Consideration on interoperability of different wireless access networks using the IEEE 802.21 approach

Octavian Fratu¹, Eduard C. Popovici², Simona V. Halunga³, Emil Stănescu⁴

Abstract – The paper describes the main issues regarding the interoperability between some representative wireless access networks and the corresponding vertical handover in wireless hybrid access networks. The proposed approach conforms to the IEEE 802.21 Media Independent Handover (MIH) standards.

Keywords – Hybrid wireless access network, Media Independent Handover (MIH), interoperability, convergence.

I. INTRODUCTION

A major trend in the development of wireless access systems is to integrate different wireless communications technologies into a common hybrid “easy to use” communication infrastructure. A possible approach in achievement this objective is the use of Media Independent Handover (MIH) procedures. IEEE is building a standard based on this concept, the IEEE 802.21 one [1] [2].

Following this development, the RIWCoS (Reconfigurable Interactive Wireless Communications Systems, SIP-982469) project was proposed into Science for Peace NATO framework, focused on the communications issues in emergency cases. Later a new research project was started (Wireless Hybrid Access System with Unique Addressability – SAWHAU, no. 12-126/2008, funded by Romanian Government through the National Company for Project Management) focused on optimizing an hybrid wireless access network in order to reduce or, at least, to optimise the wireless communications access costs for small and medium enterprises.

Both of these projects are considering the IEEE 802.21 standard and its Media Independent Handover (MIH) concept as the main technological solution to solve their main objectives.

II. MEDIA INDEPENDENT HANDOVER STANDARD

The purpose of the IEEE 802.21 standard is to ensure seamless handover for a mobile unit between networks corresponding to IEEE 802 set of standards and 3GPP/3GPP2 standards. In particular, these types of mechanisms presented in the IEEE 802.21 standard can be used also for handover in homogenous networks, not only in heterogeneous ones. This new standard was finalized in November 2008 based on the D14 draft version which was approved. But this version is focused on the background features, so new tasks are now open: the 802.21a group focused on security and authentication issues, as well as 802.21b, proposed for interworking between access networks and broadcasting networks (especially DVB-T and DVB-H).

The Media Independent Handover (MIH) functionality facilitates handover decision making by placing itself between the network dependent link layer and upper layers that make this decision based on messages from MIH Functions (MIHF). On the other side, the MIH Function communicates with the link layer through technology specific interfaces.

The most important functionalities that IEEE 802.21 standard proposes are (see figure 1):

- a framework that enables a certain mobile unit that has support for MIH functionality to initiate a vertical handover between different types of networks, without the upper communication layers (starting with Layer 3) to lose service continuity.
- set of handover-enabling functions that work with different protocol stacks from Layer 2 and Layer 1 of different networks.

Fig. 1. Media Independent Handover framework

- implementation of a Service Access Point (MIH-SAP) between Layer 2 and Layer 3 of each independent
media independent services or link layers, or even predict changes of the link properties that have been registered to such events.

- **Media Independent Command Service** which transmits messages from protocol independent upper layers to the protocol dependent link layer for every network – enables also MIH users to command link properties during handover and switching between networks.

- **Media Independent Information Service** that provide information about different types of network and their Quality of Service implementation, which can be useful during handover decisions.

- **MIH-SAP** definitions of associated primitves for every type of network link layer technology, helping collecting information and control during handover.

The MIHF provides synchronous and asynchronous service through SAP for link layers and MIH users, that can use the three Services provided by MIHF to manage, determine and control the underlying interfaces.

Before providing these services, the MIH functions must be configured in several steps:

- **Discovery procedure** between a MIH user and a local or remote MIH function in terms of MIH capabilities and services, MIH protocol or media dependent (layer 2) broadcast messages.

- **Registration Procedure** between two MIH entities for making their presence known to their peers, mandatory only for command services.

- **Event Subscription** used by a MIH user to subscribe for events originating from a local or remote MIH function.

