FACTA UNIVERSITATIS
Ser: **Elec. Energ.** Vol. 25, N° 1, April 2012, pp. 43 - 58
DOI: 10.2298/FUEE1201043K

RELIABLE ARCHITECTURE FOR HETEROGENEOUS HOME-NETWORKS: THE OMEGA I-MAC APPROACH *

Rolf Kraemer[1], Marcin Brzozowski[1], Stefan Nowak[2]

[1]Members of the Wireless Systems Design Department of the IHP, Frankfurt (Oder), Germany
[2]Faculty of Electrical Engineering at the Chair of Communication Technology,
Technical University of Dortmund, Germany

Abstract. *Home networks are becoming more and more popular. Today's state of the art is that several different technologies, like Wireless LAN (WLAN), Power Line Communication (PLC), or Ethernet, are being used concurrently to connect home devices. If any connection fault happens, the communication stops and new connectivity has to be established. Since the fault detection can last several seconds, it reduces the "Quality of Experience" dramatically, and ongoing transmissions can be disturbed. Therefore, reliability of transmission is a necessary precondition to fulfil customers' expectations. In our approach, we suggest an additional protocol layer, dubbed layer 2.5, which manages any available connectivity and automatically chooses a new connection with the correct properties, for instance HDTV stream. It balances load situations and can also be used to intelligently distribute traffic between nodes. The system has been proven to work in standard LINUX kernel implementation with a speed up to 1 Gb/s and with an extreme low latency. The topology control and signalling components have been implemented in LINUX user space and work on best effort bases. Within this paper, we outline the architectural considerations and show the initial results. The work has been used as the starting point for a new IEEE standardization, i.e., IEEE1905.1. This group started with the OMEGA I-MAC architecture, introduced here. The work described here has received funding from the European Commission's Seventh Framework Programme FP7/2007-2013 under grant agreement no. 213311, also referred to as OMEGA.*

Key words: *home networks, reliable networks, heterogeneous networks, I-MAC, IEEE1905.1, OMEGA*

Received April 15, 2012

Corresponding author: Rolf Kraemer
Members of the wireless systems design department of the IHP, Im Technologiepark 25, 15236 Frankfurt (Oder), Germany • E-mail: kraemer@ihp-microelectronics.com

* **Acknowledgement**. The authors would like to thank the whole OMEGA research team for the good co-operation during the challenging FP7 project. We would also like to express our gratitude to the European Commission that supported the work with funding under grant agreement n0.213311 also referred to as OMEGA.

1. Introduction

Several concepts and architectures are currently being developed for home network applications. These networks are all addressing the service provision from sources, such as video servers, to sinks, such as television-displays or monitors. However, most nodes in the network have more than one connection port, for instance, WLAN and Ethernet. The user has to set-up his topology appropriately to allow the best connectivity for each service to be used. Each port has its own IP address, and if a service or connection fails, a new connection has to be established manually to continue the interrupted service. This happens only after the TCP or UDP session fault has been discovered. Moreover, it is often not clear how the required properties of a communication, like the maximum delay-jitter or the maximum packet latency, can be achieved via any of the available network connections. To address this issue, that is, enabling uninterrupted services with appropriate QoS, we introduced a layer above LLC and below IP, referred to as I-MAC. This new layer deals with all heterogeneity, reliability and QoS in home-area network (HAN) applications. That is, I-MAC determines multi-hop paths with heterogeneous technologies from sources to sinks, e.g. from a film server to TV-sets. To fulfil certain QoS requirements, such as short network delays required by HDTV (High-definition television) streams, I-MAC examines the quality of single links leading to the TV-set. In this case, I-MAC selects links with delays short enough to guarantee a high quality of HDTV streams. In case of connection problems, I-MAC discovers quickly an alternative path, switches seamlessly the network traffic (handover), possibly to other technology. As a result, users do not notice connectivity problems and continue using services, like watching films. Some previous works [1-4] introduced briefly the I-MAC layer.

The idea of media-independent routing and handover is also in the scope of IEEE standards 802.21 [5, 6]. It relates, however, to wireless access technologies only, and does not define the handover mechanism or the selection of networking technologies. Moreover, although meshed network topologies are predominant in HANs, they are not considered in IEEE 802.21, since the main idea of this standard is "vertical handover" in public networks, i.e., between LTE, and IEEE802.16 or IEEE802.11x. Since too many issues are not addressed, the use of IEEE802.21 is not appropriate for typical HANs, and therefore we introduced I-MAC.

The rest of this paper is organized as follows. Section II describes background of selecting the IEEE802.2 as the appropriate interface. In Section III the OMEGA I-MAC architecture is described. This section outlines also the function of the three planes: the data plane, the signalling plane, and the management plane. In Section IV we discuss the implementation, and in Section V present the measurements results of our prototype system. Section VI introduces our primary demonstration platform. Finally, we conclude the paper in Section VII.

2. Choosing the Correct Layer for HAN Extensions

After discussing the pitfalls of current HANs, we aimed to optimize the QoE (Quality of Experience), and it led to several options. Mainly we wanted the system to find automatically a way-out in case of communication problems, and to continue the service. Moreover, legacy devices needed to be supported seamlessly, even if they did not support the extensions of the new architecture. To keep the cost of the systems as low as possible all currently available applications should be supported without any change. This request addressed especially UPnP based systems in homes. The QoS support of this approach sends discovery messages into the network to determine the best connectivity [7, 8]. Since these messages are implicitly embedded into the IP message flow, we have to be able to recognize them and react appropriately. Besides, we wanted to be able to accept also future physical layers like PLC, free space optical communications, or 60 GHz based communications according to IEEE802.11ad and IEEE802.15.3c. All these new ideas fit into the IEEE802 concept. In **Fig. 1** the TCP/IP stack is outlined. The two lower layers address the physical layer and the associated MAC layers. The third layer, the LLC (Logical Link Control) is a layer of common functionality like flow control, ARQ (automatic-repeat-request) etc. All IP based services have to support the interface according to IEEE802.2 between LLC and the higher layers. Therefore, if we define the OMEGA interfaces to be conformant to 802.2, all current applications can be supported completely unchanged.

