| | |
|---|---|
| Internet Engineering Task Force | M. Wasserman, Ed. |
| Internet-Draft | Sandstorm Enterprises |
| Intended status: Informational | March 25, 2009 |
| Expires: September 26, 2009 | |


TOC

**Current Practices for Multiple Interface Hosts**
**draft-mrw-mif-current-practices-02**

**Status of this Memo**



**Copyright Notice**



**Abstract**


An increasing number of hosts are operating in multiple-interface environments, where different network interfaces are providing unequal levels of service or connectivity. This document describes how some common operating systems cope with the related challenges.

**Table of Contents**



---

## 1. Introduction

TOC

Multiple-interface hosts face several challenges not faced by single-interface hosts, some of which are described in [I-D.blanchet-mif-problem-statement] (Blanchet, M. and P. Seite, “Multiple Interfaces Problem Statement,” June 2009.). This document collects and analyzes publicly-available information about the multiple-interface solutions implemented in some widely used operating systems, including: Nokia S60 [S60] (Nokia Corporation, “S60 Platform: IP Bearer Management,” 2007.), Microsoft Windows Mobile [WINDOWSMOBILE] (Microsoft Corporation, “SDK Documentation for Windows Mobile-Based Smartphones: Connection Manager,” 2005.), Blackberry [BLACKBERRY] (Research In Motion Limited, “BlackBerry Java Development

Environment - Fundamentals Guide: Wireless gateways," 2009.), Google Android [ANDROID] (Google Inc., "Android developers: package android.net," 2009.), Microsoft Windows, Apple Mac OS X, Linux and BSD-based operating systems.
In section 3, common solutions implemented in different operating systems are identified and generalized.
NOTE: Further input on the behaviour of these or other operating systems (including newer versons) is very welcome.

---

## 2. Current practices of some operating systems

TOC

The following sections briefly describe the current multiple-interface host implementations on some widely-used operating systems. Please refer to the References section for pointers to original documentation on most of these systems, including further details.

---

### 2.1. Nokia S60 3rd Edition, Feature Pack 2

TOC

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.

---

```
          Web server                                        MMS Gateway
               |                                                 |
              -+--Internet----            ----Operator network--+-
                      |                           |
                  +-------+                   +-------+
                  |WLAN AP|                   | GGSN  |
                  +-------+                   +-------+
                      |        +--------+         |
                      +--------|Cellular|---------+
                               |device  |
                               +--------+
```

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.).

S60 uses the concept of an Internet Access Point (IAP) [S60] (Nokia Corporation, “S60 Platform: IP Bearer Management,” 2007.) 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, it it possibly for the operating system to 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 and 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).

Please see the source documentation for more details and screenshots: [S60] (Nokia Corporation, “S60 Platform: IP Bearer Management,” 2007.).

---

### 2.2. Microsoft Windows Mobile 2003 Second Edition

TOC

A Connection Manager architecture is described in [WINDOWSMOBILE] (Microsoft Corporation, “SDK Documentation for Windows Mobile-Based Smartphones: Connection Manager,” 2005.). 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.

---

### 2.3. BlackBerry

TOC

In BlackBerry devices [BLACKBERRY] (Research In Motion Limited, “BlackBerry Java Development Environment - Fundamentals Guide: Wireless gateways,” 2009.) Java applications can use one of two wireless gateways to proxy the connection to the Internet or to a corporate network. The application can be designed to always use the default Internet gateway, or to use a more preferred enterprise gateway when available. The intent is to hide connectivity issues from users.
DISCUSS: How does the Blackberry decide when a WLAN interface, a cellular interface or some other physical interface is used?

---

### 2.4. Google Android

TOC

The Android reference documentation describes the android.net package [ANDROID] (Google Inc., “Android developers: package android.net,” 2009.) and the ConnectivityManager class that applications can use to request a route to a specified destination address via a specified network interface (Mobile or Wifi). 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.
DISCUSS: Are applications bound to use one network type at a time, or can one application use multiple network features in parallel?

---

### 2.5. Arena Connection Manager

TOC

The Arena Connection Manager is described in [I-D.zhang-mif-connection-manager-arena] (Zhang, Y., Sun, T., and H.

