MODULAR LINK LAYER FUNCTIONS OF A GENERIC PROTOCOL STACK FOR FUTURE WIRELESS NETWORKS

Lars Berlemann, Arnaud Cassaigne, Ralf Pabst, Bernhard Walke
(ComNets, RWTH Aachen University, Aachen, Germany)
ber|adc|pab|walke@comnets.rwth-aachen.de

ABSTRACT

Multi-mode capable wireless networks are a key issue in future wireless communication. This paper introduces therefore the realization and application of a generic protocol stack as common part of a multi-mode capable communication protocol software. This can be regarded as an extension of the field of software defined radios with its origin in the physical layer on the upper layers of the protocol stack. The generic protocol stack compromises common functionality and behavior of the communication protocols that is extended through specific parts of a dedicated radio access network technology. In a bottom-up approach, this paper considers fundamental protocol functions realized as parameterizable modules. These protocol functions originally correspond to the data link layer of the ISO/OSI reference model. The system specific aspects of the protocol software are realized through adequate parameterization of the modules' behavior. Further specific functionality and behavior can be added to the generic protocol stack through inheritance or insertion of system specific modules. Thus, the generic protocol stack enables an efficient as well as flexible realization of the protocol software as part of a future communication network of multiple radio access technologies.

1. INTRODUCTION

The radio access of future ubiquitous communication networks will be released from the constrains of cellular wireless networks, as for instance Universal Mobile Telecommunication System (UMTS), or Wireless Local Area Networks (WLANs). Wireless mobile broadband systems, providing a patchy coverage in densely populated urban areas, will play an important role. For details on such a fixed and planned relay based radio network see [1] and [2]. The addressed future wireless network will have to combine several Radio Access Technologies (RATs): Consequently, multi-mode capable terminals as well as base stations are required to enable the seamless interworking between these RATs. Multi-mode architectures can already be found in existing systems, like IEEE 802.16 [3].

Software Defined Radios (SDRs), [4] and [6], are a promising approach towards these multi-mode devices. The recent technological progress enables an extension of the key issues in research of SDR from the signal processing of the physical layer on the complete communication chain used for wireless communication as enabling technology for cognitive radios [4], [5] and [7]. Therefore, this paper extends the focus of SDRs on the communication protocol software in considering the Data Link Layer (DLL) and network layer corresponding to the ISO/OSI reference model [8]. This work supplements the research of the integrated project E^2R [9] dealing with end-to-end reconfigurability.

After introducing the idea of a generic protocol stack in Section 2, its application for protocol convergence in the context of a multi-mode capable protocol architecture is outlined. The realization of a generic protocol stack on basis of fundamental protocol functions that can be parameterized is summarized in Section 3. The composition of specific protocol layers of adequately parameterized modules is shown in Section 4 followed by an evaluation of the segmentation/reassembly module at the example of channel utilization with and without concatenation in Section 5. This paper ends with a summary and outline of future work as part of the conclusion.

2. THE GENERIC PROTOCOL STACK

The rationale for approaching a generic protocol stack is that all communication protocols share much functional commonality, which can be exploited to build an efficient multi-mode capable wireless system. The aim is to gather these common parts in a single generic stack and specialize this generic part following particular requirements of the targeted RAT, as depicted in Fig. 1. The targeted advantages of this concept are: runtime reconfigurability and maintainability, code/resource sharing and protocol development acceleration through reusability.

The initial step towards a generic stack is a detailed, layer-by-layer analysis of communication protocols to

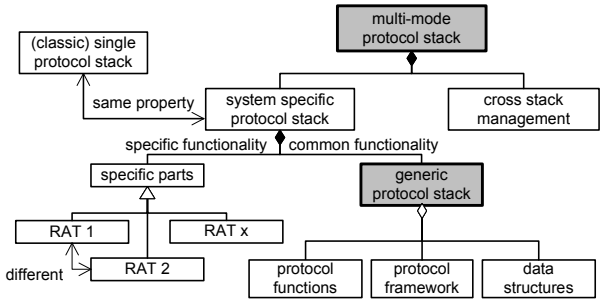

Fig. 1: UML diagram of the generic protocol stack in the context of multi-mode capable networks.

identify their similarities. Their elaborated realization of the generic parts is crucial for the success of the proposed concept in the face of a tradeoff between genericity, i.e. general usability, and implementation effort. As depicted in Fig. 1, the generic protocol stack comprises fundamental protocol functions, data structures and an architectural framework, which form, together with RAT specific parts, a system specific protocol stack. An efficient multi-mode capable stack is realized in adding cross stack management related functions. The cross stack management of the generic protocol stack for enabling protocol reconfigurability is introduced in detail in [14].