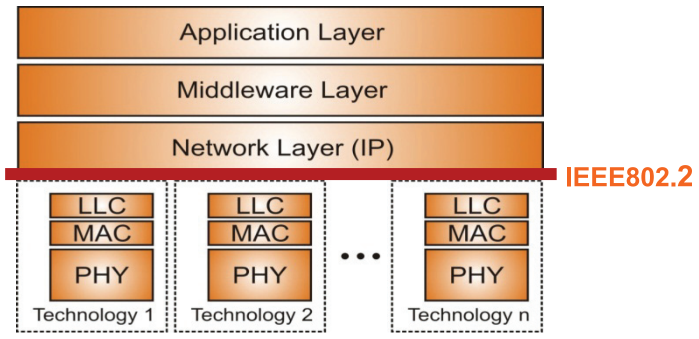

Fig. 1. TCP/IP stack according to IEEE 802.2

Moreover all IEEE802 conformant technologies do already support IEEE802.2 to allow for layer-2 bridging in heterogeneous networks. So e.g. are common standard bridge like a WLAN access point uses this approach by bridging between Ethernet and WLAN using an LLC relay function.

IEEE802.2 therefore seems to be the ideal layer to fulfil the mentioned requirements. It is today's most used interface, since almost all networks are IEEE802.2 conformant.

Unfortunately, we need more functionality than the current LLC layer supports. Therefore, we introduced a new layer above LLC and below IP that support IEEE802.2 interfaces to both sides. By this little trick we can introduce new functionality into a completely defined stack without violating electrical or logical interfaces. However, we have to assure that the expected behaviour from both lower and higher layers is fulfilled, and that additional functions do not intercept the message flow in an illegal form.

The introduction of the additional layer has a couple of consequences that have to be explained. First, we can consider all layers below IEEE802.2 as a common technology layer independent of what it really contains. So we can introduce a virtual MAC address that represents all communication interfaces to the higher layers making the concrete physical interface transparent for IP streams. This also allows us to have only one IP address for the whole set of potentially available physical communication interfaces.

Second, the IP layer does not longer know anything about the properties of the lower layers. It assumes that a message sent to any MAC address will reach its destination within this sub-network. Consequently, we can support any kind of topologies within the sub-layers, we can do any kind of routing, relaying, hand-over, as long as we fulfil the interface requirements of 802.2. Third, we can now redefine the HAN to be a heterogeneous network that supports QoS, reliability and connectivity for any IEEE802 conformant legacy technologies. It is transparent for any IP based application traffic.
The new layer introduced can realize any kind of topologies like star- and meshed-networks. Especially meshed networks are of particular interest, since they use the multi connectivity to allow multiple paths through the network from a source to a destination. This in turn allows increasing the reliability of a network significantly.

3. THE OMEGA I-MAC ARCHITECTURE

Having described the interface considerations, we will now describe the architecture that performs the additional layer 2.5 functionality. Fig. 2 shows the I-MAC architecture.

The I-MAC architecture consists of three independent planes:

- The Data Plane
- The Control Plane, named also the Signalling Plane
- The Management Plane

The data plane performs the transport of all data flows. It accepts data from each technology port and relays it to another port, or it terminates the flow by handing the packets to the higher layer. Moreover, the data plane provides statistics about the individual packet flows to allow the QoS management. In case of link failures, link overload etc., the data plane informs the control plane to initiate measures for fault correction, re-routing, or load balancing. To insure the security within the network, the data plane encrypts and decrypts packet stream. Since the plane supports end-to-end security, data flows are not encrypted or decrypted on intermediary nodes. Additionally, the data plane provides authentication to avoid the participation of un-authorized nodes. If legacy devices are connected to the OMEGA network, they have to connect via a device port that provides the necessary authentication for traffic flows. The switching of data packets is performed by the forwarding engine based on information stored in the forwarding table. The forwarding engine itself is controlled by the control plane.

The control plane organizes the complete OMEGA network topology. It controls the admission of flows, determines routes through the OMEGA network based on QoS demands of the individual streams. Moreover, all topology changes are managed by the control plane. To this end a monitoring engine keeps information about the available neighbour connectivity, the performance and load of each connection.

Within the I-MAC terminology all technology dependent ports of I-MAC are called south-interfaces, whereas the connection to the higher layer port is called the north-interface. South interface protocols are MAC protocols. North interface protocols are IP protocols. This reflexes the interaction over the IEEE802.2 interface to lower and upper layers.