The network communication functions provide transport services over the data plane on the local node for MIH messages between local and remote nodes. For transport services over layer 2, the MIH_NET service access point utilizes primitives specified by the MIH_LINK service access point. Transport services over layer 3 are specified by the MIH_NET service access point.

**Media Independent Event Services** – may indicate, asynchronously, changes in state and transmission behaviour of the physical, data or link layers, or even predict changes of these layers.

The originators of these service messages can be MIH function for generating events or lower link layer.

The destinations can be MIH function for receiving events or upper link layers that have registered to such events.

Its main usages are enabling upper layers to detect the need for handover, carrying additional data such as layer 2 and 3 identifiers and IP address acquisition indications needed for handover.

**Media Independent Command Services** - the upper layers can, through these services, control the lower layers (physical, data and logical links). These MIH functions are mandatory by nature and are expected to be obeyed at reception.

The originators can be the upper layers or the MIH function itself - its destinations may be the lower layers or even the MIH function.

An example of commands exchange can be during network selection, when the mobile unit and the networks exchange messages. If a valid network has been found, the upper layers command the handover for the lower layers through the use of MIH services.

**Media Independent Information Services** - provides a framework and corresponding mechanisms by which a MIH entity may discover and obtain network information in order to facilitate the handovers.

It uses a set of information elements alongside with the information structure and its representation and a request - response type of mechanism for information transfer. All the information it transports can be available for both link and network layers through secure or non-secure ports.

It typically provides static link layer parameters (channel information, MAC addresses, security information of a point of access - PoA) as well as information about higher layers services that can be helpful when making a handover.

**Media Independent Handover Framework** - specifies a communication model in which a mobile unit exchanges MIH information through reference points with its Point of Service, or Point of Attachment in a specific network.

Different types of reference points are defined, depending on where the MIH function resides:

- **Reference Point 1 (RP1)** – communication procedures between a mobile unit’s MIHF and its Point of Service (PoS).

- **Reference Point 2 (RP2)** – communication procedures between a mobile unit’s MIHF and a Point of Service candidate for attachment during handover procedures.

- **Reference Point 3 (RP3)** – communication procedures between a mobile unit’s MIHF and a Point of Service that cannot be Point of Attachment for that specific unit, but has specific upper layer information that can be helpful during handover.

- **Reference Point 4 (RP4)** – communication procedures between a PoS in a network entity and non-PoS device in another network, containing only upper layers information.

- **Reference Point 5 (RP5)** – communication procedures between two MIH PoS in different network entities, containing also only upper layer information.

### III. RIWCoS General Architecture

The RIWCoS system is considering few wireless access technologies: UMTS, WiMAX and WLAN and can be extended to DVB-T/H with return channel, (i.e. a hybrid WiMAX-DVB-T/H or UMTS-DVB-T/H ones) [3]. The SAWHAU project is maintaining the main solutions coming from RIWCoS architecture, but the system is optimized in different manner (but a compatible one), i.e. it is focused on the access cost reduction.

According to the IEEE 802.21 standard, the RIWCoS project is considering a Convergence Stratum, containing an
Interoperability Sub-Stratum and a Resource Management Sub-Stratum (fig. 2) [3].

An Interoperability Manager module was introduced for each wireless communication system, representing the Interoperability Sub-Stratum. Its main function is mobility management. In a MIH capable network it should communicate and interoperate with MIH Function convergence layer (Fig. 2) and ensures the cooperation between the component technologies. The new convergence layer is considered according with the 802.21 requirements.

A distributed Resource Management System is also considered and integrates the resource modules. It should cooperate with Interoperability Manager Modules in order to distribute the traffic in the network that best fulfil the user requirements and capabilities and also the network policies.

The two Sub-Stratums are not at the same level. The Interoperability Sub-Stratum is sub-ordinate to the Resource Management Sub-Stratum. The following chapter is describing in detail the Interoperability Sub-Stratum.

