A Software Architecture for Network-Assisted Handover in IEEE 802.21

Claudio Cicconetti, Francesco Galeassi, Raffaella Mambrini
Intecs S.p.a., Via Egidio Giannessi, 5, I-56121 Pisa, Italy
Email: {claudio.cicconetti, francesco.galeassi, raffaella.mambrini}@intecs.it

Abstract— **The IEEE 802.21 is a recent standard for enabling handover in heterogeneous wireless networks, which defines protocols and messages for mobile-to-node and node-to-node communication in a technology-neutral and flexible manner. The need arises because of the widespread diffusion of different technologies for wireless communications (e.g., WiFi, WiMAX, LTE), which are often coexisting in the same geographical area and supported by the same mobile device, but seldom interoperable. In this paper we propose the architecture of a Media Independent Information Service (MIIS) and the procedures for a network-assisted handover, which are left unspecified by the standard, aimed at reducing the energy consumption of mobile nodes due to scanning. A detailed description of the software modules involved and their interactions is included. Testbed results are reported to show the effectiveness of the proposed solution with COTS devices.**

Index Terms— **media-independent handover; seamless handover; IEEE 802.21**

I. Introduction

Many access networks are based on wireless technologies and their popularity is rapidly increasing while the users become more and more familiar with ubiquitous services. Also, mobile devices supporting multiple radio technologies are now common [1], and achieving the best Quality of Experience (QoE) requires a means to experience smooth handover procedures both intra-technology (horizontal) and inter-technology (vertical) so as to keep the mobile device connected to the most suitable access network at a given time and place.

For these reasons, the IEEE 802.21 standard [2] for Media Independent Handover (MIH) services has been published in January 2009. The document is aimed at defining the mechanisms to enable and optimize handover between heterogeneous IEEE 802 networks, including those where handover is not otherwise defined, and facilitate handover between IEEE 802 networks and cellular networks, as well as 4G wireless networks [3]. This goal is achieved by defining new entities and services that must be implemented into the mobile and the network devices and an extensible communication protocol. However, as it is often the case with communication standards, the procedures and algorithms are left unspecified so as to promote competition by differentiation of equipment capabilities and services. In this paper we propose the detailed procedures to enable network-assisted handover, where the mobile devices are instructed by the network itself on the best possible serving access points, thus removing the need for scanning. This way both the handover latency and the energy consumption of the mobile device are reduced, and benefits are especially significant if multiple radios coexist in the same device, since only one of them needs to be powered on at any time. A proof-of-concept experimental evaluation, with two overlapping non-interoperable IEEE 802.11a and IEEE 802.11g networks, has been carried out to show the feasibility and effectiveness of the proposed solution.

The rest of the paper is structured as follows. An overview of the IEEE 802.21 standard is provided in Section II. A survey of the recent state of the art is reported in Section III. In Section IV we describe in details the proposed network-assisted handover, which is validated through testbed experiments in Section VI. Conclusions are drawn in Section VII.

II. IEEE 802.21 Overview

In this section we describe the aspects of the IEEE 802.21 that are more closely related to this work. We refer the interested reader to the recent tutorial paper [4] for a gentle introduction and to the standard document [2] for all details.

The standard consists of i) a framework that allows seamless transition of Mobile Node (MN) between networks with (possibly) technology; ii) a new entity called MIH function (MIHF); iii) a MIH Service Access Point (SAP), called MIH_SAP, and its associated primitives; iv) a link-layer SAP, called MIH_LINK_SAP, for each network technology; v) an abstract media dependent interface that provides transport services over the data plane on the local node, called MIH_NET_SAP; vi) a new entity called MIH user (MIH_USR), which is the functional entity that employs MIH services.

The MIH reference model is illustrated in Fig. 1. The MIH_LINK_SAP is used to collect link information and

This paper is based on "Network-Assisted Handover for Heterogeneous Wireless Networks," by C. Cicconetti, F. Galeassi, and R. Mambrini, which appeared in the Proceedings of the Workshop on Seamless Mobility, co-located with IEEE Globecom'10, Miami, FL, USA, December 2010. © 2010 IEEE.

The research leading to these results has been partially funded by the European Community's Seventh Framework Programme (FP7/2007-2013) under Grant Agreement n. 233679. The SANDRA project is a Large Scale Integrating Project for the topic AAT.2008.4.4.2 (Integrated approach to network centric aircraft communications for global aircraft operations). The project has 31 partners and started on 01/10/09.

Manuscript received November 15, 2010; accepted January 13, 2011.

doi:10.4304/jcm.6.1.44-55

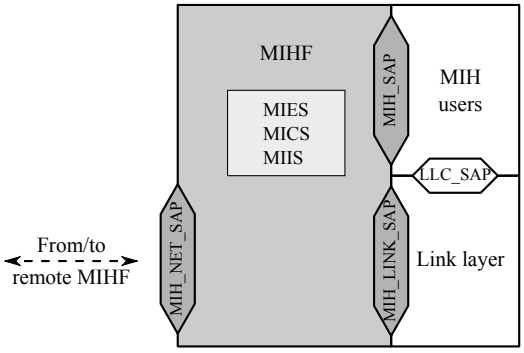

Figure 1. MIH reference model and SAPs.

control link behavior during handover. For each existing network technology, amendments are needed to interact with the basic primitives defined in the IEEE 802.21 standard. Such work is being carried out for the IEEE 802.3, IEEE 802.11, IEEE 802.16, 3GPP and 3GPP2 in the respective task groups or technical committees. On the other hand, it is envisaged that new technologies will incorporate suitable primitives for IEEE 802.21 natively. The MIHF provides the following services. *Media Independent Event Service* (MIES): The events are notifications that are generated by the link-layer, typically involving the link quality and status. A MIH user can subscribe to receive such notifications, both from the local lower layers and from remote entities, through the local MIHF. It is possible for multiple users to register to the same event, in which case the notification is sent to all the subscribers.

Media Independent Command Service (MICS): Commands are sent from the MIH users to the lower layers. Commands are used to configure, control, and retrieve information from the lower layers. Like events, also commands can be directed to remote entities through the local MIHF.

Media Independent Information Service (MIIS): It defines a framework for acquiring, storing, and retrieving information useful for handover decisions within a geographical area. Support for many Information Elements (IEs) is included to encompass the different types of mobility and supported technologies. The IEs can be encoded either in a binary form, using Type-Length-Value (TLV) structures, or with a Resource Description Framework (RDF)[1] representation, similar to the Extensible Markup Language (XML). With RDF the basic schema is provided in the Annex H [2], which can be easily extended to include user- or vendor-specific pieces of information. Data retrieval is done by means of TLV or SPARQL[2] queries, for each respective encoding method.

