Providing Movement Information to Applications in Wireless IPv6 and Mobile IPv6 Terminals

Jarno Kalliomäki, Bilhanan Silverajan, and Jarmo Harju

Institute of Communications Engineering,
Tampere University of Technology,
P.O. Box 553,
33101 Tampere Finland
firstname.lastname@tut.fi

Abstract. Innovative, adaptive and context-aware applications today are poised to take advantage of their immediate surroundings for interaction, both with the user as well as with other surrounding devices. Often, these applications reside in mobile devices, and the need for obtaining movement detection information is placed at a premium. However, very little work actually exists in bringing this information to such applications in a uniform way. In this paper, we address the lack of consistent application level support for obtaining timely information related to network-level movement detection and location awareness, by presenting the design and architecture of a movement notification system to support advanced mobile applications. This system was designed primarily for use for wireless devices moving in IPv6 and Mobile IPv6 spaces, although it can also be used for IPv4. Future work on the system is also discussed.

Keywords: Movement Detection, Mobile Computing, IPv6.

1 Introduction

Future wireless and pervasive computing environments are envisaged to support the ability for a diverse range of portable wireless peripherals to roam seamlessly across different wireless networks. The gradual development of higher wireless data rates in the next few years, and the proliferation of mobile devices such as laptops, phones and PDAs, point to a distant but realistic possibility: the emergence of networks which only need a wired infrastructure for access points, switches, routers, aggregators and other network elements, but not to the devices themselves. Everything ranging from video streaming to VoIP, seamless and transparent handovers between WiFi and 3G networks and location-based services are being touted as killer applications for these devices.

IPv6 and Mobile IPv6 will be the most likely technologies to drive the deployment for such future networks. IPv6 offers automatic address configuration, and lifts the NAT tax by supplying a global address space supporting billions of unique hosts. With the inclusion of IPsec, end-to-end reachability can be securely obtained. With Mobile IPv6, mobile terminals can always remain reachable via home addresses, independent of their location, but also receive temporary Care-Of Addresses.

A. Pras and M. van Sinderen (Eds.): EUNICE 2007, LNCS 4606, pp. 25–32, 2007.

This extra degree of mobility and the desire for seamless connectivity will give rise to new breeds of network applications and services no longer relying on traditional abstractions that Internet services have long been modeled on. The proper functioning of these applications might depend on information gathered from the immediate vicinity though often, when roaming into network spaces for the first time they will have no previous knowledge of the types of services and other devices resident in the new network space. Also, network attachment points may change when the devices discover new access networks or lose connectivity to previously attached networks.

Therefore, one of the primary issues in developing such advanced applications is to address the lack of consistent application level support for obtaining timely information related to network-level movement detection and location awareness, which may be inherently and readily available at the local interfaces of these devices.

This paper highlights our activities in raising the importance of supporting rapid movement detection as an underlying foundation for the development of advanced mobile applications, and the importance of exporting this information for use in a consistent manner that may reside in these devices. A movement notification system named Mobinfo was designed and implemented to support advanced mobile applications.

Section 2 provides a quick survey of how movement notification can benefit different classes of applications. Section 3 provides a detailed design of Mobinfo. Section 4 outlines the future direction of work with our movement notification system, and Section 5 draws some conclusions.

2 Movement Notification: Usage Scenarios

Movement detection and notification would prove beneficial to many classes of applications. Some of these are described in the following paragraphs.

Mobile applications running in Mobile IPv6 mobile nodes would remain unaware they have moved, as the IP layer shields movement from upper layer. In cases where these applications have roamed into foreign networks and attempt to initiate unicast or multicast communication using UDP, source address selection procedures stipulate the home address be used by default [1]. To avoid suboptimal routing, it would be beneficial to notify applications to enable them to be movement aware, so that their care-of-addresses can instead be used. This would also be especially useful to applications that are location- and context-sensitive which require rediscovery of essential services for proper operation, such as finding or using a multicast DNS server [2], printing and file services as well as proxy settings.

When multicast-aware applications and devices roam into new networks, they would first need to detect movement into the new network space and subsequently rejoin multicast groups they were previously members of. This reduces latency and the perceived interruption in service, by not waiting for Multicast Listener Queries from the local router in the foreign network before responding and rejoining the multicast groups. Examples are multicast audio/video streaming receivers and agents using multicast-based service discovery protocols such as SLP [3].

Overlay networks such as application-level P2P networks build a routing topology, the performance of which is dependent on efficient connectivity paths among nodes in

the underlying network layer. Some overlay or P2P networks are also self-organising and location-aware [4], taking into account changes by nodes entering or leaving the network. Optimal routing decisions in these networks can be achieved if movement information is available to reflect the changing network paths between nodes.

