resolv.conf` and the default routes each time new DHCP information is received on any interface. However, the `-R` flag can be used to instruct the client to not replace the DNS servers in `/etc/resolv.conf`. However, this flag is a global flag for the DHCP server, and is therefore applicable to all interfaces. When `dhcpcd` is called with the `-R` flag, the DNS servers are never replaced.

The `pump` client [`PUMP`] also behaves similarly to the ISC client. It replaces the DNS servers in `/etc/resolv.conf` and the default routes each time new DHCP information is received on any interface. However, the `nodns` and `nogateway` options can be specified on a per interface basis, enabling the user to define which interface should be used to obtain the global configuration information.

The `udhcp` client [`UDHCP`] is often used in embedded platforms based on `busybox`. The `udhcp` client behaves similarly to the ISC client. It rewrites default routes and the DNS server list each time new DHCP information is received.

Redhat-based distributions, such as Redhat, Centos and Fedora have a per-interface configuration option (`PEERDNS`) that indicates that the DNS server list should not be updated based on configuration received on that interface.

The most configurable DHCP clients can be set to define a primary interface to use only that interface for the global configuration data. However, this is limited, since a mobile host might not always have the same set of interfaces available. Connection managers may help in this situation.

Some distributions also have a connection manager. However, most connection managers serve as a GUI to the DHCP client, therefore not changing the functionality described above.

4. Acknowledgements

Authors of the document would like to thank following people for their input and feedback: Dan Wing, Hui Deng, Jari Arkko, Julien Laganier and Steinar H. Gunderson.

5. IANA Considerations

This memo includes no request to IANA.

6. Security Considerations

This document describes current operating system implementations and how they handle the issues raised in the MIF problem statement. While it is possible that the currently implemented mechanisms described in this document may affect the security of the systems described, this document merely reports on current practice. It does not attempt to analyze the security properties (or any other architectural properties) of the currently implemented mechanisms.

7. Contributors

The following people contributed most of the per-Operating System information found in this document:

- o Marc Blanchet, Viagenie
- o Hua Chen, Leadcoretech, Ltd.
- o Yan Zhang, Leadcoretech Ltd.
- o Shunan Fan, Huawei Technology
- o Jian Yang, Huawei Technology
- o Gabriel Montenegro, Microsoft Corporation
- o Shyam Seshadri, Microsoft Corporation
- o Dave Thaler, Microsoft Corporation
- o Kevin Chin, Microsoft Corporation
- o Teemu Savolainen, Nokia
- o Tao Sun, China Mobile
- o George Tsirtsis, Qualcomm.
- o David Freyermuth, France telecom.

- o Aurelien Collet, Altran.
- o Giyeong Son, RIM.

8. References

8.1. Normative References

[I-D.ietf-mif-problem-statement]
Blanchet, M. and P. Seite, "Multiple Interfaces and Provisioning Domains Problem Statement", draft-ietf-mif-problem-statement-15 (work in progress), May 2011.

8.2. Informative References

[ANDROID] Google Inc., "Android developers: package android.net", 2009, <<http://developer.android.com/reference/android/net/ConnectivityManager.html>>.

[ANDROID-RFC3484]
Gunderson, S., "RFC 3484 support for Android", 2010, <<http://gitorious.org/0xdroid/bionic/commit/9ab75d4cc803e91b7f1b656ffbe2ad32c52a86f9>>.

[BLACKBERRY]
Research In Motion Limited, "BlackBerry Java Development Environment - Fundamentals Guide: Wireless gateways", 2009, <http://na.blackberry.com/eng/deliverables/5827/Wireless_gateways_447132_11.jsp>.

[I-D.montenegro-mif-multihoming]
Montenegro, G., Thaler, D., and S. Seshadri, "Multiple Interfaces on Windows", draft-montenegro-mif-multihoming-00 (work in progress), March 2009.

[I-D.yang-mif-connection-manager-impl-req]
Yang, J., Sun, T., and S. Fan, "Multi-interface Connection Manager Implementation and Requirements", draft-yang-mif-connection-manager-impl-req-00 (work in progress), March 2009.

[I-D.zhang-mif-connection-manager-arena]
Zhang, Y., Sun, T., and H. Chen, "Multi-interface Network Connection Manager in Arena Platform", draft-zhang-mif-connection-manager-arena-00 (work in progress), March 2009.

progress), February 2009.

- [ISCDHCP] Internet Software Consortium, "ISC DHCP", 2009,
<<http://www.isc.org/software/dhcp>>.
- [NRPT] Windows, "Name Resolution Policy Table", February 2010, <
[http://technet.microsoft.com/en-us/magazine/
ff394369.aspx](http://technet.microsoft.com/en-us/magazine/ff394369.aspx)>.
- [OPENBSDDHCLIENT] OpenBSD, "OpenBSD dhclient", 2009,
<<http://www.openbsd.org/>>.
- [PHYSTECHDHCPD] Phystech, "dhcpcd", 2009,
<<http://www.phystech.com/download/dhcpcd.html>>.
- [PUMP] RedHat, "PUMP", 2009, <<http://redhat.com>>.
- [RFC1122] Braden, R., "Requirements for Internet Hosts -
Communication Layers", STD 3, RFC 1122, October 1989.
- [RFC3484] Draves, R., "Default Address Selection for Internet
Protocol version 6 (IPv6)", RFC 3484, February 2003.
- [RFC4311] Hinden, R. and D. Thaler, "IPv6 Host-to-Router Load
Sharing", RFC 4311, November 2005.
- [RFC5113] Arkko, J., Aboba, B., Korhonen, J., and F. Bari, "Network
Discovery and Selection Problem", RFC 5113, January 2008.
- [S60] Nokia Corporation, "S60 Platform: IP Bearer Management",
2007, <[http://www.forum.nokia.com/info/sw.nokia.com/id/
190358c8-7cbl-4be3-9321-f9d6788ecae5/
S60_Platform_IP_Bearer_Management_v1_0_en.pdf.html](http://www.forum.nokia.com/info/sw.nokia.com/id/190358c8-7cbl-4be3-9321-f9d6788ecae5/S60_Platform_IP_Bearer_Management_v1_0_en.pdf.html)>.
- [UDHCP] Busybox, "uDhcp", 2009, <[http://sources.busybox.net/
index.py/trunk/busybox/networking/udhcp/](http://sources.busybox.net/index.py/trunk/busybox/networking/udhcp/)>.
- [WINDOWSMOBILE] Microsoft Corporation, "SDK Documentation for Windows
Mobile-Based Smartphones: Connection Manager", 2005,
<<http://msdn.microsoft.com/en-us/library/aa457829.aspx>>.
- [WNDS-RFC3484] Microsoft Corporation, "SDK Documentation for Windows
Mobile-Based Smartphones: Default Address Selection for
IPv6", 2005,

<<http://msdn.microsoft.com/en-us/library/aa925716.aspx>>.

Authors' Addresses

Margaret Wasserman
Painless Security, LLC
356 Abbott Street
North Andover, MA 01845
USA

Phone: +1 781 405-7464
Email: mrw@painless-security.com
URI: <http://www.painless-security.com>

Pierrick Seite
France Telecom - Orange
4, rue du clos courtel BP 91226
Cesson-Sevigne 35512
France

Email: pierrick.seite@orange-ftgroup.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 13, 2012

W. Dec, Ed.
Cisco Systems
T. Mrugalski
ISC
T. Sun
China Mobile
B. Sarikaya
Huawei USA
September 10, 2011

DHCPv6 Route Options
draft-ietf-mif-dhcpv6-route-option-03

Abstract

This document describes DHCPv6 Route Options for provisioning IPv6 routes on DHCPv6 client nodes. This is expected to improve the ability of an operator to configure and influence a nodes' ability to pick an appropriate route to a destination when this node is multi-homed and where other means of route configuration may be impractical.

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].

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 March 13, 2012.

Copyright Notice

Copyright (c) 2011 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. Problem overview	3
3. DHCPv6 Based Solution	4
3.1. Default route configuration	4
3.2. Configuring on-link routes	5
3.3. Deleting obsolete route	5
3.4. Applicability to routers	5
3.5. Updating Routing Information	5
3.6. Limitations	6
4. DHCPv6 Route Options	7
4.1. Next Hop Option Format	7
4.2. Route Prefix Option Format	8
5. DHCPv6 Server Behavior	10
6. DHCPv6 Client Behavior	10
7. IANA Considerations	11
8. Security Considerations	11
9. Contributors and Acknowledgements	12
10. References	12
10.1. Normative References	12
10.2. Informative References	12
Authors' Addresses	13

1. Introduction

The Neighbor Discovery (ND) protocol [RFC4861] provides a mechanism for hosts to discover one or more default routers on a directly connected network segment. Extensions to the Router Advertisement (RA) protocol defined in [RFC4191] allow hosts to discover the preferences for multiple default routers on a given link, as well as any specific routes advertised by these routers. This allows network administrators to better handle multi-homed host topologies and influence the route selection by the host. This ND based mechanism however is sub optimal or impractical in some multi-homing scenarios, where DHCPv6 [RFC3315] is seen to be more viable.

This draft defines the DHCPv6 Route Options for provisioning IPv6 routes on DHCPv6 clients. The proposed option is primarily envisaged for use by DHCPv6 client nodes that are capable of making basic IP routing decisions and maintaining an IPv6 routing table, broadly in line with the capabilities of a generic host as described in [RFC4191].

Throughout the document the words node and client are used as a reference to the device with such routing capabilities, hosting the DHCPv6 client software. The route information is taken to be equivalent to static routing, and limited in the number of required routes to a handful.

2. Problem overview

The solution described in this document applies to multi-homed scenarios including ones where the client is simultaneously connected to multiple access network (e.g. WiFi and 3G). The following scenario is used to illustrate the problem as found in typical multi-homed residential access networks. It is duly noted that the problem is not specific to IPv6, occurring also with IPv4, where it is today solved by means of DHCPv4 classless route information option [RFC3442], or alternative configuration mechanisms.

In multi-homed networks, a given user's node may be connected to more than one gateway. Such connectivity may be realized by means of dedicated physical or logical links that may also be shared with other users nodes. In such multi-homed networks it is quite common for the network operator to offer the delivery of a particular type of IP service via a particular gateway, where the service can be characterised by means of specific destination IP network prefixes. Thus, from an IP routing perspective in order for the user node to select the appropriate gateway for a given destination IP prefix, recourse needs to be made to classic longest destination match IP

routing, with the node acquiring such prefixes into its routing table. This is typically the remit of dynamic Internal Gateway Protocols (IGPs), which however are rarely used by operators in residential access networks. This is primarily due to operational costs and a desire to contain the complexity of user nodes and IP Edge devices to a minimum. While, IP Route configuration may be achieved using the ICMPv6 extensions defined in [RFC4191], this mechanism does not lend itself to other operational constraints such as the desire to control the route information on a per node basis, the ability to determine whether a given node is actually capable of receiving/processing such route information. A preferred mechanism, and one that additionally also lends itself to centralized management independent of the management of the gateways, is that of using the DHCP protocol for conveying route information to the nodes.

3. DHCPv6 Based Solution

A DHCPv6 based solution allows an operator an on demand and node specific means of configuring static routing information. Such a solution also fits into network environments where the operator prefers to manage Residential Gateway (RG) configuration information from a centralized DHCP server. [I-D.ietf-v6ops-ipv6-multihoming-without-ipv6nat] provides additional background to the need for a DHCPv6 solution to the problem.

In terms of the high level operation of the solution defined in this draft, a DHCPv6 client interested in obtaining routing information request the route options using the DHCPv6 Option Request Option (ORO) sent to a server. A Server, when configured to do so, provides the requested route information as part of a nested options structure covering; the next-hop address; the destination prefix; the route metric; any additional options applicable to the destination or next-hop.

3.1. Default route configuration

Defined mechanism may be used to configure default route. Default route may be specified in two ways.

In bandwidth constrained networks, server MAY send NEXT_HOP option without any RT_PREFIX options. NEXT_HOP option that does not contain any RT_PREFIX options designate default router. Second way of defining default route is to convey RT_PREFIX option that specifies ::/0 route, included as suboption in NEXT_HOP. First approach has the benefit of consuming less bandwidth, while the second one allows definition of default route lifetime and metric.

Server MUST NOT define more than one default prefix (i.e. both defined configuration methods are mutually exclusive). Unless there are significant bandwidth restrictions, mechanism that uses `::/0` RT_PREFIX option SHOULD be used.

3.2. Configuring on-link routes

Server may also configure on-link routes, i.e. routes that are available directly over the link, not via routers. To specify on-link routes, server MAY include RTPREFIX option directly in Advertise and Reply messages.

3.3. Deleting obsolete route

There are two mechanisms that allow removing a route. Each defined route has a route lifetime. If specific route is not refreshed and its timer reaches 0, client MUST remove corresponding entry from routing table.

In cases, where faster route removal is needed, server SHOULD return RT_PREFIX option with route lifetime set to 0. Client that receives RT_PREFIX with route lifetime set to 0 MUST remove specified route immediately, even if its previous lifetime did not expire yet.

3.4. Applicability to routers

Contrary to Router Advertisement mechanism, defined in [RFC4861] that explicitly limits configuration to hosts, routing configuration over DHCPv6 defined in this document may be used by both hosts and routers.

One of the envisaged usages for this solution are residential gateways (RG) or Customer Premises Equipment (CPE). Those devices very often perform routing. It may be useful to configure routing on such devices over DHCPv6. One example of such use may be a class of premium users that are allowed to use dedicated router that is not available to regular users.

3.5. Updating Routing Information

Network configuration occasionally changes, due to failure of existing hardware, migration to newer equipment or many other reasons. Therefore there a way to inform clients that routing information have changed is required.

There are several ways to inform clients about new routing information. Every client SHOULD periodically refresh its configuration, according to Information Refresh Time Option, so

server may send updated information the next time client refreshes its information. New routes may be configured at that time. As every route has associated lifetime, client is required to remove its routes when this timer expires. This method is particularly useful, when migrating to new router is undergoing, but old router is still available.

Server MAY also announce routes via soon to be removed router with lifetimes set to 0. This will cause the client to remove its routes, despite the fact that previously received lifetime may not yet expire.

Aforementioned methods are useful, when there is no urgent need to update routing information. Bound by timer set by value of Information Refresh Time Option, clients may use outdated routing information until next scheduled renewal. Depending on configured value this delay may be not acceptable in some cases. In such scenarios, administrators are advised to use RECONFIGURE mechanism, defined in [RFC3315]. Server transmits RECONFIGURE message to each client, thus forcing it to immediately start renewal process.

See also Section 3.6 about limitations regarding dynamic routing.

3.6. Limitations

Defined mechanism is not intended to be used as a dynamic routing protocol. It should be noted that proposed mechanism cannot automatically detect routing changes. In networks that use dynamic routing and also employ this mechanism, clients may attempt using routes configured over DHCPv6 even though routers or specific routes ceased to be available. This may cause black hole routing problem. Therefore it is not recommended to use this mechanism in networks that use dynamic routing protocols. This mechanism SHOULD NOT be used in such networks, unless network operator can provide a way to update DHCP server information in case of router availability changes.

Discussion: It should be noted that DHCPv6 server is not able to monitor health of existing routers. As there are currently more than 60 options defined for DHCPv6, it is infeasible to implement mechanism that would monitor huge set of services and stop announcing its availability in case of service outage. Therefore in case of prolonged unavailability human intervention is required to change DHCPv6 server configuration. If that is considered a problem, network administrators should consider using other alternatives, like RA and ND mechanisms (see [RFC4861]).

4. DHCPv6 Route Options

A DHCPv6 client interested in obtaining routing information includes the NEXT_HOP and RT_PREFIX options as part of its Option Request Option (ORO) in messages directed to a server (as allowed by [RFC3315], i.e. Solicit, Request, Renew, Rebind or Information-request messages). A Server, when configured to do so, provides the requested route information using zero, one or more NEXT_HOP options in messages sent in response (Advertise, and Reply). So as to allow the route options to be both extensible, as well as conveying detailed info for routes, use is made of a nested options structure. Server sends one or more NEXT_HOP options that specify the IPv6 next hop addresses. Each NEXT_HOP option conveys in turn zero, one or more RT_PREFIX options that represents the IPv6 destination prefixes reachable via the given next hop. Server includes RT_PREFIX directly in message to indicate that given prefix is available directly on-link. Server MAY send a single NEXT_HOP without any RT_PREFIX suboptions or with RT_PREFIX that contains ::/0 to indicate available default route. The Formats of the NEXT_HOP and RT_PREFIX options are defined in the following sub-sections.