From the software engineering perspective, there are in general two possibilities for approaching the generic protocol stack: (1.) Parameterizable functional modules and/or (2.) inheritance, depending on the abstraction level of the identified protocol commonalities. As introduced above this paper focuses more on the modular approach while the inheritance-based approach is in considered in [10] and [11]. Additionally, [12] takes up the idea of a generic protocol stack in focusing on a generic link layer for the cooperation of different access networks at the level of the data link layer. However, not only the link layer protocols have to be considered in a multi-mode capable network but also higher layer functions as for instance the control and management of the radio resources as well as mobility.

2.1. Identifying Commonalities

The evolution of the digital cellular mobile radio networks originated in the Global System for Mobile Communication (GSM) toward systems of the third generation, as for example UMTS, has shown that in their standardization it is fallen back on well-proven functions and mechanisms which are adopted to the specific requirements of their application. The approach towards an efficient multi-mode protocol stack, introduced in this paper, is based on these protocol commonalities.

As the architecture of modern communication protocols cannot be forced into the classical layered architecture of the ISO/OSI reference model, it is rather difficult to identify similarities and attribute these to specific layers. Therefore, this paper deepens the level of examination and considers fundamental protocol functions as one basis for a generic protocol stack, contrary to [10] and [11] where complete protocols are analyzed for genericity. The identified protocol functions correspond mainly to the DLL as specified in the ISO/OSI reference model. Nevertheless they can be found in multiple layers of today's protocol stacks as for instance the function of segmentation and reassembly or an Automatic Repeated Request (ARQ) protocol for error correction which is located in the Radio Link Control (RLC) as well as in the transport layer, namely in the Transmission Control Protocol (TCP).

2.2. Enabling Protocol Convergence of Future Wireless Networks

The generic protocol stack enables the protocol convergence of future wireless networks. The convergence of such multi-mode protocol stacks has two dimensions: On the one hand the convergence between two adjacent layers, in the following referred to as vertical convergence, and on the other hand the convergence between layers located in the different modes of the protocol stack which have the same functions: In the following referred to as horizontal convergence. The generic protocol stack, as introduced above, enables both the horizontal as well as the vertical protocol convergence.

Fig. 2 depicts the transition between two protocol stacks of different air interface modes of a future wireless network. The different Physical Layer (PHY) options of IEEE 802.16 could serve as an example, see [3]. In present, these PHY options are not envisaged to co-exist in terminal or access equipment, although they share a common Medium Access Control (MAC) protocol with very little option-specific extensions to the MAC. The protocol stack is separated into the user and control plane (u- and c-plane) on the one hand and the management plane (m-plane) on the other hand. The split between common and specific parts of the protocol is here exemplary depicted for the MAC layer, as explained above. This split may be also necessary in the RLC or higher layers, depending on the targeted protocol architecture and functional flexibility of the common part. A cross stack management logically connects the protocol stacks of the different modes on the m-plane.

A seamless interworking and optimized transition between mode 1 and mode 2 has certain requirements to the cross stack management: The protocol data, as for instance the protocol status information of the existing connections has to be transferred between the two modes.