A link-layer element which can provide the MN with network access is called Point of Attachment (PoA). The PoA currently in use by a MN is said its *serving PoA*, while the PoAs under evaluation for possibly becoming PoA in the near future are called *candidate PoA*, and, finally, the PoA that has been selected to become the next serving PoA is the *target PoA*. The MIHF of the MN exchanges MIH messages with a peer in the network that is called its Point of Service (PoS). For maximum flexibility, the communication protocol between MIHF entities is specified by the standard for both layer 2 and layer 3: layer 2 transport is allowed with the EtherType value set to that for MIH Protocol; layer 3 transport is supported for the Transmission Control Protocol (TCP), the User Datagram Protocol (UDP), and the Stream Control Transmission Protocol (SCTP). An acknowledgment service can be enabled to add reliability to message exchange if the transport method adopted does not already provide this.

[1]http://www.w3.org/RDF/

[2]http://www.w3.org/TR/rdf-sparql-query/

III. Related word

The motivation for a network-assisted handover is provided by the previous work in the literature, though not involving the IEEE 802.21 standard. For instance, the benefits in terms of handover delay and fair service satisfaction have been demonstrated through testbed experiments in [5], at least for the case of homogeneous IEEE 802.11 networks. On the other hand, the impact of scanning is investigated in [6], where the authors propose an analytical model to predict the successful scan probability in a heterogeneous IEEE 802.11/IEEE 802.16 network, also taking into account some practical service charge functions.

With regard to the specific area of IEEE 802.21, in [7] the authors present a simulation of handover between 3G and Wireless Local Area Networks (WLANs), observing a trade-off between service continuity and WLAN utilization. Analysis and simulation have also been employed in [8] [9] to verify that the overall signaling can be significantly reduced by employing the IEEE 802.21 capabilities in vehicular networks.

With regard to practical implementations, the early work in [10] describes a prototype implementation of a Mobility Manager for handling not only handover, but also power management and adaptation. However, the Mobile IPv6 for Linux (MIPL) was used instead of a pure IEEE 802.21 because the latter was still in an early draft stage. In [11] is proposed a framework for the implementation of the IEEE 802.21 MIH standard and its performances are evaluated through experiments in integrated IEEE 802.11/802.16e networks. From the experimental results in real testbeds, the authors demonstrate that the MIH services can be used to reduce packet losses, the Access Points (APs) discovery time and the energy consumption of mobile nodes during a vertical handover.

More recently, practical considerations for implementing IEEE 802.21 in Linux have been discussed in [12], which also includes a blueprint of the functional blocks required. This implementation focuses on the support of several access technologies including IEEE 802.11, IEEE 802.16 and IEEE 802.3 using off-the-shelf Linux capabilities.

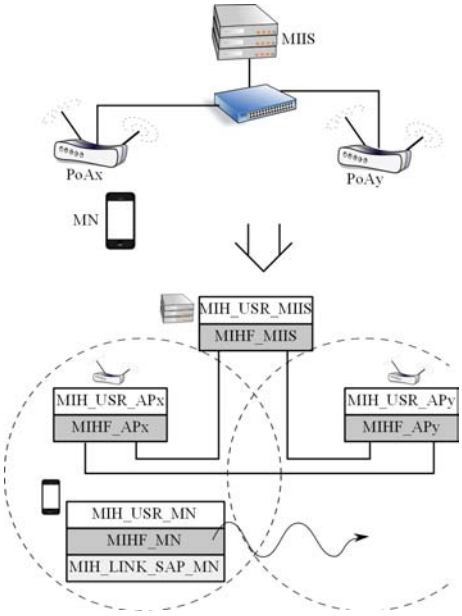

Figure 2. Logical view of the MIH elements in a sample network.

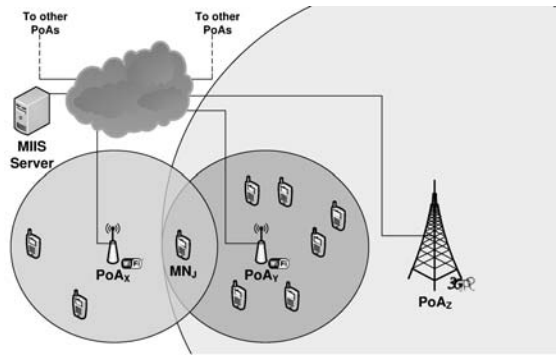

Figure 3. Example scenario.

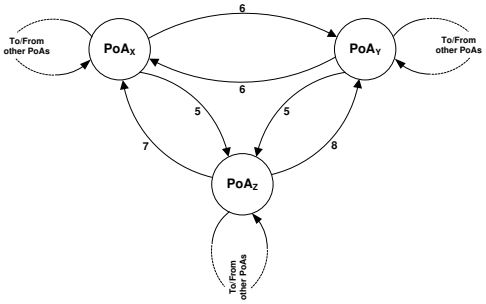

Figure 4. Connectivity graph for the example scenario.

The approach of [13] is complementary and aims at providing a generic MIH services framework, encompassing the standard's communication model, transport and state machines. An illustrative scenario for proactive pre-authentication using COTS devices is also presented to demonstrate how network selection and handover preparation can be leveraged by such an implementation. A pre-authentication method combined with IEEE 802.21 is also presented in [14]. In [15] is illustrated the design and implementation of a MIH middleware on a Linux platform so as to support a high quality Voice over IP (VoIP) system.

Finally, in [1] the authors present a vertical handover framework built around the concept of IEEE 802.21 and complemented by modules inspired by IEEE 1900.4 [16] for context information collection and efficient handover decision making and control.

IV. Network-Assisted Handover

In this section we describe how to enable network-assisted handover with IEEE 802.21 nodes. For simplicity, we assume that all the infrastructure nodes can act as PoA with co-located PoS, and that they are all willing to cooperate and share information with a node with MIIS functions. In practice this is straightforward when the infrastructure nodes, including the MIIS node, are managed by the same entity, e.g., the Mobile Network Operator (MNO).

The logical view of a sample network with two PoAs, with co-located PoSs, and the MIIS is illustrated in Fig. 2, which also shows the MIHF, MIH_USR, and MIH_LINK_SAP entities in all the network elements involved. Note that all the MIHF blocks of the infrastructure nodes, including the MIIS node, are bound together in a full mesh mode. This is to allow network-to-network exchanges to favor the handover, and can be done by means of discovery procedures or static configuration or any other method. How such connections are created is not relevant to this work and, hence, is not discussed here. In the following we describe the data structures used by the MIIS to assist handover, then provide a detailed overview of the bootstrap and handover procedures.

A. MIIS handover decision

The MIIS has two data structures that it uses to take network-wide handover decisions. We refer in the following to the example in Fig. 3, with three overlapping heterogeneous wireless networks.

First, in the *connectivity graph* each PoA is represented as a node and the weight of the edge between node i and node j is proportional to the likelihood that a MN currently associated to node i will roam to node j. The connectivity graph for the example scenario looks like that depicted in Fig. 4, where WiFi APs are preferred to the 3G PoA from the point of view of the network operator. It is important that the connectivity graph is updated regularly, otherwise handover decisions might not be sub-optimal.

