Mobile IP API

Aki Yokote, Alper E. Yegin, Muhammad Mukarram Bin Tariq,
Guangrui Fu, Carl Williams, Atsushi Takeshita

DoCoMo Communications Laboratories USA, Inc. 181 Metro Drive, Suite 300, San Jose CA, 95110, USA
{yokote, alper, tariq, fu, carlw, takeshita}@docomolabs-usa.com

Abstract **- Mobile IP is focused on providing transparent connectivity to mobile nodes in an IP based network environment. While transparent mobility support is essential for compatibility with applications that are geared towards fixed network environment, it is a handicap for applications that can potentially benefit from mobility awareness. In this paper, we present Mobile IP API which is an interface between the applications and the mobility management module at mobile terminals and their correspondent nodes. Our main focus is basic Mobile IP API which primarily enables 'reading' of mobility information for applications on mobile nodes and correspondent nodes. We present usage scenarios, design, and implementation of the API. We are also currently pursuing standardization of Mobile IP API through the IETF.**

Keywords: Mobile communication; Programming; Operating Systems; Internet; Protocols; Network Interfaces.

1. INTRODUCTION

Ubiquity and freedom provided by mobile computing resulted in its enormous popularity. As users moved from wired to wireless devices, they wanted to run their traditional applications which were primarily geared towards fixed network connectivity on mobile devices. This led to need for transparent mobility support by mobile networks and mobile devices. Protocols like Mobile IP [4] [2] were devised to address this issue. However, mobility becomes a norm rather than exception, the strict transparency from mobility is becoming a handicap for the applications. Mandating such transparency means that the much of the fascinating and useful information related to mobility can never be available to applications that can potentially exploit such information and provide new and exciting services. Our effort is to provide a comprehensive interface through which the interested applications can acquire and manipulate mobility related information. The number of new applications that can make use of such information is enormous and limited only by imagination of application developers.

We share our vision of having Mobile IP [4] [2] as the primary mobility management protocol in future networks. We have focused on designing an API which is specific to Mobile IP functionality. Mobile IP hides the effect of change in network attachment point from the transport and higher layers, by assigning two kinds of IP address to the mobile host: the home address which is the host's own (permanent) address, and a care-of address which is an (temporary) assigned address by the visited networks. The binding between the home address and the care-of address ensures that a mobile host remains "virtually" connected to its home network address regardless of its current point of attachment. It allows applications to operate over mobile nodes without any modifications.

Binding between the home address and the care-of address of a mobile host is the source of location and movement information. Although such information is available on the mobile host and its correspondent hosts, it is hidden in the IP module, therefore not visible to upper layers. Making this information available to applications is the goal of the Mobile IP API.

Large sets of usage scenarios are enabled by simply allowing applications to acquire mobility and location related information. As a first step we have focused on a basic Mobile IP API which primarily enables applications on mobile node (MN) and correspondent node (CN) to 'read' mobility information. Section 2 presents a functional overview of Mobile IP API. Usage scenarios for basic Mobile IP API are presented in Section 3. Section 4 is dedicated to definition and implementation of basic Mobile IP API. Section 5 briefly discusses the standardization efforts. We present the future works in Section 6. Section 7 presents a related work, and the paper concludes with Section 8.

2. FUNCTIONAL OVERVIEW OF MOBILE IP API

Mobile IP API is an interface between mobility management and application layer as shown in Fig. 1. Basic Mobile IP API primarily deals with mobility and (IP topological) location awareness for the applications running at the mobile nodes and the correspondent nodes. In short, basic Mobile IP API provides following four kinds of mobility and location awareness.

For applications running on MNs:

(a). Awareness of MN's own location in the Internet topology

(b). Awareness of MN's own movement, and new location.