IV. ARCHITECTURAL DESCRIPTION OF THE LINK INTEROPERABILITY MODULE

The RIWCoS Link Interoperability Modules (fig. 3) have as main responsibility to adapt specific, media dependent link layer management interface (collectively denoted here as L2_Specific_SAP) to the generic, media independent link layer management interface (MIH_LINK_SAP) [1] [10]. Besides adapting different media dependent link layer management interfaces to a common media independent interface, the Link Interoperability Modules realize other link management related activities, supported by the internal components described as follows.

Fig. 2. The RIWCoS hierarchical architecture, according 802.21 requirements

Fig. 3. Link Interoperability Module Components for Link Parameters and Events Management
The Event Subscription Handler component supports the MIHF Event Subscription mechanism (which allows an MIH User to subscribe for a particular set of events) for those events of interest that originate the local MIHF (Interoperability Manager). The Event Subscription Handler component implements two complementary Command Service (MICS) primitives, Link_Event_Subscribe and Link_Event_Unsubscribe, by using the L2 Event Monitoring local component (fig. 4).

The L2 Events Monitoring component centralizes the event subscription/notification process (fig. 5) both from the native link layer event sources (L2 Event Sources) and from local event sources (parameter threshold crossing or reporting interval passing). The Events Monitoring component keeps track of the subscribed event through a list of events of interest.

Link Events are defined as events that originate from event source entities (L2 Event Sources) below the MIHF (Interoperability Manager) and may terminate at the MIHF. Entities generating Link Events include, but are not limited to, various IEEE 802-defined, 3GPP-defined, and 3GPP2-defined interfaces. The event service may be used to detect the need for handovers. For example, an indication that the link will cease to carry MAC SDUs at some point in the near future may be used by MIH Users to prepare a new point of attachment ahead of the current point of attachment ceasing to carry frames. This has the potential to reduce the time needed to handover between attachment points.

The State Events Handler component implements seven Event Service (MIES) primitives related to changes of link state, Link_Detected, Link_Up, Link_Going_Down, Link_Event_Rollback, Link_Down, Link_Handover_Iminent and Link_Handover_Complete. It informs in real-time the Interoperability Manager about changes of link states that are of interest for MIHF local or remote entities (those for which at list one MIHF entity subscribed for).

The Media Independent Event Service (MIES) supports several categories of link events:
1) MAC and PHY State Change events: These events correspond to changes in MAC and PHY state. For example, Link_Up event is an example of a state change event.
2) Link Parameter events: These events are due to changes in link layer parameters. For example, the primitive Link_Parameters_Report is a Link Parameter event.
3) Predictive events: Predictive events convey the likelihood of a change in the link properties in the near future based on past and present conditions. For example, decay in signal strength of a WLAN network may indicate a loss of link connectivity in the near future. In case predictive events are incorrect they may be retracted.
4) Link Handover events: These events inform upper layers about the occurrence of L2 handovers/link switches if supported by the given media type.
5) Link Transmission events: These events indicate the link layer transmission status (e.g., success or failure) of upper layer PDUs. This information may be used by upper layers to improve buffer management for minimizing the upper layer data loss due to a handover.

As mentioned before, the L2 Events Monitoring component centralizes the event notification process from local event sources (parameter threshold crossing or reporting interval passing). It receives parameter report notifications from L2 Parameters Monitoring component, which centralizes the link parameters monitoring activities.

The L2 Parameters Monitoring component periodically requests parameter measurements from L2 Parameters Measurement Tools, and use the acquired parameter values to request the checking of threshold crossing from Threshold Comparison component. If at least one of the monitored parameters crossed a threshold, L2 Parameters Monitoring component will issue an alert report toward the L2 Events Monitoring component, which in turn notifies the Parameters Events Handler component (fig. 6).

The Parameters Events Handler component implements one Event Service primitives, Link_Parameters_Report, which are used to report in real-time to Interoperability Manager of any changes of link parameters values that are of interest for MIHF local or remote entities (those for which at list one MIHF entity subscribed for).
When requested by the Report Timer component (each time the reporting interval passed) the L2 Parameters Monitoring component will issue toward the L2 Events Monitoring component a periodic report, containing the last measured parameters values. Again, the Parameters Events Handler component is notified and issues a Link_Parameters_Report toward the Interoperability Manager (fig. 7).

