A Mechanism for Host Mobility Management supporting Application Awareness

Arjan Peddemors Hans Zandbelt Mortaza Bargh
Telematica Instituut
Enschede, The Netherlands
{Arjan.Peddemors, Hans.Zandbelt, Mortaza.Bargh}@telin.nl

ABSTRACT

Many approaches exist today that address the issues that arise when a mobile node changes its point(s) of attachment to the Internet. Mobile IP takes care of host mobility at the IP layer; others at the transport layer (Mobile SCTP) or at the application layer (SIP with re-invite). In practice, most of these approaches rely on functionality residing on the mobile host that scans, detects and activates the networks available through one or more network interfaces.

The mechanism proposed in this paper takes into account that multiple of these approaches may be applied at the same time. It provides the applications on the mobile host with information about the state of the lower-layer mobility management protocols (such as Mobile IP) as well as the state and characteristics of the available network resources. Applications may consecutively adapt their behavior depending on this mobility process information and thus accommodate to the changed network connectivity conditions, possibly in an application specific manner. In this paper, we present the architecture of our mobility management mechanism. We also describe the implementation of our prototype and the results of experiments with the mechanism, thereby addressing the complexities of an integrated application-aware mobility management system.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design – *Network communications, Wireless communication*; D.4.4 [**Operating Systems**]: Communications Management – *Network communication*

General Terms

Management, Design, Experimentation

Keywords

Mobility Management, Host Mobility, Application Awareness

MobiSys'04, June 6-9, 2004, Boston, Massachusetts, USA.

1. INTRODUCTION

The advances in mobile device capabilities have been remarkable in recent years. Mobile phones, PDAs, and notebooks have become more powerful, more versatile and better usable and this progress is likely to continue in the near future. At the same time, the availability and applicability of new network technologies such as the IEEE 802.11 Wireless LAN series, and Bluetooth, but also new developments in 2.5G and 3G cellular network standardization, make clear that we live and continue to live in a world of heterogeneous network technologies. The Internet Protocol is the binding factor between these network technologies: even the traditional applications that strongly rely on circuit-switched networks characteristics, such as voice calls, are more and more used in an IP environment. For this paper, we will therefore consider an all-IP network situation.

With the evolution of mobile technologies towards more advanced devices and faster networks, the networked applications running in mobile environments will further evolve from applications known and implicitly targeted at fixed workstations towards applications that can be regarded as "truly" mobile. True mobile applications explicitly take into account the fact that they run on a mobile device that is connected to one or more dynamically changing networks. They will not be part of a niche domain but will be running on consumer-market devices such as advanced mobile phones.

Mobile applications must take into account that they typically operate in a highly dynamic network environment. The mobile device may connect and disconnect dynamically to, mostly, wireless networks. The wireless networks in general expose a more dynamic behavior in terms of variation of available bandwidth and delay than their fixed counterparts; they occasionally may not even be available at all for a short period of time (when the device owner is driving through a tunnel, for example). These dynamic network characteristics are inherent to the mobile environment and they impose a specific set of constraints on the connectivity handling functionality in the mobile terminal on both the network (IP) and application level.

The mobile device may have various means to provide so-called seamless mobility to the applications. It may implement Mobile IP Mobile Node (MN) functionality to supply mobility management at the IP layer. Or it may provide a Mobile Stream Control Transmission Protocol (Mobile SCTP) implementation handling mobility management at the transport layer (see also the next section). A well functioning real-world mobile system, however,

must make sure that whenever networks appear or disappear, the mobility management functionalities are triggered to adjust to the new situation. We call this process of network activation and deactivation, and the updates in the state of the mobility management components the *mobility process*.

This paper proposes a system-level mobility management mechanism for mobile hosts in a heterogeneous network environment. The mechanism provides means to inform the applications running on the mobile host about the state and the events in the mobility process and about the state and characteristics of the available network resources. It cooperates and interacts with existing protocols and mechanisms, such as Mobile IP or SCTP, which are specifically designed for or can be applied to accommodate node mobility within the Internet.

The remainder of this paper is organized as follows. In the next section, we will give a description of the background, motivation and our guiding principles. Section 3 focuses on the architecture; section 4 and 5 discuss the implementation of our mechanism and the experiments we have done for its initial validation. Section 6 evaluates and discusses our results and section 7 provides an overview of the work that relates to our approach. The final section presents our conclusions and thoughts about our future work.

2. BACKGROUND AND MOTIVATION

This section describes the mobile device context that we consider to be representative for devices in future all-IP network environments. We propose a classification of applications that may run on the mobile terminal. Additionally, we discuss a number of principles that we used as guidelines for the construction of the mechanism architecture.

2.1 Mobile device context

Many kinds of wireless technologies currently do co-exist and it is likely that their number will further increase in the near future. Mobile devices, also low-end models, will be equipped with multiple network interfaces that facilitate access to different kinds of wireless networks. This requires that, on the mobile host, functionality is in place that manages the activation of network access through the network interfaces. For uninterrupted IP services, it is important that this activation can be performed without explicit interaction with the user.

Much effort has been spent on solving the issues that arise when a node becomes mobile and changes its point of attachment to the Internet. This has led to the development of various protocols and protocol implementations that address these problems. To date, the most notable of these solutions is Mobile IP [16][9]. This protocol ensures that the mobile node does not need to change its primary IP address that is used in connectivity sessions. However, it is not obvious that Mobile IP will be applied in all (mobile) circumstances. It requires at least one extra entity in the network for keeping track of the mobile connectivity state: the Home Agent (HA). Its deployments may be problematic, for instance in case of coexistence with firewalls and Virtual Private Networks (VPNs). Mobile IP will only be a viable solution when it can be applied in the majority of access networks that the user normally visits.

Other existing protocols and techniques can be applied to address attachment issues at the transport layer (Mobile SCTP [18] or TCP-Migrate [19]) or at the application layer (for instance, the Session Initiation Protocol (SIP) with re-invite [23]). Often, these follow an end-to-end approach, which does not require extra functionality in the network, except, when needed, for location management.

SCTP supports the reliable transfer of data between two endpoints over a so-called association. Contrary to a TCP connection, an SCTP association may use multiple IP addresses per peer (multi-homing). The set of IP addresses per peer may be altered during the lifetime of the association (through Dynamic Address Reconfiguration [20]), which can be used to sustain the association while one or both peers roam between access networks.

We believe that, at this point in time, it is unclear whether there will be a single dominant protocol or technique for mobility management. Additionally, one specific mobility management technique may be more suitable than another for certain kinds of applications [12]. Or, different mobility management techniques may better suit a single application in changing circumstances [11]. Therefore, we want our mechanism to simultaneously support multiple solutions.

When the mobile device connects to a network, it usually must supply credentials for authentication and authorization. In general different and mostly non-cooperating parties administer these networks. In many cases, this means that the user must supply credentials before the network can be accessed, which, naturally, may harm the objective of seamless connectivity. We assume, however, that an average user will not enter new networks, governed by unknown parties, very frequently, and that it is acceptable to occasionally provide new credentials.

2.2 Application classification

From the perspective of mobility management, different types of applications can run on the mobile device. We define two characteristics that we use for our classification of applications: 1) whether or not the application is aware of the mobility process and 2) whether or not the application itself is dealing with the management of its connections and session in the event of changing connected networks.

Table 1: application classification

	Aware of mobility process	Managing own connection / sessions
Type I application	no	no
Type II application	yes	no
Type III application	yes	yes

A type I application is not aware of any changes at the network level. It assumes that, in normal circumstances, a connection to another node on the Internet is not disrupted. This is the default behavior of many applications that normally run in a non-mobile environment.

Applications of type II are aware of the mobility process, but do not want to maintain their connections themselves. They do not

want to implement their own strategy for the handover of their connections from one interface to another, but rather rely on system level functionality for this purpose. The classical example is a video stream player that tells the video server to adapt its bandwidth usage, depending on the characteristics of the available networks. The player depends on a mechanism such as Mobile IP to ensure that the same UDP connection remains established between server and client even while the client has switched to a different access network and thus has changed its point of attachment.