For instance, the network operator can initialize the graph based on the location of PoAs and their technology, then learning algorithms can be implemented in the MIIS to adapt the weights based on the actual network dynamics (e.g., if handover of MNs between two nodes repetitiously fails, then the corresponding weight can be reduced or the edge in the graph can be removed). The detailed way how the connectivity graph is initialized and updated is out of the scope of this work.

On the other hand, the *utilization table* stores for each PoA a value between 0 and 1, where 0 means totally unused and 1 means overloaded. The way how this value is computed depends on the link-layer technology; the table is kept up-to-date by the PoAs themselves, which piggyback the current utilization value in the MIH_Get_Information.request messages (see detailed procedure below). Table I shows a possible utilization table for the example scenario, where PoA_z is totally unused, while PoA_y is heavily overloaded. The scope of this table is to balance the load due to MNs among the available PoAs so as to obtain optimal resource utilization, maximize throughput, minimize response time and avoid overload. By combining the data in the connectivity graph and the utilization table, the MIIS can then take handover decisions for the MNs taking into account both long-term and short-term inputs, respectively.

For instance, as shown in Fig. 3, assume that the MN_j is associated with PoA_x and that it is moving towards PoA_y. If the MIIS server uses only the connectivity graph for the handover decision, PoA_y will be indicated as the next target PoA. However, since the latter is overloaded, this would lead to a sub-optimal resource utilization. On the other hand, by combining the data in the connectivity graph and the utilization table, the MIIS server can rightfully select PoA_z as the candidate PoA for the MN_j.

The data structures proposed in this work are only meant as the foundations to define sophisticated algorithms for global network optimization, possibly including the prediction of the MN path and direction, which also depend significantly on the environment. For instance, consider a scenario where the MN is constrained along pre-determined paths, e.g., a motorway. In that case the only pieces of information needed for accurate prediction are the serving PoA and the MN direction: the former can be easily derived keeping track of the past PoAs that the MN left, the latter is indicated by the PoA itself when querying the MIIS.

PoA ID	Utilization Value
PoA_x	0.3
PoA_y	0.9
PoA_z	0
...	...

TABLE I.
UTILIZATION TABLE FOR THE EXAMPLE SCENARIO.

B. Boostrap procedure

When the MN first enters the network, e.g., it is switched on, it enters a *bootstrap* stage. This is the only phase when the MN is required to scan through all the interface to find a suitable PoA. After association and IP connectivity establishment, the MIHF of the MN discovers the MIHF of the PoA and binds it to itself. In our implementation, discovery is implemented by means of UDP broadcast. After binding, the MIHF of the PoA registers to the link parameters events, so as to be kept informed on changing conditions that might lead to a handover, and it sends a command to switch off all the unused wireless interfaces, in order to reduce power consumption. Until the MN remains in the network covered by the MIIS, it will never be required to perform scan again and it will use only one network interface at a time.

C. Handover procedure

Let us see now what happens when a handover occurs. The handover procedure is triggered by the MIH_LINK_SAP of the MN, which sends an event to the MIHF_MN when the link quality is degrading and, hence, a handover is imminent. The precise definition of the link quality degradation is technology-dependent and must be implemented in the link-layer specific part of the MIH_LINK_SAP. The trigger is conveyed by the MIHF_MN to the PoA, which performs a SPARQL query on the MIIS to retrieve the set of candidates PoAs. The request message contains also a SPARQL/Update[3] query used by the serving PoA to keep its value in the utilization table of the MIIS server up-to-date. The query result from the MIIS, encoded in XML[4], is based on its internal handover algorithm, which takes as input the connectivity graph, the utilization table, the last and the serving PoA of the MN. In our current implementation, a baseline simple algorithm is employed, which seeks the candidate PoAs by pruning all the PoAs with their utilization above a configurable threshold and sorts the remaining ones in decreasing order of their weights in the connectivity graph. If the last PoA of the MN is in the candidate list, it is indicated as last choice. The ordered list of PoA is passed to the serving PoA and, eventually, to the MN. Then, the MIHF of the MN tries all the candidate PoAs until association succeeds with one of them, which becomes the target PoA. At this point, the MN notifies both the serving PoA and the target PoA, which now becomes the serving PoA. The latter, also completes the handover procedures with network-to-network messages with both the previous PoA (in case the MN could not notify it due to excessive link quality degradation) and the MIIS (so that it can learn from the outcome of its handover decisions).

The detailed sequence diagram is illustrated in Fig. 5, where PoAx is the serving PoA and PoAy is the target one. The ordered list of PoAs contained in the MIH_Net_HO_Commit.indication messages requires vendor-specific extensions, which can be easily accommodated due to the flexibility of IEEE 802.21 messages. In the diagram some blocks of messages are enclosed with dashed boxes marked with letters from A to D, because they can happen simultaneously and, hence, their order in the figure is arbitrary.

[3]http://www.w3.org/Submission/SPARQL-Update/.
[4]http://www.w3.org/TR/rdf-sparql-XMLres/.