The DHCPv6 Route Options format borrows from the principles of the Route Information Option defined in [RFC4191].

4.1. Next Hop Option Format

Each IPv6 route consists of an IPv6 next hop address, an IPv6 destination prefix (a.k.a. the destination subnet), and a host preference value for the route. Elements of such route (e.g. Next hops and prefixes associated with them) are conveyed in NEXT_HOP option that contains RT_PREFIX suboptions.

The Next Hop Option defines the IPv6 address of the next hop, usually corresponding to a specific next-hop router. For each next hop address there can be zero, one or more prefixes reachable via that next hop.

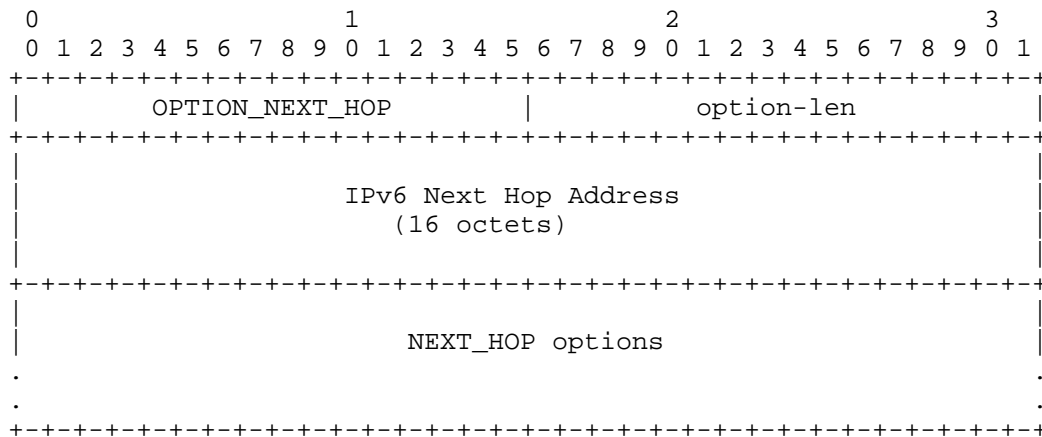


Figure 1: IPv6 Next Hop Option Format

option-code: `OPTION_NEXT_HOP` (TBD).

option-len: 16 + Length of `NEXT_HOP` options field.

IPv6 Next Hop Address: 16 octet long field that specified IPv6 address of the next hop.

`NEXT_HOP` options: Options associated with this Next Hop. This includes, but is not limited to, zero, one or more `RT_PREFIX` options that specify prefixes reachable through the given next hop.

4.2. Route Prefix Option Format

The Route Prefix Option is used to convey information about a single prefix that represents the destination network. The Route Prefix Option is used as a sub-option in the previously defined Next Hop Option. It may also be sent directly in message to indicate that route is available directly on-link.

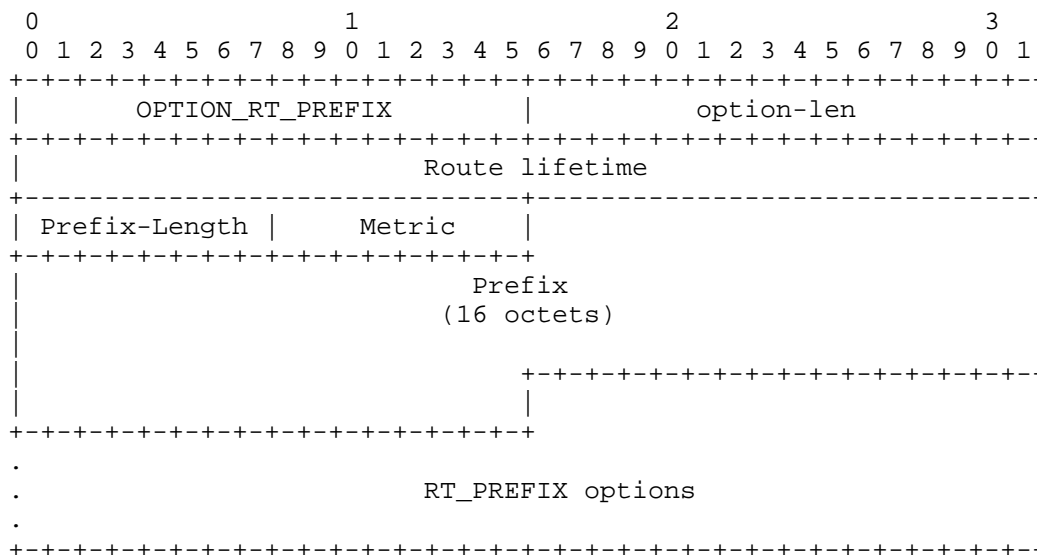


Figure 2: Route Prefix Option Format

option-code: OPTION_RT_PREFIX (TBD).

option-len: 18 + length of RT_PREFIX options.

Route lifetime 32-bit unsigned integer. Specifies lifetime of the route information, expressed in seconds. There are 2 special values defined. 0 means that route is no longer valid and must be removed by clients. 0xffffffff means infinity.

Prefix Length: 8-bit unsigned integer. The length in bits of the IP Prefix. The value ranges from 0 to 128. This field represents the number of valid leading bits in the prefix.

Metric: Route Metric. 8-bit signed integer. The Route Metric indicates whether to prefer the next hop associated with this prefix over others, when multiple identical prefixes (for different next hops) have been received.

Prefix: Fixed length 16 octet field containing an IPv6 prefix.

RT_PREFIX options: Options specific to this particular prefix.

5. DHCPv6 Server Behavior

When configured to do so, a DHCPv6 server shall provide the Next Hop and Route Prefix Options in ADVERTISE and REPLY messages sent to a client that requested the route option. Each Next Hop Option sent by the server must convey at least one Route Prefix Option.

Server includes NEXT_HOP option with possible RT_PREFIX suboptions to designate that specific routes are available via routers. Server includes RT_PREFIX options directly in Advertise and Reply messages to inform that specific routes are available directly on-link.

If there is more than one route available via specific next hop, server MUST send only one NEXT_HOP for that next hop, which contains multiple RT_PREFIX options. Server MUST NOT send more than one identical (i.e. with equal next hop address field) NEXT_HOP option.

Servers SHOULD NOT send Route Option to clients that did not explicitly requested it, using the ORO.

Servers MUST NOT send Route Option in messages other than ADVERTISE or REPLY.

Servers MAY also include Status Code Option, defined in Section 22.13 of the [RFC3315] to indicate the status of the operation.

Servers MUST include the Status Code Option, if the requested routing configuration was not successful and SHOULD use status codes as defined in [RFC3315] and [RFC3633].

The maximum number of routing information in one DHCPv6 message depend on the maximum DHCPv6 message size defined in [RFC3315]

6. DHCPv6 Client Behavior

A DHCPv6 client compliant with this specification MUST request the NEXT_HOP and RT_PREFIX Options in an Option Request Option (ORO) in the following messages: Solicit, Request, Renew, Rebind, and Information-Request. The messages are to be sent as and when specified by [RFC3315].

When processing a received Route Options a client MUST substitute a received 0::0 value in the Next Hop Option with the source IPv6 address of the received DHCPv6 message. It MUST also associate a received Link Local next hop addresses with the interface on which the client received the DHCPv6 message containing the route option. Such a substitution and/or association is useful in cases where the

DHCPv6 server operator does not directly know the IPv6 next-hop address, other than knowing it is that of a DHCPv6 relay agent on the client LAN segment. DHCPv6 Packets relayed to the client are sourced by the relay using this relay's IPv6 address, which could be a link local address.

The Client SHOULD refresh assigned route information periodically. The generic DHCPv6 Information Refresh Time Option, as specified in [RFC4242], can be used when it is desired for the client to periodically refresh of route information.

The routes conveyed by the Route Option should be considered as complimentary to any other static route learning and maintenance mechanism used by, or on the client with one modification: The client MUST flush DHCPv6 installed routes following a link flap event on the DHCPv6 client interface over which the routes were installed. This requirement is necessary to automate the flushing of routes for clients that may move to a different network.

Client MUST confirm that routers announced over DHCPv6 are reachable, using one of methods suitable for specific network type. The most common mechanism is Neighbor Unreachability Detection (NUD), specified in [RFC4861]. Client SHOULD use NUD to verify that received routers are reachable before adjusting its routing tables. Client MAY use other reachability verification mechanisms specific to used network technology. To avoid potential long-lived routing black holes, client MAY periodically confirm that router is still reachable.

7. IANA Considerations

A DHCPv6 option number of TBD for the introduced Route Option. IANA is requested to allocate three DHCPv6 option codes referencing this document: OPTION_NEXT_HOP and OPTION_RT_PREFIX.

8. Security Considerations

The overall security considerations discussed in [RFC3315] apply also to this document. The Route option could be used by malicious parties to misdirect traffic sent by the client either as part of a denial of service or man-in-the-middle attack. An alternative denial of service attack could also be realized by means of using the route option to overflowing any known memory limitations of the client, or to exceed the client's ability to handle the number of next hop addresses.

Neither of the above considerations are new and specific to the proposed route option. The mechanisms identified for securing DHCPv6 as well as reasonable checks performed by client implementations are deemed sufficient in addressing these problems.

It is essential that clients verify that announced routers are indeed reachable, as specified in Section 6. Failing to do so may create black hole routing problem.

This mechanism may introduce severe problems if deployed in networks that use dynamic routing protocols. See Section 3.6 for details.

Reader is also encouraged to read DHCPv6 security considerations document [I-D.ietf-dhc-secure-dhcpv6].

9. Contributors and Acknowledgements

This document would not have been possible without the significant contribution provided by: Arifumi Matsumoto, Hui Deng, Richard Johnson, and Zhen Cao.

The authors would also like to thank Alfred Hines, Ralph Droms, Ted Lemon, Ole Troan, Dave Oran, Dave Ward, Joel Halpern, Marcin Siodelski and Alexandru Petrescu for their comments and useful suggestions.

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [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.
- [RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, December 2003.

10.2. Informative References

- [I-D.ietf-dhc-secure-dhcpv6]
Jiang, S. and S. Shen, "Secure DHCPv6 Using CGAs",
draft-ietf-dhc-secure-dhcpv6-03 (work in progress),

June 2011.

- [I-D.ietf-v6ops-ipv6-multihoming-without-ipv6nat]
Troan, O., Miles, D., Matsushima, S., Okimoto, T., and D. Wing, "IPv6 Multihoming without Network Address Translation",
draft-ietf-v6ops-ipv6-multihoming-without-ipv6nat-01 (work in progress), August 2011.
- [RFC3442] Lemon, T., Cheshire, S., and B. Volz, "The Classless Static Route Option for Dynamic Host Configuration Protocol (DHCP) version 4", RFC 3442, December 2002.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, November 2005.
- [RFC4242] Venaas, S., Chown, T., and B. Volz, "Information Refresh Time Option for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 4242, November 2005.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

Authors' Addresses

Wojciech Dec (editor)
Cisco Systems
Haarlerbergweg 13-19
1101 CH Amsterdam
The Netherlands

Email: wdec@cisco.com

Tomasz Mrugalski
Internet Systems Consortium, Inc.
950 Charter Street
Redwood City, CA 94063
USA

Phone: +1 650 423 1345
Email: tomasz.mrugalski@gmail.com

Tao Sun
China Mobile
Unit2, 28 Xuanwumenxi Ave
Beijing, Xuanwu District 100053
China

Phone:
Email: suntao@chinamobile.com

Behcet Sarikaya
Huawei USA
1700 Alma Dr. Suite 500
Plano, TX 75075
United States

Phone: +1 972-509-5599
Fax:
Email: sarikaya@ieee.org
URI:

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: December 24, 2011

T. Savolainen
Nokia
J. Kato
NTT
T. Lemon
Nominum, Inc.
June 22, 2011

Improved DNS Server Selection for Multi-Homed Nodes
draft-ietf-mif-dns-server-selection-03

Abstract

A multi-homed node can be connected to multiple networks that may utilize different DNS namespaces. The node commonly receives DNS server configuration information from all connected networks. Some of the DNS servers may have information about namespaces other servers do not have. When the multi-homed node needs to utilize DNS, it has to choose which of the servers to contact to. This document describes a policy based method for helping on selection of DNS server, for both forward and reverse DNS lookup procedures, with help of DNS suffix and IPv6 prefix information received via DHCPv6 or DHCPv4.

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 December 24, 2011.

Copyright Notice

Copyright (c) 2011 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	4
1.1. Requirements Language	4
2. Problem description for local namespaces with multi-homed nodes	4
2.1. Fully qualified domain names with limited scopes	5
2.2. Network interface specific IP addresses	6
2.3. A problem not fully solved by the described solution	7
3. Deployment scenarios	7
3.1. CPE deployment scenario	8
3.2. Cellular network scenario	8
3.3. VPN scenario	8
3.4. Dual-stack accesses	9
4. Improved DNS server selection	9
4.1. Procedure for prioritizing DNS servers and handling responses	9
4.2. DNS server selection DHCPv6 option	11
4.3. DNS server selection DHCPv4 option	13
4.4. Limitations on use	15
4.5. Coexistence with RFC3646	15
4.6. Interactions with OPTION_DOMAIN_LIST	16
4.7. CNAME/DNAME record considerations	16
5. Example of a node behavior	16
6. Scalability considerations	19
7. Considerations for network administrators	19
8. Acknowledgements	19
9. IANA Considerations	19
10. Security Considerations	20
11. References	20
11.1. Normative References	20
11.2. Informative References	21
Appendix A. Best Current Practice for DNS server selection	22
A.1. Sending queries out on multiple interfaces in parallel	22
A.2. Search list option for DNS forward lookup decisions	22
A.3. More specific routes for reverse lookup decision	23
A.4. Longest matching prefix for reverse lookup decision	23
Appendix B. DNSSEC and multiple answers validating with	

different trust anchors	23
Authors' Addresses	24

1. Introduction

A multi-homed node faces several problems a single-homed node does not encounter, as is described in [I-D.ietf-mif-problem-statement]. This document studies in detail the problems local namespaces may cause for multi-homed nodes and provides a solution for IPv6 domain. The node may be implemented as a host or as a router.

When multiple namespaces are visible for a node, some DNS servers have information other servers do not have. Because of that, a multi-homed node cannot assume every DNS server is able to properly answer for any query, but instead the node must be able to ask right server for the information it needs.

An example of an application that benefits from multi-homing is a web browser that commonly accesses many different destinations and needs to be able to dynamically communicate over different network interfaces.

In deployments where multiple namespaces are present, selection of correct route and destination and source addresses for the actual IP connection is crucial as well, as the resolved destination's IP addresses may be only usable on the network interface over which the name was resolved on. Hence solution described in this document is assumed to be commonly used in combination with tools for delivering additional routing and source and destination address selection policies.

The Appendix A describes best current practices possible with tools preceding this document and on networks not supporting the solution described in this document. As it is possible to solve the problem with less efficient and less explicit manners, the new solution may be considered as an optimization. However, in some environments this solution is considered essential.

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].

