

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 14, 2011

D. Liu
China Mobile
Yuri. Ismailov
Ericsson
Z. Cao
China Mobile
March 14, 2011

Socket API Extension for MIF Host
draft-liu-mif-api-extension-04

Abstract

API (Application Program Interface) is an interface implemented by a software program that enables it to interact with other software. In IP network communication area, Socket API is the de facto standard and is widely used in many systems. Recently, multiple interfaces or multiple connection capable devices become more and more common. But due to the limitation of the default route of the host and other issues such as DNS selection etc prevent the utilization of multiple interfaces/connections benefit. Moreover, there is no API level support for the application developer to utilize the benefit of the host's multiple interfaces/connections. Starting with the requirement of MIF API extension, this document describes a new set of abstraction APIs to provide additional services to applications running on hosts attached to multiple provisioning domains. These services could assist advanced applications in having greater control over first-hop, source address and/or DNS selection issues.

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 September 8, 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. Conventions used in this document	3
3. MIF API Concept	3
4. API extension for MIF host	4
4.1. MIF API Requirement	5
4.2. Definition of New Socket APIs	5
4.3. Connection Manager API extension	7
5. Using the extended APIs	7
6. Security Considerations	7
7. IANA Considerations	8
8. Acknowledgments	8
9. References	8
9.1. Normative References	8
9.2. Informative References	8
Authors' Addresses	8

1. Introduction

API is an software interface which is used to interact with other softwares. In IP network area, Socket API is widely used in many operating systems.

Socket API has been extended according to the evolution of TCP/IP protocol stack. For example, Socket API was extended to support IPv6 in RFC 3493 [RFC3493]. Nowadays, there are more and more devices that have the ability to attach to multiple provisioning domains. Due to the limitation of Socket API and other issues such as default route and DNS selection, the application can not directly control the multiple interfaces/connection capability of the devices. This document describes a new set of abstraction APIs to provide additional services to applications running on hosts attached to multiple provisioning domains. These services could assist advanced applications in having greater control over first-hop, source address and/or DNS selection issues.

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

Figure 1 demonstrates the MIF API concept. MIF API is mostly extended based on current Socket API. MIF API is an abstract API which describes the abstract function and not depend on specific platform. MIF provides the following function:

1.Address selection

MIF host may has multiple IP addresses. for example, different interface may have different IP addresses been configured. Further more, some protocols, for example: mobile IP protocol may assign home address to the host. Application need to control which IP address it want to use. MIF API may provide the address selection function in this case.

2.Routing selection

Noramlly, host only has a default route which points to one default gateway. for MIF host, there may be different connections using multiple interfaces and each connection has its own default gateway.

MIF API provide the capability for the MIF host to control the first hop route.

3. DNS Selection

The MIF host may have the DNS split issues. MIF API may provide the capability for the application to select and control which DNS server it want to use.

4. Connection Manager API

Connection Manager is widely used in current mobile terminals. The lack of standard guide of connection manager will lead to non-consistency of user experience. Connection manager API aims to provide a set of abstract function description of connection manager.

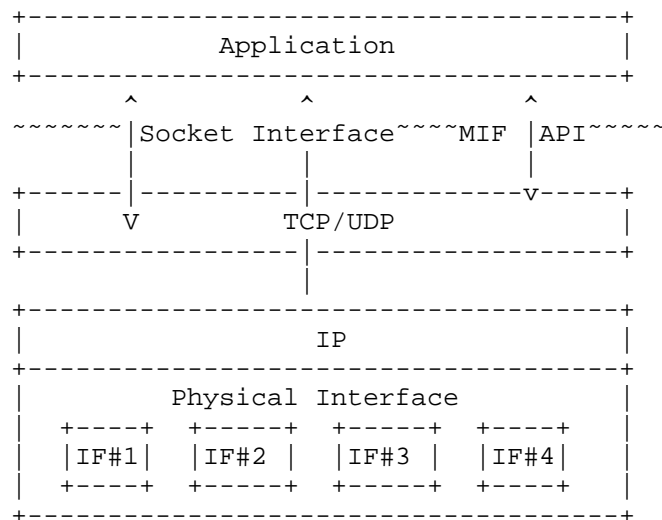


Figure 1. MIF API Concept

Figure 1

4. API extension for MIF host

This section discusses the possible ways to design the MIF API.

