

Towards a GNU/Linux IEEE 802.21 Implementation

Esa Piri and Kostas Pentikousis
VTT Technical Research Centre of Finland
Kaitoväylä 1, FI-90571 Oulu, Finland
firstname.lastname@vtt.fi

Abstract—Multiaccess mobile devices and overlapping wireless network deployments have emerged as a next generation network fixture. To make the most of all available networks, mobile devices should be capable of handing over between heterogeneous networks seamlessly and automatically. At the same time, operators should be able to steer network attachment based on their criteria. Although several cross layer mechanisms have been proposed in recent years, only the Media Independent Handover (MIH) Services framework has advanced in any of the established standardization bodies. This paper presents a blueprint for a GNU/Linux implementation of IEEE 802.21. We review the salient points of the standard, introduce our software implementation architecture, detail information gathering in GNU/Linux, and show how our prototype implementation can be used in practice. In contrast with prior published work, this paper presents a real IEEE 802.21 implementation, not an abstracted or reduced MIH-like framework, tested and empirically evaluated over real heterogeneous networks.

I. INTRODUCTION

Mobile device hardware currently supports a range of network access technologies; devices often come with integrated personal, local, metropolitan and cellular network interfaces. Once merely referred to as network “terminals,” mobile hosts may now capitalize on the proliferation of several overlapping heterogeneous networks. However, in practice, state-of-the-art devices still lack sophisticated simultaneous multiaccess coordination capabilities. This is partly due to the absence of standards for facilitating seamless media-independent mobility management. To address this need, the IEEE 802.21 working group (see www.ieee802.org/21) has specified an open standard called Media Independent Handover (MIH) Services [1]. IEEE 802.21 provides mechanisms to gather information from various link types and associated networks in a timely and consistent manner, and deliver it to upper layer entities.

Although IEEE 802.21 deployment is anticipated in 2009-2010, there is still no reference implementation available. This paper partially fills this gap by presenting a blueprint for an IEEE 802.21 implementation. We explain how link layer information can be collected in GNU/Linux systems and demonstrate how IEEE 802.21 services can be used by upper layers to decide on handovers and steer traffic and application adaptation. For example, we show how a scalable video receiver can use IEEE 802.21 information to request streaming rate adaptation and prepare for handovers in advance. The focus of this work is to develop methods for gathering information specified in IEEE 802.21 in a generic manner without driver modification or vendor support. We

aim at a GNU/Linux implementation that uses distribution-independent and native kernel properties as much as possible.

To the best of our knowledge this is the first GNU/Linux IEEE 802.21 implementation reported in the open literature. We expect that researchers and practitioners alike will benefit from the material presented in this paper, which is organized as follows. Section II presents IEEE 802.21 and reviews salient previous work. Section III provides a high-level implementation overview and details link information gathering methods. Section IV presents empirical evaluation results and Section V concludes this paper outlining future work items.

II. IEEE 802.21 OVERVIEW AND RELATED WORK

The main design elements of IEEE 802.21 can be classified into three categories: a) a framework for enabling seamless service continuity while handing over between heterogeneous access technologies; b) a set of handover-enabling functions; and c) a set of Service Access Points (SAPs). Service continuity is enabled by gathering all necessary information needed to affiliate with a new point of attachment (PoA) before breaking up the currently used access. The handover-enabling functions are specified with respect to existing network elements in the protocol stack. IEEE 802.21 introduces a new logical entity to the protocol stack, namely the MIH Function (MIHF). The primary role of MIHF is to assist in handovers and handover target decision-making by providing all necessary information to the network selector or mobility management entities, referred to as MIH Users (MIHUs).

Fig. 1 illustrates the IEEE 802.21 general reference model. The MIH_SAP lets MIHUs access three MIHF services. The Media Independent Event Service (MIES) provides event reporting about, for example, dynamic changes in link conditions, status, and quality. Events may originate locally or from (remote) peer MIHFs. The Media Independent Command Service (MICS) enables MIHUs to manage and control parameters related to link operation and handovers. The information obtained via MICS is dynamic in nature. Finally, the Media Independent Information Service (MIIS) caters static information about the characteristics and services of the serving network and networks in range. This information can assist decision-making about handover target accesses. Service management allows for peer-MIHF session configuration.