For applications running on CNs:

(c). Awareness of MN's location

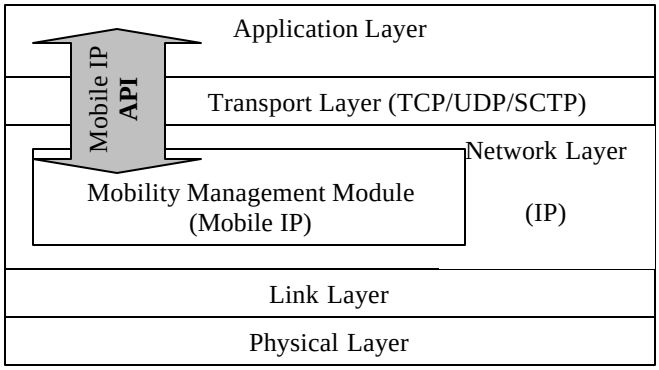

Fig. 1. Network Stack and Mobile IP API

(d). Notification of MNs movement, and new location

Mobile IP API relies on existing Mobile IP protocol for providing movement and location awareness to applications and as such no additional signaling is generated between the mobile node and the correspondent node or any other entity. In essence, at the mobile node, acquiring of a new care-of address triggers the notification to the application about the movement of mobile node. On the correspondent node side, the receipt of a binding update message triggers notification to the interested applications about the movement of the mobile node. We will discuss the working in more details later in Section 4.

2.1 *Scope of Basic Mobile IP API*

It is possible that multiple location awareness applications running on the same host at the same time, and all these applications interact with the same mobility management module. We have taken special care that basic Mobile IP API provides 'read-only' functions, in which the use of API by one application does not affect the operation of any other application on the same host. At later stage, we will define advanced Mobile IP API that will provide more advanced functionalities to applications in privileged mode. Advanced API is briefly described in section 6.

2.2 *Security Considerations for Basic Mobile IP API*

Location privacy is a concern with use of Mobile IP API. As stated above, implementation of this API does not generate extra signaling between the mobile node and the correspondent node. For providing information to the applications running on the correspondent node, the API only relies on receipt of binding update messages on the correspondent node. If the mobile node does not send a binding update message for any reason including location privacy, the API will not provide any information to applications running on the correspondent node. As such, the API does not reveal any information other than what MN is willing to provide.

It is noteworthy that the location information provided by Mobile IP API is strictly in terms of IP address. The mapping of IP addresses to geographical locations is out of scope of this work.

3. USAGE SCENARIOS

In this section, we will describe several scenarios as examples to illustrate the location-aware and mobility-aware applications and services enabled by Mobile IP API. The usage scenarios cover all four kinds of mobility and location awareness as listed in Section 2.

3.1 *Mobile Node Makes Decision Based on its Location Information*

This section describes the scenario wherein the web browser of a mobile client decides whether to download all embedded objects of a web page or not based on the mobile client's location, see Fig. 2.

When a user is roaming in a visited network, the network access fee is usually higher than the fee when user is at home. It avoid downloading large size of image, audio or other types of information than the plain text version of the page at visited networks if those contents are not needed right away. The web browser on MN get mobility information (MN's location) from mobility management module, and initiates web request that includes downloading preference according to the current location. Depending on the request, web pages are downloaded from the server in text only format or with all the embedded objects. For example, in the Fig. 2 above, a browser on MN queries MN's location from MM-layer (1). The application initiates downloading text only web pages (3)

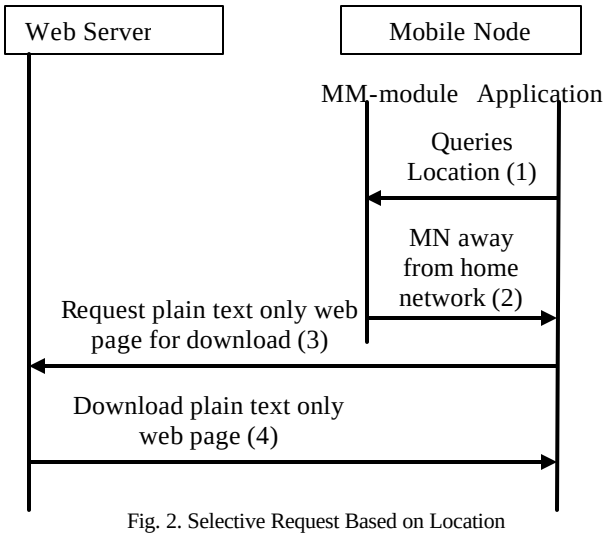

Fig. 2. Selective Request Based on Location

when it realizes the MN is away from home. Web server responds to send text only pages to the client (4).