The parameter event subscription takes the form of local event sources configuration, by means of the Link_Configure_Thresholds primitive (fig. 8). The Parameter Commands Handler component, which implements two Command Service (MICS) primitives, Link_Get_Parameters and Link_Configure_Thresholds, is responsible for configuration of link parameters thresholds or report intervals, as well as for providing the last measured values of such parameters at request.

Thus, through the Link_Configure_Thresholds primitive the Interoperability Manager (MIHF implementation) can set the parameter thresholds (minimum and/or maximum permitted values) into the Threshold Comparison components and the reporting intervals into the Report Timer components. At the same time, the list of parameters of interest inside the Parameter Monitoring component is updated.

Parameter Commands Handler component is responsible for processing the Link_Get_Parameters requests used by MIHF (Interoperability manager) to asynchronously acquire the current values of measured and calculated link parameters (the later by means of Derivated Parameters Computing component) (fig. 9).

The Interoperability Manager can obtain at any time the current content of the list of events of interest from the Events Monitoring component and the current content of the list of parameters of interest from the Parameter Monitoring component by issuing a Link_Capability_Discover primitive.

In a RIWCoS Link Interoperability Module, the Link Capability Discovery Handler component implements the Link_Capability_Discover primitive of Command Service (MICS), performing capabilities discovery at link level (available parameters and events) for local entities (fig. 10).

One last component of a RIWCoS Link Interoperability Module is Action Command Handler, which implements only one Command Service primitive, Link_Action, which in turn enables Interoperability Manager to ask from the link layer to execute an action (such as a scan for handover candidate networks). Besides asking for a candidate network scan, this component can request to the L2 Specific Management any available action for handover execution.

An overview of all components of RIWCoS Link interoperability Module and their relations is presented in figure 11.
V. CONSIDERATIONS ON IMPLEMENTATION OF WLAN INTEROPERABILITY MODULE

We consider the case of Media Specific Interoperability Module for WLAN wireless communication system, as a subcomponent of Interoperability Manager Module from Interoperability Sub-Stratum. For implementing the main functions of this sub-stratum, which is mobility management, we analyse the implementation approach for interoperability modules using NDIS support and specific Microsoft Windows Driver Architecture.

A. Overview on Windows Driver Support

For implementing the interoperability module on L2 network layer we have to consider the layered driver architecture supported by Windows operating systems. Every networking device is serviced by a driver stack which is build by a chain of drivers [8]. A driver stack could contain all or some of the types of drivers depicted in Fig 12. A user application uses Win32 API to initiate I/O operations, sending the request to I/O Manager. The I/O Manager builds I/O request packets (IRPs) in response to these I/O requests. IRPs can also be created by the plug-and-play manager, power manager, and other system components. The drivers can also create IRPs and then pass to other drivers. The IoCallDriver routine sends an IRP to the driver associated with a specified device object.

![Fig. 11. Components and their relations in a RIWCoS Link Interoperability Module](image)

![Fig. 12. Layered driver architecture](image)

The following routines are required for all drivers:
- **DriverEntry** is the first routine called after a driver is loaded, and is responsible for initializing the driver.
- **AddDevice** routine is responsible for creating functional device objects (FDO) or filter device objects (filter DO).
- DispatchXxx routine services I/O request packets (IRPs) using I/O function codes.
- Unload routine performs any operations that are necessary before the system unloads the driver.

In next sections we refer to the components of this architecture.

**B. Windows Driver Model**

The most recent driver model for Microsoft Windows family of operating Systems is Windows Driver Foundation (WDF) [5].

The previous drivers were built using Windows Driver Model (WDM), or other device class-specific driver models for common classes such as networking.

WDM is a framework for device drivers and it includes important features: asynchronous I/O, driver layering, Plug and Play, power management, Windows Management Instrumentation (WMI), and specific device objects. Communication between drivers and with the operating system in layered WDM hierarchy is achieved via kernel mode data structures, named I/O request packets (IRPs).