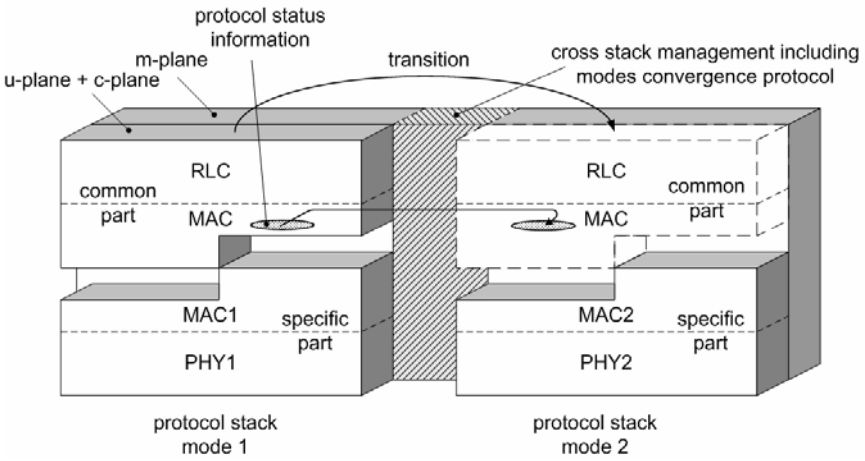

Fig. 2: A multi-mode capable protocol architecture under consideration of an optimized protocol convergence. Protocol information is transferred between two modes, here exemplary depicted on the level of the MAC layer as it is the case in IEEE 802.16 [3], with the help of a of a cross stack management.

This horizontal convergence is performed by a mode convergence protocol. Therefore, protocol functions of the c-plane, common to the different modes, are necessary to access u-plane status information. These common functions rely on a well-defined interface towards the mode specific part of the layer. For further details on the proposed protocol architecture and corresponding infrastructure see [13].

The cut between the common and specific part has a step to indicate that there is the need for (1.) a mode-specific interface to the PHY as said interface may inherently not be common, and (2.) a MAC layer internal interface for the communication between the common and specific part to enable for instance the vertical protocol convergence.

The generic protocol stack is the realization of the common part, illustrated in Fig. 3, and implements its common functions based on modules. These modules are concreted through parameterization adequate to the targeted specific mode.

3. MODULAR APPROACH TOWARDS THE REALIZATION OF A GENERIC PROTOCOL STACK

The generic protocol stack is the realization of the common parts, as illustrated in Fig. 1, and implements its common functions based on modules. These common protocol functions get their system specific behavior based on parameterization. Once specified, these modules can be repeatedly used with a different set of parameters corresponding to the specific communication system. The modules of generic protocol functions - form together with system specific modules - a complete protocol layer, as depicted in Fig. 3. The communication inside said layer is done based on generic service primitives and generic Protocol Data Units (PDUs) which are also considered as being a part of the generic stack, see again Fig. 1.

A unique manager as well as interfaces for the Service Access Points (SAP) to the adjacent layers complete the fully functional protocol layer as depicted in Fig. 3. In detail, the mentioned components have the following tasks:

- **Functional Module** (generic or RAT specific): Realizes a certain fundamental functionality as black box. In case of a generic module, a list of parameters for characterizing the functionality is given and the underlying functionality is hidden. The comprehensiveness of the fulfilled function is limited to fit straightforward into a single module.
- **Manager**: Composes and administers the layer during runtime. This implies the composition, rearrangement, parameterization and data questioning of the functional modules. Additionally, the manager administers the layer internal communication, as for instance the connection of the layer's modules through generic service primitives. The manager is the layer's counterpart of the protocol reconFiguration manager as introduced above in Fig. 2. The manager realizes the protocol convergence of the layer.
- **Generic interface**: Translates the generic service primitives with specific protocol information as payload to system specific ones and enables thus the vertical as well as horizontal integration of the system specific parts of the layer.
- **Service Access Point (SAP)**: Here, services of the layer are performed for the adjacent layers. The layer may communicate via generic primitives without a translation interface to an adjacent layer if said layer has the same modular composition. The interface is

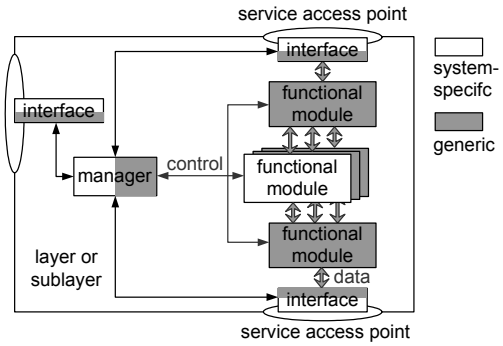

Fig. 3: Composition of a protocol specific layer or sublayer based on generic and system-specific functional modules.