3.2 *Movement as a Trigger of Service Re-discovery for Applications Running at Mobile Node*

This section describes the scenario where a MN can trigger its service discovery process upon detecting network-layer movement. It is the case that the MN is notified of its own movement and new location.

SLP [2] can be used for discovering local services and servers. An application using SLP for service discovery needs to initiate the discovery process before it can use particular service.

When a mobile node moves to new IP subnet, it needs to discover local services again. It can either wait until the next request for initiating the service discovery or perform discovery as soon as it detects the movement. The latter provides better performance as the service location is already discovered before the client issues the first service request.

In the usage scenario shown in Fig. 3, an application running on the MN is using SLP for discovering local printers. As MN performs a network-layer handover, it acquires a new care-of address (CoA) (1). Application is notified of this movement (2) and initiates local printer discovery (3). Local printers are discovered using SLP and the information is cached (4). When the application needs to send a request to a local printer (5), it does not need to start a discovery process and it can use one of the cached printer servers.

3.3 *Location Based Content Delivery*

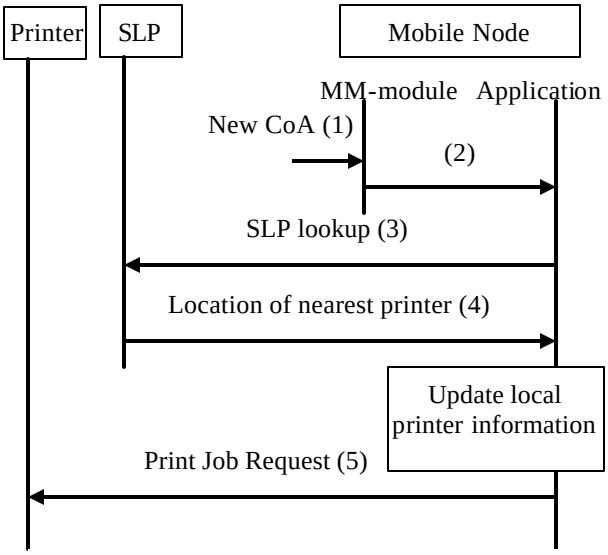

Fig. 3. Selective Request Based on Location

This section describes the scenario where content delivered to a MN can be customized based on its location. It is the case where a CN get notification of MN's movement, and new location.

In this scenario, the CN is a content server capable of serving location based content. The CN keeps track of MN's movement by receiving a binding update from MN. In Fig. 4 below, each time CN receives a binding update form MN (1), content server application get mobility information from mobility management module on CN (2). The application can map the MN's IP address to its geographic location (3), and dispatch location-based content to MN.

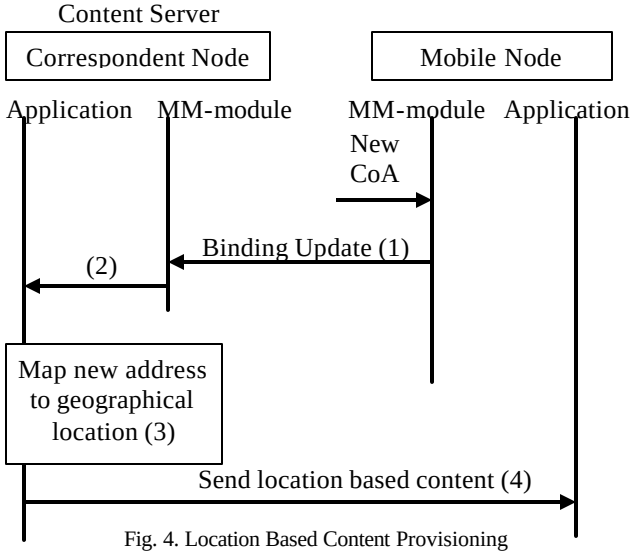

Fig. 4. Location Based Content Provisioning

4. DEFINITION AND IMPLEMENTATION OF MOBILE IP API

We have defined the Mobile IP API in a way that it is applicable to both Mobile IPv4 [4] and Mobile IPv6 [3]. Following are data structures and functions interfaces in C language.