IEEE 802.21 defines another two SAPs, along with the corresponding primitives, between i) the link layers and MIHF (MIH_LINK_SAP) and ii) MIHF and L2 and L3 transport services (MIH_NET_SAP). LLC_SAP is outside the scope of

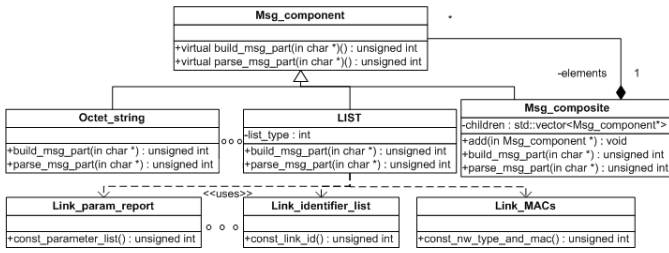


Fig. 4. IEEE 802.21 message construction using Composite Design Pattern

Communication between peer MIHFs occurs at L2 and L3. For L2 communication over IEEE 802.3 and 802.11, PF_PACKET sockets [12] allow us to access network devices directly, thus bypassing the general protocol stack packet handling and avoiding kernel protocol processing. The prototype uses SOCK_RAW sockets which allow complete control over the Ethernet headers [13]; for Wi-Fi, packets are converted into “fake” Ethernet packets by the drivers. Listening to a SOCK_RAW socket by default leads to receiving all packets from the current network broadcast range which, in a large network, leads to significant performance degradation. In order to reduce the amount of processing already at lower layers we employ LPF filters in the PF_PACKET protocol-processing routines. LPF filters are written in the Berkeley packet filter (BPF) language [13]. This enables the MIHF module to efficiently specify the messages it is interested in. In the current version of the prototype we use UDP for L3 communication. IEEE 802.21 also allows the use of media-specific control messaging, such as Wi-Fi beacons. This is not implemented in our prototype as it requires driver modifications. MIIS queries to the information server are executed at L3 using UDP sockets and all MIIS messages are encoded in TLV or XML.

IEEE 802.21 does not prescribe how internal messages need to be encoded, but specifies their high-level content. We opted to stick with the TLV encoding for internal messages, similar with the MIH Protocol messages. This permits us to unify message construction and parsing for both internal and remote IEEE 802.21 messages. However, it is not reasonable to keep the MIH Protocol header in internal messages and we thus have replaced it with a simpler version, which includes only the service and message type, the action related to the service type, and the destination MIHF ID. We used a composite design pattern methodology to construct both internal and MIH Protocol messages in order to effectively deal with their complex structure. Fig. 4 illustrates the simplified class diagram of the message constructors.

A. Link Information Collector

LIC, illustrated in Fig. 5, collects both dynamic and static local link information and controls all local links. Due to space restrictions, we can only detail link information gathering in this paper. Nevertheless, the methods for controlling links are similar with those employed for information gathering.

When a new link is detected by a Static Info Collector (SIC) it initiates a Dynamic Info Collector (DIC) thread

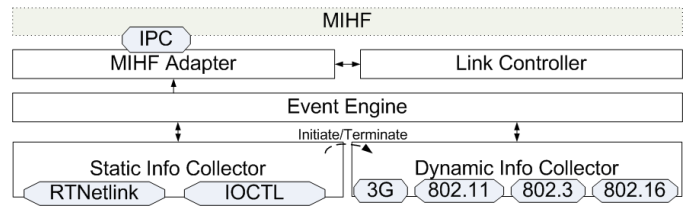


Fig. 5. The Link Information Collector

for gathering dynamic information about the link. Media-independent parameters defined in IEEE 802.21 include data rate, relative signal strength, Signal to Interference plus Noise Ratio (SINR), throughput in a particular period of time, and packet error rate. Moreover, the standard defines some media-specific parameters such as the Wi-Fi Received Signal Strength Indication (RSSI), discussed below. SINR is not provided by the prototype. These parameter values are provided to MIHUs as events in addition to events, such as, Link_Up, Link_Down, and the predictive Link_Going_Down.

Event triggering is handled by the Event Engine module (EE). EE processes both static and dynamic information received from SIC and DIC and generates IEEE 802.21 events. In addition, we opted to let EE monitor the battery state of charge, although this is clearly outside the scope of IEEE 802.21. The logic for predictive events that anticipate the upcoming break-down in link connectivity in the near future is based on constant monitoring of the link parameters and battery capacity. Since there may be short-term peaks, for example, in signal strength, EE uses a low pass filter to monitor trends during the previous five seconds. Melia *et al.* [14] considered the use of more complex algorithms, which use weighted coefficients for previous values in longer-term parameter calculation and argued that their use could increase the accuracy and reliability of predictive events. We plan to evaluate such algorithms in practice using our prototype implementation in different scenarios. Further, our unified information gathering methods revealed differences in signal strength values when testing adapters from different vendors. Static LIC configuration led to connection breakdowns rendering predictive events very inaccurate. Thus, we suggest the adoption of adaptive event triggering in LIC which updates its thresholds for predictive events based on the operation of the currently used network card.

