Generic Protocol Functions for Design and Simulative Performance Evaluation of the Link-Layer for Re-configurable Wireless Systems

Lars Berlemann, Arnaud Cassaigne, Bernhard Walke
Chair of Communication Networks, Aachen University (RWTH),
Kopernikusstraße 16, D-52074 Aachen, Germany,
e-mail: {ber|adc|walke}@comnets.rwth-aachen.de

***Abstract*— This paper discusses first steps towards the realization and application of a generic protocol stack as part of the software for re-configurable wireless communication systems. This focus on the protocol software extends the field of software defined radios with its origin in the physical layer. The generic protocol stack compromises common protocol functionality and behavior which is extended through specific parts of the targeted radio access network technology. Following a bottom-up approach this paper considers parameterizable modules of basic protocol functions corresponding to the data link layer of the ISO/OSI reference model. System specific aspects of the protocol software are realized through adequate parameterization of the modules. Further functionality and behavior can be added through the insertion of system specific modules or inheritance. The generic protocol stack enables an efficient realization of re-configurable protocol software as part of a completely re-configurable wireless communication system. The presented modules of the generic protocol stack can also be regarded as tool-box for the accelerated development of future communication protocols.**

***Keywords*— Generic Protocol Stack, Link Layer Functions, Modular Layer Composition, Re-configurability, Software Defined Radio**

I. Introduction

The evolution of the digital cellular mobile radio networks originated in the GSM toward systems of the third generation, as e.g. the UMTS, has shown that in their standardization it is fallen back on well-proved functions and mechanisms that are adopted to the specific requirements of their application. Looking at recent developments, terminals tend to integrate GSM as fallback for uncovered areas on the one hand and additionally integrate WLANs for high capacity needs on the other hand. In this context it is obvious that re-configurability is the one of the key issues for future wireless communication systems beyond 3G. Consequently a completely re-configurable mobile terminal in a re-configuration supporting radio network is demanded.

Software Defined Radios (SDRs) [1] and [2] are a promising approach towards this re-configurability. For a long term, the hardware has been the key issue of research in the field of SDR. But the recent technical progress enabled an expanding of research efforts on the complete communication chain with the communication software and the protocol stack as essential part of it [3] and [4].

This paper focuses on protocol functions of the Data Link Layer (DLL), the layer two corresponding to the ISO/OSI reference model [5]. Related work considering the network layer, the layer three, has been published in [6]. The data link layer consists of (1.) the Medium Access Control (MAC) layer for the coordinated access to the medium as well as (2.) a layer for the error protection and error free transfer of data, often referred to as Logical Link Control (LLC) layer, as for instance in GSM. Alternatively or as part of it, the corresponding layer may be also referred to as Radio Link Control (RLC) layer, as for example in UMTS.

This paper is outlined as follows: The idea of a generic protocol stack on the basis of fundamental, parameterizable functional modules of the DLL in the context of protocol re-configurability is introduced in Section II. The necessary additional elements to realize a fully functional specific protocol layer are outlined thereafter. Section III introduces the composition of system specific layers at the example of the UMTS RLC, the Transmission Control Protocol (TCP) and the IEEE 802.11 MAC layer, which differ in their development history as well as in their layer classification corresponding to the ISO/OSI reference model. The latter two layers are validated and evaluated analytically and simulative in Section IV.

II. The Generic Protocol Stack

The primary idea of the generic protocol stack is that all communication protocols share a lot of functional communalities that should be exploited to build an efficient re-configurable wireless system [7]. The aim is to gather these common parts in a single generic stack and specialize this generic part following the particular requirement of the targeted Radio Access Technology (RAT), as depicted in Fig. 1. The targeted advantages of this concept are: runtime re-configurability and maintainability, code/resource sharing and protocol development acceleration through reusability.

The initial step towards a generic stack is a detailed, stepwise analysis of communication protocols to 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

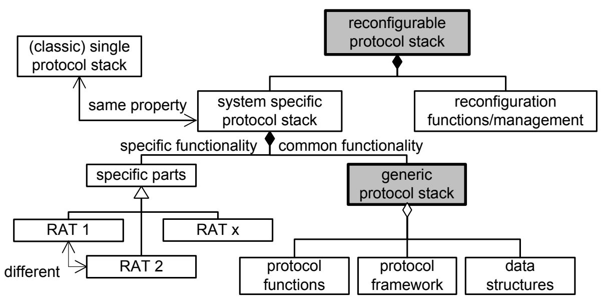


Fig. 1: UML diagram of the generic protocol stack in the context of protocol re-configurablility.

implementation effort. From the software engineering perspective, there are in general two possibilities for approaching the generic protocol stack: Parameterizable functional modules and/or 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 considered in [6] and [8].

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 re-configurable stack is realized in adding re-configuration related functions and management, as further outlined in the next but one section.

A. Generic Protocol Functions of the Data Link Layer

As the architecture of modern communication protocols can not 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 in the search for similarities and considers fundamental protocol functions, contrary to [6] and [8] where complete protocols are analyzed for genericity. Though these protocol functions mainly correspond to the DLL as specified in the ISO/OSI reference model, they can be found in multiple layers of today's protocol stacks as shown below. The following functions are considered for the generic protocol stack:

- Error handling with the help of Forward Error Correction (FEC) or Automatic Repeated reQuest (ARQ) protocols as for instance Send-and-Wait ARQ, Go-back-N ARQ* or Selective-Reject ARQ
- Flow control*
- Segmentation, concatenation and padding of Protocol Data Units (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 with a star* marked functions are considered in the following section while the other functions are considered in [9].