There are works in IETF relatd to socket API extensions,such as: IPv6 socket API extension, HIP socket API extension, shim6 socket API

extensions. This section gives initial proposals regarding to how this MIF Socket APIs should be designed.

4.1. MIF API Requirement

R1: Compatible with existing Socket APIs

The extension of Socket APIs should remain compatible with current existing Socket APIs which means this extension will not affect the existing application and the application developer can chose whether to use this extension or not.

R2: Support for multiple Connections

The newly introduced APIs can support the host applications to utilize multiple connections. If some connections fail to work, applications can failover to other available interfaces.

R3: No Changes to Existing Socket APIs

The newly introduced APIs cannot change the behavior of existing network APIs available on the host operating system.

R4: Easy for Implementation

The implementation details for the newly introduced APIs should be as neat as possible. It keeps the underlying implementation details simple, and makes the application developers easy to use these APIs.

4.2. Definition of New Socket APIs

This section gives the definition and description of MIF API.

Address Selection API

In current Socket API definition, `setsockopt()` is used to set the option of socket. It has the following definition:

```
int setsockopt(int socket, int level, int option_name, const void
*option_value, socklen_t option_len);
```

The `setsockopt()` function shall set the option specified by the `option_name` argument, at the protocol level specified by the `level` argument, to the value pointed to by the `option_value` argument for the socket associated with the file descriptor specified by the `socket` argument.

The `option_name` argument specifies a single option to set. The

option_name argument and any specified options are passed uninterpreted to the appropriate protocol module for interpretations. The options are as follows:

SO_DEBUG

SO_DONTROUTE

SO_SNDBUF

.....

This document proposes to extend the current setsockopt() function as follows:

Extend new option type used for address selection:

SO_ADDR

When setsockopt() set this parameter, the application will use the IP address specified by option_value as the source IP address.

Route Selection API

Extend new option type parameter for setsockopt() function:

SO_FSTHOP

When this option is set, the packet will send via the gateway that specified by option_value.

DNS Selection API

gethostbyname() and gethostbyaddr() is the APIs used for DNS queries. Current version of gethostbyname() has the following definition:

```
struct hostent *gethostbyname(const char *name)
```

This document proposes to extend this function as follows:

```
struct hostent *gethostbyname(const char *name, struct sockaddr
*DNS_addr)
```

DNS_addr is used to specify which DNS server the application want to use for the DNS query.

4.3. Connection Manager API extension

Connection manager is a common practice for MIF host to perform provisioning domain selection and interface configuration [I-D.draft-seite-mif-connection-manager]. High level API of connection manager may differ across different operating systems and/or platforms. It is hence preferred to have more consistent connection manager API specification and this will be beneficial for both applications developers, users and operators.

Get Interface status/configuration

Connection Manager may need to get the interface's status and its configuration parameters, for example, the connection manager in MIF host may need to know all the interface's current status (e.g. on/off), interface type, IP address etc.

A new structure is defined in this document:

```
Struct InterfaceInfo {  
  
    char * name; /*interface name*/  
  
    struct parameter; /*IP address, address type, address scope(e.g.  
    HoA, CoA, temp address), MAC address, Interface type (WiFi, cellular)  
    etc*/  
  
    uint32_t status; /*interface status*/  
  
}
```

function struct Interfaceinfo getInterfaceInfo() is used to get the status and parameters of an interface.

Function setInterfaceInfo(struct InterfaceInfo parameter) is used to set parameters associated with a specific interface.

5. Using the extended APIs

To enjoy the benefit of multiple interfaces, the developers can write programs according to the APIs introduced in this document. Detail example need to be further specified.

6. Security Considerations

TBD

7. IANA Considerations

None

8. Acknowledgments

The authors want to thank Teemu Savolainen from Nokia, Dayi Zhao from Bitway and others for their useful suggestions and discussions.

9. References

9.1. Normative References

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

9.2. 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

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