RTnetlink [15] provides an effective way to gather and control kernel media-independent network information. RTnetlink is a subset of Netlink [16] which provides for full-duplex communication between kernel modules and user-space processes through PF_NETLINK sockets [12]. The information provided by RTnetlink can be classified into link layer interface settings, network layer interface settings, network layer routing tables and rules, neighbor cache, queuing settings, traffic classes, and traffic filters. Via RTnetlink, LIC can *get*, *set*, or *delete* all these information elements. RTnetlink enables joining various multicast groups for getting notifications about changes in states and parameters from all information classes mentioned

above. This reduces the overall implementation workload considerably, spares user-space applications from having to constantly poll the kernel for new information, and allows for an elegant MIHF implementation.

B. IEEE 802.3

Ethernet provides the bare minimum information about link conditions. Besides the RTnetlink information, other necessary information is the maximum attainable capacity and the current capacity. This information can be obtained via ETHTOOL [17], which is a well defined API for determining the status of an Ethernet link. There exists only a single *ioctl* command, SIOCETHTOOL, but several subcommands expand its functionality. Useful subcommands include ETHTOOL_GSET for getting the currently supported bitrate and ETHTOOL_GWOL for getting the maximum supported bitrate.

C. IEEE 802.11

The Linux Wireless Extensions API (WE) [18] is a generic interface to obtain statistics and configurations of a Wi-Fi link. WE methods are based on a set of *ioctl* calls and the */proc* file system. WE provides both dynamic and static information about the Wi-Fi card and the associated link condition. A straightforward way to identify a network interface as a Wi-Fi one is to test if WE exists for that particular interface.

The centralized entry */proc/net/wireless* provides wireless specific dynamic statistics from the driver that can be classified into two categories: quality and discarded packets. This entry is used indirectly via WE. Link quality statistics include link quality, level, and noise. Link level, indicating the current signal strength, and link noise level are measured at the receiver and are given either in dBm or in relative values. According to [19], the one byte long signal strength value indicating the received RF energy is defined as RSSI. Unfortunately, we found that, in practice, the signal noise value was unrealistic for some tested Wi-Fi cards. Thus, the RSSI value given as a proportional value to the signal noise can be meaningless in practice for an IEEE 802.21 implementation. Note that IEEE 802.21 does not define whether the RSSI value should be given in dBm, dB, or as a proportional value to the radio signal strength.

The link quality metric, also briefly defined in [19], indicates how good the received signal is. According to WE, it is a combination of different measurements, such as percentage of retries, signal-to-noise ratio (SNR), and missed beacons. Our hands-on experience with various Wi-Fi cards indicates that the implementation of this parameter varies widely between different drivers. The link quality metric does provide a good additional indication about the link quality to WE with some cards but our test measurements showed that RSSI provides more accurate and reliable event triggering thresholds.

D. IEEE 802.16

Despite WiMAX Forum (see www.wimaxforum.org) efforts, implementations of WiMAX equipment still differ significantly from each other. In particular, there is no standardized way to gather information about WiMAX interfaces in

GNU/Linux. Our solution was to employ a separate WiMAX adapter, presented in [20], which reduces the complexity of LIC considerably. The adapter interfaces the Convergence Layer with the technology-dependent layers of the WiMAX Access Networks (DLC/MAC). The adapter's primary role is to facilitate information gathering from fixed WiMAX (based on IEEE 802.16-2004) using SNMP queries. It comprises two parts: the Generic Adapter (GA) and Vendor Specific Adapters (VSAs). GA provides a generic interface for LIC to collect link information from WiMAX equipment of different vendors. GA converts the information query to SNMP query according to OID conversion table (based on the queried vendor's SNMP MIB) and directs the query to VSA. VSA executes the actual information query on the equipment. Mobile WiMAX (based on IEEE 802.16e-2005) has recently emerged as a potent telecommunications offering. Unfortunately, there are currently no publicly available GNU/Linux drivers for mobile WiMAX adapter cards, and thus a Dynamic Information Collector for mobile WiMAX is left for future work.

E. 3GPP

The Dynamic Information Collector for 3GPP collects transmission statistics via RTnetlink as 3G connections with PCMCIA cards are established in Linux using the point-to-point protocol (PPP). For media-specific information, AT commands provide a straightforward way to retrieve information about the cellular link directly from the adapter card. The most useful AT commands are +COPS? and +CSQ [21].