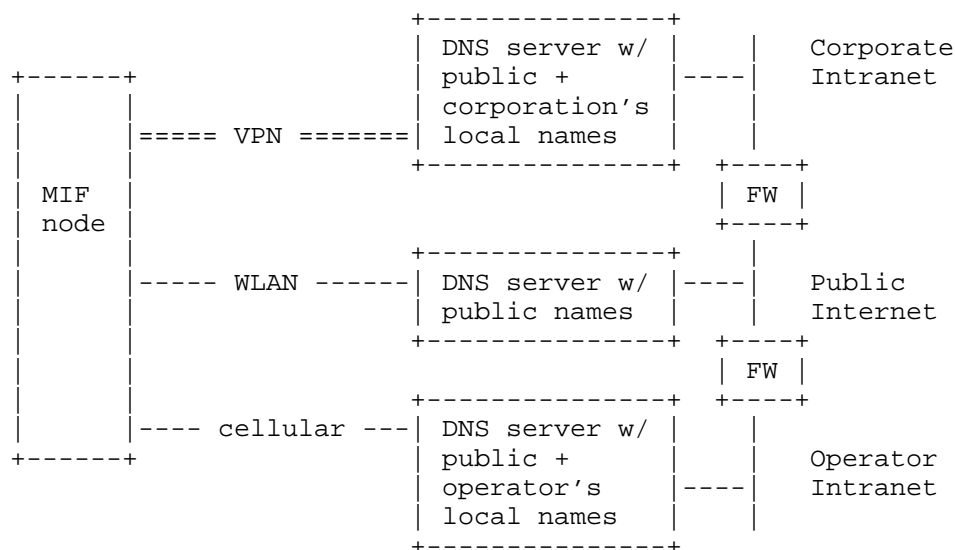
2. Problem description for local namespaces with multi-homed nodes

This chapter describes two host multi-homing related local namespace scenarios for which the procedure described in chapter 3 provides a solution for. Essentially the same challenges may be faced by Consumer Premises Equipment as is described in

[I-D.ietf-v6ops-multihoming-without-nat66]. This chapter additionally describes a related problem for which this document provides only partial solution.

2.1. Fully qualified domain names with limited scopes

A multi-homed node may be connected to one or more networks that are using local namespaces. As an example, the node may have simultaneously open a wireless LAN (WLAN) connection to the public Internet, cellular connection to an operator network, and a virtual private network (VPN) connection to a corporate network. When an application initiates a connection establishment to an FQDN, the host needs to be able to choose the right DNS server for making a successful DNS query. This is illustrated in the figure 1. An FQDN for a public name can be usually resolved with any DNS server, but for an FQDN of corporation's or operator's service's local name the node needs to be able to correctly select the right DNS server for the DNS resolution, i.e. do also network interface selection already before destination's IP address is known.

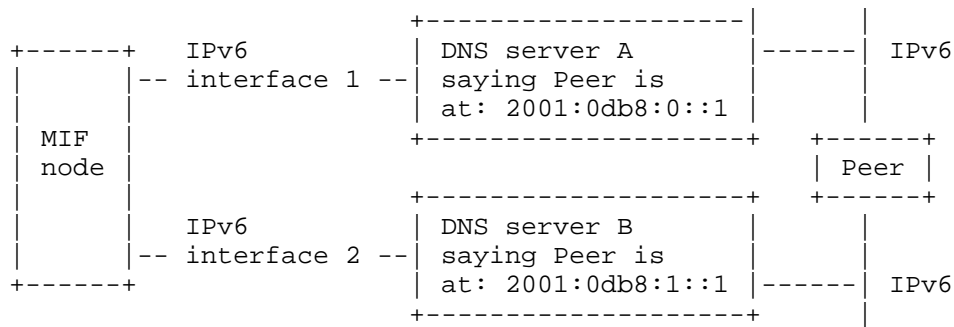


Local DNS namespaces illustrated

Figure 1

2.2. Network interface specific IP addresses

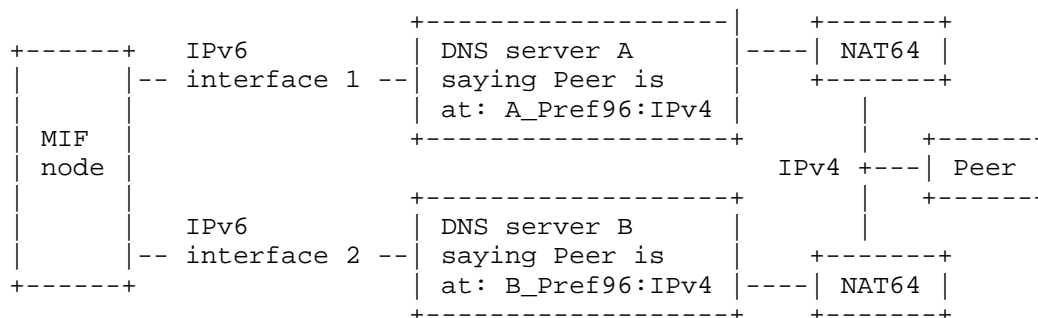
In the second problem an FQDN is valid and resolvable via different network interfaces, but to different and not necessarily globally reachable IP addresses, as is illustrated in the figure 2. Node's routing and source and destination address selection mechanism must ensure the destination's IP address is only used in combination with source IP addresses of the network interface the name was resolved on.



Local DNS namespaces and different IP addresses for an FQDN on interfaces 1 and 2.

Figure 2

Similar situation can happen with IPv6 protocol translation and AAAA record synthesis [RFC6147]. A synthesised AAAA record is guaranteed to be valid only on a network interface it was synthesized on. Figure 3 illustrates a scenario where the peer's IPv4 address is synthesized into different IPv6 addresses by DNS servers A and B.



AAAA synthesis results in interface specific IPv6 addresses.

Figure 3

A thing worth noting is that interface specific IP addresses can cause problems also for a single-homed host, if the host retains its DNS cache during movement from one network interface to another. After the interface change a host could have both positive and negative DNS cache entries no longer valid on the new network interface. Because of this the cached DNS information should be considered network interface local instead of node global.

2.3. A problem not fully solved by the described solution

A more complex scenario is an FQDN, which in addition to possibly resolving into network interface specific IP addresses, identifies on different network interfaces completely different peer entities with potentially different set of service offerings. In even more complex scenario, an FQDN identifies unique peer entity, but one that provides different services on its different network interfaces. The solution described in this document is not able to tackle these higher layer issues. In fact, these problems may be solvable only by manual user intervention.

However, when DNSSEC is used, the DNSSEC validation procedure may provide assistance for selecting correct responses for some, but not all, use cases. A node may prefer to use the DNS answer that validates with the preferred trust anchor.

3. Deployment scenarios

This document has been written with three particular deployment scenarios in mind. First being a Consumer Premises Equipment (CPE) with two or more uplink VLAN connections. Second scenario involves a

cellular device with two uplink Internet connections: WLAN and cellular. Third scenario is for VPNs, where use of local DNS server may be preferred for latency reasons, but corporate DNS server must be used to resolve private names used by the corporation.

3.1. CPE deployment scenario

A home gateway may have two uplink connections leading to different networks, as is described in [I-D.ietf-v6ops-multihoming-without-nat66]. In this scenario only first uplink connections lead to Internet, while second uplink connection leads to a private network utilizing private namespace.

It is desirable that the CPE does not have to send DNS queries over both uplink connections, but instead CPE should send default queries to the DNS server of the interface leading to the Internet, and queries related to private namespace to the DNS server of the private network.

In this scenario the legacy hosts can be supported by deploying DNS proxy on the CPE and configuring hosts in the LAN to talk to the DNS proxy. However, updated hosts would be able to talk directly to the correct DNS servers of each uplink ISP's DNS server. It is deployment decision whether the updated hosts would be pointed to DNS proxy or to actual DNS servers.

Depending on actual deployments, all VLAN connections may be considered secure.

3.2. Cellular network scenario

A cellular device may have both WLAN and cellular network interfaces up. In such a case it is often desirable to use WLAN by default, except for those connections cellular network operator wants to go over cellular interface. The cellular network may utilize private names and hence the cellular device needs to ask for those through the cellular interface.

In this scenario cellular interface can be considered secure and WLAN often insecure.

3.3. VPN scenario

Depending on a deployment, there may be need to use VPN only for traffic destined to a corporate network. The corporation may be using private namespace, and hence related DNS queries should be send over VPN to the corporate DNS server, while by default a DNS server of a local access network may be used.

In this scenario VPN interface can be considered secure and local access network insecure.

3.4. Dual-stack accesses

A node may be connected to one or more dual-stack capable access networks. In such a case both or either of DHCPv4 and DHCPv6 can be used to learn DNS server selection information.

4. Improved DNS server selection

This chapter describes a procedure that a (stub / proxy) resolver may utilize for improved DNS server selection in face of multiple namespaces and multiple simultaneously active network interfaces.

4.1. Procedure for prioritizing DNS servers and handling responses

A resolver SHALL build a priority list of DNS servers it will contact to depending on the query. To build the list in an optimal way, a node SHOULD ask with DHCP which DNS servers of each network interface are most likely able to successfully serve forward lookup requests matching to specific DNS suffixes or reverse (PTR record) lookup requests matching to specific IPv6 prefixes. For security reasons the DNS server selection information MUST be used only when it is safe to do so, see section 4.3 for details.

The node SHOULD create a host specific route for the DNS server addresses learned via DHCP. The route must point to the interface DNS server address was learned on. This is required to ensure DNS queries are sent out via the right interface.

A resolver lacking more explicit information shall assume that all information is available from any DNS server of any network interface. The DNS servers learnt by other DNS server address configuration methods MUST be handled as medium priority default servers.

When a DNS query needs to be made, the resolver SHOULD give highest precedence to the DNS servers explicitly known to serve matching suffixes or prefixes. However, the resolver SHOULD take into account different trust levels of pieces of DNS server selection information the resolver may have received from node's network interfaces. The resolver SHOULD prefer DNS servers of trusted interfaces. The DNS servers of trusted interfaces may be of highest priority only if trusted interfaces specifically configure DNS servers to be of low priority. The non-exhaustive list on figure 4 illustrates how the different trust levels of received DNS server selection information

SHOULD influence the DNS server selection logic.

A resolver SHOULD prioritize between equally trusted DNS servers with help of the DHCP option preference field. The resolver SHOULD NOT prioritize less trusted DNS servers higher than trusted, even in the case of less trusted server would apparently have additional information. In the case of all other things being equal the resolver shall make the prioritization decision based on its internal preferences.

Information from from more trusted interface A	Information from less trusted interface B	Resulting DNS server priority selection
1. Medium priority default	Medium priority default	Default: A, then B
2. Medium priority default	High priority default High priority specific	Default: A, then B Specific: A, then B
3. Low priority default	Medium priority default	Default: B, then A
4. Low priority default High priority specific	Medium priority default	Default: B, then A Specific: A, then B

Figure 4: DNS server selection in case of different trust levels

The resolver SHOULD avoid sending queries to different interfaces in parallel as that may waste resources, sometimes significantly, and would also unnecessary reveal information about ongoing communications. Independently of whether DNS queries are sent in series or parallel, replies for DNS queries MUST be waited until acceptable positive reply is received, all replies are received, or time out occurs.

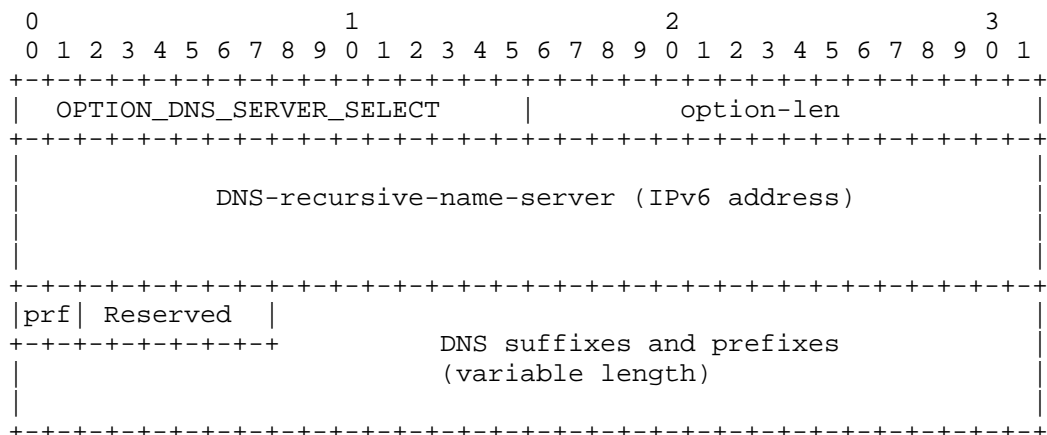
Because DNSSEC provides cryptographic assurance of the integrity of DNS data, data that can be validated under DNSSEC is necessarily to be preferred over data that cannot be. It follows that, if validation is not performed by the host making the decision about whether to trust the DNS data from a given interface, it cannot make a decision to prefer data from any interface with any great assurance: any response could be forged, and there is no way to detect it without DNSSEC. Specifically, the validating security aware host MUST NOT proceed with a reply that cannot be validated with DNSSEC if DNS queries sent to other servers are still pending.

In the case of a trusted DNS server replying negatively to a question having matching suffix, it will be for implementation to decide whether to consider that as a final response, or whether to ask also from other DNS servers. The implementation decision may be based, for example, on deployment or trust models.

(DISCUSS: What about those DNS servers that instead of negative answer always return positive reply with an IP address of some captive portal?)

4.2. DNS server selection DHCPv6 option

DHCPv6 option described below can be used to inform resolvers which DNS server should be contacted when initiating forward or reverse DNS lookup procedures.



option-code: OPTION_DNS_SERVER_SELECT (TBD)

option-len: Length of the option in octets

DNS-recursive-name-server: An IPv6 address of a DNS server

prf: DNS server preference, for selecting between
 equally trusted DNS servers:
 01 High
 00 Medium
 11 Low
 10 Reserved

Reserved: Flags reserved for the future. MUST be set to zero.

DNS suffixes and prefixes: The list of DNS suffixes for forward DNS
lookup and prefixes for reverse DNS lookup the DNS server
has special knowledge about. Field MUST be encoded as
specified in section "Representation and use of
domain names" of [RFC3315].
Special suffix of "." is used to indicate
capability to resolve global names and act as a
default name server. Lack of "."
suffix on the list indicates DNS server only has
information related to listed suffixes and prefixes.
Prefixes for reverse mapping are encoded as
defined for ip6.arpa [RFC3152].

DHCPv6 option for explicit DNS suffix configuration

Figure 5

A node SHOULD include an OPTION_ORO option in a DHCPv6 request with the OPTION_DNS_SERVER_SELECT option code to inform the DHCPv6 server about the support for the improved DNS server selection logic. DHCPv6 server receiving this information MAY then choose to provision DNS server addresses only with the OPTION_DNS_SERVER_SELECT.

The OPTION_DNS_SERVER_SELECT contains one or more DNS suffixes the related DNS server has particular knowledge of. The option can occur multiple times in a single DHCPv6 message, if multiple DNS servers are to be configured.

IPv6 prefixes should cover all the DNS suffixes configured in this option. Prefixes should be as long as possible to avoid potential collision with information received on other option instances or with options received from DHCPv6 servers of other network interfaces. Overlapping IPv6 prefixes are interpreted so that the resolver can use any of the DNS servers for queries matching the prefixes.

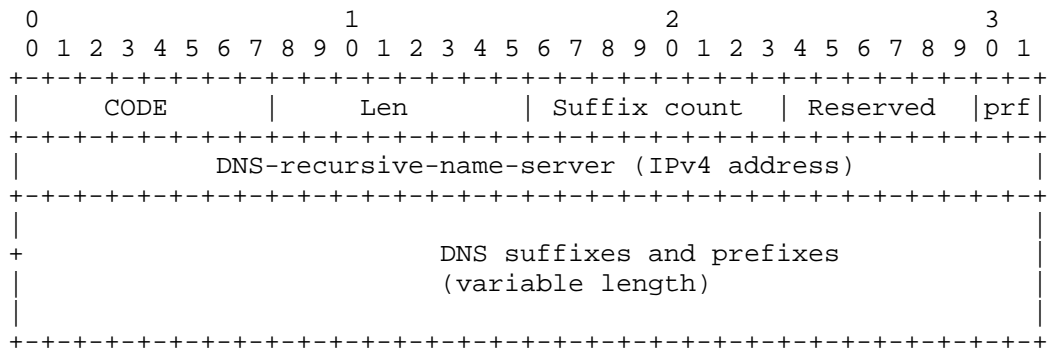
If the OPTION_DNS_SERVER_SELECT contains a DNS server address already learned from other DHCPv6 servers and possibly through other network interfaces, the node MAY append new prefixes and suffixes to the information received earlier. The node MUST NOT remove previously obtained information. However, the node SHOULD NOT extent lifetime of earlier information either. In the case conflicting DNS server address and related information is learned from less trusted interface, the node MAY choose to ignore the option.

As the DNS options of [RFC3646], the OPTION_DNS_SERVER_SELECT option MUST NOT appear in any other than the following DHCPv6 messages: Solicit, Advertise, Request, Renew, Rebind, Information-Request, and Reply.