The type III application is aware of the mobility process and managing its own connections. In the example above, it would tear down the UDP connection over the old network and reinitiate it over the newly available network. This application level mobility management may be beneficial when no lower level mobility management functionality is available, or when it is relatively easy to reinitiate the connection in the scope of the work that needs to be done anyway to adapt to the new situation.

The trade-off when developing a type II or type III application is between implementation effort and optimized application behavior during handovers. A type II application relies on the system mobility functions that typically are configured to meet the common denominator between all application and user requirements and thus cannot deal with any specific application requirements. By controlling the network attachment strategy and handover timing themselves, different applications can manage their own connections taking into account their different characteristics. Examples are a low bandwidth Instant Messaging utility that preferably stays connected to a WAN all the time as opposed to a video streaming tool that prefers a network connection with the highest bandwidth available even if this induces frequent handovers with occasional packet loss.

2.3 Guiding principles

This subsection provides the set of most important principles that we used as a guideline when constructing the mobility management mechanism:

a) Applications on the mobile host can often benefit from information about the mobility process. They may need information about the estimated bandwidth for currently active access networks, upcoming changes in the set of available access networks, etc, to provide the best possible service to the user. This means that, from the perspective of some of the applications, we are moving away from IP as the single provider of abstraction towards beneath-IP layer information. An additional abstraction of mostly link and IP layer information must be introduced that is capable of expressing what is going on in the mobility process and capable of describing the (dynamic) characteristics of the available network resources.

b) Simultaneously support the cooperation with and incorporation of existing mobility management protocols such as Mobile IP, Mobile SCTP, and SIP re-invite, at the network, transport and application layer. This provides flexibility towards applications running on the mobile host: they can select the protocol that matches their current needs best. Additionally, it provides alternatives when one protocol cannot be used for a certain network configuration.

c) Control is at the end point – the mobile host – with a clear division of responsibilities for the operating system and the mobile applications. The operating system makes decisions about the activation of access networks, routing settings for these networks, and the state and usage of below application layer mobility protocols. No entity in one of the networks influences these decisions (naturally, the user can be denied access to a network). The applications running on the mobile host do not influence the operating system decisions either: they may run their own application layer mobility management and may use the available network resources, but they do not, for instance, take part in the selection of access networks. In general, it may be possible to construct a mechanism that lets the operating system and applications share responsibilities. We think, however, that enabling application influence will quickly lead to conflicts when applications have opposing interests. At this stage, we focus on a clear division of responsibilities.

d) The mobile host provides one or more mobility management components to support seamless mobility. One aspect of seamless mobility is real-time behavior. The real-time characteristics vary per protocol and continue to be a topic of research. The end-to-end nature of Mobile SCTP and SIP re-invite, for example, may results in high latency handovers. For IP layer mobility, a number of alternatives and improvements to Mobile IP have been proposed that essentially group IP subnets into mobility domains and minimize the scope (and latency) of most updates to the domain only: Cellular IP (CIP) [4], HAWAII [17], Mobile IP Regional Registration (MIP-RR) [6], Mobile IP with Location Registers (MIP-LR) [8], etc. The real-time requirements for a single protocol are much less strict when switching from one overlay network to another. In that case, the mobile host simultaneously has access to two or more networks: the handover time is not influenced by the time needed to establish link layer connectivity, authentication and authorization. Additionally, many applications do not need very fast handovers, because they are not real-time by nature or they can occasionally cope with short network interruptions. Here, we do not focus on real-time aspects.

3. ARCHITECTURE

The main functional entity in our architecture is the mobility manager. The mobility manager is a system-level service (or daemon) that resides on the mobile host as a background process. It is responsible for the management of the host's network mobility. The applications on the mobile host can connect to the mobility manager to receive events that describe the mobility process and the available network resources.

The mobility manager listens for events from the operating system about changes in the network interfaces, the availability of network scan results, timeouts, etc. Based on these events and based on the configuration of the mobility manager, it takes actions to, for instance, set the IP configuration for network interfaces, manipulate the host's routing table, initiate scanning on

certain network interfaces, etc. Additionally, it controls any available IP layer and transport layer mobility management components on the mobile host. The mobility manager is not aware of components that take care of application layer mobility management: these typically come in the form of libraries that are linked with applications. These components connect to the mobility manager to decide in close agreement with the application logic to, for example, move a session from one active network to another.

The most obvious candidate for an IP layer mobility management solution is Mobile IP. We therefore choose Mobile IP to discuss how our architecture deals with IP layer mobility management functionality. A candidate for transport layer mobility management is less obvious: the protocols and mechanism at this layer are less well known and not (yet) part of solid standards and implementations. We choose Mobile SCTP, mainly because its embedding in the regular SCTP standard is on its way. The usage of the two protocols as part of our mobility management mechanism is depicted in Figure 1 and Figure 2.

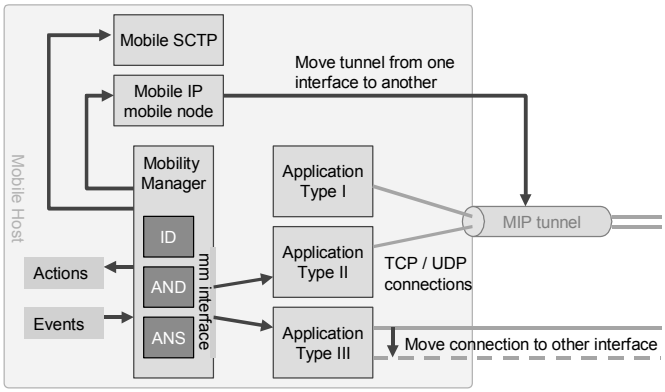

Figure 1: Host mobility management with Mobile IP usage example; applications of type I and II rely on Mobile IP and applications of type III manage their own connections.

When looking at Mobile IP usage (Figure 1) we see the following interaction. Suppose that at a certain moment, the mobile host is connected to two wireless networks A and B and the user is walking out of range of A. Suppose that Mobile IP runs in tunneled mode with a co-located care-of address (CoA), all traffic from and to the CoA is received over the bi-directional tunnel and the tunnel is realized over network A. The mobility manager then indicates towards the Mobile IP mobile node functionality that the tunnel must be moved from network A to network B. Typically, a type I application, which does not receive information about the current state of the mobility process from the mobility manager, implicitly uses the CoA when it initiates connections to other nodes on the Internet, because the default route is over the Mobile IP tunnel. All its connections are setup through the tunnel and remain established when the tunnel moves from one interface to another. The connections for a type II application are similar to the ones for a type I application; they remain established without intervention by the application. The type II application, however, is aware of the fact that the tunnel is moved from one interface to another and is additionally aware of the types and characteristics of both network A and B. It therefore can decide to change, for example, the amount of traffic it generates over the established connections. The type III application receives information from the mobility manager that the connection with network A is lost. It decides to stop the connections it had over network A and re-establishes them over network B.

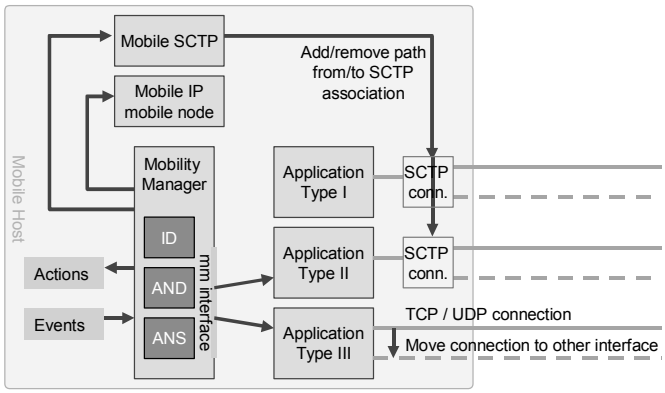

Figure 2: Host mobility management with Mobile SCTP usage example; applications of type I and II rely on Mobile SCTP and applications of type III manage their own connections.

