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

Table of Contents

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:

- Centralized connection management
- Per-application connection settings
- 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:

- DNS resolution issues
- Routing
- 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.

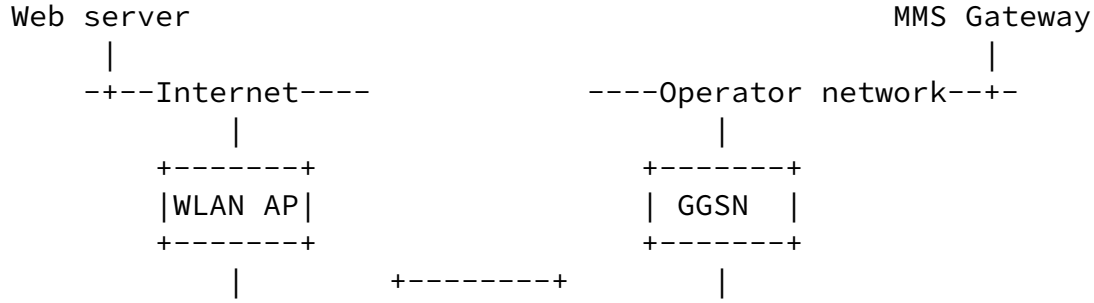
3. Current Practices in Some Operating Systems

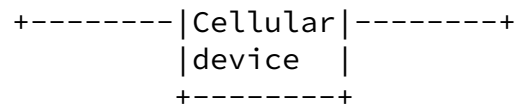
The material presented in this section is derived from contributions from people familiar with the Operating Systems described, and those people are listed in Section 7. The authors and the IETF take no position about the Operating Systems described, and understand that other Operating Systems also exist. 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. 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:

- Statically: e.g. from a configuration interface, via client provisioning/device management system, or at build-time.

- 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.

- 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 adress 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 muliple IP stacks are associated to multiple interfaces, the application can select the source address correponding 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:

- 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.

- 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:

- 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.

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

The Phystech dhcpcd client [PHYSTECHDHCPC] 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 dhcpd 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:

- Marc Blanchet, Viagenie
- Hua Chen, Leadcoretech, Ltd.
- Yan Zhang, Leadcoretech Ltd.

- Shunan Fan, Huawei Technology
- Jian Yang, Huawei Technology
- Gabriel Montenegro, Microsoft Corporation
- Shyam Seshadri, Microsoft Corporation
- Dave Thaler, Microsoft Corporation
- Kevin Chin, Microsoft Corporation
- Teemu Savolainen, Nokia
- Tao Sun, China Mobile
- George Tsirtsis, Qualcomm.
- David Freyermuth, France telecom.

- Aurelien Collet, Altran.
- 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), 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>.

[OPENBSDDHCLIENT]
OpenBSD, "OpenBSD dhclient", 2009, <http://www.openbsd.org/>.

[PHYSTECHDHCPC]
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/190358c8-7cb1-4be3-9321-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/>.

[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