The information conveyed in OPTION_DNS_SERVER_SELECT is considered valid until changed or refreshed by general events that trigger DHCPv6 action. In the event that it is desired for the client to request a refresh of the information, use of generic DHCPv6 Information Refresh Time Option, as specified in [RFC4242] is envisaged.

4.3. DNS server selection DHCPv4 option

DHCPv4 option described below can be used to inform resolvers which DNS server should be contacted when initiating forward or reverse DNS lookup procedures.



option-code: OPTION_DNS_SERVER_SELECT (TBD)

option-len: Lenght of the option in octets

Suffix count: Number of suffixes and prefixes included

DNS-recursive-name-server: An IPv4 address of a DNS server

prf: DNS server preference, for selecting between
 equally trusted DNS servers:
 01 High
 00 Medium
 11 Low
 10 Reserved

Reserved: Flags reserved for the future. MUST be set to zero.

DNS suffixes and prefixes: The list of DNS suffixes for forward DNS lookup and prefixes for reverse DNS lookup the DNS server has special knowledge about. Field MUST be encoded as specified in section "Representation and use of domain names" of [RFC3315].
 Special suffix of "." is used to indicate capability to resolve global names and act as a default name server. Lack of "." suffix on the list indicates DNS server only has information related to listed suffixes and prefixes.
 Prefixes for reverse mapping are encoded as defined for in-addr.arpa [RFC2317]. Trailing zeros shall be added until next octet boundary.

DHCPv4 option for explicit DNS suffix configuration

Figure 6

The `OPTION_DNS_SERVER_SELECT` contains one or more DNS suffixes the related DNS server has particular knowledge of. The option can occur multiple times in a single DHCPv4 message, if multiple DNS servers are to be configured.

If multiple instances of `OPTION_DNS_SERVER_SELECT` are present, then the data portions of all the options are concatenated together as specified in "Encoding Long DHCP Options in the Dynamic Host Configuration Protocol (DHCPv4)" [RFC3396].

If the `OPTION_DNS_SERVER_SELECT` contains a DNS server address already learned from other DHCPv4 servers and possibly through other network interfaces, the node MAY append new prefixes and suffixes to the information received earlier. The node MUST NOT remove previously obtained information. However, the node SHOULD NOT extent lifetime of earlier information either. In the case conflicting DNS server address and related information is learned from less trusted interface, the node MAY choose to ignore the option.

4.4. Limitations on use

A node MAY use `OPTION_DNS_SERVER_SELECT` in any of the following four cases. In other cases the node MUST NOT use `OPTION_DNS_SERVER_SELECT` unless the node is specifically configured to do so.

1. The server selection option is delivered across a secure, trusted channel.
2. The server selection option is not secured, but the client on a node does DNSSEC validation.
3. The server selection option is not secured, the resolver does DNSSEC validation, and the client communicates with the resolver configured with server selection option over a secure, trusted channel.
4. The DNS server IP address that is being recommended in the server selection option is known and trusted by the client; that is, the server selection option serves not to introduce the client to a new server, but rather to inform it that a server it has already been configured to trust is available to it for resolving certain domains.

4.5. Coexistence with RFC3646

The `OPTION_DNS_SERVER_SELECT` is designed to coexist with `OPTION_DNS_SERVERS` defined in [RFC3646]. The DNS servers configured via `OPTION_DNS_SERVERS` MUST BE considered as default name servers with medium preference. When both options are received from the same

network interface and the `OPTION_DNS_SERVER_SELECT` contains default DNS server address, the resolver MUST make the decision which one to prefer based on preferences. If `OPTION_DNS_SERVER_SELECT` defines medium preference then DNS server from `OPTION_DNS_SERVER_SELECT` SHALL be selected. All default servers are assumed to be able to resolve queries for global names.

If both `OPTION_DNS_SERVERS` and `OPTION_DNS_SERVER_SELECT` contain the same DNS server(s) IPv6 address(es), only one instance of each DNS servers' IPv6 addresses shall be added to the DNS server list.

If a node had indicated support for `OPTION_DNS_SERVER_SELECT` in DHCPv6 request, the DHCPv6 server may choose to omit sending of `OPTION_DNS_SERVERS`. This enables offloading use case where network administrator wishes to only advertise low priority default DNS servers.

4.6. Interactions with `OPTION_DOMAIN_LIST`

A node may be configured with DNS search list with `OPTION_DOMAIN_LIST`. Resolution for the name containing any dots SHOULD first be attempted with DNS servers of all interfaces as described earlier. Only if the resolution fails the node SHOULD append the name with search list suffix(es) and then utilize improved DNS server selection algorithm again to decide which DNS server(s) to contact next. A name without any dots SHALL immediately be appended with suffix(es) and improved DNS server selection be utilized on resolution.

4.7. CNAME/DNAME record considerations

If a node receives a reply with a canonical name (CNAME) or delegation name (DNAME) the follow-up queries MUST be sent to the same DNS server irrespectively of the FQDN received. Otherwise referrals may fail.

5. Example of a node behavior

Figure 6 illustrates node behavior when it initializes two network interfaces for parallel usage and learns DNS suffix and prefix information from DHCPv6 servers.

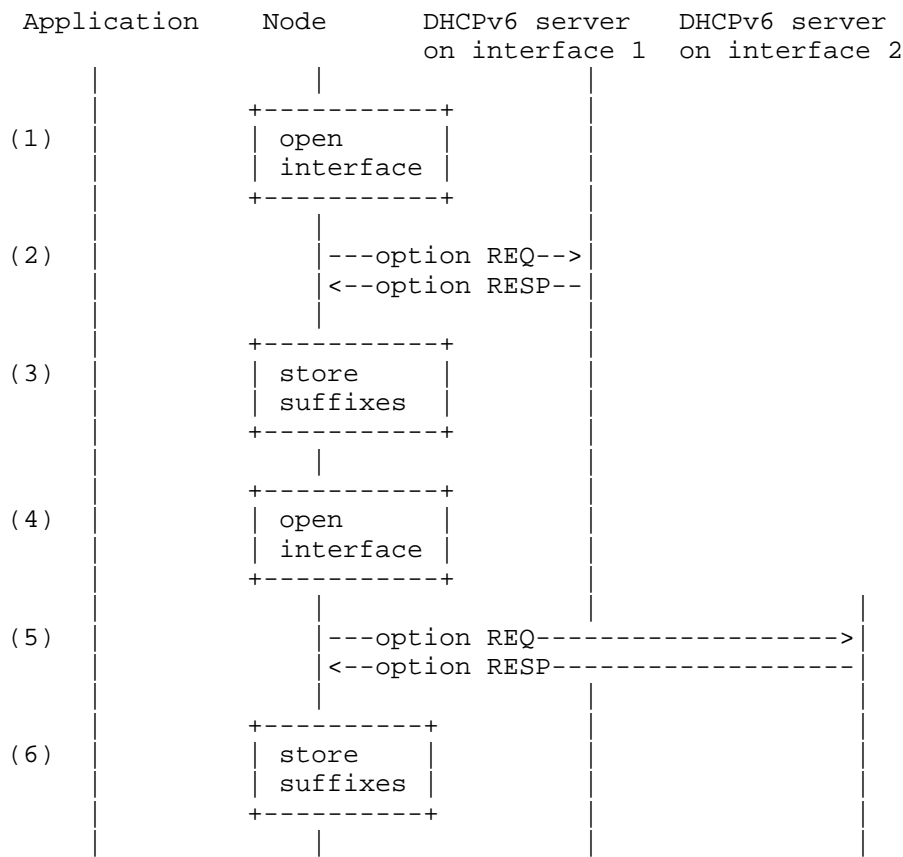


Illustration of learning DNS suffixes

Figure 7

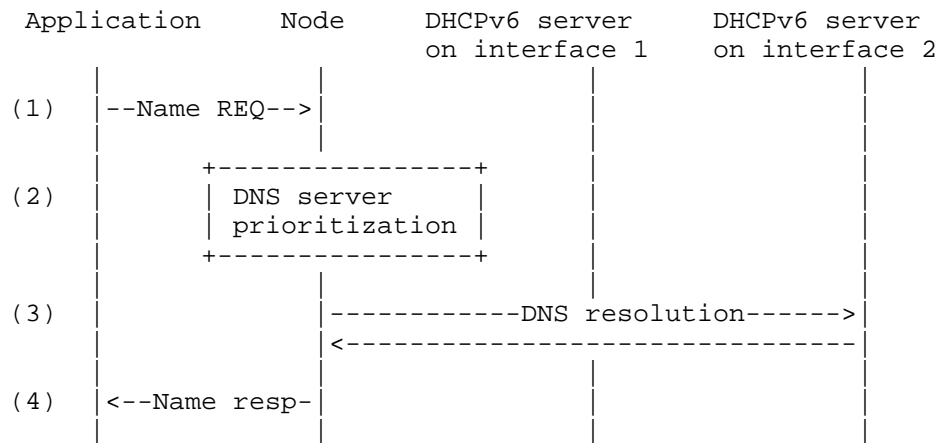
Flow explanations:

1. A node opens its first network interface
2. The node obtains DNS suffix and IPv6 prefix information for the new interface 1 from DHCPv6 server
3. The node stores the learned DNS suffixes and IPv6 prefixes for later use
4. The node opens its seconds network interface 2
5. The node obtains DNS suffix, say 'example.com', and IPv6 prefix information, say '8.b.d.0.1.0.0.2.ip6.arpa' for the new interface

2 from DHCPv6 server

6. The node stores the learned DNS suffixes and prefixes for later use

Figure 7 below illustrates how a resolver uses the learned suffix information. Prefix information use for reverse lookups is not illustrated, but that would go as the figure 7 example.



Example on choosing interface based on DNS suffix

Figure 8

Flow explanations:

1. An application makes a request for resolving an FQDN, e.g. 'private.example.com'
2. A node creates list of DNS servers to contact to and uses configured DNS server information and stored DNS suffix information on prioritization decisions.
3. The node has chosen interface 2, as from DHCPv6 it was learned earlier that the interface 2 has DNS suffix 'example.com'. The node then resolves the requested name using interface 2's DNS server to an IPv6 address
4. The node replies to application with the resolved IPv6 address

6. Scalability considerations

The size limitations of DHCPv6 messages limit the number of suffixes and prefixes that can be carried in a configuration option. Including the suffixes and prefixes in a DHCPv6 option is best suited for deployments where relatively few carefully selected suffixes and prefixes are adequate.

7. Considerations for network administrators

Network administrators deploying private namespaces should assist advanced hosts in the DNS server selection by providing information described in this document for nodes. To ensure nodes' routing and source and destination IP address selection also works correctly, network administrators should also deploy related technologies for that purpose.

The solution described herein is best for selecting a DNS server having knowledge of some namespaces. The solution is not able to make the right decision in a scenario where the same name points to different services on different network interfaces, as described in section 2.3. Network administrators are recommended to avoid overloading of namespaces in such manner.

To mitigate against attacks against local namespaces, administrators utilizing this tool should deploy DNSSEC for their zone.

8. Acknowledgements

The author would like to thank following people for their valuable feedback and improvement ideas: Mark Andrews, Jari Arkko, Marcelo Bagnulo, Stuart Cheshire, Lars Eggert, Tomohiro Fujisaki, Peter Koch, Suresh Krishnan, Edward Lewis, Kurtis Lindqvist, Arifumi Matsumoto, Erik Nordmark, Steve Padgett, Fabien Rapin, Dave Thaler, Margaret Wasserman, Dan Wing, and Dec Wojciech. Ted Lemon and Julien Laganier receive special thanks for their contributions to security considerations.

This document was prepared using xml2rfc template and the related web-tool.

9. IANA Considerations

This memo includes a new DHCPv6 option that requires allocation of a new code point.

10. Security Considerations

It is possible that attackers might try to utilize `OPTION_DNS_SERVER_SELECT` option to redirect some or all DNS queries sent by a resolver to undesired destinations. The purpose of an attack might be denial-of-service, preparation for man-in-the-middle attack, or something akin.

Attackers might try to lure specific traffic by advertising DNS suffixes and prefixes from very small to very large scope or simply by trying to place attacker's DNS server as the highest priority default server.

The main countermeasure against these attacks is to use this option only when safe to do so, see section 4.3 for details. The safest approach is for nodes to implement validating DNSSEC aware resolvers. Trusting on validation done by a DNS server is a possibility only if a host trusts the DNS server and can use a secure channel for DNS messages.

Decision on trust levels of network interfaces depends very much on deployment scenario and types of network interfaces. For example, unmanaged WLAN may be considered less trustworthy than managed cellular or VPN connections.

A node that accepts DNS server selection rules from non-trusted interfaces and implements DNSSEC validation **SHOULD** send queries also to (all) other known DNS servers in case a non-validatable response is received from the preferred DNS server. This protects against possible redirection attacks.

11. References

11.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2317] Eidnes, H., de Groot, G., and P. Vixie, "Classless IN-ADDR.ARPA delegation", BCP 20, RFC 2317, March 1998.
- [RFC3152] Bush, R., "Delegation of IP6.ARPA", BCP 49, RFC 3152, August 2001.
- [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.

- [RFC3396] Lemon, T. and S. Cheshire, "Encoding Long Options in the Dynamic Host Configuration Protocol (DHCPv4)", RFC 3396, November 2002.
- [RFC3736] Droms, R., "Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6", RFC 3736, April 2004.
- [RFC4242] Venaas, S., Chown, T., and B. Volz, "Information Refresh Time Option for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 4242, November 2005.

11.2. Informative References

- [I-D.ietf-mif-problem-statement]
Blanchet, M. and P. Seite, "Multiple Interfaces and Provisioning Domains Problem Statement",
draft-ietf-mif-problem-statement-15 (work in progress),
May 2011.
- [I-D.ietf-v6ops-multihoming-without-nat66]
Troan, O., Miles, D., Matsushima, S., Okimoto, T., and D. Wing, "IPv6 Multihoming without Network Address Translation",
draft-ietf-v6ops-multihoming-without-nat66-00 (work in progress), December 2010.
- [I-D.wing-behave-dns64-config]
Wing, D., "IPv6-only and Dual Stack Hosts on the Same Network with DNS64", draft-wing-behave-dns64-config-03
(work in progress), February 2011.
- [RFC3397] Aboba, B. and S. Cheshire, "Dynamic Host Configuration Protocol (DHCP) Domain Search Option", RFC 3397, November 2002.
- [RFC3646] Droms, R., "DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3646, December 2003.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, November 2005.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, October 2005.
- [RFC5006] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli, "IPv6 Router Advertisement Option for DNS Configuration", RFC 5006, September 2007.

[RFC6147] Bagnulo, M., Sullivan, A., Matthews, P., and I. van Beijnum, "DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", RFC 6147, April 2011.

Appendix A. Best Current Practice for DNS server selection

On some local namespace deployments explicit policies for DNS server selection are not available. This section describes ways for hosts to mitigate the problem by sending wide-spread queries and by utilizing possibly existing indirect information elements as hints.

A.1. Sending queries out on multiple interfaces in parallel

A possible current practice is to send DNS queries out of multiple interfaces and pick up the best out of the received responses. A host SHOULD implement DNSSEC in order to be able to reject responses that cannot be validated. Selection between legitimate answers is implementation specific, but replies from trusted servers should be preferred.

A downside of this approach is increased consumption of resources. Namely power consumption if an interface, e.g. wireless, has to be brought up just for the DNS query that could have been resolved also via cheaper interface. Also load on DNS servers is increased. However, local caching of results mitigates these problems, and a node might also learn interfaces that seem to be able to provide 'better' responses than other and prefer those - without forgetting fallback required for cases when node is connected to more than one network using local namespaces.

Another downside is revealing to all DNS servers the names a host is connecting to. For example, a DNS server of a public hotspot could learn all the private names host is trying to connect on other interfaces.

A.2. Search list option for DNS forward lookup decisions

A host can learn the special DNS suffixes of attached network interfaces from DHCP search list options; DHCPv4 Domain Search Option number 119 [RFC3397] and DHCPv6 Domain Search List Option number 24 [RFC3646]. The host behavior is very similar as is illustrated in the example at section 3.3. While these DHCP options are not intended to be used in DNS server selection, they may be used by the host as hints for smarter DNS server prioritization purposes in order to increase likelihood of fast and successful DNS query.