The interaction is mostly the same when Mobile SCTP is used (see Figure 2). Let's assume the same network situation as above. Now, the applications of type I and II use an SCTP connection to reliably transfer data from and to another node on the Internet. First, the SCTP connection uses a single path over network A to the other node. When the connection with network A is lost, the mobility manager indicates towards the Mobile SCTP functionality that the SCTP connections must use a new path, over network B, for communication with the other nodes. The situation for a type III application is equal to above. Naturally, a type III application may use an SCTP connection as well, but then obtains control over this connection to add or remove paths to the SCTP association by itself.

In our opinion, SCTP can be a good alternative to mobility management functionality at other layers. It provides a good replacement for Mobile IP when end-to-end characteristics are important; it can provide system-level mobility management when an application does not want to maintain its own connections and Mobile IP cannot be applied. To our knowledge, however, implementations of Mobile SCTP are few and still in their early stages. Additionally, we feel that Mobile IP is more widely recognized as an important mobility management protocol than Mobile SCTP. We therefore do not further consider Mobile SCTP in the rest of the description of our architecture and implementation, but focus on Mobile IP when it comes to the interaction of our mechanism with existing mobility management protocols.

The mobility manager consists of a number of components (as depicted in Figure 1 and Figure 2). The communication between these components is message based. Every component defines a set of messages: outgoing messages that provide information towards the other components and incoming messages that are interpreted by the component. The following components are

defined: *Interface Detection* (ID), *Access Network Detection* (AND), and *Access Network Selection* (ANS). The configuration of the mobility manager and its components is static. Currently, no provisions are made to dynamically change configuration parameters at runtime.

The applications that need to stay up to date with the state of the mobility process and the characteristics of the network resources connect to the mobility manager using the mobility manager interface. The messages generated by ID, AND, and ANS are passed on to the clients in unaltered form: our approach is that the abstractions made by the components is sufficient to suit both the other components and the applications. The application can than filter out the messages that are of particular interest. Obviously, the application may link with a library, for instance to provide SIP re-invite based mobility management, that connects to the mobility manager and interprets the messages.

In a multi-homed situation, a subtle relation exists between the usage of the IP layer mobility management functionality (Mobile IP), the selection of the outgoing interface, the selection of the source address, and the default route. Applications typically use the Sockets Application Programming Interface (API) to setup connections with other Internet entities. By binding to a local address while initiating a connection, the application can choose the IP-level mobility management: the connection will be managed by Mobile IP when binding to the Mobile IP address; the connection must be managed by the application when binding to another address than the Mobile IP address (not considering transport layer mobility management here). Type I applications do not explicitly bind to a local address when initiating outgoing connections, but leave this to the operating system. Normally, when connecting to a node, which is not directly accessible through one of the local networks, the source address becomes the address configured for the interface that is set as the default route. By setting the default route to the Mobile IP tunnel (when running in tunneled mode), outgoing TCP and UDP connections will, by default, be managed by Mobile IP. When an application explicitly binds to a local address for a TCP or UDP connection initiated to a global node, we assume that the outgoing network interface for this connection is the interface that is configured with this local address.

The remainder of this section describes the components of the mobility manager and the messages that they can send and receive. Furthermore, it provides a description of the interaction between the mobility manager and its clients, the applications.

3.1 Interface detection

The mobility manager must be aware of the network interfaces that are currently available in the operating system. The Interface Detection (ID) component supplies information about the creation and deletion of network interfaces in the OS. It indicates the type of the network interface as an abstraction of the operating system interface type: we currently define the types `eth` (Ethernet), `wlan` (Wireless LAN), and `dialup` (PPP dialup). It also indicates the expected maximum bandwidth for a network interface. Note that this is not the same as the actual available bandwidth. Additionally, ID indicates whether a network interface is up or down. Finally, it provides information about the current IP address that is set for a network interface.

We define different types of interfaces, because a type can be associated with one or more inherent characteristics. Obviously, this is important to the other mobility manager components, but it is also relevant to the clients. An application may, for example, choose to use a network through one type of interface because it knows the costs of usage are low compared to the networks available through interfaces of other types. In that case, the application associates a characteristic – cost of usage – with an interface type, as a supplement to the characteristics defined by ID.

Table 2: messages defined for Interface Detection

ID message	description
`NEW_EVT`	A new interface is available in the OS
`CHG_EVT`	An existing interface has changed: up or down, IP address change
`DEL_EVT`	An interface is removed from the OS
`LIST_REQ`	Request a list of current interface
`LIST_EVT`	A list of current interfaces (including characteristics)

3.2 Access network detection

It is important that the mobility manager is aware of the networks that can currently be made available through the network interfaces existing in the operating system. For some interface types, there is not much to choose: an Ethernet adapter is either connected to the Ethernet or not. For wireless adapters this is often different: multiple networks can be available through the same adapter. A Wireless LAN interface can be instructed to scan for available networks, although it can only be connected (associated) with a single network at a given time. The scanning results normally contain information about the quality of the available networks.

Table 3: messages defined for Access Network Detection

AND message	description
`NEW_EVT`	A new network is available
`CHG_EVT`	An existing network has changed: link quality change
`DEL_EVT`	A network is no longer available
`SCAN_START_REQ`	Start scanning on a specific interface (in addition to automatic scans)
`SCAN_READY_EVT`	Scan is ready for a specific interface
`LIST_REQ`	Request a list of available networks
`LIST_EVT`	A list of currently available networks (including characteristics)

The Access Network Detection (AND) component is responsible for the scanning of available networks for `wlan` interfaces at regular intervals. It provides a translation from Wireless LAN specific indications of network quality, such as signal and noise values, to a qualitative indication of network quality. We define four levels of quality: poor, medium, good, and excellent. AND

uses the information from ID about the available network interfaces. Whenever the quality of a network chances, it notifies the other components by sending a network change event.

3.3 Access network selection

Most of the control functionality of the mobility manager resides at the Access Network Selection (ANS) component. ANS deals with a number of different issues. For each interface it activates the preferred network at the link layer. Additionally, it makes sure that IP parameters such as the IP address are obtained for each interface with an active network, and it takes care of setting these parameters. Furthermore, it is responsible for controlling the system level mobility management functionality such as Mobile IP. ANS moves the Mobile IP tunnel from one physical interface to another if appropriate given the current network situation. Finally, it controls which interface is used as the default route.

The activation of link layer connections, the actual "access network selection", is executed for those network interfaces that support connecting to different networks. The interface type `wlan` is an example. For each such an interface, a list of networks exists in the ANS configuration that indicates the preferred activation order. The first entry in the list has the highest preference and will be activated when available. If AND indicates that this network is not available, or if the quality of that network is below a certain (configured) threshold value, the second entry will be considered, etc. An entry in the preferred network list may be a wildcard. On reaching a wildcard entry, ANS selects and activates the network with the best quality. So, for unknown networks the default behavior is to select the one with the highest quality.

In many cases, ANS must supply credentials for authentication and authorization to setup a link layer connection. For `wlan`, this may involve setting the Wired Equivalent Privacy (WEP) key. Alternatively, ANS may use a method like IEEE 802.1X with various kinds of authentication over the Extensible Authentication Protocol (EAP). The ANS configuration stores the credentials for the networks: a set of default credentials may be specified for networks that can be accessed using 802.1X and networks that are not known at startup.

After network activation at the link layer, it is necessary to obtain the IP settings for the interface. The IP parameters such as IP address, network prefix and gateway may be specified in the ANS configuration. In most cases, however, the IP settings are obtained dynamically using a DHCP client. ANS takes care of controlling the DHCP client for interfaces that are actively connected at the link layer. It waits for the DHCP client to supply and set the IP parameters for newly activated networks. The DHCP client also must take care of setting the network route for the new network. ANS, however, sets the default route (and gateway).