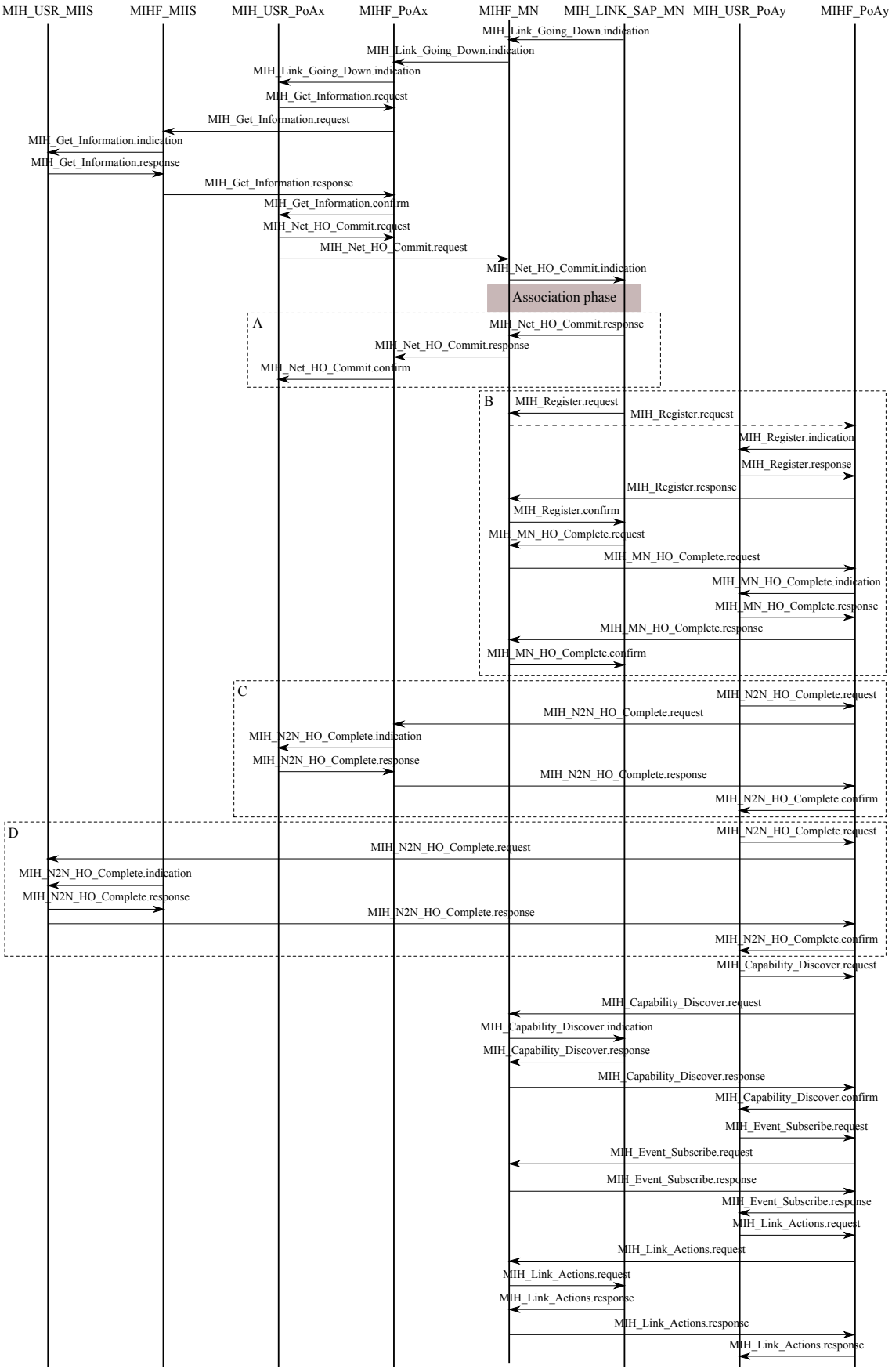

Figure 5. Sequence diagram of a network-assisted handover.

V. IMPLEMENTATION

In this section we describe the implementation of the proposed network-assisted handover procedure and provide the functional architecture of the MIIS server (Section V-A), the PoA (Section V-B), and the MN (Section V-C).

We employed as a basis the MIHF developed within the ODTONE project [5], with some necessary modifications for both optimization and extending those functional modules needed but not fully implemented (e.g., MIHF registration and discovery). ODTONE stands for Open Dot Twenty ONE and is an open source implementation of the Media Independent Handover framework from the IEEE 802.21 Media Independent Handover Services standard, using C++ APIs. The project started in July 2009 at the University of Aveiro, in Portugal, and it is released under the LGPL license. ODTONE's implementation provides a basic MIHF for implementing own MIH_SAP and MIH_LINK_SAP modules and handling MIH Protocol messages and state transitions.

A. MIIS server architecture

The MIIS server is composed of two modules, as shown in Fig. 2, i.e., the MIHF_MIIS and the MIH_USR_MIIS. The MIH_LINK_SAP is not necessary since the server does not need to interact with the layers below the MIHF, for instance, to configure a network interface or to receive notifications from the network devices. The main function of the MIHF_MIIS is to pass all the messages received from MIHF peers (e.g., MIHF_AP) to the MIIS server and to handle MIHF registration, in order to enable communication among remote MIH users. The MIIS server was implemented as an MIH user (i.e., MIH_USR_MIIS) and it has a local database which contains network information essential for handover decisions. It uses the framework supplied by the MIHF and the APIs provided by the MIH_SAP, to assist the MNs on handover decisions.

The MIIS server was implemented as a multi-thread process, as shown in Fig. 6, so as to handle multiple requests from the PoAs in parallel. One thread waits for incoming messages from the local MIHF; whenever a new message arrives, it creates a new thread to process the received message. The connectivity graph, the utilization table and the RDF schemas are contained in the *RDF/XML Data*. Information on the last PoA for each MN is instead contained in the data structure of the *Last PoAs*. The listening thread is composed of the following modules: MIHF Registration Handler, Handover Notification Handler, and Handover Assistant.

The *MIHF Registration Handler* deals with registration and de-registration of new MIHF entities and is responsible for sending the MIH_Push_Information.request to the registered peers, which are informed about supported IEs. The *Handover Notification Handler* manages notification on handover completion and is responsible for registering the last serving PoA in the internal data structure. The *Handover Assistant*, which helps PoA in handover decisions. For each MIH_Get_Information.request received, this module creates a new thread, which processes the message. This thread is in turn composed of the following two modules: an *Information Updater*, which is responsible for updating the utilization table using information contained in the SPARQL/Update query, and a *Handover Decisor*, which executes the SPARQL queries, so as to obtain the list of candidate PoAs for a specific MN.

B. PoA architecture

Each PoA is composed of two modules: the MIHF_AP and the MIH_USR_AP. As in the case of the MIIS server, the MIH_LINK_SAP is not necessary. The main function of the local MIHF is to pass all the messages received from MIHF peers (e.g., others MIHF_AP or MIHF_MIIS) to the MIH user and to handle MIHF registration, so as to enable communication among remote MIH users.

The MIH_USR of the PoAs was implemented as a multi-thread process, as shown in Fig. 7, so as to manage multiple requests from MNs in parallel. It uses the framework supplied by the MIHF and the APIs provided by the MIH_SAP, to assist associated MNs in handover decisions.

As shown in Fig. 7, the MIH_USR_POA is composed of the following logical modules: MIHF Registration Handler, MN Handler, Event Handler, Handover Assistant, and Handover Notification Handler and Notificator.

The *MIHF Registration Handler* deals with registration and de-registration of new MIHF entities. The *MNs Handler* is responsible for all the operations needed to register a new MN. After MIHF registration, it discovers the supported MIH services and the available interfaces on the MN. This information is stored in an internal data structure and is subsequently used by the Handover Assistant to assist the MN on handover. Afterwards, the MNs Handler registers the PoA MIH user to the link parameters events and switches off the unused interfaces. Optionally, it can set a threshold on the associated wireless interface. The *Event Handler* waits for event notification from the associated MNs. When this module receives a new notification, it means that handover is imminent for that MN. This way, it contacts the Handover Assistant, which will query the MIIS server to obtain the candidate PoAs list. The *Handover Assistant* assists associated MNs during handover. The handover process is triggered by the Event Handler, which communicates the MN that needs handover to the Handover Assistant. At this point, on the bases of the information contained in the internal data structure, the Handover Assistant prepares the SPARQL/Update and SPARQL queries, and sends them in the MIH_Get_Information.request to the MIIS server. This way, it obtains the candidate PoAs list, which is finally passed to the MIH_USR_MN. The *Handover Notification Handler and Notificator* is responsible for communicating the handover completion to the MIIS server and to the old serving PoA. Moreover, it manages notification from other PoAs and uses the paired MIHF

[5] http://helios.av.it.pt/projects/odtone/.