3 Mobinfo: Motivation and Design

Work on Mobinfo draws upon lessons learnt in previously designing a Mobile IPv6 movement detection library specifically for Linux 2.4-based systems, which hooked directly into the kernel source code [5]. For applications, the IETF mip6 (Mobility for IPv6) working group also standardised API support that allows Mobile IPv6 applications and implementations access to Mobility binding messages and Return Routability messages [6], thereby extending the Advanced Socket API for IPv6 [7]. Research has also been done optimising movement detection in Mobile IPv6 [8].

It can be observed that much of the related work in IPv6 movement detection either pertains directly to Mobile IPv6 such as with [5], [6] and [8], requires applications to execute with root privileges such as with [6], or inflexibly supports only a very specific architecture, such as with [5]. In contrast, our aim is for the design of Mobinfo to be more resilient, portable not just across different series of Linux kernels, but also across a range of UNIX and Windows operating systems. It should also provide both movement and interface information independent of whether Mobile IPv6, native fixed IPv6 or transitional IPv6 (such as 6to4 or Teredo) is being used by the terminal. Function calls should be provided for both synchronous (blocking) and asynchronous (non-blocking) modes, to applications needing movement notification. Additionally, Mobinfo should not impose any design or execution constraints onto applications interested in receiving movement information, such as requiring them application to be multithreaded or to execute with root privileges.

Mobinfo is written in ANSI C++. It has been developed and tested using Linux-based nodes. These nodes, apart from having native IPv4 and IPv6 functionality, also support Mobile IPv6 via MIPL [9].

The notification system comprises two components. Firstly it has a shared library which any user program requiring interface information or movement detection is linked against. Secondly it also has a service daemon called Mobinfod which is responsible for obtaining low level information from the interfaces of the device and passing it on to the library. In order to access and examine low-level information, Mobinfod uses a third party packet capture library called libpcap [10]. The daemon and the shared library execute in separate processes; the daemon, together with libpcap runs in one process with privileged permissions, while the shared library, together with the associated application executes in a normal user process. Consequently, FIFO queues are used between the two parts, as FIFOs provide a fairly good implementation for loosely decoupled interprocess communication.

There is a well-known inbound queue for the daemon part which all library instances use in sending their subscription requests. Every library instance also creates an individual queue for receiving messages from the daemon. If necessary, the library instance spawns a thread to handle all non-blocking instances. Should a thread be spawned, it will also receive messages from the daemon via its own FIFO.

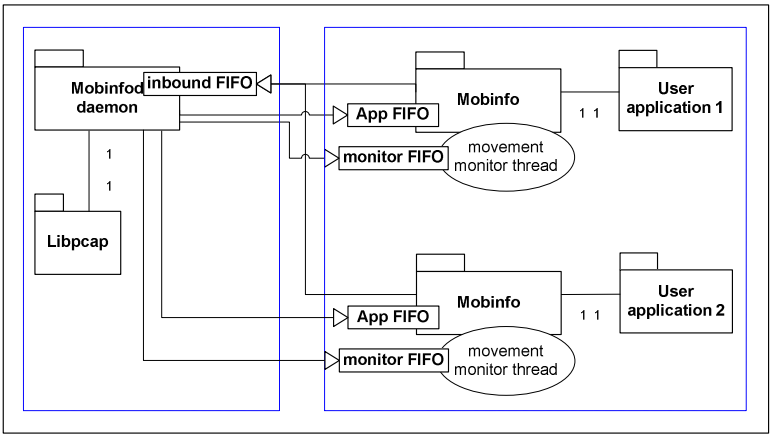

Fig. 1. Architecture of Mobinfo

The overall design and architecture of the movement notification mechanism in UML is depicted by Figure 1, while the facilities of the 2 components are described in greater detail in the following subsections.

3.1 Mobinfo Shared Library Component

The library provides several structures and member functions to applications, to best represent or select the different types of information they need. The following code describes the datatypes defined by the library that applications can use:

```
struct MobinfoAddrInfo {
   short type;
   PrefixType prefix;
   union {
      sockaddr_in ip4_addr;
      sockaddr_in6 ip6_addr;
   } ip_addr;
};

struct MobinfoIface {
   std::string name;
   std::vector< MobinfoAddrInfo > addresses;
};

enum PrefixType {UNKNOWN, UNSPECIFIED, LOOPBACK,
GLOBAL_UNICAST, UNIQUE_LOCAL_UNICAST,LINK_LOCAL_UNICAST,
MULTICAST, _6TO4, TEREDO};
```

The *MobinfoAddrInfo* structure encapsulates IP address information obtained from the daemon to be passed to the application. It currently contains a discriminator to distinguish between an IPv4 or IPv6 address. In addition it contains an enumerated value categorising common address types.

The *MobinfoIface* structure encapsulates the device's interface information. Each structure contains the name of the interface as well as a vector of all addresses associated with the interface.