Every time a network is activated or deactivated for one of the interfaces, ANS checks if it is necessary to change the settings of the system level mobility management components (here, only Mobile IP). We assume that Mobile IP runs in tunneled mode with a co-located CoA. We also assume that the available Mobile IP mobile node functionality is "passive": it does not actively scan for available network on Wireless LAN interfaces and does not change the path of the tunnel by itself.

A list in the ANS configuration provides the order in which interface/network combinations are preferred to supply the Mobile IP CoA. The combination of interface and network is important, because multiple interfaces may connect to the same network. In certain situations it may be desirable to prefer one interface over the other, for instance, because the interface has better power consumption characteristics. Again, the first combination in the list has the highest preference and will be chosen as the path for the tunnel. If the first combination does not exist, ANS looks at the second, etc. The list of preferred Mobile IP CoA suppliers does not influence the activation of a network for a network interface: this is solely determined by the list of preferred networks for that interface. Additionally, an interface/network combination can only be chosen when IP layer connectivity is realized over that network (otherwise, there would be no point in trying to establish the tunnel over this interface/network). After the selection of the preferred combination, the Mobile IP component is signaled to move the tunnel to this new path.

Table 4: messages defined for Access Network Selection

ANS message	description
`AN_SELECT`	A network is selected for a certain interface
`AN_DESELECT`	A network is deselected for a certain interface
`DEFAULT_IF_SELECT`	An interface is selected as default interface (default route)
`MIP_COA_SELECT`	An interface is selected to supply the Mobile IP CoA
`MIP_COA_DESELECT`	An interface is deselected to supply the Mobile IP CoA
`STATUS_REQ`	Request the current ANS status
`STATUS_EVT`	Supply the current ANS status (network selection per interface, selected default route, selected CoA supplier)

The ANS component manipulates the routing table of the mobile host. Obviously, the default route entry must exist in the routing table and it changes over time as the mobile host connects to different networks. For this reason, the ANS configuration holds a list that describes the order in which interface/network combinations are preferred to provide the default route (similar to the preference for CoA supplier). On a change in the configuration of the active networks, ANS checks whether the default route must be changed using this preference list.

The type III applications may choose to setup connections to other Internet nodes over active networks that are not used as the default route. This can be done by binding the connection to the IP number associated with the (non-default) interface, provided that a gateway is configured to route the traffic off this local network. Normally, the default route provides the gateway. The routing mechanism on the mobile host must be instructed to use another route when the source address is not the address of the regular default interface. It is important to realize, however, that routing

on the mobile host typically is based on the destination address only.

In contrast to the ID and AND components, ANS must be extended when a new below-application layer mobility management protocol is added. This is inevitable, as ANS is the primary controlling entity in the mobility manager. For Mobile IP, it has knowledge about Mobile IP specific actions and concepts such as binding updates to the home agent, the CoA, and the virtual interface for the tunnel. It must control the functionality for new mobility management protocols such as Mobile SCTP in a similar way. It is highly likely that this new protocol will introduce new AND messages to describe the mobility process state to the applications.

3.4 Client interfacing

An application connects to the mobility manager interface to stay informed about the current state of the mobility process and the network resources. Strictly speaking, it could obtain this information directly from the operating system. The components of the mobility manager, however, provide an abstraction of below-application layer state of the network resources and their configurations. The AND component provides, amongst others, an abstraction of the current state of the mobility management protocols at the IP and transport layer. Furthermore, the mobility manager supplies a single mechanism to receive all this information: there is no need for the application to use one system API for getting the available network interfaces, another for obtaining information about the available wireless networks, and a third for retrieving the routing configuration.

The abstraction hides many details of the layers below the application layer. Without this abstraction, the application must deal with many parameters that are specific for a certain layer or technology. For example, to obtain a quality measure for a network activated through an 802.11b interface, the application would have to be capable of interpreting 802.11b scan results and translate signal and noise parameters to an indication of network quality. To make things worse, this information is often driver and hardware specific.

The level of abstraction should fit the usage. Here, we want the application to:

- Have information about the local network path followed by the IP packets for connections with other nodes; it must know through which access network the connection traffic is routed.
- Know the available below-application layer mobility management protocols that it can use for new connections.
- Know the system default behavior for mobility management, e.g. when regular connections are initiated, will they be managed by Mobile IP or maybe not at all.
- Have knowledge of the current descriptive parameters of the interfaces (type, IP settings) as well as the active network accessible through each interface.
- Obtain a notion of the quality of the available access networks, when applicable, e.g. Wireless LAN link quality.

We believe that the messages generated by the mobility manager components provide a level of abstraction that is sufficient for the applications to obtain the above information. The interaction between the mobility manager and a typical 'aware' application is illustrated by two examples given below.

3.4.1 Interaction Example 1

In example 1, the application (of type II) has a Mobile IP based UDP connection with another Internet node. The mobile host running the application has access to a fixed Ethernet and an 802.11b WLAN. When the user moves around, the mobile host looses its access to the fixed network and the application wants to adapt the bandwidth consumption based on the expected network resources available in the new situation. The following actions can be discerned:

a) The application starts and connects to the mobility manager.

b) The application sends an ID `LIST_REQ` to obtain a list of currently available interfaces and their types, maximum available bandwidths, up or down states, and IP parameters. The mobility manager replies with an ID `LIST_EVT` message including the interface information.

c) The application sends an ANS `STATUS_REQ` to request the current ANS status. The mobility manager replies with an ANS `STATUS_EVT` indicating that the `eth` interface supplies the Mobile IP CoA and that the default route is set to the Mobile IP tunnel: now the application knows that is has Mobile IP facilities at its disposal and that normal connections (initiated without binding to a specific local IP address) will be managed by Mobile IP. Additionally, it knows that the tunnel traffic will go over the `eth` physical interface.

d) The application initiates the UDP connection to the other node using the regular Sockets API. This connection is, by default, managed by Mobile IP. The application starts using the connection with a bandwidth consumption that can be supported by an `eth` interface with the indicated maximum bandwidth.

e) After a while, the mobile host disconnects from the fixed network. The mobility manager detects that the link is down and sends an AND `DEL_EVT` for the `eth` interface. The application does not react to this information. ANS, however, uses this trigger to move the Mobile IP tunnel from the `eth` to the `wlan` interface: it first sends out an ANS `MIP_COA_DESELECT` message for `eth` to indicate to the applications that the tunnel is not over the fixed network anymore, immediately followed by an ANS `MIP_COA_SELECT` message for `wlan` to indicate that the tunnel is up again over the wireless network.

f) Now the application knows that the packets for its connection go over the `wlan` interface; it adapts the bandwidth consumption for the connection to a level that is appropriate for the wireless network.

Note that the application has no guarantees about the bandwidth available on a certain network. The network resources may have to be shared with other users and applications. Each network *type* can

be related to a rough indication of its empirically determined available bandwidth. Naturally, the difference between types may be multiple orders of magnitude: the average available bandwidth for a fixed FastEthernet network can be 20 Mbits/s while for a GPRS network this may be 15 Kbits/s. We assume that the applications are most interested in changes in this order of magnitude (for any given parameter, not just bandwidth). It is up to the application to assume certain average values for parameters that are associated with network types; the mobility manager only provides an indication of the network type itself.