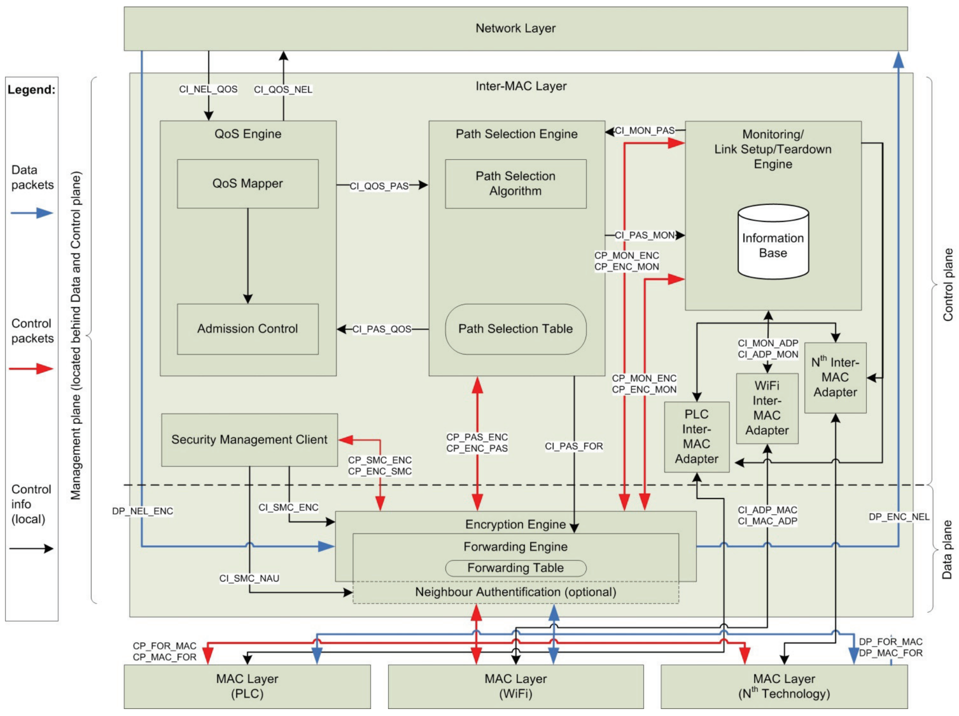

Fig. 2. The OMEGA I-MAC architecture

The core element of the control plane is the path-selection engine. This object takes care of the management of all communication flows. If a new connection has to be established, the path selection engine checks the availability of physical links to the destination, calculates optimal performance and load conditions based on the flows demand. Moreover, the path selection engine communicates with the path selection engine of the destination to set up end-to-end flow. To determine optimal routes, the path selection engine consults the monitoring engine to give information about the available capacity of each physical connection, and to evaluate the performance change if the newly requested flow should be granted. In case of performance degradation, the path selection engine has to re-arrange the flows to achieve the admitted QoS values for each flow. Thus, the degradation of a single data-flow leads not only to a re-routing of this flow, but might also affect the arrangement of other flows. The main data structure of the path selection engine is the path selection table. This table contains all end-to-end connections for each flow together with the granted QoS parameters.

The monitoring engine observes the network traffic, and on performance degradation it signals the path selection engine. The monitoring engine interacts with technology dependent MAC adapters to maintain QoS statistics about underlying physical connections. The MAC adaptors represent a technology port within the I-MAC. Their main task is to convert technology-specific QoS representation, such as channel load in WLAN, to the common QoS parameters of the OMEGA network. The main data structure of the monitoring engine is the information base, which includes the current communication flows, their QoS parameters, and statistics about the QoS trends within the network. For example, a proactive change of a path can be triggered, if a QoS trend shows that the QoS threshold will be reached because of systematic quality degradation. This can happen if an OMEGA device is connected by visual light communication.

The QoS engine represents the interface to the network layer. For example, an end device, like a set-top-box, initiates a new flow to a film server. This request is signalled to the QoS engine. To setup this flow, the I-MAC checks if enough capacity is available, and whether the new flow would not harm ongoing flows. The OMEGA I-MAC supports different priorities such that a newly requested flow can lead to the stop of another, less important flow, if not enough capacity is available. The new flow has to be explicitly admitted. This also includes a security check. Legacy devices are therefore always connected via this interface. This signalling interface is not part of IEEE802.2. Therefore, only OMEGA devices can use the QoS interface of the I-MAC. To allow interoperation with legacy technologies, such as UPnP, an additional sniffer has to be integrated into the QoS engine. The sniffer intercepts UPnP flows, understands signalling messages, and extracts QoS information. This information is used to perform the invocation of the QoS interfaces and achieve the same effect as explicit call via the QoS interface messages. If any other mechanism for implicit QoS negotiation on higher layers is introduced, an appropriate I-MAC application adapter has to be provided too. Otherwise, technologies without adapters will be handled with the best-effort QoS.

The management plane of the OMEGA I-MAC architecture provides information about the topology of the network, flow based QoS information, fault conditions etc. All management information is directly retrieved from the main data structures of both the data- and control planes. This information is mainly used to visualize the network for management purposes. The management plane provides a management interface to the outside world. Within the project this interface has been used to instrument the demonstrator and to show how the systems worked. Topology profiles can be loaded into the system to pre-configure nodes and interfaces. In future applications the management plane can be used for remote maintenance via e.g. TR069 protocols.

4. IMPLEMENTATION

Within the OMEGA project several implementations of the I-MAC have been accomplished. That is, the I-MAC runs on various hardware platforms, such as PCs with Linux operating system or on FPGA boards with PowerPC processors. To avoid complex re-programing, a clear implementation flow has been introduced, which allowed the utmost re-use of implemented code, dubbed cross-platform. By doing so, each hardware platform provides only an adapter and executes already available I-MAC functionality.

Fig. 3 depicts the I-MAC implementation structure. As introduced in the previous section, we divided the I-MAC into two major blocks: Data Plane (DP) and Control Plane (CP). The main functionality of both blocks was implemented as a cross-platform code in ANSI C. That is, it includes only plain C instructions, without any hardware-specific code.