A classification of types of WDM drivers is following:

- **Device function drivers** – are typically written by the device vendor and can service one or more devices. A function driver provides the operational interface for its device. The function driver for a device can be implemented by two subtypes:
  - Class drivers – provide interfaces between different levels of WDM architecture and include the common functionality for other class and miniport drivers.
  - Miniport drivers are hardware specific but control access to the hardware through a specific bus class driver.

- **Bus drivers** -handles bus-specific I/O operations and service bus controllers such as PCI, USB and FireWire.

- **Filter drivers** are optional drivers. They modify the behaviour of a device and may be non-device drivers. A filter driver can be located at any layer of driver stack above the hardware bus driver.

Network drivers have their own terminology which is detailed in section (D), about NDIS support. In Windows network architecture, the Logical Link Layer (LLC), network and transport layers are implemented by software drivers known as protocol drivers, referred, also, as transport drivers. Depending on type of device and the bus to which it connects, the driver stack includes the appropriate drivers.

WDM has some limitations: it is low/level, and complex, interfaces are not designed to allow versioning, there are too many miniport models, drivers running in kernel mode are treated as part of operating system, leading to possible causes of system crash, and, also, testing of drivers is difficult to perform.

**C. Windows Driver Foundation**

WDF provides an object-oriented, event driven model, and responds to some of difficulties of WDM and other previous models. For example, network-connected devices (Universal Plug and Play compatible) and some USB devices require third-party support for proprietary protocols, and it is preferable that the drivers for these devices to run in user mode.

Objects defined in WDF are modified through well-defined interfaces. They work as building blocks for the driver. Each type of objects can be affected by a set of events. The behaviour for each event is defined by a framework or by specific callback routines that override the defaults.

Windows Driver Foundation includes Kernel-Mode Driver Framework (KMD) and User-Mode Driver Framework (UMDF). Windows Vista supports both WDM and the newer Windows Driver Foundation. KMDF and UMDF are also available for Windows XP. KMDF provides interfaces that are simpler to use than WDM interfaces.

**D. Network Driver Interface Specification (NDIS)**

NDIS (Network Driver Interface Specification) describes the interface by which one or more NIC drivers communicate with one or more underlying network interface cards, with one or more overlying protocol drivers (such as TCP/IP), and with the operating system [7]. The NDIS is a Logical Link Control (LLC) that forms the upper sublayer of the OSI data link layer.

NDIS supports the following types of network drivers [8] (Fig. 13):

- A miniport driver directly manages a network interface card (NIC) and provides an interface to higher-level drivers, such as intermediate drivers and protocol transport drivers.
- An intermediate driver interfaces between upper-level protocol drivers, such as transport driver, and a miniport driver. Intermediate drivers can manage hardware when they are configured as a miniport-intermediate driver.
- An upper-level protocol driver implements a Transport Driver Interface (TDI), or an application-specific interface at its upper edge to provide services to users of the network. At its lower edge, a protocol driver provides a protocol interface to pass packets to and receive incoming packets from the next-lower driver.

For Windows operating systems it has been developed successive NDIS versions. The most important NDIS versions for actual Windows operating systems are NDIS 5.x and NDIS 6.0. NDIS 5.1 is supported by Windows XP, Windows Server 2003 and Windows Embedded CE. NDIS 6.0, supported by Windows Vista and later, includes functionality for Filter Drivers. Filter Drivers filters information on the interface between protocol drivers and miniport drivers. They are, typically transparent for other drivers. The monitoring filters can collect statistics which can be used by upper layer drivers for different actions they take in time.

For our driver implementation, the NDIS objects are very important. They are MIB elements, identified by OIDs of OID_Xxx form. They are managed by NDIS drivers. These objects are organized in subgroups by underlying device types.