3.4.2 Interaction Example 2

In example 2, the application (of type III) has a UDP connection to another Internet node and re-initiates this connection when necessary given the changes in network resources. In this case, the mobile host has access to a wide-area cellular network with a low bandwidth and substantial usage costs. The application would rather use more bandwidth, but only wants to switch to another network if it has a good quality, for instance a network of type `eth` or of type `wlan` with a good quality indication. By switching only when another high quality network is available, the application reduces the number of connection re-initiations. The following interaction takes place:

a) As interaction example 1.

b) As interaction example 1.

c) The application sends an ANS `STATUS_REQ` to obtain the current ANS status. The mobility manager replies with an ANS `STATUS_EVT` indicating that the cellular network is available through the `dialup` interface; the `eth` and `wlan` interface have no active networks. Additionally, it may indicate the Mobile IP and default route status, but this is of no interest to the application.

d) The application sets up the UDP connection to the other node using the Sockets API. Before connecting, it binds its local socket to the IP address of the `dialup` interface. The traffic for this connection will therefore run over the `dialup` interface, regardless of the changes in the Mobile IP state.

e) The application receives AND `NEW_EVT`, `DEL_EVT`, and `CHG_EVT` messages indicating that 802.11b networks have become in or out of reach, or that their qualities have changed. The application keeps track of the quality of these networks. ANS uses this information to activate one of these networks through the `wlan` interface, depending on preference (as configured) and quality.

f) The mobility manager sends an ANS `AN_SELECT` to notify that one 802.11b network has been selected and activated through the `wlan` interface. The application checks the current quality of the accessible 802.11b network and learns that it is medium. The application decides that this quality is not high enough to justify a re-initialization of the connection.

g) The mobility manager sends an AND `CHG_EVT` indicating that the quality of the currently available 802.11b network has gone up from medium to good. Now the application decides that the quality is good enough. It tears down the connection over the `dialup` interface, and initiates a new connection over the `wlan` interface (by binding its local socket to the IP number of the `wlan` interface).

h) The application adapts the bandwidth consumption to a level that is appropriate for the newly chosen network.

4. IMPLEMENTATION

This section describes our prototype implementation. The mobility manager as described above is implemented in C as a user space daemon for the Linux operating system. Our primary target device is an HP iPAQ running Familiar Linux v7.x, although the implementation also works on other, perhaps less mobile, environments such as Linux Redhat 9 on a laptop. The prototype supports Ethernet, 802.11b, and PPP/dialup physical link types to connect the mobile host to different kinds of networks. All communication is IPv4 based. The Mobility Manager implementation consists of the components identified in the previous section with one addition: a component that implements the Mobile IP mobile node functionality. The parameters that are used to configure the mobility manager are stored in a configuration file. This file is read during startup of the mobility manager.

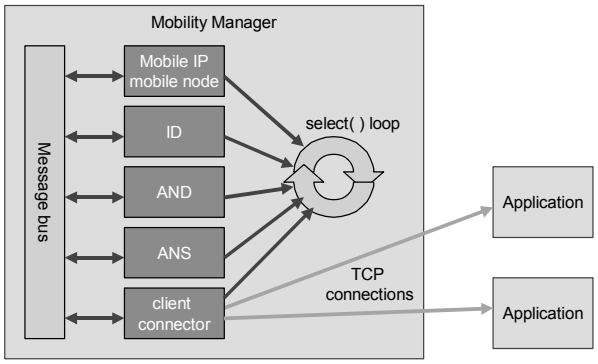

Figure 3: the internals of the Mobility Manager: the components use the message bus for communication between each other and the `select()` loop for the subscription to asynchronous events

As depicted in Figure 3, the components communicate with each other using a message bus. A message sent to the message bus is received by all connecting components. All messages consist of a header part indicating type and size and a data part which is a plain C `struct`. The Mobility Manager runs single threaded: the main loop is implemented using the `select()` Unix system call. All components have the opportunity to add file descriptors to this `select()` loop. This allows them to be notified of asynchronous events.

4.1 Interface detection

The detection of interface changes is realized using a Linux `rtnetlink` socket. The kernel sends interface creation, deletion, and change events through this socket to the ID component. Additionally, this socket is used to receive changes in IP address settings for an interface. The ID component translates these

system-specific events into ID messages. ID inserts the socket into the `select()` loop and will be notified whenever the kernel sends an interface event.

4.2 Mobile IP mobile node functionality

This component implements the Mobile IP mobile node functionality for the mobile host. It is essentially a stripped and modified Dynamics Mobile IP mobile node daemon [5]. All active parts – the scanning of Wireless LANs, the setting of the default route – are removed. It is tested to run in tunnelled mode with a co-located CoA, although it may run in other modes (e.g. Foreign Agent mode) as well. This component is not configured through the Mobility Manager, but re-uses the configuration file that comes with the original Dynamics mobile node daemon. It defines a single message, `UPDATE_REQ`, which can be used to move the tunnel from one interface to another.

4.3 Access network detection

The detection of 802.11b networks is accomplished using the Linux Wireless Extensions. Not all drivers support scanning, however. We have successfully used `hostap_cs` (v0.1.0) for cards using the Intersil PrismII chipset, a patched `orinoco_cs` (v0.11b, morinoco patch) and a patched `wvlan_cs` (v1.07, proprietary patch) for Orinoco and Avaya cards.

For most driver and card combinations it is necessary to set the 802.11b SSID to 'any' to obtain scanning results for all reachable access points. This means that during scanning, the association with the current network is lost and must be re-established when the scan has finished. Our experience is that this does not influence the state of the layers above the link layer; TCP connection, for instance, do not break when the association is temporarily interrupted at the link layer. The `hostap_cs` driver supports a continuing association during scanning for certain card firmware versions.

During scanning it is not possible to communicate (even when remaining associated). Therefore the intermediate time between two consecutive scans should be long. However, to have an up-to-date picture of the qualities of the currently available networks, it is important to scan often. These two conflicting interests must be balanced for the actual scan interval. Our experience is that the time to execute a full scan can be quite diverse given different cards and drivers. A scan for the Avaya/`orinoco_cs` combination can take around 400 ms, while a Linksys/`hostap_cs` scan sometimes takes up to 1200 ms to complete. Our default setting is a scan interval of 10 seconds.

We have noticed that the outcome of successive scans can differ significantly. Sometimes an access point 'disappears' even when it is obvious that it is well within reach. With the next scan it is often back in the scan results. AND keeps track of the scan history in a sliding window fashion. The quality is calculated using a weighted average, where recent results have a higher weight. Our default window size is 5. The signal and noise parameters are transformed to the qualitative indicators of network quality using a set of threshold values. These threshold values are, unfortunately, specific for a certain hardware and driver combination.

4.4 Access network selection

The implementation of the ANS component also uses the Wireless Extensions. When a `wlan` network is selected, the association with the access point is created with this functionality.

After link layer activation, the IP parameters must be obtained and set. When not specified in the configuration file, the IP parameters are obtained using an external (outside of the Mobility Manager process) DHCP client. We currently use dhcpcd, version 1.3.22pl4.

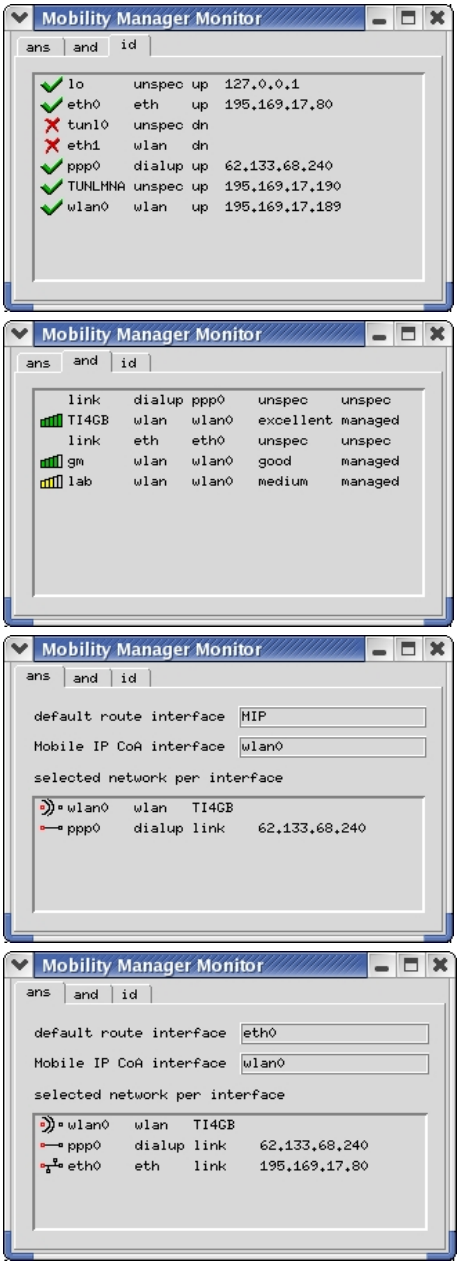

Figure 4: the Mobility Manager Monitor showing the state of different components of the Mobility Manager

To manipulate the default route, ANS uses a Linux `rtnetlink` socket. To move the Mobile IP tunnel, ANS uses the Mobile IP component.