Chen, “Multi-interface Network Connection Manager in Arena Platform,” February 2009.) and [I-D.yang-mif-connection-manager-impl-req] (Yang, J., Sun, T., and S. Fan, “Multi-interface Connection Manager Implementation and Requirements,” March 2009.). The arena connection manager provides a means for applications to register their connectivity requirement with the Connection Manager. The Connection Manager can then choose an interface that matches the application's needs while considering other factors such as availability, cost and stablility. Also, the Connection Manager can handle multi-interface issues such as connection sharing.

---

### 2.6. Microsoft Windows

TOC

The multi-interface functionality currently implemented in Microsoft Windows operation systems is described in more detail in [I-D.montenegro-mif-multihoming] (Montenegro, G., Thaler, D., and S. Seshadri, “Multiple Interfaces on Windows,” March 2009.).

---

#### 2.6.1. Routing

TOC

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 behaviour 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] (Hinden, R. and D. Thaler, “IPv6 Host-to-Router Load Sharing,” November 2005.) is used for both IPv4 and IPv6.

---

#### 2.6.2. Outbound and Inbound Addresses

TOC

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] (Draves, R., “Default Address Selection for Internet Protocol version 6 (IPv6),” February 2003.) 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] (Braden, R., “Requirements for Internet Hosts - Communication Layers,” October 1989.) 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.

---

### 2.6.3. DNS Configuration

TOC

Host-wide DNS configuration is input via static configuration or, in sites that use Active Directory, Microsoft's Group Policy. The host-wide configuration consists of a DNS suffix to be used for the local host, as well as a list of domain names 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 precendent over per-interface DNS configuration.
Interface specific DNS configuration can be input via static configuration or via DHCP. It includes:

*An interface-specific suffix list.

*A list of DNS server IP addresses.

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.

---

## 2.7. Linux and BSD-based Operating Systems

TOC

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] (Internet Software Consortium, “ISC DHCP,” 2009.) and its derivative for OpenBSD [OPENBSDDHCLIENT] (OpenBSD, “OpenBSD dhclient,” 2009.) 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.

The Phystech dhcpcd client [PHYSTECHDHCPC] (Phystech, “dhcpcd,” 2009.) 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] (RedHat, “PUMP,” 2009.) 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] (Busybox, “uDHCP,” 2009.) 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 . TODO: Verify all connection managers.

TODO: DHCPv6 clients

---

TOC

### 2.8. Apple Mac OS X

This section is based on testing Mac OS X (version 10.5.6).
When using multiple interfaces on Mac OS X, global configuration data such as default routes and the DNS server list are taken from the DHCP data received on the primary interface. Therefore, the order in which the interfaces receive their configuration data is not relevant. For example, if the primary interface receives its configuration data first, then the second interface receives its configuration data, the interface-specific information for the second interface will be configured, but the global configuration information such as the DNS server list and default routes is not overwritten.

---

## 3. Common solutions

TOC

Essentially all operating systems use the same types of information to make decisions about multiple-interface operation: user input, operator/administrator provided information, and what has been statically configured or hard-coded. It is possible to design clever ways for tackling the problems related to multi-homing from the set of dynamically available information, vendor specific policies and design decisions. However, limitations on available information also set limits on what different operating systems can theoretically achieve. This section describes what common solution approaches the analyzed operating systems are known to utilize.

---

### 3.1. Centralized connection management

TOC

It seems to be common practice to have a centralized connection manager entity, which does the network interface selection based on application input. The information used by the connection manager may be programmed into an application, learned from the users, or provisioned.
Routing tables are not typically used for network interface selection, 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.

---

TOC

### 3.2. Per application connection settings

As each application has its particular connectivity needs, applications are able to request what kind of connectivity they need.

---

## 4. Common problems

TOC

The current solutions are limited by the information available. This section discusses what types of information IETF-designed protocols could provide to allow implementations to improve functionality in multi-inteface scenarios.
Please also see MIF problem statement document [I-D.blanchet-mif-problem-statement] (Blanchet, M. and P. Seite, “Multiple Interfaces Problem Statement,” June 2009.).
DISCUSS: is this section 4 required in this document, or should we fully leave the problem descriptions to problem statement, or have here some general problems, and in problem-statement more detailed problems?

---

### 4.1. Selection of an interface providing access to a destination

TOC

As different network interfaces may provide connectivity to different destination networks, a host needs to be able to choose a correct network interface (or a set of interfaces, if the application can use multiple interfaces in parallel) to allow the application or IP flow to reach the intended destination.