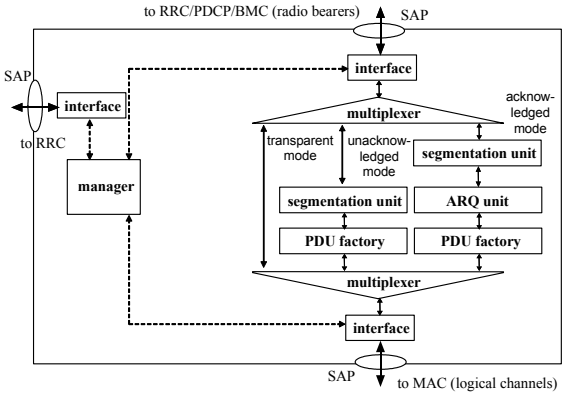

Fig. 4: UMTS RLC layer based on the functional modules of the generic protocol stack.

needed if it is demanded that the layer appears as a classic layer fitting into an ordinary protocol stack.

- **PDU factory** (as functional module, later depicted in Fig. 4-6): Composes layer specific protocol frames and places them as payload in generic PDUs.

The introduced modular concept enables the simulation and performance evaluation of the protocol software on several levels of abstraction: A single (sub-) layer as well as a complete protocol stack can be composed out of the introduced functional modules.

3.1. Generic Protocol Functions of the Data Link Layer

This section introduces generic protocol functions of the data link layer, realized as parameterizable modules, as initial step in the implementation of a protocol software for a multi-mode capable future wireless system as introduced in Section 2.

The following link layer functions may be part of the generic protocol stack

- Error handling with the help of ARQ protocols: Send-and-Wait ARQ, Go-back-N or Selective-Reject ARQ or Forward Error Correction (FEC)
- Flow control*
- Segmentation, concatenation and padding of PDUs*
- Discarding of several times received segments*
- Reordering of PDUs*
- Multiplexing/De-Multiplexing of the data flow, as for instance the mapping of different channels*
- Dynamic scheduling
- Ciphering
- Header compression

The functions marked with a star* are considered in the following section while further functions are considered in [14]. Segmentation is used if the length of a

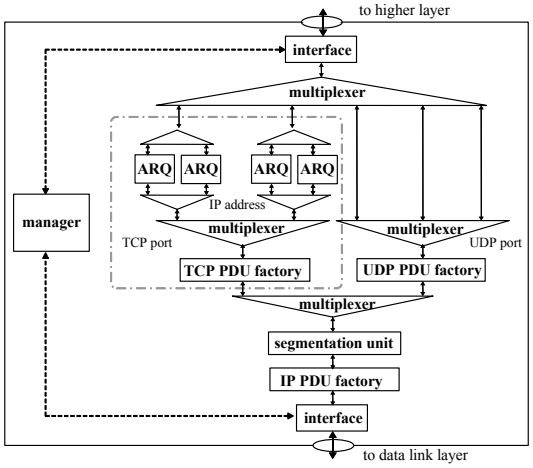

Fig. 5: TCP, IP and UDP layer based on the functional modules of the generic protocol stack. The TCP layer (gray dash-dotted line) is considered in [14].

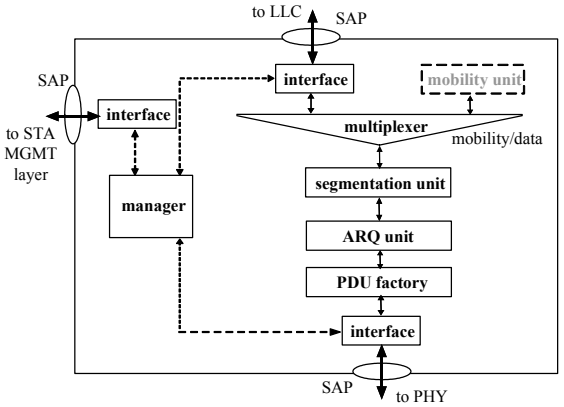

Fig. 6: 802.11 MAC layer based on the functional modules of the generic protocol stack.

PDU exceeds a certain predefined limit, given for instance through the capacity of a physical channel. The PDU is separated into different segments that are transmitted separately. Contrary the concatenation, where several

upper-layer SDUs are combined in the same data packet to reach a predefined PDU length.

3.2. Parameterization of Functional Modules

In this context parameterization implies not only specific values, as for instance the datagram size of a segmentation module, but also a conFiguration of behavior and characteristics of a module, as for example the concretion of an ARQ module as a Go-back-N ARQ protocol with specified window sizes for transmission and reception. This implies as well a conFiguration of the modules' interface to the outside. The parameterization of functional modules may imply (1.) a specification of certain variables, (2.) the switching on/off of certain functionality/behavior and (3.) an extension of the module's interface to the outside.