4.5 Client communication

Applications (clients of the Mobility Manager) set up a TCP connection to localhost on port 9898. The client connector component is responsible for handling the client connections. All messages that appear on the message bus are forwarded to the clients. An example of a client is the Mobility Manager Monitor (see Figure 4).

All applications running on the mobile host can connect to the mobility manager in this way, irrespective of the user accounts under which the applications run. This may result in unprivileged users obtaining system information that is usually not available to them, such as (an abstraction of) 802.11 scanning results. Many mobile hosts, however, effectively run as single user systems. Choosing a slightly different type of communication could probably solve unprivileged access, for instance through Unix domain sockets that are associated with file system permissions. For reasons of simplicity, we have not further considered security issues for the communication between the mobility manager and its clients.

4.6 Routing

By default, the Linux kernel selects the outgoing interface for an IP packet based on the destination address only (i.e. for the kernel version 2.4 that we used). We assumed, however, that when an application wants to initiate a connection to a host elsewhere on the Internet and it wants to force the traffic for this connection over a specific network interface, it binds the local socket for that connection explicitly to the IP address associated with the preferred interface. In this case, the kernel always forwards the outgoing IP packets to the gateway directly reachable through the default interface, even when the socket is bound to an IP number associated with a non-default interface. The packets leave the host through the default network interface with a source address that belongs to another interface. While this may work in terms of end-to-end communication, it results in a return path that is different from the outbound path. More importantly, the communication partly goes through a non-preferred local network.

Fortunately, Linux supports more advanced ways of route selection in the form of *policy based routing*. This functionality (available from kernel version 2.2 and up) uses multiple routing tables and a routing policy database (RPDB) to select routes based on a number of criteria such as the source address. It supports the configuration of multiple routing tables. To route an outbound IP packet, the kernel selects a table based on a number of attributes associated with the packet.

The mobility manager can configure the policy based routing facilities in such a way that the outgoing IP packets leave the mobile host through the interface that is associated with the source address. For every network with a gateway to other networks, typically to the Internet, it configures a dedicated routing table that is essentially a copy of the main routing table except for the default route. The dedicated routing table has a default route that points to the gateway reachable through this specific network and interface. A rule in the RPDB prescribes that the dedicated routing table must be used when the source address matches the address associated with this interface. For applications that do not explicitly set the source address, the default route in the main routing table applies. The mobility manager changes the default route in the main table when it decides that the regular default route must point to another network. A network without a gateway does not get a dedicated routing table, because it is used only for local communication.

5. EXPERIMENTS

This section describes the experiments we have done for an initial validation of the mobility manager concepts and its implementation. We want to verify that our mechanism works for different kinds of applications. Given the classification we provided in section 2, this means testing the mobility manager implementation against all three types of applications. Applications of type I, however, use so little mobility manager functionality that we only consider type II and type III applications in this section. Type I applications are a trivial case: running them successfully would merely prove that the Mobile IP functionality works.

We have developed a streaming video application that interfaces with our mobility manager implementation in order to react to changes in the state of below-application level mobility protocols (Mobile IP) and to changes in the network resources. The application consists of a video streaming server and client that adapt the video stream running between them, when necessary. It can run as a type II and as a type III application. The specific behavior of the application during handovers is configurable thereby enabling us to validate different handover scenarios.

5.1 Video Streaming Server

The video streaming server is a Linux host running a VIdeo Conference tool (VIC) [22], a rather well-known MBone [7] videoconferencing application. The VIC tool is configured in transmit-only mode, thus used only for sending out a video stream, not receiving or listening for peer streams. The host is equipped with a TV-card that has its input connected to the local cable television broadcast network. The VIC server application captures the TV signal through the TV card, encodes the signal in the h.261 format and sends it out over the network as a RTP unicast UDP stream to a specific target IP address. The original VIC server code has been extended with (scripted) functionality that opens a control socket on which it listens for messages that instruct the VIC server to adapt its configuration. These messages conform to a simple proprietary protocol whose semantics are comparable to the usage of a SIP re-invite transaction, used for adaptation of an existing session. The possible forms of adaptation are:

- increase or decrease the network bandwidth used for the RTP stream
- change the number of frames per second that is encoded
- modify the encoded picture size (small or large)
- modify the encoded picture quality,
- alter the target IP address for the unicast RTP stream

- change the TV channel (this feature is actually unrelated to mobility management),

This functionality allows a peer client to dynamically reconfigure the server and adapt the running streaming session characteristics. Although the server is only able to handle one client simultaneously (unicast), this is sufficient for our purposes.

5.2 Video Streaming Client

The video streaming client running on the mobile terminal is also a VIdeo Conferencing Tool (VIC) that is configured in receive-only mode, thus not sending its own video. We have used the VIC client during the validation tests on both notebook and Hewlett Packard iPAQ (type 3870) hardware running the Linux operating system. The VIC client receives the RTP packets, decodes the h.261 frames and displays the video stream in a local window. The original VIC code has been extended with (scripting-only) code that communicates with both the mobility manager and the peer streaming server. VIC opens a client socket to the mobility manager on which it receives messages that it uses to determine its mobility strategy. Furthermore it is capable of sending control messages (in the previously described proprietary format) to the VIC server in order to instruct its peer to adapt the characteristics of the multimedia session; this is typically done upon interpreting the messages from the mobility manager about any changes in the network environment and the state of Mobile IP.

5.3 Networks and Handovers

During the validation tests we connected our mobile terminals (both the notebook and the iPAQ) to the following networks simultaneously:

- An 802.11b Wireless LAN, by using a PCMCIA WLAN card
- A GPRS wide-area cellular network, by connecting via a GPRS enabled mobile phone using Bluetooth[1].
- A fixed Ethernet (over a host-to-host USB cable for the iPAQ)

We have used the GPRS network connection as an "always on" low priority connection. The connection to this network was setup and configured statically and assumed to be permanently available. The mobility manager dynamically configured the WLAN and Ethernet interfaces on the mobile host. Handovers were triggered by moving the mobile terminal in and out of range of a WLAN access point or, alternatively, manually reduce the transmit power of the access point[2]. Additionally, handovers were initiated by plugging in an Ethernet cable (notebook only) or by putting the iPAQ in its cradle, which enabled it to use a host-to-host Ethernet connection over its USB interface. Furthermore, we were able to force a handover by manually bringing WLAN and Ethernet interfaces up or down (for use in test scripts).

[1] We have also executed preliminary tests over a UMTS network by connecting to a test network via a UMTS phone using Bluetooth. Unfortunately we were not able to use a globally routable IP addresses for our mobile host so we could not use the RTP streaming application.

[2] This was done using a laptop with an Intersil PrismII WLAN chipset in so-called HostAP mode, thus serving as a wireless access point. This setup allows access to controls that are inaccessible in most commercial hardware access points.

5.4 Application Type II Streaming

When running as a type II application, the VIC client interprets mobility manager Access Network Selection (ANS) messages about changes in the Mobile IP CoA (`MIP_COA_SELECT`). Such a message indicates that, based on the strategies configured in the mobility manager, Mobile IP traffic runs over a new network, and possibly over a new interface. Notice that the Mobile IP CoA has changed, but since VIC acts as a type II application, it has bound its sockets to the Mobile IP Home Address and as such the stream sent by the VIC server will still reach the Mobile Host through Mobile IP tunnel over the CoA interface. However, since we are on a new network, the layer 2 type of the network may have changed; for example we may have switched from a WLAN network to a GPRS network. If this is the case, as indicated by the ANS message, the characteristics of the new network may dramatically differ from those of the old network. In order to accommodate for such a change, the VIC client maps the layer 2 type of the network to a predetermined maximum bandwidth that the RTP stream is allowed to use and it notifies the peer VIC server of this bandwidth by sending a (proprietary) adaptation message. In case of a handover to a GPRS network this would be about 30kb/s.