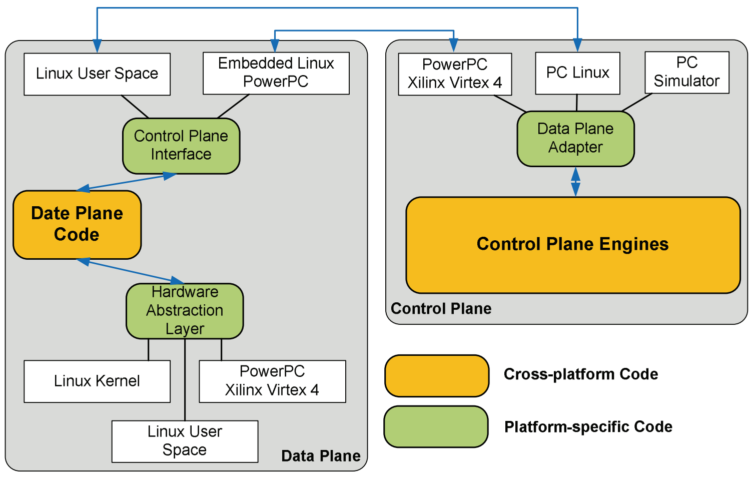

Fig. 3. Cross-platform implementation of the I-MAC

To provide flexibility, we did not couple CP with DP, but implemented them as separate applications. Therefore, certain CP instances work with various DP versions, and vice versa. For example, the CP running in Linux user-space works with two DP versions: in user-space and in kernel. We defined only a generic interface between DP and CP (see *Control Plane Interface* and *Data Plane Adapter* in Fig. 3), and each instance of DP and CP provides the interface implementation.

As previously mentioned, we implemented a few versions of the I-MAC during the project. In the following we present briefly each of them, that is:

1. PC simulator
2. FPGA with PowerPC
3. PC with Linux.

4.1. PC simulator

To allow efficient development of CP engines, such as routing protocols of the Path Selection Engine, we provided the PC Simulator version of the I-MAC. In this case, developers start several instances of CP on a single computer (see Fig. 4), define the network topology, and test their protocols. Since several I-MAC nodes are simulated on a single computer, it speeds up the debugging process, especially when using common debugging tools, like GNU Debugger.

Although our approach eases the I-MAC development, it also has its limitations. We implemented only a simple DP emulation, i.e., frames transmitted by a sender are immediately received, and there are no packet losses. To simulate link problems, developers must implement extra functionality on their own. Nonetheless, the PC simulator speeded up the development process significantly, as the main functionally was tested with it, and developers fixed major software errors. Only when their software ran without problems on the simulator, did they start tests on real hardware platforms.

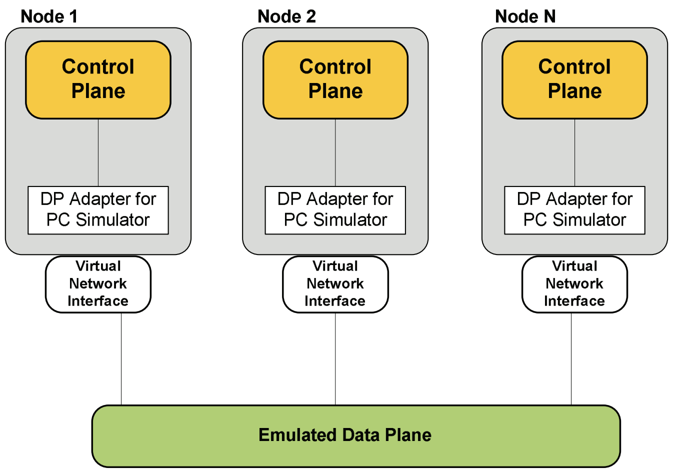
Node 1
Node 2
Node N
Control Plane
Control Plane
Control Plane
DP Adapter for PC Simulator
DP Adapter for PC Simulator
DP Adapter for PC Simulator
Virtual Network Interface
Virtual Network Interface
Virtual Network Interface
Emulated Data Plane

Fig. 4. Several I-MAC instances running on PC Simulator

4.2. FPGA with PowerPC

This I-MAC version runs on FPGA boards based on Xilinx Virtex 4. However, this version is still a software solution, and not the I-MAC running as a hardware approach (VHDL or Verilog). The rationale behind this was to find bottlenecks of the I-MAC, and then implement part of the I-MAC as hardware blocks. Such an approach supports flexibility, as most of the I-MAC functionality runs as software, and only high-speed task are realized in hardware.

During the project, we provided only the software version of the I-MAC, as with the Linux I-MAC version, introduced in the next paragraph, we achieved our goal: support of Gb/s data rate. Nonetheless, we will focus on hardware accelerators in future research to achieve a speed beyond Gb/s.

Fig. 5 illustrates the I-MAC architecture running as software on FPGA boards. There are two PowerPC processors synthesized in this FPGA. One of them works with an embedded Linux operating system and runs CP. Another PowerPC executes DP as a standalone application. We realized communication between CP and DP with a shared memory:

1. Common CP-DP communication is performed with message queues. That is, CP writes a command to the queue, DP reads it and carries out a specific task. The communication in the reverse direction, from DP to CP, works in a similar way.
2. To ensure consistency of the Forwarding Table, only DP has the write access. If the CP needs to update the table, it issues a command to the DP by writing it to the message queue. On the reception, DP updates the table.

The FPGA boards include six Ethernet ports: two with GEth and four with 100 Mbps speed. We configured one GEth as an north interface. I.e., it received traffic from higher layers (IP), or from devices that do not support I-MAC, like off-the-shelf set-top-boxes. Other interfaces were connected to the OMEGA network.