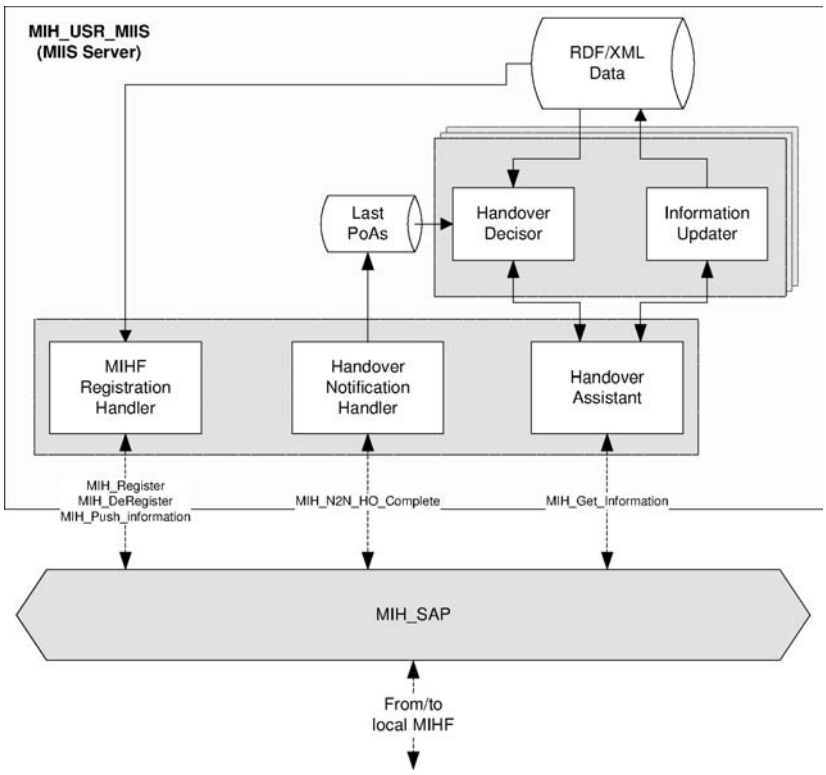

Figure 6. MIIS internal architecture.

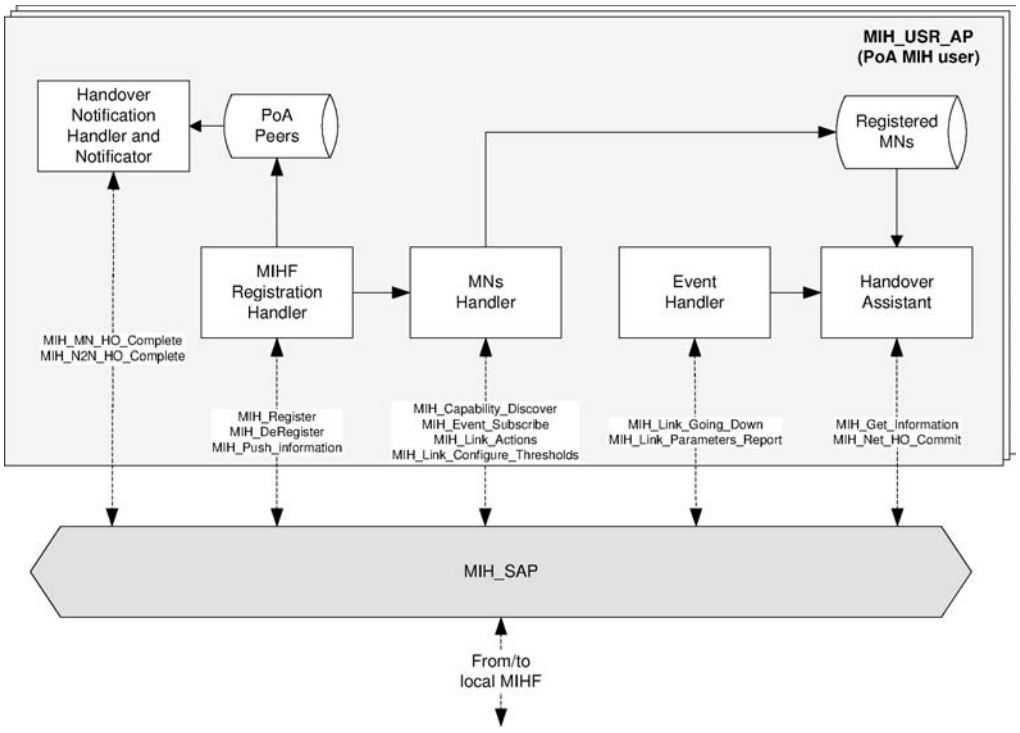

Figure 7. PoA internal architecture.

ID and the link address information, stored in the internal data structure, to contact the other PoAs.

The data structure of the *Registered MNs* contains the list of network interfaces for each associated MN, so as

to support it when handover is needed. Instead the data structure of the *PoA Peers* contains the paired list of link addresses and MIHF IDs for each registered PoA peer.

C. MN architecture

Each MN is composed of three modules: MIH_LINK_SAP_MN, the MIHF_MN, and the MIH_USR_MN. The main function of the MIH user is to start the bootstrap phase when the MN is first switched on, and, when handover is needed, to try all the candidate PoAs until an association succeeds.

Given the limited number of functions performed by the MIH user and in order to simplify the architecture of the MN, the MIH_USR_MN and the MIH_LINK_SAP have been combined. This way, the MIH_USR_MN no longer exists on the MN since its tasks are now performed by the MIH_LINK_SAP_MN, whose functional view is shown in Fig. 8.

The MIH_LINK_SAP_MN is a link-layer dependent MIH_LINK_SAP for MNs with support for IEEE 802.11 wireless interfaces. It has been developed for Linux, using the RTNetlink user space/kernel interface, and it is composed of the following modules: MIHF Registration Handler, Command Handler, RTNetlink Message Handler, Signal Level Checker, Bootstrap Handler, and Handover Handler.

The *MIHF Registration Handler* deals with registration and de-registration of new MIHF entities: it is informed by the RTNetlink Message Handler when an association with a new PoA has been completed, so as to start the registration process with the serving PoA, and it manages the de-registartion process with the old serving PoA when the MN completes a handover.

The *Command Handler* receives and processes MIH commands from the serving PoA, e.g., MIH_Capability_Discover.request messages. Furthermore, it is responsible for powering on/off the network interfaces and commanding a wireless scan, using ioctl system calls, as well as setting a threshold on the signal level of the associated interface. This threshold is then stored in the appropriate data structure and used by the Signal Level Checker module.