5.5 Application Type III Streaming

When running as a type III application, the VIC client interprets mobility manager Access Network Selection (ANS) messages indicating changes in the default route interface (`DEFAULT_IF_SELECT`). Such a message indicates that the MM has chosen a new network and/or interface as the default interface for outgoing traffic. This means that our (default) IP address has changed and perhaps the layer 2 type of the network as well. Since our type III application does not use Mobile IP, the old IP address may become unreachable and the client decision is to change the target endpoint of the current RTP connection to the new IP address. (Alternatively the application could decide to defer this handover until mobility manager notifies that the old network is actually down). The VIC client rebinds its receiving socket to the new IP address and notifies its peer server of the address change, and possibly the new bandwidth parameters as determined by the layer 2 network type.

6. EVALUATION AND DISCUSSION

In this section we summarize our experiences with and thoughts about the concepts, implementation and the initial validation of our mobility management mechanism. Since we have not done detailed measurements and analysis, we cannot present quantitative results. Instead we describe our experiences from a qualitative perspective.

6.1 Multiple mobility management protocols

The experiments with the video streaming application show that our mechanism works for a situation where Mobile IP is used as well as for a situation where a proprietary application-level mobility protocol is used. We strongly feel that the availability of multiple means of mobility management provide the kind of flexibility needed in a heterogeneous network environment.

To experiment with a high level of flexibility, however, it would be good to have more mobility management protocols available. Mobile SCTP is a prominent candidate because it supplies mobility management at the transport level, which is something we have not been able to test in our current implementation. Furthermore, adding Mobile SCTP would allow us to investigate the modularity of our architecture and implementation.

6.2 Interface Abstraction

From the architecture description in section 3, it is clear that the interface provided by the mobility manager towards the applications is hiding many aspects and parameters from layers below the application layer. At the same time, it is clear that applications that need to be mobility aware have to deal with quite a number of elements that have *not* been abstracted away. They need to have, for example, an understanding of the capabilities of the different interface types: networks accessible through a `wlan` interface can have a certain link quality, but this is not the case for networks reachable through an `eth` interface. Also, the entities defined as data elements in the messages, *interface, network, bandwidth, quality, etc*. have a certain relationship towards each other. The entities and their relationships define a model that has to be understood, at least partly, by the application.

The experiments in the previous section show that for a particular configuration with a particular application, the level of abstraction and the associating model is sufficient to support this application running in both a type II and a type III mode. This is an initial form of validation, but for a more thorough validation many more applications and configurations must be used to determine that the abstraction and semantics are right.

For instance, we probably could extend the set of Access Network Selection (ANS) messages with a message that notifies the applications upfront that a new network will be activated instead of the current active network for a particular network interface. Applications then have the opportunity to take some preparing actions, for instance signaling their peers, while their current connections are still up and running. This appears like a logical addition, but we're not sure it will be used by many applications.

Looking at the actual mechanism for passing messages from the mobility manager to the application, C `structs`, we realize it is necessary to have a more generic approach, because our current implementation is too much C programming language specific. We would suggest using a human readable simple text-based scheme.

6.3 Implementation: Handover Delay

The experienced handover delay is only very marginally influenced by the mobility manager architecture or the chosen mobility protocol. It largely depends on the following factors:

a) Network interface scan specifics

The time needed for scanning new networks (here 802.11b WLAN) depends on the (radio) hardware, the driver and the firmware of the network interface card. Our measurements with different WLAN PCMCIA cards vary between 0.4 and 3 seconds needed for one scan. This parameter has a major effect on the reaction time of the mobility manager, which can only react after gathering and processing network scanning results.

b) Network latency

Communication with remote entities is required in several layers of the mobile terminal protocol stack. The application layer needs to transport session adaptation parameters to the peer entity, the network layer needs to re-register for location management purposes (Mobile IP registration), and possibly even layer 2 authentication may be needed (e.g. 802.1x). These steps need to finish before the handover can finish. The time needed for this is largely dependent on the round-trip delay of the newly chosen network. For example, our measurements on the GPRS network show a round-trip delay of nearly 1 second, which adds a 2 second delay to a type 2 application handover scenario (Mobile IP registration, session adaptation, no dynamic authentication).

This calls for integration of network authentication, location management and session adaptation. As a possible alternative, we could use SIP for that purpose.

c) Authentication

The authentication to the new network itself may cause a delay in the time that is needed to configure the network interface. An example is the usage of 802.1x with EAP-(T)TLS for wireless LAN connectivity. The processing overhead involved for setting up the TLS traffic and processing certificates may be significant for small client devices (see paragraph e)). Moreover the server side processing can add to this delay, possibly by contacting other servers (Radius). Our experiences show that the usage of 802.1x as opposed to using WEP keys, introduces a noticeable delay in handover times in the order of 1 second (this includes both processing and communication overhead).

d) IP address configuration

In our tests we have used the DHCP protocol for configuring IP addresses on the network interfaces. Our measurements show that DHCP may need between < 50 ms and 5 seconds to obtain and configure an IP address on an interface. Of course this depends on the configuration and implementation of the DHCP client and server, but we suspect that network equipment such as routers and switches play an important role here as well. This experience shows that IP address configuration is an important factor nevertheless. A possible improvement would be to use statically assigned, pre-configured IP addresses when roaming to well-known networks.

e) Client device processing power

The limited processing power available on mobile devices is important for all of the tasks that need to be performed during a handover. This includes executing operating system tasks, network related tasks, mobility management processing and application processing. Although this seems like an obvious remark, its effect is not to be underestimated. We have seen a more than remarkable reduction of "user experienced" handover times when comparing notebook performance against iPAQ performance. The average user experience on the iPAQ is "a matter of seconds", on the laptop it could be called "truly seamless".

f) Application characteristics

The application that is actually being used during the handover, can in itself contribute to the seamlessness of the handover. This is typically done by buffering: when the application buffer is filled

before entering the handover phase, and the handover is finished before this buffer is empty, a user will not notice any interruption at the application level. We have noticed the benefits of this effect when using VIC video.

Note that the negative effects of the first five factors mentioned above can be avoided if the new network can be configured before the session is transferred from the old network. However this is not possible when the network connectivity is lost unexpectedly or when performing a handover to a new network connected to the same interface as the old network (e.g. WLAN-to-WLAN handovers using a single network interface card).

7. RELATED WORK

This section provides an outline of the work that relates to the issues presented in this paper. It gives an overview of related work focusing on mechanisms that allow mobile applications to be network context aware (mobility and network aware applications). Additionally, we look at an approach for letting applications influence the mobility process (application aware mobility management).

The Odyssey platform for adaptive mobile data access described in [14], with experiences presented in [15], models the adjustment of applications to general changes in resources around the high-level concepts of agility and fidelity. It supports a division of adaptation functionality between operating system and application. We limit our scope to the awareness of the mobility process and the network resources, where the applications may choose at which layer their connections are managed for mobility. We do not support, however, a situation where the application and the system jointly decide about for example the selection of the Mobile IP CoA (which may have a large impact on available bandwidth for existing Mobile IP connections). We feel that such a joint responsibility requires an interaction with a high complexity (and overhead) between applications and system, also in the light of conflicts that might occur. Additionally, we think that the abstraction and the model of the information that makes applications aware are by itself more important than the mechanism for the transfer of this information. Such an abstraction and model are mostly domain specific. Odyssey is complementary because it deals with the actual application adaptation, while our approach focuses more on supplying the information for this adaptation.