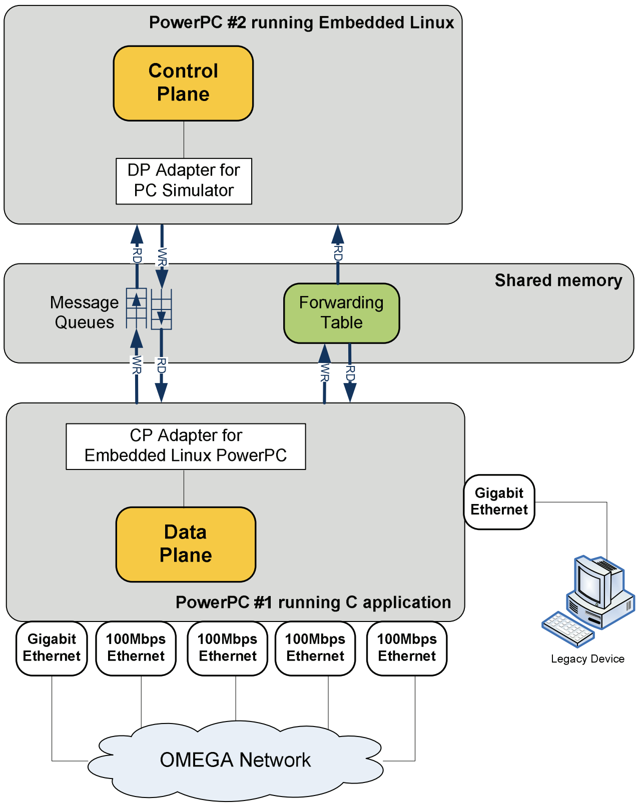

Fig. 5. I-MAC running on FPGA Xilinx Virtex 4 with PowerPC processors

4.3. PC with Linux

We ported the I-MAC to the Linux platform, although we did not plan it at the beginning of the project. Since we did not expect the I-MAC software version to achieve a Gb/s forwarding speed, we intended to use FPGAs as our primary hardware platform and provide hardware accelerators. However, it turned out that the I-MAC based on software achieve a forwarding speed of about 200 Mbps on FPGA, with PowerPC processors and without extra hardware accelerators. Since off-the-shelf computers include processors a few times faster than the PowerPC, we hoped they could increase the forwarding speed to Gb/s. In addition, with the Linux version, developers did not have to buy dedicated FPGA boards, and could run the I-MAC on common computers. Finally, the I-MAC based on Linux achieved Gb/s forwarding speed, and became our primary demonstration platform.

The CP part of the I-MAC runs completely in the Linux user-space, as it does not carry out time-critical tasks. We provided two version of the DP: kernel and user-space. The former supports a forwarding speed of Gb/s, and we used it in our demonstration platform. The latter is mainly used to ease the debugging process, since it is more efficient to find software errors in user-space programs than in the Linux kernel. In this section, however, we do not introduce the user-space version of the DP, since it is not part of the final I-MAC solution.

4.3.1. Communication between control plane and data plane

The communication between the CP and the DP is realized with the common Linux socket interface in two components: *DataPlane Adaptor* and *ControlPlane Interface* (see Fig. 6). They provide both asynchronous and synchronous communication types. In the former case, the sender issues a command, such as triggering the DP to transmit an I-MAC frame, and does not care about the result. However, with the synchronous communication, the sender waits for a receiver's reply. For example, the CP asks the DP to provide a certain entry from the Forwarding Table.

4.3.2. Forwarding table

Each time the DP forwards frames, it looks up the Forwarding Table to find the outgoing interface and the receiver's address. To speed up the table look ups, the Forwarding Table is located in the memory of the kernel space. Therefore, the CP cannot access it directly but sends read commands to the DP, the DP reads the Forwarding Table and sends back a reply.

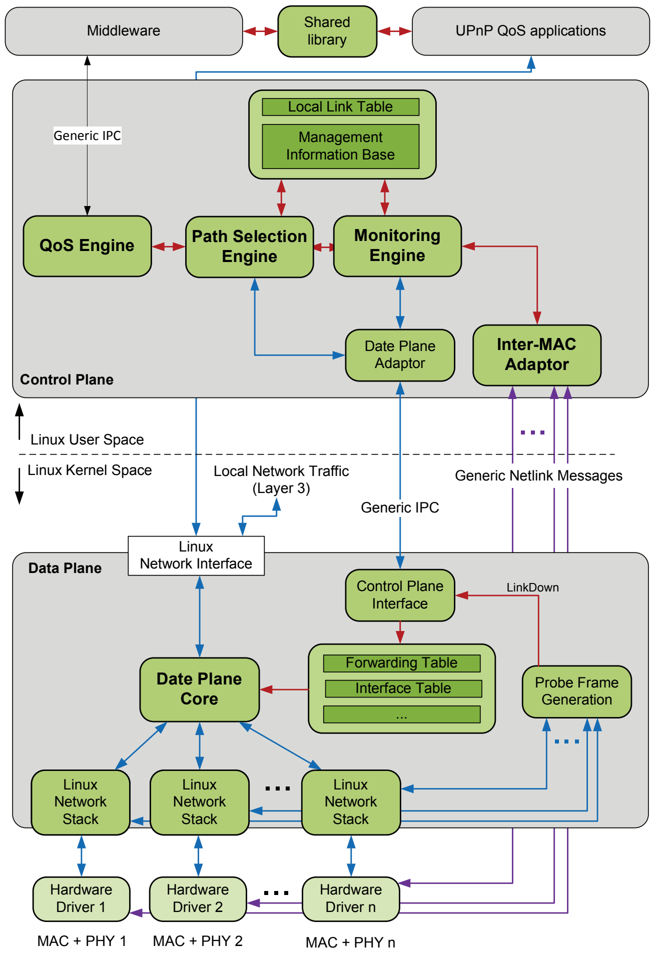

Fig. 6. I-MAC implementation based on Linux

4.3.3. Probe frames

The Monitoring Engine (ME), a part of the CP, estimates link statistics and discovers broken links with probe frames. However, if the ME generates probe frames on its own, passes them to the DP for transmission, it will result in a significant processing overhead. Therefore, we implemented the Probe Frame Generator in the DP. This component provides various link statistics, like current throughput on the interface basis, to the CP, and triggers the ME when a broken link is detected.