---

### 4.2. Prioritization of interfaces for the same destination

TOC

As several interfaces may lead to the same destination network, a host has to choose which one to use. It may be that if one network has connectivity limitations, such as firewalls or NATs, and the other network does not, it may be preferable to use the interface to the less restricted network. This is not always the case, e.g. if access to the restricted network is faster or cheaper, it might be desirable to use that interface, if it can support the application and reach the desired destination. Could the network somehow indicate what limitations it is imposing, or could there be new technologies that would help to determine the connectivity properties of a network?

Improved source/destination address selection (underway in the 6man address selection design team), more specific routes (RFC4191), or other (TBD) technologies may be helpful in this area.

---

### 4.3. Enablers for application multihoming

TOC

When applications are bound to use single network interface at a time, they are unable to benefit from technologies developed for multihoming purposes. Therefore technologies that help to free applications from being bound into a single network interface would be useful. Essentially this means advanced ways to ensure that applications use the right network interface, and that applications do not accidentally use interfaces that will not support the application or provide connetivity to the destination. Can improvements be designed for IPv4 in spite of overlapping private IPv4 address spaces, or only for IPv6?

---

## 5. Acknowledgements

TOC

Authors of the document would like to thank following people for their input and feedback: Hui Deng, Jari Arkko.
This document was prepared using xml2rfc template and related web-tool.

---

## 6. IANA Considerations

TOC

This memo includes no request to IANA.

---

## 7. Security Considerations

TOC

This draft describes current multiple-interface host implementations on some common operating systems without any focus on security considerations.

---

TOC

## 8. Change Log

The following changes were made between versions -00 and -01:

*Number of authors passed five, so moved to Contributors section.

*Added information on Microsoft Windows.

*Added information on Arena Connection Manager.

---

## 9. Contributors

TOC

The following people contributed most of the information found in this document:

*Marc Blanchet, Viagenie

*Hua Chen, Leadcoretech, Ltd.

*Shunan Fan, Huawei Technology

*Gabriel Montenegro, Microsoft Corporation

*Teemu Savolainen, Nokia

*Shyam Seshadri, Microsoft Corporation

*Tao Sun, China Mobile

*Dave Thaler, Microsoft Corporation

*Jian Yang, Huawei Technology

*Yan Zhang, Leadcoretech Ltd.

---

## 10. References

TOC

---

### 10.1. Normative References

TOC


| | |
|---|---|
| [I-D.blanchet-mif-problem-statement] | Blanchet, M. and P. Seite, "Multiple Interfaces Problem Statement," draft-blanchet-mif-problem-statement-01 (work in progress), June 2009 (TXT). |


---

### 10.2. Informative References

TOC


| | |
|---|---|
| [ANDROID] | Google Inc., "Android developers: package android.net," 2009. |
| [BLACKBERRY] | Research In Motion Limited, "BlackBerry Java Development Environment - Fundamentals Guide: Wireless gateways," 2009. |
| [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 (TXT). |
| [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 (TXT). |
| [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 (TXT). |
| [ISCDHCP] | Internet Software Consortium, "ISC DHCP," 2009. |
| [OPENBSDDHCLIENT] | OpenBSD, "OpenBSD dhclient," 2009. |
| [PHYSTECHDHCPC] | Phystech, "dhcpcd," 2009. |
| [PUMP] | RedHat, "PUMP," 2009. |
| [RFC1122] | Braden, R., "Requirements for Internet Hosts - Communication Layers," STD 3, RFC 1122, October 1989 (TXT). |
| [RFC3484] | Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)," RFC 3484, February 2003 (TXT). |
| [RFC4311] | Hinden, R. and D. Thaler, "IPv6 Host-to-Router Load Sharing," RFC 4311, November 2005 (TXT). |
| [S60] | Nokia Corporation, "S60 Platform: IP Bearer Management," 2007. |
| [UDHCP] | Busybox, "uDHCP," 2009. |

| | |
|---|---|
| [WINDOWSMOBILE] | Microsoft Corporation, “SDK Documentation for Windows Mobile-Based Smartphones: Connection Manager,” 2005. |


---

## Author's Address

TOC


| | |
|---|---|
| | Margaret Wasserman (editor) |
| | Sandstorm Enterprises |
| | 14 Summer Street |
| | Malden, MA 02148 |
| | USA |
| Phone: | +1 781 333 3200 |
| Email: | mrw@lilacglade.org |
| URI: | http://www.sandstorm.net |