The Mobiware Toolkit presented in [2], provides an adaptive mobile networking environment. It defines a service model where part of the functionality is at the mobile device and parts are in the network (routers/switches). It has a strong focus on quality of service (QoS) aspects and supplies handover functionality. In contrast, our approach assumes the availability of existing mobility management mechanisms and standards (that support handovers). Mobiware defines a utility function that indicates how well an application can stand changes in bandwidth, and an application adaptation policy. Our mechanism "simply" provides the appropriate information about the mobility process and network resources to the applications, such that they can apply their mobility strategy in a model of their choosing. As an enhancement to Mobiware, the programmable handoff architecture in [10] supports multiple styles of handoff control, and suggests, as we do, to use a system-level network detection mechanism for multiple mobility management protocols. The architecture has an elegant modular structure for handoff adapters, but does not focus on providing information towards the applications about the mobility process.

The framework described in [3] supports the development of network-aware applications. It provides the application with a feedback loop that informs about changes in the network resources and that helps mapping from network centric quality measures to application centric quality measures. The states of below-application layer mobility protocols are not part of the network resources. Additionally, network communication goes through the feedback loop layer (adaptation layer). Our mechanism allows an application to use the regular operating system facilities for network communications.

Although the Congestion Manager presented in [1] does not deal with mobility management issues, it provides a mechanism (API) that allows applications to be aware of changing network conditions. The API replaces the Sockets API, whereas our interface towards the applications stands side by side with the Sockets API. The work in [21] uses ICMP messages to provide information about the link conditions and network environment to the layers above the network layer. However, it does not define messages describing the mobility process.

The mechanism proposed in [24] is an example of application aware mobility management. It adapts the routing behavior of the mobile host depending on the kind of connection initiated by the application. This allows the operating system to decide whether Mobile IP will handle a connection or not. In contrast, we let the application make this decision: by binding a socket to the IP address of an interface, the application can influence the selection of the applied mobility management protocol. Additionally, our mechanism not only incorporates the usage of mobility management protocols at the network layer, but also at the transport and application layer.

8. CONCLUSIONS AND FUTURE WORK

We have introduced a mechanism for host mobility management in an environment of heterogeneous networks that cooperates and interacts with existing mobility management protocols at different layers. This mechanism provides means to make the applications running on the mobile host aware of the states and events of the mobility process and aware of the state and characteristics of the available network resources. We have proposed an application classification to better understand the way applications react to information about the mobility process and the network resources. Furthermore, we have implemented a prototype of this mechanism in the form of the mobility manager, showing the complexities of an integrated application-aware mobility management system. We provided an initial validation of the mechanism by using the prototype with a streaming video player implemented as both a type II and a type III application.

Our future work will focus on the level of abstraction offered by the mobility manager interface towards the applications.

9. AVAILABILITY

The implementation of the mobility manager is available at [13] in the form of source code.

10. ACKNOWLEDGMENT

We would like to thank the members of the 4gplus project, especially our colleague Ronald van Eijk and our co-workers from Lucent Bell Labs in The Netherlands, for their contributions to the fruitful discussions we had about the concepts and implementations of mobility management on the mobile host.

This research has been supported by the Dutch "Freeband Kennisimpuls" Research Programme for Telecommunication Applications (4gplus project).

11. REFERENCES

[1] D. Andersen, D. Bansal, D. Curtis, S. Seshan, and H. Balakrishnan, "System Support for Bandwidth Management and Content Adaptation in Internet Applications", In *Proceedings of the Fourth Symposium on Operating Systems Design and Implementation (OSDI 2000)*, Oct. 2000.

[2] O. Angin, A. Campbell, M. Kounavis, and R. Liao, "The Mobiware Toolkit: Programmable Support for Adaptive Mobile Networking," *IEEE Personal Communications Magazine*, pp. 32–43, Aug. 1998.

[3] J. Bolliger, and T. Gross, "A Framework-Based Approach to the Development of Network-Aware Applications", *IEEE transactions on Software Engineering*, vol. 24, no. 5, pp. 376-390, 1998.

[4] A. T. Campbell, J. Gomez, and A. Valko, "An Overview of Cellular IP", In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC'99)*, September 1999.

[5] Dynamics Mobile IP, Retrieved on 24/3/2004 from http:// dynamics.sourceforge.net/.

[6] E. Gustafesson, A. Jonsson, and C. Perkins, "Mobile IPv4 Regional Registration", *IETF Internet Draft*, draft-ietf-mobileip-reg-tunnel-08.txt (work in progress), Nov. 2003.

[7] Introduction to the MBone, Retrieved on 24/3/2004 from http://www-itg.lbl.gov/mbone/.

[8] R. Jain, T. Raleigh, C. Graff and M. Bereschinsky, "Mobile Internet Access and QoS Guarantees Using Mobile IP and RSVP with Location Registers", In *Proceedings of the IEEE International Conference on Communications (ICC '98)*, Atlanta, GA, June 1998.

[9] D. Johnson, C. Perkins, and J. Arkko, "Mobility Support in IPv6", *IETF Internet draft*, draft-ietf-mobileip-ipv6-24.txt (work in progress), June 2003.

[10] M. E. Kounavis, A. T. Campbell1, G. Ito, and G. Bianchi, "Design, implementation and evaluation of programmable handoff in mobile networks", *Mobile Networks and Applications 6*, pp. 443–461, 2001.

[11] T. T. Kwon, M. Gerla, S. Das, and S. Das, "Mobility Management for VoIP: Mobile IP vs. SIP", *IEEE Wireless Communications Magazine*, vol. 9, no. 5, pp. 66-75, Oct. 2002.

[12] A. Misra, S. Das, and P. Agrawal P, "Application-centric analysis of IP-based mobility management techniques", *Journal of Wireless Communications and Mobile Computing*, vol. 1, issue 3, Aug. 2001.

[13] Mobility Manager source code, http://moma.telin.nl/.

[14] B. Noble, "System Support for Mobile, Adaptive Applications", *IEEE Personal Communications*, pp. 44-49, Oct. 2000.

[15] B. Noble and M. Satyanarayanan, "Experience with adaptive mobile applications in Odyssey", *Mobile Networks and Applications 4*, pp. 245-254, 1999

[16] C. Perkins (Ed.), "IP Mobility Support for IPv4", IETF RFC 3344, Aug. 2002.

[17] R. Ramjee, T. La Porta, S. Thuel, K. Varadhan and S.Y. Wang, "HAWAII: A Domain-Based Approach for Supporting Mobility in Wide-Area Wireless Networks", In *Proceedings of the Seventh Annual International Conference on Network Protocols (ICNP '99)*, Oct. 1999.

[18] M. Riegel, and M. Tuexen, "Mobile SCTP", *IEFT Internet draft*, draft-riegel-tuexen-mobile-sctp-03.txt (work in progress), Aug. 2003.

[19] A. Snoeren, and H. Balakrishnan, "An End-to-End Approach to Host Mobility", in *Proceedings of the 6th International Conference on Mobile Computing and Networking (MobiCom 2000)*, Aug. 2000.

[20] R. Stewart, M. Ramalho, Q. Xie, M. Tuexen, I. Rytina, M. Belinchon, and P. Conrad, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", *IETF Internet draft*, draft-ietf-tsvwg-addip-sctp-08.txt (work in progress), Sep. 2003.

[21] P. Sudame and B.R. Badrinath, "On Providing Support for Protocol Adaptation in Mobile Wireless Networks", *Mobile Networks and Applications 6*, 43-55, 2001.

[22] Video Conferencing Tool, Retrieved on 7/11/2003 from http://www-mice.cs.ucl.ac.uk/multimedia/software/vic/.

[23] E. Wedlund, and H. Schulzrinne, "Mobility Support Using SIP", In *Proceedings of 2nd ACM/IEEE International Conference on Wireless and Mobile Multimedia (WoWMoM'99)*, Seattle, USA, Aug. 1999.

[24] X. Zhao, C. Castelluccia, and M. Baker, "Flexible Network Support for Mobility", In *Proceedings of the Fourth Annual International Conference on Mobile Computing and Networking (MobiCom'98)*, Dallas, Texas, October 1998.