4.3.4. Inter-MAC adaptor

The main task of Inter-MAC Adaptors is to provide technology-specific statistics, which cannot be obtained with probe frames. For the time being, we implemented such an adaptor for the WLAN interface. The adaptor reads RSSI (Received Signal Strength Indication) value with Netlink messages from the WLAN driver, and this value is used in the CP to perform a handover to a better Access Point, if available.

4.3.5. Frame reception

I-MAC frames are encapsulated into Ethernet frames with the *EtherType* field set to 0x9999. Therefore, the DP registers a handler for receiving I-MAC frames. In other words, if the kernel received an Ethernet frame of 0x9999 type, it passes the frame to the DP handler. However, some DP interfaces may support devices that do not include the I-MAC functionality, e.g., off-the-shelf network devices like set-top-boxes. These devices send ordinary Ethernet frames, the DP adds the I-MAC header and forwards to the OMEGA network. In this case, the DP must receive not only I-MAC frames, but Ethernet frames of any kind. For such interfaces, the DP registers another RX handler.

Common network interface drivers raise the RX interrupt on frame reception. That is, the kernel interrupts the program execution, switches the execution context, and starts the RX interrupt service routine (ISR). The ISR reads the frame from the buffer of the network device and handles it. After the ISR finishes, the kernel resumes the program previously interrupted. Although the whole process takes usually dozens of microseconds, it may be too slow to support a Gb/s forwarding with the I-MAC. For example, with a data rate of 900 Gb/s and 1400-byte frames, the kernel must handle each frame with 12.5 microseconds and less. Clearly, such an RX interrupt model may be a bottleneck in high-speed applications.

Linux kernel version 2.5 introduced a new API (application programming interface) for handling incoming frames, dubbed NAPI. It addresses applications with high data rates and shortens the processing time of received frames. In short, when the kernel detects that the frequency of frame reception goes beyond a certain threshold, it switches to the NAPI. In this mode, the kernel disable RX interrupts but checks the network cards (polling) if they received frames. By doing so, the kernel reduces the overhead caused by RX interrupts and shortens the processing time of RX frames. The I-MAC layer based on NAPI needs about 7.5 microseconds to process a single received frame. With such speed, the I-MAC is theoretically capable of forwarding frames with 1.5 Gb/s, provided frames are at least 1400 bytes long.

5. Evaluation

Fig. 7 depicts the set-up that we used for the evaluation. Two nodes, UDP client and server, were end-devices that generated network traffic. First, they were directly connected with a Gigabit Ethernet (GEth) link for comparison reasons, dubbed as the native network. Second, we connected both end devices with two OMEGA nodes, each running the I-MAC. Each OMEGA device had four network interfaces: two GEth (based on Intel 82571EB), PLC (Netgear Powerline AV+ 200) and WLAN (D-Link DW-556 with AR922X chipset). Both OMEGA nodes were PCs (HP Compaq 8100) with Intel Core i5-660 Processors (3.33 GHz) running the Linux Ubuntu version 10.04 with kernel 2.6.32. This evaluation was partly included in our previous work [1].

5.1. Data rate limit

To estimate the highest data rate supported by the I-MAC, we measured the packet error rate (PER) for various data rates. In case the I-MAC does not support a certain data rate, RX buffers of receivers become full, the following frames are discarded, and the PER increases.

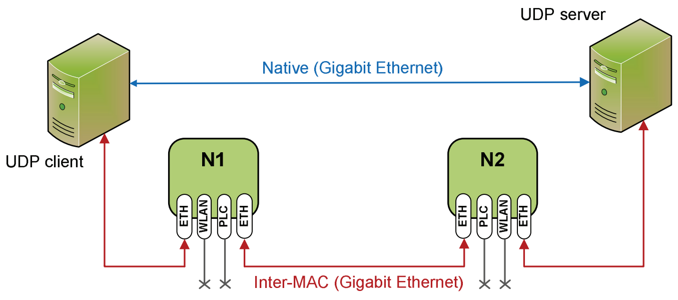

Fig. 7. Measurement set-up for performance evaluation

To test the set-up with various data rates, we ran *iperf* tool on both UDP client and server. In short, the client transmitted UDP datagrams with a specific data rate, and the server estimated the PER. The client generated traffic with two different frame sizes, since smaller sizes results in more transmitted frames for a given data rate. As the I-MAC layer causes a certain forwarding delay, a higher number of frames may result in a worse performance. Therefore, we considered short 100-byte frames and long ones with 1400 bytes, which is close to the default size limit of Ethernet networks (1500 bytes).

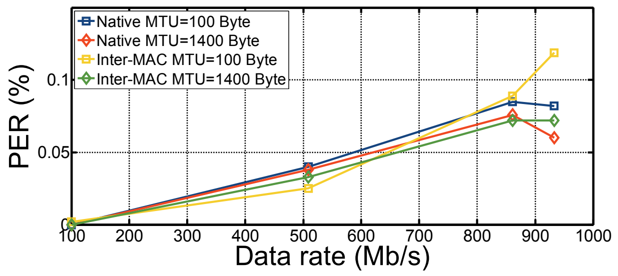

Fig. 8. Packet Error Rate (PER) of the native network (Gigabit Ethernet) and of the network based on I-MAC nodes

Our evaluation revealed that I-MAC with this set-up supports data rates close to Gb/s. For example, when the UDP client was sending data with a rate of 930 Mbit/s, we did not notice any significant differences between the native and I-MAC networks (see Fig. 8). In the worst case (data rate of 930 Mbit/s), the PER of the native network sending 1400-byte frames was 0.06%, and the I-MAC increased the PER to 0.072%. Moreover, the I-MAC layer does not result in a significant penalty when sending short frames. For instance, with the frame size of 100 bytes and the data rate of 930 Mbit/s, the nodes were transmitting more than a million frames a second. Nonetheless, the I-MAC layer did not increase the PER significantly, meaning it supports Gb/s data rates with short frames too. Besides, with lower data rates, such as 860 Mbps, there was no difference in PER between both networks at all. It shows that the I-MAC layer is capable of supporting Gb/s network traffic.