The Public Land Mobile Network (PLMN) selection read command COPS? returns, among others, the currently employed access technology. According to the 3GPP specification [21], the last digit of the received response message conveys the current access technology as follows: 0 \equiv GSM/GPRS; 1 \equiv GSM Compact; 2 \equiv UTRAN; 3 \equiv EGPRS; 4 \equiv HSDPA; 5 \equiv HSUPA; 6 \equiv HSDPA and HSUPA (also called HSPA).

The signal quality command +CSQ [21] returns the relative RSSI and channel Bit Error Rate (BER). RSSI values are on a scale of 0-31 and 99, whereas BER return values (RXQUAL) are in the range 0-7 and 99. The RSSI value can be converted to dBm according to [21]. 3GPP [22] defines the conversion table for RXQUAL values to actual BER percentage values. Unfortunately, none of the cards we tested supports BER.

AT commands are sent to 3G cards through a serial port. The serial port connection address depends on the driver and the card used. Because of variety of cards, vendors, and different attachment techniques the device location and serial port name is easiest to determine by going through the kernel messages.

IV. IEEE 802.21 IN PRACTICE

The main aim of IEEE 802.21 is to facilitate heterogeneous handovers. Still, it can also be used to adapt network applications according to varying link conditions and characteristics of the serving network. For example, we used the prototype implementation to control Skype sessions and adapt synthetically generated video and VoIP streams based on the IEEE 802.21 information. The Skype API (see developer.skype.com)

defines a way for a third party software to steer Skype behavior such as, for example, to change user status and initiate voice calls, and get timely information about Skype events such as call and chat attempts. We explored a scenario where voice calls were restricted according to IEEE 802.21 information about link conditions and network characteristics. For instance, using the cost information obtained from the information server via MIIS, we allow the Skype client to restrict VoIP calls and reply to call attempts via instant messaging according to internal policy rules, saving bandwidth. This is automated and carried out without user intervention.

Delay savings when adapting a 1 Mb/s video stream over a Wi-Fi link based on IEEE 802.21 services were also demonstrated. Even if no handover was performed, a tolerable video quality could be sustained in the weakening link by only adapting the bitrate. For VoIP, we experimented with emulated voice sample aggregation and ROHC as means to improve VoIP quality. These methods were triggered based on IEEE 802.21 events. For example, bundling two 24-byte voice samples into one VoIP packet can reduce total throughput (including RTP/UDP/IPv4 headers) of a stream sending 32 packets/s by 31%. ROHC improves this rate even further by limiting the overhead introduced by RTP, UDP, and IP headers. Interested readers can refer to [23] for further details on the benefits of ROHC and VoIP aggregation over fixed WiMAX and Wi-Fi links.

V. CONCLUSION AND FUTURE WORK

This paper presents a blueprint for an IEEE 802.21 implementation. We introduced the main building blocks of our implementation, namely the MIHF and LIC modules, which gather and disseminate link and network information to handover decision makers, system components and applications. We showed how one can gather link information in a unified manner in various Linux distributions, without driver modifications, and from various network adapters by different vendors. We find that the information obtained from network adapters and drivers by different vendors can vary considerably; delivering information in a consistent manner requires careful and knowledgeable implementation. We also evaluated the benefits for application adaptation based on the information from an IEEE 802.21 implementation. We showed that IEEE 802.21 services may bring significant advantages in areas other than heterogeneous handovers.

Our future work agenda includes an evaluation of different algorithms for triggering Link_Going_Down events. We are also considering driver modifications and porting of the MIHF and LIC implementations to kernel space. The increased accuracy and efficiency, however, need to be traded off with loss in the generality and portability of our implementation. Finally, we are currently working on networked multimedia adaptation with the assistance of IEEE 802.21 services.

ACKNOWLEDGEMENT

This work has been supported by the IST FP6 Integrated Project WEIRD (IST-034622-IP) and the ICT FP7 Integrated

Project OPTIMIX (grant agreement no. 214625), which were partially funded by the Commission of the European Union.