Taking the example of the segmentation/reassembly module, the parameterization may imply among other things:

- Use of concatenation
- Use of padding, i.e. filling up of the PDU to reach a certain size
- Transmitter or/and receiver role
- Buffer size for SDUs concatenated in a single PDU
- Size of PDU after handling
- Behavior in case of error, i.e. interworking with ARQ module

4. COMPOSITION OF SYSTEM SPECIfiC LAYERS

As introduced above, the link layer functions are not limited in their appearance to the DLL. To illustrate the applicability of the modular approach, a composition of three exemplary protocol layers, all differently localized in a protocol stack corresponding to the ISO/OSI reference model, is introduced in the following: (1.) A UMTS RLC layer in Fig. 4, (2.) a TCP, IP and UDP layer in Fig. 5 and (3.) a IEEE 802.11 MAC layer in Fig. 6. We limit the consideration of Fig. 5 to the TCP layer, marked through the gray dash-dotted rectangle. The medium access of the Distributed Coordination Function (DCF) of 802.11 may be regarded as a Send-and-Wait ARQ, simply realized in the ARQ module by a Go-Back-N ARQ with a window length of *1* [14].

5. SIMULATIVE EVALUATION AND VALIDATION OF THE FUNCTIONAL MODULES

The parameterizable functional modules are implemented in the Specification and Description Language (SDL), and evaluated with the help of a Modular Object-oriented Software and Environment for Protocol Simulation (MOSEPS) that provides basic traffic generators, a model of an erroneous channel and statistical evaluation methods. Once specified, these modules can be repeatedly used with a different set of parameters corresponding to the specific communication system. This section introduces the modular approach to protocol functions with a focus on the segmentation/reassembly module at the example of the approach to a realization of an UMTS RLC as depicted in Fig. 4.

5.1. Analytical Evaluation of Segmentation/Reassembly Module

In general, segmentation is needed in all cases where higher layer PDUs, referred to as Service Data Units (SDUs), need to be separated into multiple PDUs to be further handled by lower layers. This restriction concerning the size of a PDU may result for instance from reasonable limitations of a PDU transmitted with error correction in an ARQ protocol or may be motivated through the capacity of a transport channel offered by the physical layer, using the notation of UMTS.

In case of multiple users sharing a common channel, the segmentation can increase the channels efficiency in having a multiplexing gain. The above-introduced concatenation increases thereby the channel utilization as it is outlined in Fig. 7. The channel utilization as quotient of user *payload* and available *channel capacity* is given through

$$\frac{payload}{channel\ capacity} = \frac{l_{payload}}{\left\lceil \frac{l_{payload}}{packet\ size} \right\rceil \bullet packet\ size} \qquad (1)$$

in the case of no concatenation. The *packet size* of the physical channel is assumed to be fixed and the packet length of the user data $l_{payload}$ determines if the segmented higher layer SDU(s) fit(s) into a single PDU. We assume that an additional physical channel is established if the user data requires it. Thus, additional channel capacity is provided and the available capacity is increased. In case of no concatenation the fixed-sized packet is transmitted partly empty over the physical channel. Consequently, the channels' overall utilization, i.e. the effectively used capacity compared to the amount of provided capacity, is decreased as illustrated by the solid zigzag line in Fig. 7.

5.2. Simulative Evaluation at the Example of the UMTS RLC

The introduced example of Fig. 7 focuses on the segmentation aspects of the UMTS RLC in the Unacknowledged Mode (UM). The UM of the RLC has the responsibility to concatenate SDUs to a PDU of a predefined length, here *128 Byte*. The simulative results with and without concatenation are given by the markers.

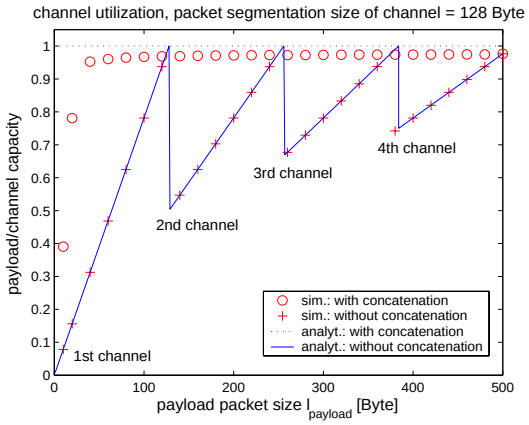

Fig. 7: Utilization of channels with a fixed packet size of *128 Byte*.