NDIS miniport drivers and protocol drivers are bound together through standard NDIS interfaces. Miniport drivers
wrap the details of the hardware implementation of the NIC such as all NIC’s for the same media (Ethernet) can be accessed using a common programming interface. NDIS provides a library of functions (called also a “wrapper”) abstracting the network hardware from network drivers. These functions can be used by MAC drivers as well as higher level protocol drivers (such as TCP/IP). The wrapper functions make development of both MAC and protocol drivers easier as well as to hide some platform dependencies [9]. NDIS also maintains state information and parameters for network drivers.

![Fig. 13. NDIS Drivers Architecture](image)

**E. Miniport and protocol driver details**

NDIS miniports and intermediate drivers communicate with its NIC and with higher-level drivers through NDIS library, which encapsulate all the operating system functions that a miniport driver needs to call. For this purpose the NDIS library exports a full set of functions (NDISXxx functions). As NDIS needs to call miniport functions for its own purposes or for higher-level drivers, it is necessary the miniport driver to export a set of entry points (MiniportXxx functions). Protocol drivers export ProtocolXxx functions. All the drivers include a DriverEntry function. For transmitting a packet, the transport driver calls a NdisXxx function exported by NDIS library. NDIS then passes the packet to the miniport driver by calling the adequate MiniportXxx function exported by miniport driver. The packet is then forwarded, by miniport driver, for transmission to the NIC, by calling the appropriate NdisXxx functions.

When the NIC receives a packet from the network, it triggers a hardware interrupt that is handled by NDIS or the NIC’s miniport driver. NDIS calls the appropriate MiniportXxx function from the miniport driver that it notifies. The miniport drivers indicate the presence of the received packet to bound higher-protocol drivers.

802.11 miniport drivers use NDIS model. They are similar to 802.3 miniport drivers, but in addition, 802.11 miniport drivers support 802.3 emulation. As a consequence, 802.11 miniport driver must support all of the mandatory Ethernet objects. Packets sent or received from driver must be in 802.3 format. For this it is necessary the driver translates the media access control (MAC) header from a format to another in interfacing to NDIS layer. The miniport driver must support all of the mandatory 802.11 WLAN objects and the media-specific status indications for 802.11 devices.

**F. Native 802.11 software architecture**

Native 802.11 is a software architecture for wireless LAN (WLAN) networks which provides a framework for an integrated and extensible set of services and drivers [8]. Native 802.11 is integrated into Windows Vista and later operating systems. This architecture is advantageous because the interface between Native 802.11 framework and underlying miniport driver provides a complete set of object identifiers (OIDs) to query or set media access control (MAC) or PHY attributes on the underlying miniport driver. These OIDs are based on IEEE 802.11 Management information base (MIB) objects. The Native 802.11 framework is notified about a variety of events pertaining to changes in MAC/PHY configuration or basic service set (BSS) network connectivity.

Native 802.11 Wireless LAN driver package includes a miniport driver that uses NDIS 6.x interfaces.

**G. Windows Embedded CE 5.0 Drivers**

Windows Embedded CE supports NDIS 5.1 and includes some NDIS technologies available for this operating system:

- **Wireless WAN support** by Object identifiers that support wireless WAN miniport drivers.
- **Signal strength** – object identifiers that request the signal strength received by NIC and set a threshold for its value.
- **Media Sense** – An interface that monitors media state.

NDISUIO provides an interface between a user-mode application and NDIS using DeviceIoControl. NDISUIO is implemented as a NDIS protocol driver. It can directly establish bindings to Ethernet network controllers, by opening a NDIS miniport driver to send requests, and to set and query NDIS Object Identifiers (OIDs).

Applications load NDISUIO from a DriverEntry routine. The application calls to CreateFile to generate a file handle, then associates the file handle with the target network device by sending IOCTL_NDISUIO_OPEN_DEVICE from DeviceIoControl. The application uses one file handle for each device that it communicates with.

**H. Some final considerations on Interoperability module implementation**

From analysis of NDIS support in Microsoft Windows environment we can conclude that exists the support for implementing Interoperability Module in a layered approach, as intermediate protocol drivers. Figure 14 depicts a possible integration of Interoperability Module in Driver Architecture.
Although exists differences between NDIS5.x and NDIS 6.x support, we have NDIS objects to measure different parameters, and to generate events.