The *RTNetlink Message Handler* manages events triggering, analyzing messages from the Linux kernel through the RTNetlink interface. RTNetlink provides an effective way of gathering and controlling kernel media-independent network information. RTNetlink is a subset of Netlink which provides full-duplex communication between kernel modules and user-space processes through AF_NETLINK sockets. RTNetlink enables joining various multicast groups for getting notifications about changes in states and parameters from all the information classes mentioned above, without the need of user-space applications to poll the kernel for new information. This way, RTNetlink Message Handler generates IEEE 802.21 events (i.e., Link_Up and Link_Down) every time there is a change in the state of a network interface: when an interface establishes L2 connectivity, the module is responsible for activating a thread which checks the signal level on the specific interface; when an interface associated loses connectivity, the module initiates a new bootstrap phase; when the L2 handover is complete, the module indicates this to the Handover Handler.

The *Signal Level Checker* controls the signal level of the IEEE 802.11 wireless interfaces. For each available interface, a specific thread is responsible for checking the Received Signal Strength Indication (RSSI) level through the Linux Wireless Extension (WE) API. WE is a generic interface for obtaining the statistics and configurations of a WiFi link. WE methods are based on a set of ioctl calls and the `/proc` file system. It provides both dynamic and static information about the WiFi card and the associated link condition. A straightforward way to identify a network interface as a WiFi is to test if WE exists for that particular interface. The centralized entry `/proc/net/wireless`, which provides wireless specific dynamic statistics from the driver, like the current signal strength and the noise level, is used indirectly via WE. WE is also used by the the MIH_LINK_SAP_MN when the MN is first switched on to obtain the list of available wireless interfaces, which is stored in a dedicated internal data structure. If the signal level crosses a configured threshold, the Signal Level Checker generates a Link_Parameters_Report event, which will be delivered to the local MIHF. Moreover, this module is responsible for generating the Link_Going_Down event when the quality of the link is degrading, and, hence, handover is needed. Note that if an interface is down or not associated, the thread responsible for checking its signal level, is not active, to mimimize waste of resources.

The *Bootstrap Handler* manages the bootstrap phase. It is activated when the MN is first switched on or if the currently associated interface loses connectivity abruptly.

Finally, the *Handover Handler* deals with the handover process. It is triggered as soon as a MIH_Net_HO_Commit.request is received from the serving PoA. Afterwards, this module tries all the candidate PoAs in the ordered list until an association succeeds with one of them. If, for any reason, the association fails with all the PoAs, the module contacts the Bootstrap Handler module, so as to start a new bootstrap phase. As already indicated, notification of handover completion is received from the RTNetlink Message Handler, upon which the Handover Handler creates the MIH_MN_HO_Complete.request message to notify the new serving PoA of handover completion, while also sending the MIH_Net_HO_Commit.response to the old serving PoA.

D. Porting and extension issues

The MIIS server and the PoA MIH user are technology-independent and they have been developed in C++. While they have been developed and tested only in Linux, the modules use only POSIX-compliant inter-process and communication facilities, which would limit the effort of porting to other platforms and architectures. On the other

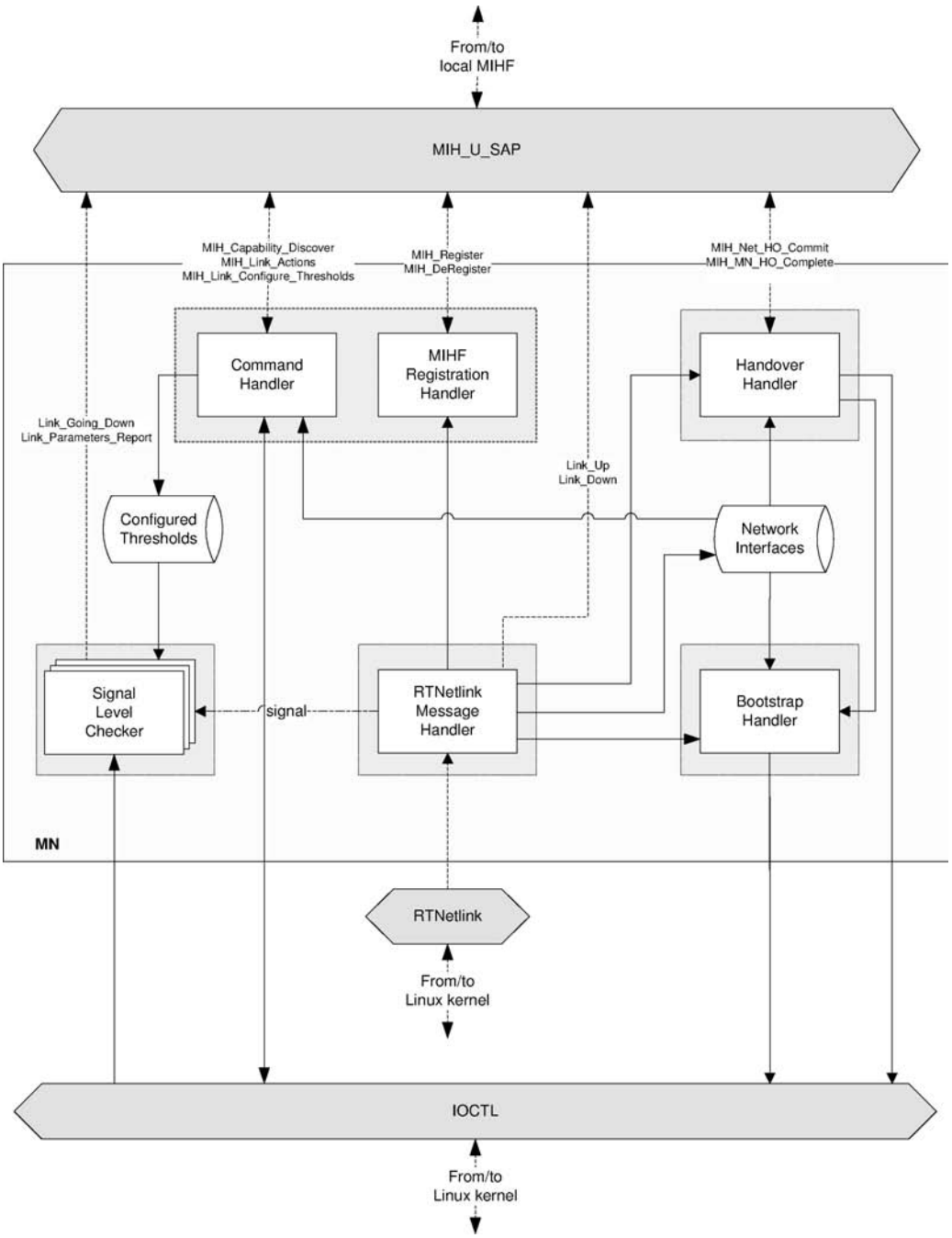

Figure 8. MIH_LINK_SAP_MN internal architecture.

hand, the MIH_LINK_SAP_MN uses Linux-specific features, hence it would require a substantial re-writing for porting to another Operating System (OS). Such an effort is comparable to what would be needed to include support for another wireless technology, e.g. IEEE 802.16, since no modifications would be required anyway to the MIIS server and the PoA MIH user.