Table 1. Handover time from Gigabit Ethernet to various technologies

	Switching from Gigabit Ethernet to		
	GEthernet	PLC	WLAN
Handover time (ms)	1.3	17.2	61.1

The above observation shows that the I-MAC layer supports the highest data rate of the Gigabit Ethernet, and such a technology is the limiting factor of the current set-up. Therefore, we intend to conduct the same measurements with 10 Gbit Ethernet cards and find the I-MAC limits.

5.2. I-MAC Delay

Since the I-MAC performs extra processing, such as looking up the forwarding table, it increases the forwarding and network delays. In this experiment, we measured the network delay by considering the same set-up as previously, i.e., with native and I-MAC networks. To consider delays stemming from the I-MAC only, we performed the experiment with GEth only. Using slower technologies, such as PLC or WLAN, would affect the results significantly, as these technologies have longer delays that GEth.

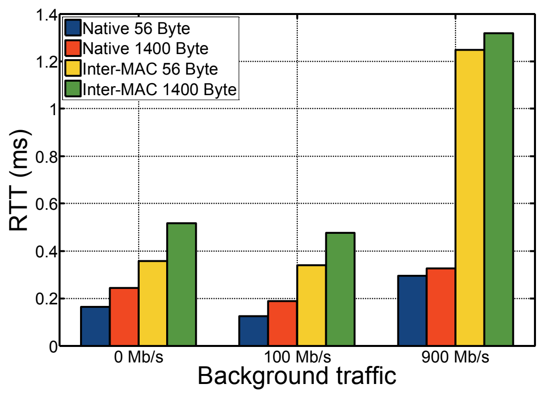

Fig. 9. Measurement results for RTT with different background traffic

In this case, the client sent 1000 ICMP request frames to the server. Upon the frame reception, the server transmitted immediately a reply to the client. For each frame received the client estimated the Round Trip Time (RTT). We carried out this test three times, with three different values of the background network traffic: no traffic at all, 100 Mbit/s and 900 Mbit/s. We also carried out tests with various frame sizes: 56 and 100 bytes.

As expected, the I-MAC causes extra network delays. For example, with background traffic of 100 Mbit/s, the I-MAC increased the RTT from 0.2 to 0.5 ms when sending 1400-byte frames (see Fig. 9) compared to the native network. When nodes send data with a data rate of 900 Mbit/s, the RTT of the I-MAC network increases to 1.3 ms. However, the RTT includes I-MAC delays of two nodes, and delays of both request and reply frames. Therefore, the processing time of a single I-MAC frame is four times smaller than depicted in Fig. 9.

Although the I-MAC increases the network delay, we can tolerate it in typical home networks. For instance, we expect a delay of less than 10 ms from a network with 20 nodes, which is still quite a large set-up for home networks. Such a delay would not affect the quality of the communication considerably.

5.3. Multi-Technology handover

After detecting a link problem, such as broken link or quality degradation, the I-MAC looks for a better connection, and then diverts the network traffic via better links, dubbed handover. Each handover includes the following steps:

1. The I-MAC detects a broken link, or a link with a better quality, with probe frames. For instance, if the I-MAC does not receive probe frames within a certain time, it assumes the link is broken and triggers Path Selection Engine (PSE) to discover another route.
2. PSE discovers a new route by broadcasting route request frames, and receiving replies. Then, PSE updates the forwarding table.
3. After updating the forwarding table, the Data Plane sends data over the new route.

We measured only the handover time (steps 2 and 3) without the delay caused by detection of broken links (step 1). This experiment considers the same set-up as previously (see Fig. 7). In this scenario, N1 was forwarding data to N2 using the GEth link. Later on, this link stopped working, and N1 diverted the traffic to other interfaces: another GEth, PLC, and WLAN.

Table 1 depicts handover times from GEth to other interfaces. Since GEth causes the shortest delays among all considered technologies, the handover time to another GEth was the shortest, 1.3 ms only. With such short times, users do not experience any video quality problems and handovers are seamless. Although handover to other technologies are longer than to GEth, they are still acceptable for small home networks, as streamed videos are only slightly affected on handovers, meaning the picture is party distorted for a time shorter than a second. The current I-MAC version discards all frames that arrive after detection of a broken link but before a new route is discovered. It causes these small picture quality problems on slower technologies.

We presented result for a 2-hop I-MAC network, but home networks may consist of more nodes. Therefore, handovers might not be seamless in typical home networks with slower technologies, such as PLC or WLAN. To counter this threat, we will optimize the I-MAC in our future work. the I-MAC will store incoming frames during the discovery, instead of discarding them, and will support seamless handovers with larger home networks too.