For example, using NDISUIO protocol driver we can obtain NIC information, retrieve NDIS object identifier values, initiate device event notification. This driver uses IOCTL_NDISUIO_OPEN_DEVICE to access to NDIS devices and processes and to receive. A similar process of access NDIS objects can be used by SAP drivers.

Some 802.11 Wireless LAN objects which can be accessed by SAP Drivers (using NDIS 6.x environment) follow:
- OID_802_11_CONFIGURATION - setting this object identifier, we request that the miniport driver set the NIC’s radio configuration parameters to specified values;
- OID_802_11_DISASSOCIATE - commands to NIC to disassociate from the current service set and turn off the radio;
- OID_802_11_RSSI - requests that the miniport driver return the received signal strength indication (RSSI) in response to a query or as a status indication event.

**Fig. 14. Integration of Interoperability module**

VI. ADDITIONAL REMARKS

The IEEE 802.21 standard offers the possibility to have a fast and seamless handover between several wireless access technologies, part of them developed by the cellular industries, part of them results of IEEE 802.xx standardisation system [10, 11]. Unfortunately the implementation of this new MIH standard needs a considerable effort: to adapt the present standards (IEEE, 3GPP or 3GPP2 ones) in order to develop MIH-compatible versions (for example IEEE 802.16m or 802.11VHT). Also, the effort to develop MIH-compatible devices will follow the standardisation effort and all this efforts increase the price of MIH-compatible equipments. In present there are not MIH-compatible equipments and the development of such equipments is very sensitive in finding solutions to make the existing equipments compatible with 802.21 standard requirements at lower costs.

Recently, ITU-R added an IEEE 802.xx wireless access system to IMT-2000’s mix. In October 2007 the ITU decided to include in its IMT family, for the first time, a wireless access system having the standard developed outside the cellular industry. The IMT-2000 / 3G family added the IEEE 802.16e (IEEE 802’s mobile WiMAX standard) as a new member.

In present it is a large effort to develop and to standardize the 4G cellular system, named by some authors as the Advanced-IMT system. LTE is a major candidate, but the experience of IMT-2000 says that several wireless access systems could coexist and cooperate inside the same cellular system. The IEEE 802 standardisation system is considering also to develop a hybrid 802.16m – 802.11VHT system using the MIH standard as convergence element.

VII. CONCLUSION

The present work described the generic design of software interoperability modules for vertical handover in wireless hybrid access networks. The proposed approach conforms to the IEEE 802.21 Media Independent Handover (MIH) standard and fits into our architecture for reconfigurable interoperability of Wireless Communications Systems, RIWCoS, as well the one of Wireless Access System with Unique Addressing, SAWHAU.

In [12] we proposed an incipient solution that uses an intermediate link layer driver (NDIS) developed by Microsoft, to provide independent access to link layer in next-generation multi-access terminals. This first approach is developed and explained in the present paper. The usage of NDIS driver gives the chance to reduce the costs of development of MIH modules especially for terminals, but the terminals are also the most sensitive in term of costs. The next step in our work will be the implementation of detailed media dependent modules, following the proposed approach and based on technologies such as NDIS, and their integration into the overall system.

ACKNOWLEDGEMENT

This work is sponsored by NATO’s Public Diplomacy Division in the framework of “Science for Peace” through the SIP-982469 “Reconfigurable Interoperability of Wireless Communications Systems (RIWCoS)” project and by Romanian Authority of Scientific Research in the framework of PNCD 2 “Partnership” through the 12-126/2008 “Hybrid wireless access system with unique addressing (SAWHAU)” project.

REFERENCES


[4] * * *, SAWHAU proposal, January 2008, PNCDI II Partnership Programme, CNMP, Bucharest, Romania


[6] * * *, “Sample Drivers for the Kernel Mode Driver Framework”, Microsoft, 2006


[8] msdn.microsoft.com