Overloading of existing DNS search list options is not without problems: resolvers would obviously use the DNS suffixes learned from search lists also for name resolution purposes. This may not be a problem in deployments where DNS search list options contain few DNS suffixes like 'example.com, private.example.com', but can become a problem if many suffixes are configured.

A.3. More specific routes for reverse lookup decision

[RFC4191] defines how more specific routes can be provisioned for hosts. This information is not intended to be used in DNS server selection, but nevertheless a host can use this information as a hint about which interface would be best to try first for reverse lookup procedures. A DNS server configured via the same interface as more specific routes is more likely capable to answer reverse lookup questions correctly than DNS server of an another interface. The likelihood of success is possibly higher if DNS server address is received in the same RA [RFC5006] as the more specific route information.

A.4. Longest matching prefix for reverse lookup decision

A host may utilize the longest matching prefix approach when deciding which DNS server to contact for reverse lookup purposes. Namely, the host may send a DNS query to a DNS server learned over an interface having longest matching prefix to the address being queried. This approach can help in cases where ULA [RFC4193] addresses are used and when the queried address belongs to a host or server within the same network (for example intranet).

Appendix B. DNSSEC and multiple answers validating with different trust anchors

When validating DNS answers with DNSSEC, a validator might order the list of trust anchors it uses to start validation chains, in terms of the host's preferences for those trust anchors. A host could use this ability in order to select among alternative DNS results from different interfaces. Suppose that a host has a trust anchor for the public DNS root, and also has a special-purpose trust anchor for example.com. An answer is received on interface i1 for www.example.com, and the validation for that succeeds by using the public trust anchor. Also, an answer is received on interface i2 for www.example.com, and the validation for that succeeds by using the trust anchor for example.com. In this case, the host has evidence for relying on i2 for answers in the example.com zone.

Authors' Addresses

Teemu Savolainen
Nokia
Hermiankatu 12 D
TAMPERE, FI-33720
FINLAND

Email: teemu.savolainen@nokia.com

Jun-ya Kato
NTT
9-11, Midori-Cho 3-Chome Musashino-Shi
TOKYO, 180-8585
JAPAN

Email: kato@syce.net

Ted Lemon
Nominum, Inc.
2000 Seaport Boulevard
Redwood City, CA 94063
USA

Phone: +1 650 381 6000
Email: Ted.Lemon@nominum.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: November 10, 2011

M. Blanchet
Viagenie
P. Seite
France Telecom - Orange
May 9, 2011

Multiple Interfaces and Provisioning Domains Problem Statement
draft-ietf-mif-problem-statement-15.txt

Abstract

This document describes issues encountered by a node attached to multiple provisioning domains. This node receives configuration information from each of its provisioning domains where some configuration objects are global to the node, others are local to the interface. Issues such as selecting the wrong interface to send traffic happen when conflicting node-scoped configuration objects are received and inappropriately used. Moreover, other issues are the result of simultaneous attachment to multiple networks, such as domain selection or addressing and naming space overlaps, regardless of the provisioning mechanism. While multiple provisioning domains are typically seen on nodes with multiple interfaces, this document also discusses single interface nodes situation.

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 November 10, 2011.

Copyright Notice

Copyright (c) 2011 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. Scope and Existing Work	4
3.1. Below IP Interaction	4
3.2. MIF node Characterization	4
3.3. Hosts Requirements	5
3.4. Mobility and other IP protocols	6
3.5. Address Selection	6
3.6. Finding and Sharing IP Addresses with Peers	6
3.7. Provisioning domain selection	7
3.8. Session management	7
3.9. Socket API	8
4. MIF Issues	9
4.1. DNS resolution issues	9
4.2. Node Routing	11
4.3. Policies conflict	12
4.4. Session management	12
4.5. Single Interface on Multiple Provisioning Domains	13
5. Underlying problems and causes	13
6. Security Considerations	15
7. IANA Considerations	16
8. Authors	16
9. Acknowledgements	16
10. Informative References	16
Authors' Addresses	20

1. Introduction

A multihomed node may have multiple provisioning domains (via physical and/or virtual interfaces). For example, a node may be simultaneously connected to a wired Ethernet LAN, a 802.11 LAN, a 3G cell network, one or multiple VPN connections or one or multiple tunnels (automatic or manual). Current laptops and smartphones typically have multiple access network interfaces and, thus, are often connected to different provisioning domains.

A multihomed node receives configuration information from each of its attached networks, through various mechanisms such as DHCPv4 [RFC2131], DHCPv6 [RFC3315], PPP [RFC1661] and IPv6 Router Advertisements [RFC4861]. Some received configuration objects are specific to an interface such as the IP address and the link prefix. Others are typically considered by implementations as being global to the node, such as the routing information (e.g. default gateway), DNS servers IP addresses, and address selection policies, herein named "node-scoped".

When the received node-scoped configuration objects have different values from each provisioning domains, such as different DNS servers IP addresses, different default gateways or different address selection policies, the node has to decide which one to use or how it will merge them.

Other issues are the result of simultaneous attachment to multiple networks, such as addressing and naming space overlaps, regardless of the provisioning mechanism.

The following sections define the multiple interfaces (MIF) node, the scope of this work, describe related work, list issues and then summarize the underlying problems.

A companion document [I-D.ietf-mif-current-practices] discusses some current practices of various implementations dealing with MIF.

2. Terminology

Administrative domain

A group of hosts, routers, and networks operated and managed by a single organization [RFC1136].

Provisioning domain

A set of consistent configuration information (e.g. Default router, Network prefixes, DNS,...) and the corresponding interface. One administrative domain may have multiple provisioning domains. Successful attachment to the provisioning domain implies that the terminal attaches to the corresponding interface with appropriate configuration information.

Reference to IP version

When a protocol keyword such as IP, PPP, DHCP is used in this document without any reference to a specific IP version, then it implies both IPv4 and IPv6. A specific IP version keyword such as DHCPv4 or DHCPv6 is meant to be specific to that IP version.

3. Scope and Existing Work

This section describes existing related work and defines the scope of the problem.

3.1. Below IP Interaction

Some types of interfaces have link layer characteristics which may be used in determining how multiple provisioning domain issues will be dealt with. For instance, link layers may have authentication and encryption characteristics which could be used as criteria for interface selection. However, network discovery and selection on lower layers as defined by [RFC5113] is out of scope of this document. Moreover, interoperability with lower layer mechanisms such as services defined in IEEE 802.21, which aims at facilitating handover between heterogeneous networks [MIH], is also out of scope.

Some mechanisms (e.g., based on a virtual IP interface) allow sharing a single IP address over multiple interfaces to networks with disparate access technologies. From the IP stack view on the node, there is only a single interface and single IP address. Therefore, this situation is out of scope of this current problem statement. Furthermore, link aggregation done under IP where a single interface is shown to the IP stack is also out of scope.

3.2. MIF node Characterization

A MIF node has the following characteristics:

- o A [RFC1122] IPv4 and/or [RFC4294] IPv6 compliant node

- o A MIF node is configured with more than one IP addresses (excluding loopback and link-local)
- o A MIF node can attach to more than one provisioning domains, as presented to the IP stack.
- o The interfaces may be virtual or physical.
- o Configuration objects come from one or more administrative domains.
- o The IP addresses may be from the same or from different address families, such as IPv4 and IPv6.
- o Communications using these IP addresses may happen simultaneously and independently.
- o Some communications using these IP addresses are possible on all the provisioning domains, while some are only possible on a smaller set of the provisioning domains.
- o While the MIF node may forward packets between its interfaces, forwarding packets is not taken into account in this definition and is out of scope for this document.

3.3. Hosts Requirements

The requirements for Internet Hosts [RFC1122] describe the multihomed node as if it has multiple IP addresses, which may be associated with one or more physical interfaces connected to the same or different networks.

The requirements states that The node maintains a route cache table where each entry contains the local IP address, the destination IP address, Differentiated Services Code Point and Next-hop gateway IP address. The route cache entry would have data about the properties of the path, such as the average round-trip delay measured by a transport protocol. Nowadays, implementations are not caching these informations.

[RFC1122] defines two host models:

- o The "Strong" host model defines a multihomed host as a set of logical hosts within the same physical host. In this model a packet must be sent on an interface that corresponds to the source address of that packet.
- o The "Weak" host model describes a host that has some embedded gateway functionality. In the weak host model, the host can send and receive packets on any interface.

The multihomed node computes routes for outgoing datagrams differently depending on the model. Under the strong model, the route is computed based on the source IP address, the destination IP address and the Differentiated Services Code Point. Under the weak model, the source IP address is not used, but only the destination IP address and the Differentiated Services Code Point.

3.4. Mobility and other IP protocols

The scope of this document is only about nodes implementing [RFC1122] for IPv4 and [RFC4294] for IPv6 without additional features or special-purpose support for transport layers, mobility, multi-homing, or identifier-locator split mechanisms. Dealing with multiple interfaces with such mechanisms is related but considered as a separate problem and is under active study elsewhere in the IETF [RFC4960], [RFC5206], [RFC5533], [RFC5648], [I-D.ietf-mptcp-architecture].

When an application is using one interface while another interface with better characteristics becomes available, the ongoing application session could be transferred to the newly enabled interface. However, in some cases, the ongoing session shall be kept on the current interface while initiating the new sessions on the new interface. The problem of the interface selection is within the MIF scope and may leverage specific node functions (Section 3.8). However, if transfer of IP session is required, IP mobility mechanisms, such as [RFC3775], shall be used.

3.5. Address Selection

The Default Address Selection specification [RFC3484] defines algorithms for source and destination IP address selections. It is mandatory to be implemented in IPv6 nodes, which also means dual-stack nodes. A node-scoped policy table managed by the IP stack is defined. Mechanisms to update the policy table are being defined [I-D.ietf-6man-addr-select-sol] to update the policy table.

Issues on using the Default Address Selection were found in [RFC5220] and [RFC5221] in the context of multiple prefixes on the same link.

3.6. Finding and Sharing IP Addresses with Peers

Interactive Connectivity Establishment (ICE [RFC5245]) is a technique for NAT traversal for UDP-based (and TCP) media streams established by the offer/answer model. The multiplicity of IP addresses, ports and transport in SDP offers are tested for connectivity by peer-to-peer connectivity checks. The result is candidate IP addresses and ports for establishing a connection with the other peer. However, ICE does not solve issues when incompatible configuration objects are received on different interfaces.

Some application protocols do referrals of IP addresses, port numbers and transport for further exchanges. For instance, applications can provide reachability information to itself or to a third party. The general problem of referrals is related to the multiple interface

problem, since, in this context, referrals must provide consistent information depending on which provisioning domain is used. Referrals are discussed in [I-D.carpenter-referral-ps] and [I-D.ietf-shim6-app-refer].

3.7. Provisioning domain selection

In a MIF context, the node may handle simultaneously multiple domains with disparate characteristics, especially when supporting multiple access technologies. Selection is simple if the application is restricted to one specific provisioning domain: the application must start on the default provisioning domain if available, otherwise the application does not start. However, if the application can be run on several provisioning domains, the selection problem can be difficult.

There is no standard method for selecting a provisioning domain but some recommendation exist while restricting the scope to the interface selection problem. For example, [TS23.234] proposes a default mechanism for the interface selection. This method uses the following information (non exhaustive list):

- o preferences provided by the user,
- o policies provided by network operator,
- o quality of the radio link,
- o network resource considerations (e.g. available QoS, IP connectivity check,...),
- o the application QoS requirements in order to map applications to the best interface

However, [TS23.234] is designed for a specific multiple-interfaces use-case. A generic way to handle these characteristics is yet to be defined.

3.8. Session management

Some implementations, specially in the mobile world, rely on higher-level session manager, also named connection manager, to deal with issues brought by simultaneous attachment to multiple provisioning domains. Typically, the session manager may deal with the selection of the interface, and/or the provisioning domain, on behalf to the applications, or tackle with complex issues such as policies conflict resolution (Section 4.3). As discussed previously in Section 3.7, the session manager may encounter difficulties because of multiple and diverse criteria.

Session managers usually leverage the link-layer interface to gather information (e.g lower layer authentication and encryption methods,

see Section 3.1) and/or for control purpose. Such link-layer interface may not provide all required services to make a proper decision (e.g. interface selection). Some OS, or terminals, already implement session managers [I-D.ietf-mif-current-practices] and vendor-specific platforms sometimes provides specific socket API (Section 3.9) a session manager can use. However, the generic architecture of a session manager and its associated API are not currently standardized, so session management behavior may differ between OS and platforms.

Multiple interfaces management sometimes relies on a virtual interface. For instance, virtual interface allows to support multihoming, inter-technology handovers and IP flow mobility in a Proxy Mobile IPv6 network [I-D.ietf-netext-logical-interface-support]. This virtual interface allows a multiple-interfaces node sharing a set of IP addresses on multiple physical interfaces and can also add benefits to multi-access scenarios such as 3GPP Multi Access PDN Connectivity [TS23.402]. In most cases, the virtual interface will map several physical network interfaces and the session manager should control both, the configuration of each one of these virtual and physical interfaces, as well as the mapping between the virtual and the sub-interfaces.

In multiple interfaces situation, active application sessions should survive to path failures. Here, the session manager may come into play but only relying on existing mechanisms to manage multipath (MPTCP [I-D.ietf-mptcp-architecture]) or failover (MIP6 [RFC3775], SHIM6 [RFC5533]). Description of interaction between these mechanisms and the session manager is out of the scope of this document.

3.9. Socket API

An Application Programming Interface (API) may expose objects that user applications, or session managers, use for dealing with multiple interfaces. For example, [RFC3542] defines how an application using the Advanced sockets API specifies the interface or the source IP address, through a simple bind() operation or with the IPV6_PKTINFO socket option.

Other APIs have been defined to solve similar issues to MIF. For instance, [RFC5014] defines an API to influence the default address selection mechanism by specifying attributes of the source addresses it prefers. [I-D.ietf-shim6-multihome-shim-api] gives another example, in a multihoming context, by defining a socket API enabling interactions between applications and the multihoming shim layer for advanced locator management, and access to information about failure detection and path exploration.

4. MIF Issues

This section describes the various issues when using a MIF node that has already received configuration objects from its various provisioning domains or when multiple interfaces are used and results in wrong domain selection, addressing or naming space overlaps. They occur, for example, when:

1. one interface is on the Internet and one is on a corporate private network. The latter may be through VPN.
2. one interface is on one access network (i.e. wifi) and the other one is on another access network (3G) with specific services.

4.1. DNS resolution issues

A MIF node (M1) has an active interface(I1) connected to a network (N1) which has its DNS server (S1) and another active interface (I2) connected to a network (N2) which has its DNS server (S2). S1 serves with some private namespace "private.example.com". The user or the application uses a name "a.private.example.com" which is within the private namespace of S1 and only resolvable by S1. Any of the following situations may occur:

1. M1 stack, based on its routing table, uses I2 to reach S1 to resolve "a.private.example.com". M1 never reaches S1. The name is not resolved.
2. M1 keeps only one set of DNS server addresses from the received configuration objects and kept S2 address. M1 sends the forward DNS query for a.private.example.com to S2. S2 responds with an error for an non-existent domain (NXDOMAIN). The name is not resolved. This issue also arises when performing reverse DNS lookup. In the same situation, the reverse DNS query fails.
3. M1 keeps only one set of DNS server addresses from the received configuration objects and kept S2 address. M1 sends the DNS query for a.private.example.com to S2. S2 asks its upstream DNS and gets an IP address for a.private.example.com. However, the IP address is not the same one S1 would have given. Therefore, the application tries to connect to the wrong destination node, or to the wrong interface of the latter, which may imply security issues or result in lack of service.
4. S1 or S2 has been used to resolve "a.private.example.com" to an [RFC1918] address. Both N1 and N2 are [RFC1918] addressed networks. If addresses overlap, traffic may be sent using the wrong interface. This issue is not related to receiving multiple configuration objects, but to an address overlap between interfaces or attaching networks.

5. M1 has resolved an FQDN to locally valid IP address when connected to N1. If the node loses connection to N1, the node may try to connect, via N2, to the same IP address as earlier, but as the address was only locally valid, connection setup fails. Similarly, M1 may have received NXDOMAIN for an FQDN when connected to N1. After detachment from N1, the node should not assume the FQDN continues to be nonexistent on N2.
6. M1 requests AAAA record from a DNS server on a network that uses protocol translators and DNS64 [I-D.ietf-behave-dns64]. If the M1 receives synthesized AAAA record, it is guaranteed to be valid only on the network it was learned from. If the M1 uses synthesized AAAA on any other network interface, traffic may be lost, dropped or forwarded to the wrong network.