4.1 *Data Structures and Constants*

The union *ip_addr* holds IPv6 address and IPv4 address and is usually defined in the <netinet/in.h> header file.

```
union ip_addr {
        struct in6_addr      ipv6_addr;
        struct in_addr       ipv4_addr;
};
```

The *mobile_node_t* stores the home address and care-of address (es) of MN.

```
struct   mobile_node_t {
```

```
        unsigned int     addr_family;
        union ip_addr    home_addr;
        unsigned int     num_coa;
        union ip_addr    *coa;
};
```

The *addr_family* field specifies whether the MN is using IPv4 or IPv6. Both home address and the care-of address (es) must belong to the same family. The value of this field should be *AF_INET* for IPv4 and *AF_INET6* for IPv6. *AF_UNSPEC* is acceptable in certain cases.

Since Mobile IPv4 supports simultaneous bindings, it is possible for a MN running Moblie IPv4 protocol to have more than one care-of addresses simultaneously. *coa* points to an array of care-of addresses. *num_coa* is the number of valid care of addresses.

4.2 *Functions*

This section briefly describes some of the functions that are part of the Mobile IP API. Detailed information can be found in [6].

4.2.1 Information about Mobile Nodes

Applications on either the mobile node or correspondent nodes can call *mip_get_one_mobile_node()* function to get information about the mobile node.

```
int mip_get_one_mobile_node(
        mobile_node_t *mobile_node);
```

The home_addr and addr_family fields are initialized with the looked up address and address family, respectively. This function will update the care-of address(es) of the *mobile_node* upon successful lookup.

4.2.2 Movement Notification

Applications on either the mobile node or correspondent nodes can call *mip_notify_movement()* function to be notified upon movement of mobile nodes. Applications can do so either in non-blocking mode by registering a callback function for notification, or in blocking mode wherein a callback function is not used rather the *mip_notify_movement()* function itself blocks until the event of interest is detected.

```
int mip_notify_movement(
        mobile_node_t *mobile_node,
        int non_blocking,
        unsigned int timeout_ms,
        long int cb_parameter,
        int (*callback)(
                mobile_node_t mobile_node,
                int event,
                long int_cb_parameter)
        );
```

The *mobile_node* specifies the mobile node on which the notification is desired. If the *addr_family* field of the *mobile_node* is set to *AF_UNSPEC*, or to any particular address family but the home address is left unspecified (i.e. 0::0 or 0.0.0.0), then it is treated as a wildcard and application will get notification for all mobile nodes. The *non_blocking* parameter indicates whether the application wants to be notified in *non_blocking* mode or not. *Timeout_ms* parameter gives the time-out value during the blocking waiting. The *cb_parameter* is the input parameter for the call back function, and the *callback* is the function pointer of the call back function. The last two parameters are only valid in non-blocking mode. The call back function itself has three parameters. The first specifies the MN on which the event has occurred, the second parameter indicates which type of event this call back function registers with, and the last one is the parameter passed from the *mip_notify_movement* function.

4.2.3 HOME Address Testing Macro

Application may use this macro to check whether the specified mobile node is at home or not.

```
int IS_AT_HOME(struct mobile_node_t* mobile_node);
```

The macro returns non-zero if the mobile node's home address is equal to its current care-of address, and zero otherwise.

4.3 *Implementation*

We have implemented the Mobile IP API in C as a shared library in Linux kernel version 2.4.16. Currently, the API is for IPv6 only, and the basic Mobile IPv6 implementation is ported from the MIPL project at Helsinki University of Technology.

In MIPL project, the MIPv6 protocol is implemented as a separated kernel module from other kernel modules, and several MIPv6 related kernel data structures have been exported to the Linux proc file for diagnostics and configuration purpose. By retrieving binding update list, binding cache, and other mobile node configuration information from the proc file, we implemented most of the MIP API functions in user space.