VI. Experimental Results

In this section we describe the implementation of our solution in order to prove its effectiveness in small-scale testbed experiments.

A. Setup of experiments

In order to validate the proposed solution, a testbed has been set up, consisting of the following entities: three PoAs, with co-located PoSs, two of which equipped with an IEEE 802.11a wireless network interface and one with an IEEE 802.11g wireless network interface; an MIIS server, supporting RDF representation and SPARQL queries; a mobile node, supporting IEEE 802.11 technology.

All PoAs consist of ALIX embedded PCs[6] equipped

[6]http://www.pcengines.ch/.

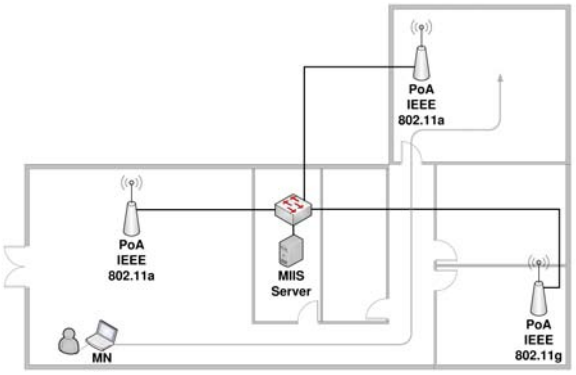

Figure 9. Testbed scenario.

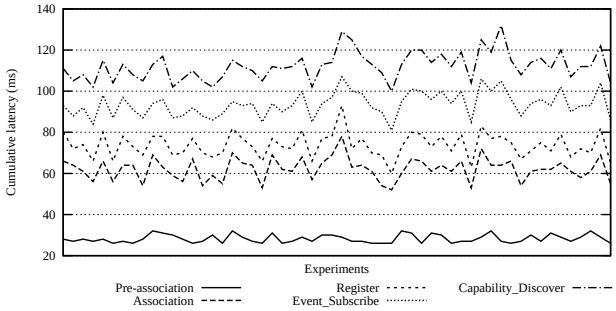

Figure 10. Latency of the handover phases (Fig. 5) in different experiments.

with Voyage Linux OS distribution[7]. The MIIS server was installed on a dedicated PC with standard hardware, connected to the PoAs via a full mesh Ethernet LAN. This is to allow network-to-network exchanges to favor the handover and inform the interested entities of handover completion, and it is done by means of discovery procedures.

With regard to the MN, experiments were run both with a laptop equipped with a single IEEE 802.11a/g wireless interface and with an ALIX embedded PC equipped with two interfaces (i.e., IEEE 802.11a and IEEE 802.11g). PoAs have been put in the topology shown in Fig. 9, with a few meters overlapping coverage areas. The MN and the serving PoA communicates toward the wireless channel, while the PoA and the MIIS server use the wired connection. Noticeable differences have not been found for the two types of MN (i.e., the laptop and the ALIX embedded PC), hence only the results obtained with the laptop are reported in the following.

Basic connectivity during handover has been successfully validated in the testbed scenario shown in Fig. 9.

When the MN is first switched on, it scans through all the interfaces to find a suitable PoA. After association and IP establishment, the MN starts to move. Afterwards, it automatically switch among available PoAs in a seamless way and without any user's intervention. However, these results are qualitative and cannot be reported here.

The handover execution (e.g., L2 and higher layers signaling) is out of the scope of the IEEE 802.21 standard and needs to be addressed by any mobility management protocols, for instance, so as to guarantee session continuity. In order to overcome this problem, in this testbed scenario each PoA automatically assign a pre-configured IP address to the MN when it completes the handover.

B. Results

In order to obtain more quantifiable and repeatable results, the PoAs were put in proximity of one another in a laboratory and the handover of the MN was triggered via a software indication (i.e., a MIH_Link_Going_Down.indication message) through an automated test procedure.

[7]http://linux.voyage.hk/.

In Fig. 10 is reported the latency of the different phases of the handover, in a cumulative manner, i.e., the latency of each phase is added to that of the previous phase. The samples obtained from more than 50 tests are included in the plot. The phases included in Fig. 10 are the following:

- *Pre-association phase*, which starts when the MN sends a MIH_Link_Going_Down.indication to the serving PoA, for indicating that the link quality is degrading and, hence, a handover is imminent. It also includes message exchanges to and from the MIIS server (i.e., MIH_Get_Information messages) for obtaining information about neighboring PoAs. This phase ends when the MN receives the MIH_Net_HO_Commit.request message from the serving PoA with the list of candidate PoAs.
- *Association phase*, which is totally dependent on the link-layer functions. This phase indicates the time necessary to the MN for establishing L2 connectivity.
- *Register phase*, where the MIHF of the MN discovers the MIHF of the PoA and binds it to itself towards MIH_Register messages. This way, the MN declares its presence to the associated PoA, so as to be later supported on handover decisions.
- *Capability Discover phase*, which represents message exchanges between the MN and the serving PoA (i.e., MIH_Capability_Discover messages). This message is used by the serving PoA to discover the supported MIH services and the available interfaces on the MN.
- *Event Subscribe phase*, where the serving PoS registers to the link parameters events, by means of a MIH_Link_Event_Subscribe.request, so as to be kept informed on changing condition that might lead to a handover.

As can be seen, the greatest contribution comes from the association phase, which is totally dependent on the link-layer functions. For instance, in the specific case of IEEE 802.11, it has been demonstrated [5] that the association time can be reduced significantly with some minor modifications to the devices. On the other hand, the pre-association phase is very small, mostly because there are no other concurrent handovers.

Note that in the handover latency are not considered node-to-network and network-to-network messages used

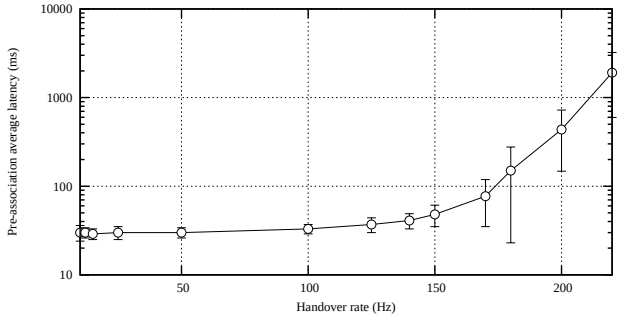

Figure 11. Average latency of the pre-association phase vs. MIIS load.

to notify the handover completion, as well as the command to switch off all the unused wireless interfaces. In fact, these messages do not contribute to the handover latency.

Therefore, it has been decided to artificially inflate the MIIS server load, by injecting 50 simulated PoA in the network, querying the MIIS server at random times. Each simulated PoA sent to the MIIS server a MIH_Get_Information.request message, containing a SPARQL/Update query used to keep its current load up-to-date, as well as a SPARQL query used to obtain the list of candidate PoAs.