In addition, the library provides function calls to applications. These function calls (and their signatures in C++) are listed below:

1. bool Mobinfo::getInterfaces(std::vector< std::string >& interfaces)
2. bool Mobinfo::getInterfaceInfo(std::string interface,
   std::vector< MobinfoAddrInfo >& addresses)
3. bool Mobinfo::getAllAddresses(bool ipv4Included, bool ipv6Included,
   std::vector< MobinfoAddrInfo >& addresses)
4. bool Mobinfo::getAllInfo(std::vector< MobinfoIface& info)
5. bool Mobinfo::notifyMovement(void (*callback)(MobinfoIface newinfo))
6. bool Mobinfo::notifyMovement(std::string interface,
   void (*callback)(MobinfoAddrInfo newAddr))
7. void Mobinfo::cancelNotification( )
8. bool Mobinfo::waitForMovement(unsigned int timeout, MobinfoIface& newInfo)
9. bool Mobinfo::waitForMovement(unsigned int timeout, std::string interface,
   MobinfoAddrInfo& newAddr)
10. void Mobinfo::setTimeout(unsigned int timeout)
11. unsigned int Mobinfo::getTimeout( )

Function calls with a *boolean* return value indicate if the requested operation was successfully carried out or not. With the exception of *notifyMovement*, all others are blocking function calls.

GetInterfaces takes a *vector* container of *string* types as a reference and populates the vector container with the names of all known interfaces of the device. *GetInterfaceInfo* accepts one *std::string* parameter representing a specific interface name, searches for an interface with the given name and provides all the addresses attached to that interface into the supplied vector. *GetAllAddresses* can be used if the caller just wants every address available, regardless of the interface. There are two *boolean* parameters that indicate which protocols the caller is interested in: IPv4 and IPv6. Lastly, *getAllInfo* supplies all addresses to the invoking application, categorized by interface.

The *notifyMovement* function calls are non-blocking function calls. The first form simply accepts a callback function as a parameter for invocation whenever any change is detected in any of the device's interfaces. The second form monitors a specific interface, and invokes the callback to return the address information pertaining only to a single interface. Registered callbacks are discarded with the *cancelNotification* function call. The *waitForMovement* function calls are the blocking counterparts to *notifyMovement*. In order to prevent them from blocking indefinitely, a timeout value can be specified in milliseconds. A value of 0 indicates the call should block until movement has occurred.

The *setTimeout* and *getTimeout* functions are used in manipulating timeout values used for communicating by the library with the daemon over the library's FIFO queue with the *poll()* system call. It is given in milliseconds.

3.2 Mobinfo Daemon

Upon startup the daemon queries for all known interfaces and addresses via the low-level API provided by libpcap and stores returned results in its cache. Periodic queries are then issued and results compared with those in the cache. In the case of a new address appearing, movement notification is sent to library instances registered to receive it. The communication between these parts is carried out by using the following sets of messages:

- SUBSCRIBE, SUBSCRIBED_ACK
- UNSUBSCRIBE, UNSUBSCRIBED_ACK
- REQUEST_INFO, NODE_INFO
- REQUEST_MOVEMENT_NOTIFICATION, MOVEMENT_NOTIFICATION
- CANCEL_MOVEMENT_NOTIFICATION, MOVEMENT_NOTIFICATION_CANCELLED_ACK
- UPDATE_QUERY, QUERY_RESPONSE

The SUBSCRIBE and SUBSCRIBED_ACK are only used when a library instance starts communicating with the daemon. Similarly the UNSUBSCRIBE and UNSUBSCRIBED_ACK are used when a library instance wants to end the interaction with the daemon.

A REQUEST_INFO message is used when a library instance wants to update its local information and requests it from the daemon. The daemon then responds with a NODE_INFO which breaks into following sub-messages:

- DATA_HEADER
- DATA
- END_OF_DATA

A DATA_HEADER will always precede a DATA message informing the receiver of which type the DATA is. The receiver can then resolve the DATA message length which is necessary when receiving messages from a FIFO queue. An END_OF DATA message informs the receiver there is nothing further to send.

A REQUEST_MOVEMENT_NOTIFICATION will register the caller to receive movement notifications. The daemon will presume a monitor thread is already running with its own FIFO in the client side and shall send it a MOVEMENT_NOTIFICATION message in the event of movement occurring.

If the library instance is willing to cancel the movement notification it will send a CANCEL_MOVEMENT_NOTIFICATION message to the daemon. The library may not close its inbound queue before the cancelling has been acknowledged by the daemon by using a MOVEMENT_NOTIFICATION_CANCELLED_ACK message.

UPDATE_QUERY message is used by the library when it needs to check if the local address information has been outdated. The daemon will respond to this query with a QUERY_RESPONSE message. Figure 2 shows an example message flow.