Some networks requires the user to authenticate on a captive web portal before providing Internet connectivity. If this redirection is achieved by modifying the DNS reply, specific issues may occur. Consider a MIF node (M1) with an active interface(I1) connected to a network (N1), which has its DNS server (S1), and another active interface (I2) connected to a network (N2), which has its DNS server (S2). Until the user has not authenticated, S1 is configured to respond to any A or AAAA record query with the IP address of a captive portal, so as to redirect web browsers to an access control portal web page. This captive portal can be reached only via I1. When the user has authenticated to the captive portal, M1 can resolve an FQDN when connected to N1. However, if the address is only locally valid on N1, any of the issue described above may occur. When the user has not authenticated, any of the following situations may occur:

1. M1 keeps only one set of DNS server addresses from the received configuration objects and kept S2 address. M1 sends the forward DNS query for a.example.com to S2. S2 responds with the correct answer, R1. M1 attempts to contact R1 by way of I1. The connection fails. Or, the connection succeeds, bypassing the security policy on N1, possibly exposing the owner of M1 to prosecution.
2. M1 keeps only one set of DNS server addresses from the received configuration objects and kept S1 address. M1 sends the DNS query for a.example.com to S1. S1 provides the address of its captive portal. M1 attempts to contact this IP address using I1. The application fails to connect, resulting in lack of service. Or, the application succeeds in connecting, but connects to the captive portal rather than the intended destination, resulting in lack of service (i.e. IP connectivity check issue described in Section 4.4).

4.2. Node Routing

A MIF node (M1) has an active interface(I1) connected to a network (N1) and another active interface (I2) connected to a network (N2). The user or the application is trying to reach an IP address (IP1). Any of the following situations may occur:

1. For IP1, M1 has one default route (R1) via network (N1). To reach IP1, M1 stack uses R1 and sends through I1. If IP1 is only reachable by N2, IP1 is never reached or is not the right target.
2. For the IP1 address family, M1 has one default route (R1, R2) per network (N1, N2). IP1 is reachable by both networks, but N2 path has better characteristics, such as better round-trip time, least cost, better bandwidth, etc.... These preferences could be defined by user, provisioned by the network operator, or else. M1 stack uses R1 and tries to send through I1. IP1 is reached but the service would be better by I2.
3. For the IP1 address family, M1 has a default route (R1), a specific X.0.0.0/8 route R1B (for example but not restricted to RFC1918 prefix) to N1 and a default route (R2) to N2. IP1 is reachable by N2 only, but the prefix (X.0.0.0/8) is used in both networks. Because of the most specific route R1B, M1 stack sends through I2 and never reach the target.

A MIF node may have multiple routes to a destination. However, by default, it does not have any hint concerning which interface would be the best to use for that destination. The first-hop selection may leverage on local routing policy, allowing some actors (e.g. network operator or service provider) to influence the routing table, i.e. make decision regarding which interface to use. For instance, a user on such multihomed node might want a local policy to influence which interface will be used based on various conditions. Some SDOs have defined policy-based routing selection mechanisms. For instance, the Access Network Discovery and Selection Function (ANDSF) [TS23.402] provides inter-systems routing policies to terminals with both a 3GPP and non-3GPP interfaces. However, the routing selection may still be difficult, due to disjoint criteria as discussed in Section 3.8. Moreover, information required to make the right decision may not be available. For instance, interfaces to lower layer may not provide all required hints to the selection (e.g. information on interface quality).

A node usually has a node-scoped routing table. However, a MIF node is connected to multiple provisioning domains; if each of these domains pushes routing policies to the node, then conflicts between policies may happen and the node has no easy way to merge or reconcile them.

On a MIF node, some source addresses are not valid if used on some interfaces. For example, an RFC1918 source address might be appropriate on the VPN interface but not on the public interface of the MIF node. If the source address is not chosen appropriately, then packets may be filtered in the path if source address filtering is in place ([RFC2827], [RFC3704]) and reply packets may never come back to the source.

4.3. Policies conflict

The distribution of configuration policies (e.g. address selection, routing, DNS selection...) to end nodes is being discussed (e.g. ANDSF in [TS23.402], [I-D.ietf-mif-dhcpv6-route-option]). If implemented in multiple provisioning domains, such mechanisms may conflict and bring issues to the multihomed node. Considering a MIF node (M1) with an active interface (I1) connected to a network (N1) and another active interface (I2) connected to a network (N2), the following conflicts may occur:

1. M1 receives from both networks (N1 and N2) an update of its default address selection policy. However, the policies are specific to each network. The policies are merged by M1 stack. Based on the merged policy, the chosen source address is from N1 but packets are sent to N2. The source address is not reachable from N2, therefore the return packet is lost. Merging address selection policies may have important impacts on routing.
2. A node usually has a node-scoped routing table. However, each of the connected provisioning domains (N1 and N2) may push routing policies to the node, then conflicts between policies may happen and the node has no easy way to merge or reconcile them.
3. M1 receives from one of the network an update of its access selection policy, e.g. via the 3GPP/ANDSF [TS23.402]. However, the policy is in conflict with the local policy (e.g. user defined, or default OS policy). Assuming that the network provides list of overloaded access network, if the policy sent by the network is ignored, packet may be sent to an access network with poor quality of communication.

4.4. Session management

Consider that a node has selected an interface and managed to configure it (i.e. the node obtained a valid IP address from the network). However, the Internet connectivity is not available. The problem could be due to the following reasons:

1. The network requires a web-based authentication (e.g. the access network is a WiFi Hot Spot). In this case the user can only access to a captive portal. For instance, the network may perform HTTP redirection or modify DNS behaviour (Section 4.1) until the user has not authenticated.

2. IP interface is configured active but layer 2 is so poor (e.g. poor radio condition) that no layer 3 traffic can succeed.

In this situation, the session management should be able to perform IP connectivity checks before selecting an interface.

Session issues may also arise when the node discovers a new provisioning domain. Consider a MIF node (M1) has an active interface(I1) connected to a network (N1) where an application is running a TCP session. A new network (N2) becomes available. If N2 is selected (e.g. because of better quality of communication), M1 gets IP connectivity to N2 and updates the routing table priority. So, if no specific route to the correspondent node and if the node implements the weak host model [RFC1122], the TCP connection breaks as next hop changes. In order to continue communicating with the correspondent node, M1 should try to re-connect the server via N2. In some situation, it could be preferable to maintain current sessions on N1 while new sessions start on N2.

4.5. Single Interface on Multiple Provisioning Domains

When a node using a single interface is connected to multiple networks, such as different default routers, similar issues as described above happen. Even with a single interface, a node may wish to connect to more than one provisioning domain: that node may use more than one IP source address and may have more than one default router. The node may want to access services that can only be reached using one of the provisioning domain. In this case, it needs to use the right outgoing source address and default gateway to reach that service. In this situation, that node may also need to use different DNS servers to get domain names in those different provisioning domains.

5. Underlying problems and causes

This section lists the underlying problems, and their causes, which lead to the issues discussed in the previous section. The problems can be divided into five categories: 1) Configuration 2) DNS resolution 3) Routing 4) Address selection and 5) session management and API. They are shown as below:

1. Configuration. In a MIF context, configuration information specific to a provisioning domain may be ignored because:
 1. Configuration objects (e.g. DNS servers, NTP servers, ...) are node-scoped. So the IP stack is not able to maintain the mapping between information and corresponding provisioning domain.

2. Same configuration objects (e.g. DNS server addresses, NTP server addresses, ...) received from multiple provisioning domains may be overwritten.
3. Host implementations usually do not keep separate network configuration (such as DNS server addresses) per provisioning domain.
2. DNS resolution
 1. Some FQDN can be resolvable only by sending queries to the right server (e.g. intranet services). However, DNS query could be sent to the wrong interface because DNS server addresses may be node-scoped.
 2. A DNS answer may be only valid on a specific provisioning domain but applications may not be aware of that mapping because DNS answers may not be kept with the provisioning from which the answer comes from.
3. Routing
 1. In the MIF context, routing information could be specific to each interface. This could lead to routing issue because, in current node implementations, routing tables are node-scoped.
 2. Current node implementations do not take into account the Differentiated Services Code Point or path characteristics in the routing table.
 3. Even if implementations take into account path characteristics, the node has no way to properly merge or reconcile the provisioning domain preferences.
 4. a node attached to multiple provisioning domain could be provided with incompatible selection policies. If the different actors (e.g. user and network operator) are allowed to provide their own policies, the node has no way to properly merge or reconcile multiple selection policies.
 5. The problem of first hop selection could not be solved via configuration (Section 3.7), and may leverage on sophisticated and specific mechanisms (Section 3.8).
4. Address selection
 1. Default Address Selection policies may be specific to their corresponding provisioning domain. However, a MIF node may not be able to manage per-provisioning domain address selection policies because default Address Selection policy is node-scoped.
 2. On a MIF node, some source addresses are not valid if used on some interfaces or even on some default routers on the same interface. In this situation, the source address should be taken into account in the routing table; but current node implementations do not support such a feature.
 3. Source address or address selection policies could be specified by applications. However, there is no advanced APIs to allow applications realizing such operations.

5. Session management and API

1. Some implementations, specially in the mobile world, have higher-level API and/or session manager (aka connection manager) to address MIF issues. These mechanisms are not standardized and do not necessarily behave the same way across different OS, and/or platforms, in the presence of the MIF problems. This lack of consistency is an issue for user and operator who could experience different session manager behaviors depending on the terminal.
2. Session managers usually leverage on interface to link layer to gather information (e.g lower layer authentication and encryption methods) and/or for control purpose. However, such link layer interface may not provide all required services (e.g. may not provide all information allowing to make a proper interface selection).
3. A MIF node can support different session managers, which may have contradictory ways to solve the MIF issues. For instance, because of different selection algorithms, two different session managers could select different domains in a same context. Or, when dealing with different domain selection policies, a session manager may give precedence to user policy while another could favor mobile operator policy.
4. When host routing is updated and if weak host model is supported, ongoing TCP sessions may break if routes changes for these sessions. When TCP sessions should be bound to the interface, the strong host model should be used.
5. When provided by different actors (e.g. user, network, default-OS), policies may conflict and, thus, need to be reconciliated at the host level. Policy conflict resolution may impact other functions (e.g. naming, routing).
6. Even if the node has managed to configure an interface, Internet connectivity could be not available. It could be due to an access control function coming into play above the layer 3, or because of poor layer 2 conditions. IP connectivity check should be performed before selecting an interface.

6. Security Considerations

The problems discussed in this document have security implications, such as when the packets sent on the wrong interface might be leaking some confidential information. Configuration parameters from one provisioning domain could cause a denial of service on another provisioning domain (e.g. DNS issues). Moreover, the undetermined behavior of IP stacks in the multihomed context bring additional threats where an interface on a multihomed node might be used to conduct attacks targeted to the networks connected by the other

interfaces.corrupted provisioning domain selection policy may induce a node to make decisions causing certain traffic to be forwarded to the attacker.

Additional security concerns are raised by possible future mechanisms that provide additional information to the node so that it can make a more intelligent decision with regards to the issues discussed in this document. Such future mechanisms may themselves be vulnerable and may not be easy to protect in the general case.

7. IANA Considerations

This document has no actions for IANA.

8. Authors

This document is a joint effort with authors of the MIF requirements draft [I-D.yang-mif-req]. The authors of this document, in alphabetical order, include: Marc Blanchet, Jacqni Qin, Pierrick Seite, Carl Williams and Peny Yang.

9. Acknowledgements

The initial Internet-Drafts prior to the MIF working group and the discussions during the MIF BOF meeting and on the mailing list around the MIF charter scope on the mailing list brought very good input to the problem statement. This draft steals a lot of text from these discussions and initial drafts (e.g. [I-D.yang-mif-req], [I-D.hui-ip-multiple-connections-ps], [I-D.ietf-mif-dns-server-selection]). Therefore, the editor would like to acknowledge the following people (in no specific order), from which some text has been taken from: Jari Arkko, Keith Moore, Sam Hartman, George Tsirtsis, Scott Brim, Ted Lemon, Bernie Volz, Giyeong Son, Gabriel Montenegro, Julien Laganier, Teemu Savolainen, Christian Vogt, Lars Eggert, Margaret Wasserman, Hui Deng, Ralph Droms, Ted Hardie, Christian Huitema, Remi Denis-Courmont, Alexandru Petrescu, Zhen Cao, Gaetan Feige, Telemaco Melia and Juan-Carlos Zuniga. Sorry if some contributors have not been named.

10. Informative References

[I-D.carpenter-referral-ps]
Carpenter, B., Jiang, S., and Z. Cao, "Problem Statement for Referral", draft-carpenter-referral-ps-02 (work in

progress), February 2011.

[I-D.hui-ip-multiple-connections-ps]

Hui, M. and H. Deng, "Problem Statement and Requirement of Simple IP Multi-homing of the Host", draft-hui-ip-multiple-connections-ps-02 (work in progress), March 2009.

[I-D.ietf-6man-addr-select-sol]

Matsumoto, A., Fujisaki, T., and R. Hiromi, "Solution approaches for address-selection problems", draft-ietf-6man-addr-select-sol-03 (work in progress), March 2010.

[I-D.ietf-behave-dns64]

Bagnulo, M., Sullivan, A., Matthews, P., and I. Beijnum, "DNS64: DNS extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", draft-ietf-behave-dns64-11 (work in progress), October 2010.

[I-D.ietf-mif-current-practices]

Wasserman, M. and P. Seite, "Current Practices for Multiple Interface Hosts", draft-ietf-mif-current-practices-11 (work in progress), April 2011.

[I-D.ietf-mif-dhcpv6-route-option]

Dec, W., Mrugalski, T., Sun, T., and B. Sarikaya, "DHCPv6 Route Option", draft-ietf-mif-dhcpv6-route-option-01 (work in progress), March 2011.

[I-D.ietf-mif-dns-server-selection]

Savolainen, T., Kato, J., and T. Lemon, "Improved DNS Server Selection for Multi-Homed Nodes", draft-ietf-mif-dns-server-selection-02 (work in progress), April 2011.

[I-D.ietf-mptcp-architecture]

Ford, A., Raiciu, C., Handley, M., Barre, S., and J. Iyengar, "Architectural Guidelines for Multipath TCP Development", draft-ietf-mptcp-architecture-05 (work in progress), January 2011.

[I-D.ietf-netext-logical-interface-support]

Melia, T. and S. Gundavelli, "Logical Interface Support for multi-mode IP Hosts", draft-ietf-netext-logical-interface-support-02 (work in progress), March 2011.

progress), March 2011.

- [I-D.ietf-shim6-app-refer]
Nordmark, E., "Shim6 Application Referral Issues",
draft-ietf-shim6-app-refer-00 (work in progress),
July 2005.
- [I-D.ietf-shim6-multihome-shim-api]
Komu, M., Bagnulo, M., Slavov, K., and S. Sugimoto,
"Socket Application Program Interface (API) for
Multihoming Shim", draft-ietf-shim6-multihome-shim-api-17
(work in progress), April 2011.
- [I-D.yang-mif-req]
Yang, P., Seite, P., Williams, C., and J. Qin,
"Requirements on multiple Interface (MIF) of simple IP",
draft-yang-mif-req-00 (work in progress), March 2009.
- [MIH] IEEE, "IEEE Standard for Local and Metropolitan Area
Networks - Part 21: Media Independent Handover Services,
IEEE LAN/MAN Std 802.21-2008, January 2009.", 2010.
- [RFC1122] Braden, R., "Requirements for Internet Hosts -
Communication Layers", STD 3, RFC 1122, October 1989.
- [RFC1136] Hares, S. and D. Katz, "Administrative Domains and Routing
Domains: A model for routing in the Internet", RFC 1136,
December 1989.
- [RFC1661] Simpson, W., "The Point-to-Point Protocol (PPP)", STD 51,
RFC 1661, July 1994.
- [RFC1918] Rekhter, Y., Moskowitz, R., Karrenberg, D., Groot, G., and
E. Lear, "Address Allocation for Private Internets",
BCP 5, RFC 1918, February 1996.
- [RFC2131] Droms, R., "Dynamic Host Configuration Protocol",
RFC 2131, March 1997.
- [RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering:
Defeating Denial of Service Attacks which employ IP Source
Address Spoofing", BCP 38, RFC 2827, May 2000.
- [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.
- [RFC3484] Draves, R., "Default Address Selection for Internet

Protocol version 6 (IPv6)", RFC 3484, February 2003.