REFERENCES

- [1] IEEE 802.21 WG. *IEEE Standard for Local and Metropolitan Area Networks. Part 21: Media Independent Handover Services*. IEEE Std 802.21-2008, January 2009.
- [2] A. Dutta, S.r Das, D. Famolari, Y. Ohba, K. Taniuchi, T. Kodama, and H. Schulzrinne. Seamless Handover across Heterogeneous Networks - An IEEE 802.21 Centric Approach. In *Proc. WPMC*, Aalborg, Denmark, September 2005.
- [3] P. Machań, S. Serwin, and J. Woźniak. Performance of Mobility Support Mechanisms in a Heterogeneous UMTS and IEEE 802.11 Network Offered under the IEEE 802.21 Standard. In *Proc. IT*, Gdansk, Poland, May 2008.
- [4] J. Mäkelä and K. Pentikousis. Trigger management mechanisms. In *Proc. ISWPC*, pages 378–383, San Juan, Puerto Rico, February 2007.
- [5] T. Ali-Yahiya, K. Sethom, and Guy Pujolle. A Case Study: IEEE 802.21 Framework Design for Service Continuity across WLAN and WMAN. In *Proc. WOCN*, pages 1–5, Grand Hyatt, Singapore, July 2007.
- [6] M. Li, K. Sandrasegaran, and T. Tung. A Multi-Interface Proposal for IEEE 802.21 Media Independent Handover. In *Proc. ICMB*, Toronto, Canada, July 2007.
- [7] U. Toseef, A. Udugama, C. Goerg, V. Pangboonyanon, and F. Pittmann. LINE: Link Information Normalization Environment. In *Proc. Mobilware*, Innsbruck, Austria, February 2008.
- [8] R. Giaffreda, K. Pentikousis, E. Hepworth, R. Agro, and A. Galis. An information service infrastructure for Ambient Networks. In *Proc. PDCN*, pages 21–27, Innsbruck, Austria, February 2007.
- [9] F. Cacace and L. Voller. Managing Mobility and Adaptation in Upcoming 802.21 Enabled Devices. In *Proc. WMASH*, pages 1–10, Los Angeles, CA, USA, September 2006.
- [10] 3GPP. 3GPP TS 23.234: 3GPP system to Wireless Local Area Network (WLAN) interworking; System description (Release 7). Technical Report V7.7.0, 3rd Generation Partnership Project, June 2008.
- [11] J. Fusco. *The Linux Programmer's Toolbox*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2007.
- [12] K. Wehrle, F. Pählke, H. Ritter, Daniel Muller, and Marc Bechler. *The Linux Networking Architecture, Design and Implementation of Network Protocols in the Linux Kernel*. Prentice Hall, Upper Saddle River, NJ, USA, 2005.
- [13] G. Insolubile. The Linux Socket Filter: Sniffing Bytes over the Network. *Linux Journal*, 86, June 2001.
- [14] T. Melia, A. de la Olivia, I. Soto, C. J. Bernardos, and A. Vidal. Analysis of the Effect of Mobile Terminal Speed on WLAN/3G Vertical Handovers. In *Proc. GLOBECOM*, pages 1–6, San Francisco, CA, USA, November 2006.
- [15] A. Udugama. Manipulating the Networking Environment Using RT-NETLINK. *Linux Journal*, 145, May 2006.
- [16] K. K. He. Why and How to Use Netlink Socket. *Linux Journal*, 130, February 2005.
- [17] J. Corbet, A. Rubini, and G. Kroah-Hartman. *Linux Device Drivers*. O'Reilly Media, Inc, Sebastopol, CA, USA, third edition, 2005.
- [18] J. Tourrilhes. Wireless Extensions for Linux. URL: http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Linux.Wireless.Extensions.html.
- [19] IEEE 802.11g Working Group, editor. *IEEE Standard for Local and Metropolitan Area Networks. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. Amendment 4: Further Higher Data Rate Extension in the 2.4 GHz Band*. IEEE Std. 802.11g-2003, June 2003.
- [20] P. Neves, T. Nissilä, T. Pereira, I. Harjula, J. Monteiro, K. Pentikousis, S. Sargento, and F. Fontes. A vendor-independent resource control framework for WiMAX. In *Proc. ISCC*, pages 899–906, Marrakech, Morocco, July 2008.
- [21] 3GPP. 3GPP TS 27.007: Technical Specification Group Core Network and Terminals; AT command set for User Equipment (UE) (Release 8). Technical Report V8.0.0, 3rd Generation Partnership Project, June 2007.
- [22] 3GPP. 3GPP TS 45.008: Technical Specification Group GSM/EDGE Radio Access Network; Radio subsystem link control (Release 7). Technical Report V7.8.0, 3rd Generation Partnership Project, May 2007.
- [23] E.Piri, J. Pinola, F. Fitzek, and K. Pentikousis. ROHC and Aggregated VoIP over Fixed WiMAX: An Empirical Evaluation. In *Proc. ISCC*, pages 1141–1146, Marrakech, Morocco, July 2008.