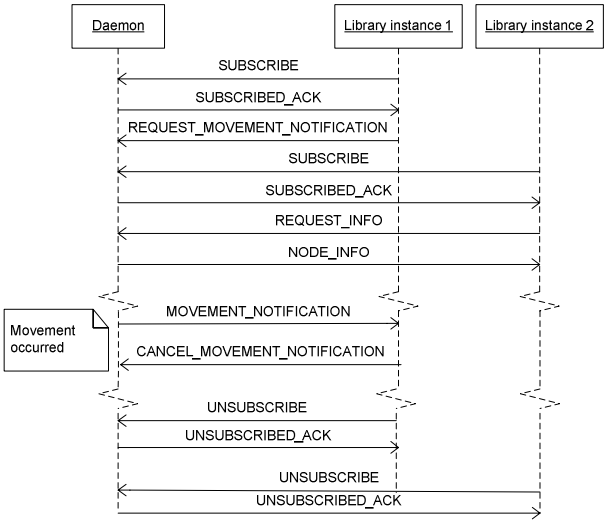

Fig. 2. Interaction between the daemon and library instances

4 Future Work

Although our primary area of interest is developing support for IPv6 (both native and transition-based) and Mobile IPv6 application development, movement notification in IPv4 space can also be easily performed. No support has yet been added to detect moving between public IP addresses and private IP addresses. Additionally, if two distinct networks both use private IPv4 addresses with the same range, difficulties detecting movement would arise when the device moves from one network to another.

Mobinfo uses libpcap to obtain interface and address information. This brings a huge advantage in that libpcap is in constant development to support new protocols, address families and interfaces, both physical and virtual (such as tunnels). Apart from supporting the monitoring of interface addresses in the Internet address families, we can also harness libpcap's experimental abilities in future to obtain information from some Bluetooth protocol stacks to extract movement from local area networks into personal area networks. In addition, we can easily substitute libpcap for winpcap (the Windows equivalent) to allow the execution of Mobinfo in Windows and Windows Mobile platforms, apart from Linux, BSD-based and Mac OS X platforms.

Apart from the function calls now provided to the applications to discover changes to interface addresses, intelligence can be incorporated to also supply more information such as the routability of a newly obtained address. A possible way of accomplishing this would be to utilize libpcap's packet capturing functionality in listening to router advertisements. However at times, this may not be as straightforward. With Mobile IPv6, a new care-of address is obtained whenever a mobile node moves into a new network. Until route optimization procedures are conducted successfully however, the home address would still be the preferred source address with packets using a bidirectional tunnel to the mobile node's home agent.

Other forms of movement detection can also be added into the notification system. An interesting development that needs to be considered is movement in geographic spaces. This could be accomplished if the mobile device possesses GPS reception capabilities.

5 Conclusion

Innovative, adaptive and context-aware applications today are poised to take advantage of their immediate surroundings for interaction, both with the user as well as with other surrounding devices. Often, the need for obtaining movement detection information is placed at a premium. However, very little work actually exists in bringing this information to such applications in a uniform way.

Mobinfo aims to bridge the gap between lower level interfaces of the device and the application by providing a comprehensive library API that preserves the richness of the information obtained for processing by applications, without application developers needing to devote time to create their own mechanisms. In doing so, it can accelerate the development, deployment time and adoption of a new generation of mobile applications.

References

1. Draves, R.: Default Address Selection for Internet Protocol version 6, RFC 3584 (February 2003)
2. Chesire, S., Krochmal, M.: Multicast DNS, IETF work in progress (August 2006)
3. Guttman, E.: Service Location Protocol Modifications for IPv6, RFC 3111 (May 2001)
4. Wu, C., Liu, D., Hwang, R.: A location-aware peer-to-peer overlay network. International Journal of Communications Systems 20(1), 83–102 (2007)
5. Borst, M., Silverajan, B.: Movement Notification in Mobile IPv6. In: Proceedings of 10th European Summer School and IFIP WG 6.3 Workshop, Tampere, Finland (June 14-16, 2004)
6. Chakrabarti, S., Nordmark, E.: Extension to Sockets API for Mobile IPv6. RFC 4584 (July 2006)
7. Stevens, W., Thomas, M., Nordmark, E., Jinmei, T.: Advanced Sockets Application Program Interface (API) for IPv6. RFC 3542 (May 2003)
8. Daley, G., Pentland, B., Nelson, R.: Movement detection optimizations in Mobile IPv6. In: Proceedings of 11th IEEE International Conference on Networks (ICON 2003), Sydney, Australia, September 28 - October 1, 2003, IEEE Computer Society Press, Los Alamitos (2003)
9. MIPL: MIPL Mobile IPv6 for Linux, http://www.mobile-ipv6.org
10. Libpcap packet capture library, http://www.tcpdump.org