In Fig. 11 is reported the pre-association latency when the total number of handover procedures per second, on average, is increased from 10 to 220. In fact, this phase includes querying the MIIS server and so it is the only affected by this test.

As can be seen, the latency remains almost constant until 150 handovers per second, i.e., until the implemented MIIS server is able to dispatch the MIH_Get_Information.response messages with no significant delay. If the load increases even further, the latency increases very sharply because the MIIS server eventually becomes saturated.

It is worth noting that, with our implementation, a standard PC is able to manage a rate of up to 150 handovers per second, i.e. hundreds to thousands of MNs, depending on their degree of mobility and on the coverage of the technologies employed.

VII. Conclusion

In this work we have proposed the detailed architecture and procedures to enable network-assisted handover with heterogeneous wireless networks, by exploiting the MIIS concept of IEEE 802.21. A prototype implementation has been developed to prove the feasibility of the solution. Preliminary experimental results show that our proposed procedure is promising to keep the handover latency small while removing the burden of scanning at the mobile device.

Acknowledgment

Francesco Galeassi contributed to the work during his internship at Intecs, under the supervision of Prof. L. Lenzini and Prof. E. Mingozzi of the University of Pisa.

References

[1] L. Sarakis, G. Kormentzas, and F. M. Guirao, "Seamless Service Provision for Multi Heterogeneous Access," *IEEE Wireless Communications*, vol. 16, no. 5, pp. 32–40, October 2009.

[2] IEEE, "IEEE Standard for Local and metropolitan area networks– Part 21: Media Independent Handover Services," *IEEE Std 802.21–2008*, January 2009.

[3] L. Eastwood, S. Migaldi, X. Qiaobing, and V. Gupta, "Mobility Using IEEE 802.21 in a Heterogenous IEEE 802.16/802.11-Based, IMT-Advanced (4G) Network," *IEEE Wireless Communications*, vol. 15, no. 2, pp. 26–34, April 2008.

[4] S. Das, M. Tauil, Y.-h. Cheng, A. Dutta, D. Baker, M. Yajnik, and D. Famolari, "IEEE 802.21: Media Independent Handover: Features, Applicability, and Realization," *IEEE Communications Magazine*, vol. 47, no. 1, pp. 112–120, January 2009.

[5] X. Chen and D. Qiao, "HaND : Fast Handoff with Null Dwell Time for IEEE 802.11 Networks," *IEEE INFOCOM 2010. 29th IEEE International Conference on Computer Communications*, pp. 1–9, March 2010.

[6] Y. Choi and S. Choi, "Service Charge and Energy-Aware Vertical Handoff in Integrated IEEE 802.16e/802.11 Networks," *IEEE INFOCOM 2007. 26th IEEE International Conference on Computer Communications*, pp. 589–597, May 2007.

[7] A. de La Oliva, T. Melia, A. Vidal, C. J. Bernardos, I. Soto, and A. Banchs, "A case study: IEEE 802.21 enabled mobile terminals for optimized WLAN/3G handovers," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 11, no. 2, pp. 29–40, April 2007.

[8] Q. B. Mussabbir, W. Yao, Z. Niu, and X. Fu, "Optimized FMIPv6 Using IEEE 802.21 MIH Services in Vehicular Networks," *IEEE Transactions on Vehicular Technology*, vol. 56, no. 6, pp. 3397–3407, November 2007.

[9] G. Gehlen, E. Weiss, S. Lukas, C.-H. Rokitansky, and B. Walke, "Architecture of a Vehicle Communication Gateway for Media Independent Handover," *3rd International Workshop on Intelligent Transportation, WIT*, March 2005.

[10] F. Cacace and L. Vollero, "Managing Mobility and Adaptation in Upcoming 802.21 enabled devices," in *International Workshop on Wireless mobile applications and services on WLAN hotspots*, 2006, pp. 1–10.

[11] W.-S. Lim, D.-W. Kim, Y.-J. Suh, and J.-J. Won, "Implementation and performance study of IEEE 802.21 in integrated IEEE 802.11/802.16e networks," *Computer Communications*, vol. 32, no. 1, pp. 134–143, January 2009.

[12] E. Piri and K. Pentikousis, "Towards a GNU/Linux IEEE 802.21 Implementation," *IEEE International Conference on Communications*, pp. 1–5, January 2009.

[13] Y. Lopez and E. Robert, "OpenMIH, an Open-Source Media-Independent Handover Implementation and Its Application to Proactive pre-Authentication," *Mobile Networks and Management: First International Conference, MONAMI 2009*, October 2009.

[14] A. Dutta, S. Das, D. Famolari, Y. Ohba, K. Taniuchi, V. Fajardo, and T. Kodama, "Secured Seamless Convergence across Heterogeneous Access Networks," *World Telecommunication Congress*, 2006.

[15] T.-Y. Chung, Y.-M. Chen, P.-C. Mao, C.-K. Tsai, S.-W. Lai, and C.-P. Chen, "The Design and Implementation of IEEE 802.21 and Its Application on Wireless VoIP," *IEEE Vehicular Technology Conference (VTC)*, pp. 1–5, May 2010.

[16] IEEE, "IEEE Standard for Architectural Building Blocks Enabling Network-Device Distributed Decision Making for Optimized Radio Resource Usage in Heterogeneous Wire-

less Access Networks," *IEEE Std 1900.4–2009*, February 2009.

Claudio Cicconetti holds a PhD in Information Engineering from the University of Pisa (2007), where he also received his MS degree in Computer Science Engineering in 2003. He has been working in Intecs S.p.a., Italy, since 2009, where he is a Research Engineer in the fields of advanced networking topics and radio communications. He has been actively involved in many R&D projects both European (SANDRA - FP7, EuQoS - FP6, Celtic - Eureka) and Italian (QuaSAR, NADIR) and has served as a member of the organization committee of several international conferences (WiOpt, European Wireless, SIMUTools, Valuetools, QoSim, NSTools). He co-authored more than 40 papers published in international journals, peer-reviewed conference proceedings and book chapters, and one international patent.

Francesco Galeassi received his MS degree in Computer Science Engineering from the University of Pisa, Italy, in 2010. He is currently a Software Engineer at Intecs S.p.a., Italy. His research interests include algorithms and protocols for efficient handover in wireless networks and Software Defined Radio (SDR) architectures.

Raffaella Mambrini received her MS degree in Telecommunications Engineering in 1998 from the University of Pisa. She is currently the Telecommunications Business Unit Leader at Intecs S.p.a., Italy. She is a member of Verification & Validation and Model Driven Engineering Intecs Excellence Teams. She has been involved in leading roles in many national and European R&D projects (currently: IPERMOB, SANDRA). Her main activities include software modeling, development, verification & validation for automotive and telecommunications systems. Her last research interests include short and long range wireless networks, next generation network, Software Defined Radio (SDR) and applications for Intelligent Transportation Systems (ITS).