6. A Real-Time Demonstrator

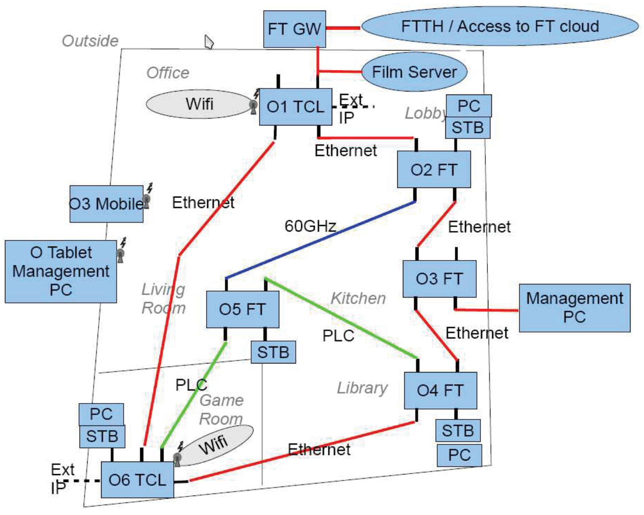

Fig. 10. OMEGA I-MAC final Demonstrator

Within the OMEGA project the I-MAC implementation was used to set up a realistic HAN with several nodes, STBs, TVs, and Servers, running several HDTV streams concurrently. Fig. 10 shows the demonstrator as used for the final demonstration in March 2011 in Rennes, France.

The Figure shows seven OMEGA nodes each connected with peer nodes with different physical connection means. For the demonstration we used Gigabit Ethernet (red), PLC (green), WLAN according to IEEE802.11n, and 60 GHz communication similar to IEEE802.15.3c (blue). Additionally we connected to the nodes several STBs, a media server and also a gateway to the Internet. A management PC was used to set up the configuration and also to visualize the network topology, the change of states and the statistics of each communication flow within the network. For the demonstration we set up 7 concurrent HDTV streams from different sources to different destinations. The I-MAC connected the streams using initial capacity and quality values. During the demonstration we disconnected some physical connections. The I-MAC recognized the missing connectivity within some ms and reconnected the stream almost seamlessly using a different route through the network. Several routes used several different media hops from source to destination. Another test demonstrated the potential of load balancing to continuously increase the traffic via certain physical connections. The I-MAC detected the risk of. QoS degradation and rerouted traffic accordingly. The third test was the demonstration of the wireless handover capability. To show this two access points were connected to two different OMEGA nodes. A thresh-hold in terms of S/N ratio was defined that indicated to the monitoring engine decreasing QoS capability. As in the test described above I-MAC detected the degradation and handed the connection over to the second access point.

The demonstration showed the high potential of the new approach clearly and with big success. To our knowledge it was the first demonstration of a really big HAN with such amount of load and self-healing capability.

7. Conclusions and Future Work

The OMEGA I-MAC approach is an innovative way to deal with heterogeneity, reliability and QoE in HANs. By using the IEEE802.2 interface we are able to use current device technology as well as current application technologies. The performance reduction measured in the reference implementation showed highly acceptable values for HANs.

Based on the OMEGA project a standardization activity has been set-up under IEEE1905.1. The inauguration meeting took place in October 2010 and the first technical session in April 2011. In the meantime more than 20 companies have shown interest in this new standard since it addresses an important aspect of future HANs.

Some technical extensions are still planned and will be realized in the near future. One of these extensions addresses the seamlessness of the rerouting within the network. As described earlier buffer handling has to be very fast to realize the high switching speed. The current technology however is not fully lossless since if a connection fails after a packet pointer has been copied into the output buffer, it is lost and the associated data will be lost too. To avoid this we plan a different form of dynamic buffer management. Here we intend to place the packet pointers into buffers that are not fixed to any output port. If the packet should be forwarded, we only apply a temporal connection for that packet.

A second improvement addressed increased reliability for data stream. Since video- and audio stream do not need any ARQ, no repetition of lost packets is necessary. We can, however split stream and duplicate packets to be send over several routs concurrently. At the destination we can merge the stream again.

Even if the described work has shown very good results, there is still a gap to bridge between the experimental work and a sellable product. The cost of a Pentium processor for a HAN-Node is too expensive. Thus we have started another HW/SW implementation to develop an ASIC based solution with an embedded low cost processor. The control and management planes could be executed by this processor running an embedded LINUX. The data plane would be implemented by a dedicated HW processor to achieve at minimum the 1 Gb/s switching speed. First estimations show that switching speed of 5 Gb/s is feasible and implementable for quite low cost using state of the art 65 nm CMOS technology.

REFERENCES

[1] M. Brzozowski, S. Nowak, F.-M. Schaefer, R. Jennen, A. Palo, "Inter-MAC- From Vision to Demonstration, Enabling Heterogeneous Meshed Home Area Networks", *14th ITG Conference on Electronic Media Technology,* Dortmund (Germany), March 2011.

[2] T. Meyer, V. Suraci, P. Langendörfer, S. Nowak, M. Bahr and R. Jennen, "An Inter-Mac Architecture for Heterogeneous Gigabit Home Networks", *IEEE PIMRC'09,* Tokyo, Sept 2009.

[3] M. Maaser, S. Nowak, P. Langendörfer, "Automated Mapping of MAC Parameters into Generic QoS Parameters by Inter-Mac Adaptors", *IEEE PIMRC'10*, Istanbul, Sept 2010.

[4] V. Suraci, A. Cimmino, R. Volella, G. Oddi, M. Castrucci, "Convergence in Home Gigabit Networks: Implementation of Inter-MAC Layer as a Pluggable Kernel Module", *IEEE 21st International Symposium on Personal Indoor and Mobile Radio Communications*, 2010.

[5] IEEE802.21, "Media Independent Handover Services", Jan 2009.

[6] K. Taniuchi et. al., "Media Independent Handover: Features, Applicability, and Realization", *IEEE Communication Magazine*, vol 47, no 1, Jan 2009.

[7] UPnP QoS Architecture 2.0, *UPnP Forum*, October 2006.

[8] *Universal Plug and Play Forum*, online: http://www.upnp.org.