The payload packet size, i.e. the user data, is increased up to *500 Byte* and the channel utilization as introduced in the previous section is evaluated.

In applying the segmentation in a communication protocol the protocol overhead comes into play: Due to this overhead, more capacity than transmitted user data is required as observable in Fig. 7 for $l_{payload}$ *=384 Byte* in the case of no concatenation. The same stands for the usage of concatenation, as the observed channel utilization matches not the ideal one. The number of SDUs prepared for transmission in a single PDU is here limited so that for small $l_{payload}$ values a PDU is not completely filled. In summary, the parameterizable segmentation/reassembly module adequately reflects the expected behavior and can be validly used in a multi-mode capable protocol stack.

6. CONCLUSION

The introduced modular concept of a generic protocol stack enables protocol software for future multi-mode capable systems under the consideration of an optimized protocol convergence. The presented modules of the generic protocol stack can be regarded as toolbox for the accelerated development resulting into dimensioning rules for an adequate support of quality of service in wireless communication from the perspective of the protocols. The additional effort for enabling genericity has to be well reasoned to guarantee in general the suitability of the generic approach. Nevertheless, especially this realization is simplified through the introduced modular approach. However, the consideration of common protocol functions and protocol convergence during the development of future protocols will increase by itself the grade of genericity and thus the efficiency of this approach.

ACKNOWLEDGMENT

The authors would like to thank the German Research Foundation (DFG) for funding the work contributed to this paper in the form of a Research College (Graduiertenkolleg).

REFERENCES

[1] B. Walke, R. Pabst and D. Schultz, “A Mobile Broadband System based on fixed Wireless Routers,” in Proc. of *ICCT’03*, pp. 1310-1317, Beijing China, 2003.

[2] R. Pabst et al., “Relay-Based Deployment Concepts for Wireless and Mobile Broadband Radio,” *in IEEE Communications Magazine*, vol. 42, no. 5, pp. 80-89, September 2004.

[3] IEEE, “Standard for Local and metropolitan area networks, Part 16: Air Interface for fixed Broadband Wireless Access Systems - Amendment 2: Medium Access Control Modifications and Additional Physical Layer Specifications for 2-11 GHz,” April 2003.

[4] J. Mitola, “The Software Radio Architecture,” in *IEEE Communications Magazine*, vol. 33, no. 5, pp. 26-38, May 1995.

[5] J. Mitola, “Cognitive Radio,” Ph.D. thesis, KTH, Stockholm, pp. 45-90, 2000.

[6] W. Tuttlebee, “Software Defined Radio: Enabling Technologies,” Wiley Series in Software Radio, ISBN 0470843187, 2002.

[7] P. Mähönen, “Cognitive Trends in Making: Future of Networks,” invited paper in Proc. of *PIMRC’04*, Barcelona Spain, 2004.

[8] ITU-T Recommendation X.200 “Information Technology – Open System Interconnection - Basic Reference Model: The Basic Model,” International Telecommunication Union (ITU), Geneva Switzerland, 1994.

[9] End-to-End Reconfigurability (E²R), IST-2003-507995 E²R, http://www.e2r.motlabs.com.

[10] M. Siebert and B. Walke, “Design of Generic and Adaptive Protocol Software (DGAPS),” in Proc. of *3Gwireless '01*, San Francisco USA, June 2001.

[11] L. Berlemann, M. Siebert and B. Walke, “Software Defined Protocols Based on Generic Protocol Functions for Wired and Wireless Networks,” in Proc. of *Software Defined Radio Technical Conference*, Orlando USA, 2003.

[12] J. Sachs, “A Generic Link Layer for Future Generation Wireless Networking,” in Proc. of *ICC’03*, Anchorage USA, 2003.

[13] B. Walke, R. Pabst, L. Berlemann and D. Schultz, “Architecture Proposal for the WINNER Radio Access Network and Protocol,” in Proc. of *11th Meeting of Wireless World Research Forum*, Oslo Norway, 2004.

[14] L. Berlemann, A. Cassaigne and B. Walke, “Generic Protocol Functions for Design and Simulative Performance Evaluation of the Link-Layer for Reconfigurable Wireless Systems,” in Proc. of *WPMC’04*, Abano Terme Italy, 2004.