We also modified mMosaic, an IPv6 enabled web browser, to support the location-aware downloading by using the Mobile IP API functions as described in Section 3.1.

First, we added a new configurable option, named "delayed at foreign network" into mMosaic, to make users have the

choice of whether to down load all the embedded objects of the web page or just the plain text version of the page when the mobile node is away from networks. Each object has an attribute named "delayed" which indicates whether this object will be downloaded or not. When the browser parses the html page, it retrieves the mobility information by calling *get_one_mobile_node()* function and checks if the mobile node is at home or not by using the *IS_AT_HOME* macro. If the "delay at foreign network" option is enabled and mobile node is at visiting network, the mobile node will set the "delayed" attribute of all objects embedded in this page to be true. The browser does not send the loading request to the web server if the object's "delayed" attribute is true.

Most web pages have rich contents. Our implementation shows that the data traffic of MN is greatly reduced in the foreign network when the embedded objects are filtered out. It is clear that location-aware web browser would benefit users in the roaming area where the service fee is usually higher than the home network.

5. Standardization Efforts

Currently, we are pursuing standardization of Mobile IP API at IETF [6]. At later stage, we will consider to make this become part of POSIX (Portable Operating System Interface) standard and UNIX standard which will be endorsed by the IEEE and ISO as well as by The Open Group. We are also looking forward to proposing it to Java Community Process (JCP) and turn it into cross platform compatible Java API.

6. Future And On Going Work

As an on going work, we are developing an advanced API which will provide much finer granularity control to the applications. This API will allow not only acquiring of mobility related information but also manipulating the behavior of mobility management module. This API is designed with a view of forth-coming network environments that will exhibit wide variety of overlapping heterogeneous access networks, multi-interface and multi-link capable devices, and possibly very ambitious applications that would want to exploit these rich network environments in every possible way. Route optimization, multilink load balancing, and binding update security on per application basis are examples of new functionalities enabled by advanced Mobile IP API.

7. Related Work

Proposal in [5] presents a similar work for creating a channel for information sharing among different OSI layers. MIBsocket provides a general tool to manipulate and acquire general network information at various modules including IP, ICMP, TCP, etc. Although it is not specifically illustrated, a similar information channel akin to Mobile IP API can be formed by using this generic interface. The essential difference between Mobile IP API and MIBsocket is that the former is specifically designed for Mobile IP by taking typical usage scenarios into consideration. Attention has been given to satisfy the needs of these applications with minimum complexity on the application side. As such, Mobile IP API provides information in its most useful form without requiring much complexity additional processing by the applications. Achieving similar results by using MIBsocket would require inserting a Mobile IP specific module before the data are processed by the applications.

8. Conclusion

Mobile IP API enables development of location and movement aware applications that can dynamically adapt themselves to their changing environments. We have designed basic Mobile IP API based on a set of usage scenarios. We have also illustrated usage of this API by modifying an application to use our API implementation. It has been observed that applications can be easily updated to take advantage of this interface. Development of intelligent applications for mobile and wireless Internet is made possible by this API.

Reference

[1] P. Bhagwat, C. Perkins, and S. Tripathi. "Transparent Resource Discovery for Mobile Computers". IEEE Workshop on Mobile Computing Systems and Applications, 1994.

[2] E. Guttman, C. Perkins, J. Veizades, M. Day. "Service Location Protocol, Version 2". RFC 2608, June 1999.

[3] D. Johnson and C. Perkins. "Mobility Support in IPv6", IETF Internet Draft, work in progress, April 2001.

[4] C. Perkins, editor. "IP Mobility Support for IPv4", RFC 3220, January 2002

[5] R. Wakikawa, K. Uehara, F. Teraoka, and J.Murai. "MIBsocket: An Integrated Mechanism to Manipulate Network Information in Mobile Communications". IEICE Trans. of Commu. Vol. E84-B, No8 pp2001-2010, August 2001.

[6] A. Yokote, A. Yegin, M. Tariq, G. Fu, C. Williams, A. Takeshita. "Mobile IP API", IETF Internet Draft, work in progress, May 2002.