- [RFC3542] Stevens, W., Thomas, M., Nordmark, E., and T. Jinmei, "Advanced Sockets Application Program Interface (API) for IPv6", RFC 3542, May 2003.
- [RFC3704] Baker, F. and P. Savola, "Ingress Filtering for Multihomed Networks", BCP 84, RFC 3704, March 2004.
- [RFC3775] Johnson, D., Perkins, C., and J. Arkko, "Mobility Support in IPv6", RFC 3775, June 2004.
- [RFC4294] Loughney, J., "IPv6 Node Requirements", RFC 4294, April 2006.
- [RFC4477] Chown, T., Venaas, S., and C. Strauf, "Dynamic Host Configuration Protocol (DHCP): IPv4 and IPv6 Dual-Stack Issues", RFC 4477, May 2006.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.
- [RFC5014] Nordmark, E., Chakrabarti, S., and J. Laganier, "IPv6 Socket API for Source Address Selection", RFC 5014, September 2007.
- [RFC5113] Arkko, J., Aboba, B., Korhonen, J., and F. Bari, "Network Discovery and Selection Problem", RFC 5113, January 2008.
- [RFC5206] Nikander, P., Henderson, T., Vogt, C., and J. Arkko, "End-Host Mobility and Multihoming with the Host Identity Protocol", RFC 5206, April 2008.
- [RFC5220] Matsumoto, A., Fujisaki, T., Hiromi, R., and K. Kanayama, "Problem Statement for Default Address Selection in Multi-Prefix Environments: Operational Issues of RFC 3484 Default Rules", RFC 5220, July 2008.
- [RFC5221] Matsumoto, A., Fujisaki, T., Hiromi, R., and K. Kanayama, "Requirements for Address Selection Mechanisms", RFC 5221, July 2008.
- [RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT)

Traversal for Offer/Answer Protocols", RFC 5245,
April 2010.

[RFC5533] Nordmark, E. and M. Bagnulo, "Shim6: Level 3 Multihoming
Shim Protocol for IPv6", RFC 5533, June 2009.

[RFC5648] Wakikawa, R., Devarapalli, V., Tsirtsis, G., Ernst, T.,
and K. Nagami, "Multiple Care-of Addresses Registration",
RFC 5648, October 2009.

[TS23.234]
3GPP, "3GPP system to Wireless Local Area Network (WLAN)
interworking; TS 23.234", 2009.

[TS23.402]
3GPP, "Architecture enhancements for non- 3GPP accesses;
TS 23.402", 2010.

Authors' Addresses

Marc Blanchet
Viagenie
2875 boul. Laurier, suite D2-630
Quebec, QC G1V 2M2
Canada

Email: Marc.Blanchet@viagenie.ca
URI: <http://viagenie.ca>

Pierrick Seite
France Telecom - Orange
4, rue du Clos Courtel, BP 91226
Cesson-Sevigne 35512
France

Email: pierrick.seite@orange-ftgroup.com

Multiple Interfaces (Mif)
Internet-Draft
Intended status: Experimental
Expires: May 3, 2012

J. Korhonen
Nokia Siemens Networks
T. Savolainen
Nokia
Y. Ding, Ed.
University of Helsinki
October 31, 2011

Controlling Traffic Offloading Using Neighbor Discovery Protocol
draft-korhonen-mif-ra-offload-03.txt

Abstract

This specification defines an extension to IPv6 Neighbor Discovery Protocol, which allows management of IPv6 traffic offloading to IPv4 and moving IPv4 traffic away from a specific interface.

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 3, 2012.

Copyright Notice

Copyright (c) 2011 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. Requirements and Terminology	3
3. Problem Background	3
4. Solution	4
4.1. Neighbor Discovery Offload Option	4
4.2. Lowering IPv4 Router Preference	5
4.3. IPv4 Offloading to Specific Routes	6
4.4. IPv4 Offloading to Default Gateway	7
4.5. Offload Lifetime	7
5. Router Behavior	7
6. Host Behavior	7
7. Security Considerations	8
8. IANA Considerations	8
9. References	8
9.1. Normative References	8
9.2. Informative References	9
Appendix A. Address Selection Approach	9
A.1. Modification to Default Address Selection	9
A.2. Address selection examples	10
A.2.1. Case 1: IPv6-only cellular and IPv4-only WLAN accesses	10
A.2.2. Case 2: WLAN access with multiple prefixes	10
A.2.3. Case 3: WLAN and cellular interface with cellular's IPv4 not default route	11
A.2.4. Case 4: Dual-stack cellular access	11
A.2.5. Case 5: Dual-stack cellular and single stack WLAN	11
A.2.6. Case 6: Coexistence with RFC4191	12
Authors' Addresses	12

1. Introduction

This specification defines an extension to Neighbor Discovery Protocol [RFC4861], which allows management of IPv6 traffic offloading to IPv4 and moving IPv4 traffic away from a specific network connection.

The described solution is intended to be used during transition towards IPv6, during which time multi-interfaced hosts are often likely to have network interfaces with IPv4-only capability. A common scenario where coexistence of IPv4 and IPv6 network interfaces is expected to occur is when a smartphone has IPv6-enabled cellular connection and IPv4-only WLAN connection active at the same time.

2. Requirements and Terminology

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].

3. Problem Background

Current Internet hosts generally prefer IPv6 addresses over IPv4 addresses when performing source and destination address selections, as is recommended in [I-D.ietf-6man-rfc3484-revise].

A multi-interfaced host may have IPv6 enabled on a more 'expensive' interface and a 'cheaper' interface may have support only for IPv4. In such a scenario it might be desirable for hosts to prefer IPv4 in communication instead of IPv6.

The above mentioned scenario can become a problem, for example, when a smartphone has simultaneously IPv6-enabled cellular connection ([I-D.ietf-v6ops-3gpp-eps]) and IPv4-only WLAN connectivity active. When connecting to dual-stack capable destinations it would oftentimes be generally more efficient to use WLAN network interface. Furthermore, a cellular network operator may want hosts to offload traffic away from cellular network whenever hosts have alternate network accesses available.

Similar issue can arise also when a host has multiple interfaces with IPv4 connectivity. The interface that provides better performance at a lower price should oftentimes be used for the communication, but it may not be clear for a host which one of the available interfaces it should prefer.

4. Solution

This document introduces a new Neighbor Discovery option that a network can use to communicate the level of router's willingness to act as a router for IPv4 traffic.

The new Neighbor Discovery option was chosen to support hosts without DHCPv6 [RFC3315] support and also to work on networks not utilizing DHCPv6.

The new Neighbor Discovery option can be used together with the Route Information option defined in [RFC4191] to communicate offloading information for specific routes.

The new Neighbor Discovery option shall be phased out when IPv4 usage diminishes.

4.1. Neighbor Discovery Offload Option

This specification defines a new Neighbor Discovery [RFC4861] option called Offload (Type TBD) to be used in Router Advertisements. The option is illustrated in Figure 1. Router and hosts implementing this specification MUST understand the Offload option.

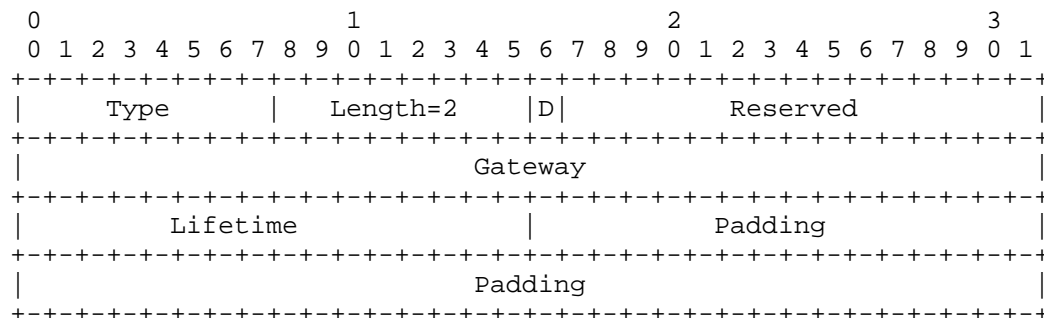


Figure 1: Router Advertisement Offload Option

Type

TBD by IANA.

Length

MUST be set to 2.

D (IPv4 Gateway Preference)

Indicates the willingness of the Dual-Stack capable router (who originated the Router Advertisement) to serve as a gateway for the IPv4 traffic. If 'D' is unset (0) then the router indicates no specific to be or not to be a gateway for IPv4 traffic. If 'D' is set (1) then the router explicitly indicates it is not willing to serve as a gateway for IPv4 traffic if there are other usable gateways present in the same or other available interfaces.

Reserved

A 15-bit unused field. It MUST be initialized to zero by the sender and MUST be ignored by the receiver.

Gateway

The address of the dual-stack router's IPv4 interface used as the next-hop from hosts point of view for sending and receiving IPv4 traffic on this link. The IPv4 address MUST belong to the same interface that originated the Router Advertisement containing this option. If the router is IPv6 only, then this field MUST be set to unspecified address (0.0.0.0) or the Neighbor Discovery Offload option MUST be omitted in all Router Advertisements originated by the router.

Lifetime

16-bit unsigned integer. The Lifetime in seconds limits the validity of state changes caused by this new option. The value of Lifetime in this option SHOULD be smaller than the value of Route Lifetime contained in the Route Information option [RFC4191], if present, in the same Router Advertisement.

Padding

The padding MUST be initialized to zero by the sender and MUST be ignored by the receiver.

The behavior of 'IPv4 Gateway Preference' (see Section 4.2) is discussed in more detail in the following sections. The usage of 'Gateway' for offloading is discussed in Section 4.3 and Section 4.4. The Offload option is only used in Router Advertisement messages.

4.2. Lowering IPv4 Router Preference

The 'D' flag bit in the Offload option indicates the willingness of Dual-Stack capable router originating the Router Advertisement to

serve as a gateway for IPv4 traffic. If 'D' is set (1), the router indicates that it SHOULD NOT be used as a gateway for IPv4 traffic, if other gateways are present in the same or other available interfaces. If 'D' is unset (0), the router does not indicate any preference of being or not being a gateway for IPv4 traffic. When 'D' is unset (0), the decision of temporarily modifying the routing status is left for hosts that receive the Offload option (see Section 4.3 and Section 4.4). The 'Gateway' field in the Offload option contains the IPv4 address of the Dual-Stack interface that originated the Router Advertisement. The address serves as the identification of the next-hop IPv4 routers.

4.3. IPv4 Offloading to Specific Routes

To enable offloading of IPv4 traffic to specific routes, both Offload option and Route Information option [RFC4191] MUST present in the same Router Advertisement. A host receiving such Router Advertisement need to maintain a set of status including specific route, Router Preference, and Lifetime. A specific route consists of an IPv4 gateway from the Offload option and an IPv4 prefix from the Route Information option. The Prefix field in the Route Information option SHOULD follow the IPv4-mapped IPv6 address format defined in [RFC4291]. The Prefix Length in the Route Information option is used to indicate the IPv4 prefix length. The Router Preference in the Route Information option indicates whether to prefer the IPv4 router associated with this prefix over others. The Lifetime in the Offload option determines how long the temporarily added specific route will be valid. The Lifetime field in Route Information option SHOULD be ignored.

When 'D' flag is unset (0) in the Offload option, the advertised specific route shall be added by hosts if there is no duplicated prefix matching to the advertised prefix and the advertised lifetime in Offload option is valid. If there is a matching prefix, such specific route will be updated or deleted according to the status of Lifetime and Router Preference. The Lifetime in Offload option determines whether the route will be deleted or updated depending on the existing routing status of the hosts. If the advertised Lifetime is set to 0, any matched prefix and the corresponding route MUST be removed. If Lifetime is valid, the Router Preference further determines whether the gateway of the existing route, if matched, will be substituted to the advertised one, or the lifetime for existing route will be updated.

When 'D' flag is set (1) in the Offload option, any existing specific routes with the next-hop router matching to the advertised gateway SHOULD be removed.

To avoid misconfiguration of offloading operation, only one Offload option is allowed in a single Router Advertisement.

4.4. IPv4 Offloading to Default Gateway

If there is no Route Information options containing IPv4-mapped IPv6 addresses in the same Router Advertisement, the default gateway for offloading can be added, updated, or deleted depending on the 'D' flag, Lifetime, and existing routing status on the hosts. When 'D' is set (1), the existing default gateway matching to the advertised one SHOULD be removed if there are other usable gateways present in the same or other available interfaces.

When 'D' is unset (0) and there is no default gateway present for the receiving interface, the advertised gateway with valid lifetime can be added. If the advertised gateway matches to the existing one on the host, depending on the advertised lifetime, the existing default gateway shall be updated to the advertised lifetime in Offload option or deleted if the lifetime is set 0. If there is a default gateway existing on the receiving interface, which does not match to the advertised gateway, the advertised one SHOULD be ignored.

4.5. Offload Lifetime

The lifetime in the Offload option determines the valid period of temporary routing changes including IPv4 gateway preferences and offloading of IPv4 traffic to specific routes and default gateway. If the router sends a new Router Advertisement without the Offload option before the router lifetime expires, it is an indication to the receiving hosts that any existing Offload option caused state/information MUST be removed.

5. Router Behavior

A router configuration SHOULD allow network administrator to add and configure this option into Router Advertisement messages. The configuration can be selectively enabled (the Offload option is included in the Router Advertisement) or disabled (the Offload option is not included in the Router Advertisement). For specific route offloading, the prefix(es) advertised in the Route Information option SHOULD follow IPv4 mapped IPv6 address (e.g. ::ffff:1.2.3.4) as described in 4.3.

6. Host Behavior

A multi-interface capable host SHOULD monitor presence of Offload

option in received Router Advertisement messages. When the Offload option is received, the IPv4 gateway preferences and offloading to default gateway shall temporarily be updated as described in 4.2 and 4.4. Depending on the presence of Route Information in the same Router Advertisement, the offloading to specific IPv4 routes shall temporarily be updated as described in 4.3. Hosts SHOULD use the lifetime value in the Offload option to determine the valid time of all routing changes caused by the Router Advertisement received.

If the host receives a Router Advertisement without the Offload option and there is an existing state created by an earlier received Offload option, then the host MUST remove all IPv4 gateway preferences and offloading modifications from the previous Router Advertisement. The removals concern the prefixes configured from router where the router advertisement was received.

7. Security Considerations

The Offload option allows malicious hosts and routers to affect a victim host's next hop and default address selection if spoofing of Router Advertisements are possible on the access link. This is a well-known and understood security threat [RFC3756] and can be mitigated using, for example, Secure Neighbor Discovery [RFC3971]. The security of utilizing the Offload option is at the equal level to solution in [RFC4191].

8. IANA Considerations

This specification defines a new Neighbor Discovery option described in Section 4.1.

9. References

9.1. Normative References

- [I-D.ietf-6man-rfc3484-revise]
Matsumoto, A., Kato, J., Fujisaki, T., and T. Chown,
"Update to RFC 3484 Default Address Selection for IPv6",
draft-ietf-6man-rfc3484-revise-04 (work in progress),
July 2011.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman,

"Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.

9.2. Informative References

- [I-D.ietf-v6ops-3gpp-eps]
Korhonen, J., Soininen, J., Patil, B., Savolainen, T., Bajko, G., and K. Iisakkila, "IPv6 in 3GPP Evolved Packet System", draft-ietf-v6ops-3gpp-eps-08 (work in progress), September 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.
- [RFC3756] Nikander, P., Kempf, J., and E. Nordmark, "IPv6 Neighbor Discovery (ND) Trust Models and Threats", RFC 3756, May 2004.
- [RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, November 2005.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.

Appendix A. Address Selection Approach

A.1. Modification to Default Address Selection

The 'lower-than-IPv4 Preference' affects the Source Address Selection Rule 3. The notation Lower(SA) returns true if the address SA was configured from the prefixes advertised by a 'lower-than-IPv4 Preference' router. Lower(SA) returns false if the address SA was configured from prefixes advertised by other than 'lower-than-IPv4 Preference' router. The notation Default(D) returns false if the address D has more specific routes (i.e. other than the default route). Default(D) returns true if the address D points only to a default route. The modified Rule 3 would be as follows:

Rule 3: Avoid deprecated addresses.

The addresses SA and SB have the same scope. If `Lower(SA) == true` and `Default(D) == true`, then mark SA temporarily as "deprecated". If `Lower(SB) == true` and `Default(D) == true`, then mark SB temporarily as "deprecated". If one of the two source addresses is "preferred" and one of them is "deprecated" (in the [RFC4862] sense), then prefer the one that is "preferred."

Similar modification also concerns the Destination Address Selection Rule 3 when checking whether a candidate source address for a given destination is deprecated.

A.2. Address selection examples

Link-local addresses are omitted in all following examples. The assumption is that possible destinations have a global scope and all IPv6 enabled interfaces have at least one global scope IPv6 address. Therefore, the default address selection would always output global scope addresses over link-local addresses.

A.2.1. Case 1: IPv6-only cellular and IPv4-only WLAN accesses

A host has obtained global IPv6 address, 2001:db8::2, on a cellular interface and with it has received Neighbor Discovery option with 'lower-than-IPv4' preference. The host also has global IPv4 address, 192.0.2.2, on a WLAN interface.

When connecting to a dual-stack enabled destination, both 2001:db8::2 and 192.0.2.2 are considered as source addresses candidates. IPv4 address is selected, because 2001:db8::2 is considered deprecated. Hence host uses WLAN for communication.

When connecting to IPv6-only destination, 2001:db8::2 is selected and cellular network used, as there are no other IPv6 addresses available.

A.2.2. Case 2: WLAN access with multiple prefixes

A host has obtained two global IPv6 addresses, one of which was from a router indicating 'lower-than-IPv4' preference. For example, 2001:db8:1::2 from router with 'lower-than-IPv4' preference and 2001:db8:2::3 from router without any special preferences.

When connecting to IPv6-only destination, both addresses are considered as source address candidates. Source address selection chooses 2001:db8:2::3 as 2001:db8:1::2 is considered deprecated (`Lower(2001:db8:1::2) == true` and `Default(D) == true`).

A.2.3. Case 3: WLAN and cellular interface with cellular's IPv4 not default route

A host has obtained IPv6 address, 2001:db8::2, and IPv4 address, 192.0.2.2, from cellular network. The network has indicated 'lower-than-IPv4' preference for IPv6 and 'not your default router' for IPv4. The host also has dual-stack WLAN access with 2001:db8:1::3 and 192.0.2.30 addresses.

When connecting to IPv4-only destination, host selects 192.0.2.30 as source address because default gateway on the interface of 192.0.2.2 address is 'not default gateway'. WLAN is used for communication.

When connecting to IPv6-only destination, host selects 2001:db8:1::3 from WLAN interface as the 2001:db8::2 is considered deprecated (Lower(2001:db8::2) == true and Default(D) == true). WLAN is used for communication.

When connecting to dual-stack destination, host selects from the four candidate addresses 2001:db8:1::3, as IPv6 is preferred in general and as that address is not deprecated. WLAN is used for communication.

A.2.4. Case 4: Dual-stack cellular access

A host has obtained IPv6 address, 2001:db8::2, and IPv4 address, 192.0.2.2, from cellular network. The network has indicated 'lower-than-IPv4' preference.

When connecting to a dual-stack enabled destination, both addresses are considered as candidate source addresses. IPv4 address is chosen, because IPv6 address is considered deprecated.

A.2.5. Case 5: Dual-stack cellular and single stack WLAN

A host has obtained IPv6 address, 2001:db8::2, and IPv4 address, 192.0.2.2, from cellular network. The network has indicated 'lower-than-IPv4' preference for IPv6 and 'not your default router' for IPv4. The host also has WLAN access with 192.0.2.30 address.

When connecting to dual-stack destination, all three addresses are considered as source address candidates. The IPv4 address from WLAN, 192.0.2.30, is selected as the IPv6 address, 2001:db8::2, is considered deprecated and as the IPv4 default route points to WLAN. Hence WLAN is used for communication.

A.2.6. Case 6: Coexistence with RFC4191

A host has obtained IPv6 address, 2001:db8:1::2/64 from cellular network. The network has indicated 'lower-than-IPv4' preference for IPv6 and a more specific route to 2001:db8:2::/48. The host also has IPv6 WLAN access with 2001:db8:3::3/64 address.

When connecting to 2001:db8:2::1 the host selects 2001:db8:1::2 from cellular interface as a source address, because `Lower(2001:db8:1::2) == true` and `Default(2001:db8:2::1) == false` and hence the 2001:db8:1::2 is not considered as deprecated address even though 'lower-than-IPv4' preference was advertised.

When connecting to 2001:db8:4::1 the host selects 2001:db8:3::3 from WLAN interface as a source address, because `Lower(2001:db8:2::1) == true` and `Default(2001:db8:3::3) == true` and hence 2001:db8:2::1 is considered as deprecated address.

Authors' Addresses

Jouni Korhonen
Nokia Siemens Networks
Linnoitustie 6
FI-02600 Espoo
Finland

Email: jouni.nospam@gmail.com

Teemu Savolainen
Nokia
Hermiankatu 12 D
FI-33720 Tampere
Finland

Email: teemu.savolainen@nokia.com

Yi Ding (editor)
University of Helsinki
P.O. Box 68
FI-00014 University of Helsinki
Finland

Email: yi.ding@cs.helsinki.fi

Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 3, 2012

D. Liu
China Mobile
Ted. Lemon
Nominum
Yuri. Ismailov
Ericsson
Z. Cao
China Mobile
October 31, 2011

MIF API consideration
draft-liu-mif-api-extension-06

Abstract

This document describes an abstract API that provides the minimal functionality required for a program to communicate effectively with peers and services on the network while running on a host that has more than one active network interface. This API is abstract: we describe the functionality that must be provided, not the bindings that should be used to provide that functionality. The functionality described here provides the building blocks from which higher-level APIs might be built, and is not intended to be used directly by typical applications.

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 3, 2012.

Copyright Notice

Copyright (c) 2011 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	4
2. Conventions used in this document	4
3. MIF API Concept	5
3.1. Provisioning Domains	5
3.2. Provisioning Domain Agnosticism	5
3.3. MIF API Elements	6
3.3.1. Application Element	6
3.3.2. High Level API	7
3.3.3. MIF API	7
3.3.4. Communications API	7
3.3.5. Network Link API	7
3.4. MIF API communication model	8
3.4.1. POST MESSAGE call	8
3.4.2. CHECK MESSAGE call	8
3.4.3. GET MESSAGE call	8
3.5. MIF Messages	8
3.5.1. Announce Interfaces	9
3.5.2. Stop Announcing Interfaces	9
3.5.3. Interface Announcement	9
3.5.4. No Interface Announcement	9
3.5.5. Announce Provisioning Domain	9
3.5.6. Stop Announcing Provisioning Domains	10
3.5.7. Provisioning Domain Announcement	10
3.5.8. No Provisioning Domain Announcement	10
3.5.9. Announce Configuration Element	10
3.5.10. Configuration Element Announcement	11
3.5.11. No Configuration Element Announcement	11
3.5.12. Announce Address	11
3.5.13. Address Announcement	12
3.5.14. No Address Announcement	12
3.5.15. Get Configuration Data	12
3.5.16. Translate Name	12
3.5.17. Stop Translating Name	13
3.5.18. Name Translation	13
3.5.19. Connect to Address	13

3.5.20. Connect to Address From Address	13
3.5.21. Connected	14
3.5.22. Not Connected	14
4. Example Usage	14
5. Security Considerations	16
6. IANA Considerations	16
7. Acknowledgments	16
8. References	16
8.1. Normative References	16
8.2. Informative References	16
Authors' Addresses	17

1. Introduction

Traditionally, hosts that communicate on the network have done so over a single network link, which is provided by a single service provider. This simple environment is relatively easy to program to, and relatively predictable.

However, this relatively simple case is no longer the norm. A typical modern host may have one or two wireless interfaces: a wireless interface connected to a broadband network, and possibly another connected to some kind of cellular network. The same host may also have a wired interface which is sometimes connected to another 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 by 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.

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

To properly handle these multiple-service interfaces, we specify the API not in terms of interfaces, but in terms of provisioning domains. So in the case of a dual-stack network attached to a single network interface, there would be two provisioning domains. If the host has a second interface that is connected to a link that only supports IPv6 service, then that host would be connected to a total of two network links, but three 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 received via DHCPv6.

****point of discussion:** it's actually possible to have two separate provisioning domains for IPv6 on the same wire. Is this a case that could happen in practice, and that we ought to support? I know that some asian countries have arrangements where the operator of the physical network is distinct from one or more operators who provide transit; I think this is all handled transparently to the host, but I don't really know the details.

****point of discussion:** is IPv4 stateless/Bonjour a separate provisioning domain? What about IPv6 ULA?

3.2. Provisioning Domain Agnosticism

Although it is possible that a high-level API built on top of this API may be able to distinguish between provisioning domains, at the level of this API, no such distinction can be made. Each provisioning domain is treated separately, and it is the responsibility of the higher-level API or of the application to decide which provisioning domain or domains to actually use.

3.3. 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.

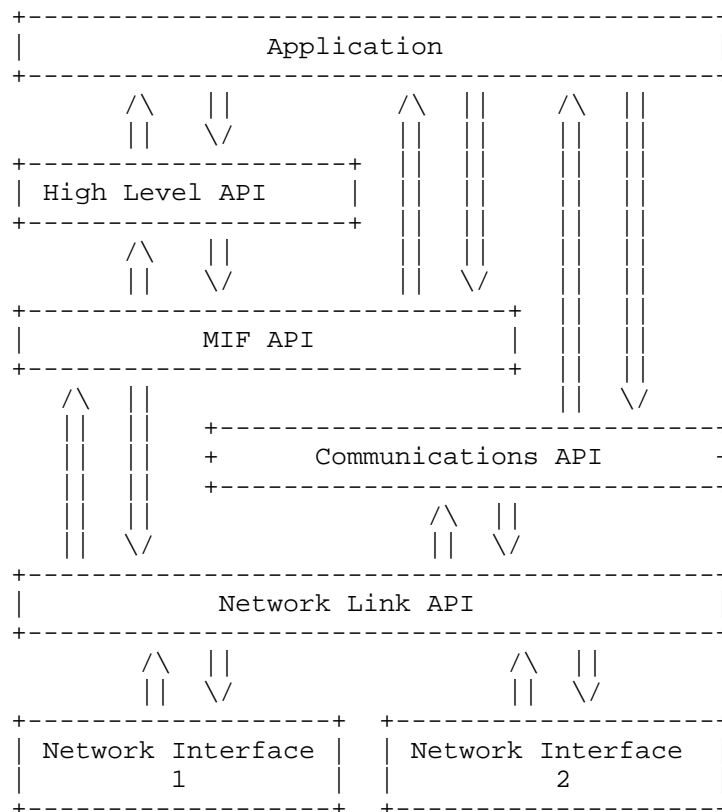


Figure 1

3.3.1. 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.

3.3.2. 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.

3.3.3. 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.

3.3.4. 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.

3.3.5. 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.

3.4. 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.

3.4.1. POST MESSAGE call

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

3.4.2. 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.

3.4.3. 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.

3.5. 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.

3.5.1. 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.

3.5.2. 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.

3.5.3. Interface Announcement

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

3.5.4. No Interface Announcement

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

3.5.5. 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.

3.5.6. 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.

3.5.7. 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.

3.5.8. 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.

3.5.9. 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 and send a No Configuration Element Announcement message for each such configuration element.

3.5.10. 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
- ...TBD...

3.5.11. 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.

3.5.12. 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 and send a No Address Announcement message for each such address.

3.5.13. 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).

3.5.14. 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.

3.5.15. 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
- ...TBD...

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.

3.5.16. 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, LLNMR, and is implementation-dependent. **need to think about this

3.5.17. 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.

3.5.18. 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.

3.5.19. 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.

3.5.20. 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.

3.5.21. 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.

3.5.22. 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.

4. Example Usage

below is an example that shows how MIF API in use:

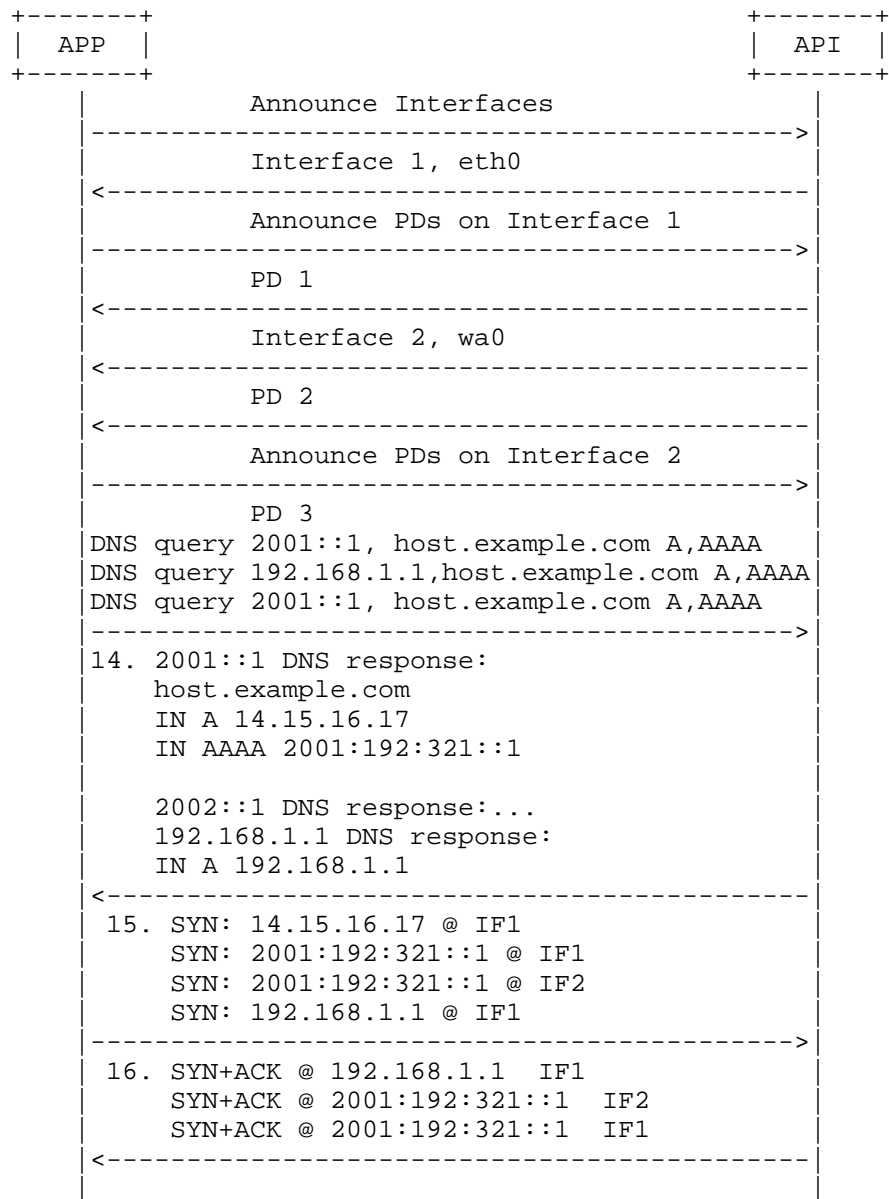


Figure 2

As described in the above communication model, the application first invoke the MIF API to query how many interfaces in the host. then, the application invokes MIF API to query how many networks attaches in each interface. application then invoke MIF API to query each DNS

configuration on each interface's attached network. application then send DNS query to each DNS server on each network. The DNS servers may return multiple IP address of the queried host name. The application then try to connect to each IP addresses of the host by sending tcp SYN packet to each destination IP addresses through multiple interfaces. Some of the destination IP address may return ACK packet some may not. The application then chose a best connection based on certain criteria. for example, the criteria may based on the quality of the link.

5. Security Considerations

TBD

6. IANA Considerations

None

7. 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.

8. References

8.1. Normative References

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

8.2. Informative References

[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

Email: yuri@ismailov.eu

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

Email: caozhen@chinamobile.com