B. Enabling Re-configurability

The generic protocol stack, with its pool of generic functions as introduced above, enables an efficient as well as flexible realization of re-configurable protocol stack. Full, end-to-end re-configurability from the modem part up to the applications requires a layer overlapping management and re-configuration functions. Therefore, as this paper considers communication protocols implemented in software, a protocol re-configuration manager is introduced in Fig. 2, which accomplishes all re-configurability related tasks of the PHYsical layer (PHY), MAC, RLC/LLC and transport layer. These system specific layers are based on the pool of generic protocol functions and

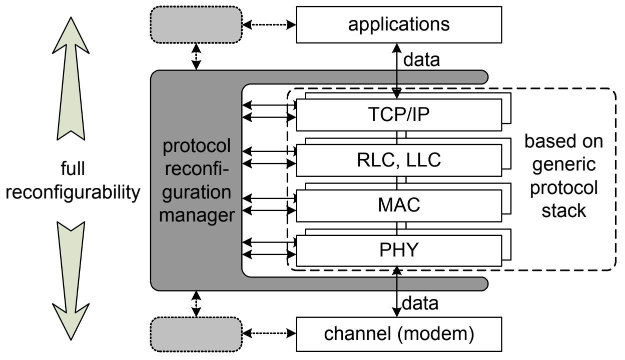


Fig. 2: A re-configurable protocol stack based on generic functional modules in the context of a completely re-configurable terminal.

mechanisms. The protocol re-configuration manager has thereby the following tasks:

- Management of the (permanently or temporally) parallel existing protocols and protocol stacks
- Creation, destruction and/or re-configuration of a single protocol or complete protocol stack
- Administration of the user data flow during the re-configuration process, as for instance the redirection of the user data from the old to the re-configured protocol stack
- Cross layer optimization, as for instance the transfer of protocol or user data within the old stack to the new one [9]
- Support and enabling of re-configuration functions of the network, as for instance a network initiated re-configuration or an update of the network information about the status of the terminal

The re-configuration of the protocol stack or single protocol, administrated through the protocol re-configuration manager, has two characteristics: (1.) the creation of a new stack/layer consisting of adequate parameterized modules of the generic stack and destruction of the existing one and (2.) the re-configuration of the existing protocol implementation in exchanging the parameterization of the corresponding modules.

C. Modular Approach – the Generic Protocol Stack as Library of Protocol Functions

The generic protocol stack is the realization of the common parts, as illustrated in Fig. 1, and implements its common functions on the basis of modules. These common protocol functions get their system specific behavior on the basis of 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 on the basis of generic service primitives and generic 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

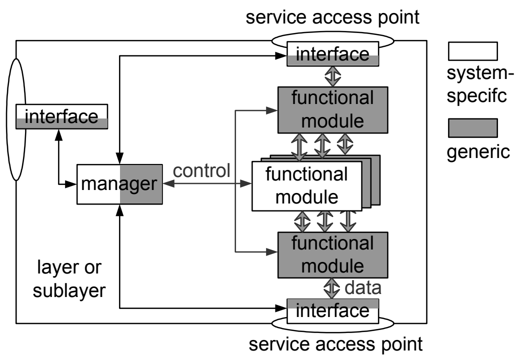


Fig. 3: Composition of a protocol specific layer or sublayer on the basis of generic and system-specific functional modules.

hidden. The comprehensiveness of the fulfilled function is limited to fit straightforward into a single module.

- **Manager**: Composes and administrates the layer during runtime. This implies the composition, rearrangement, parameterization and data questioning of the functional modules. Additionally, the manager administrates 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 re-configuration manager as introduced above in Fig. 2. The manager realizes the re-configurability 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 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.

This approach enables the simulation and performance evaluation on several levels: A single (sub-)layer as well as a complete protocol stack can be composed out of the introduced modules. To facilitate understanding the parameterization itself is introduced in the following.

D. 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 (i.) a specification of certain variables, (ii.) the switching on/off of certain functionality/behavior and (iii.) an extension of the module's interface to the outside.

At the example of the ARQ module, the parameterization may imply among other things:

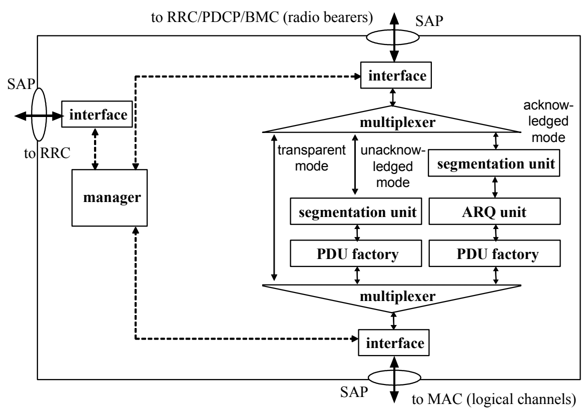


Fig. 4: UMTS RLC layer based on the functional modules of the generic protocol stack.

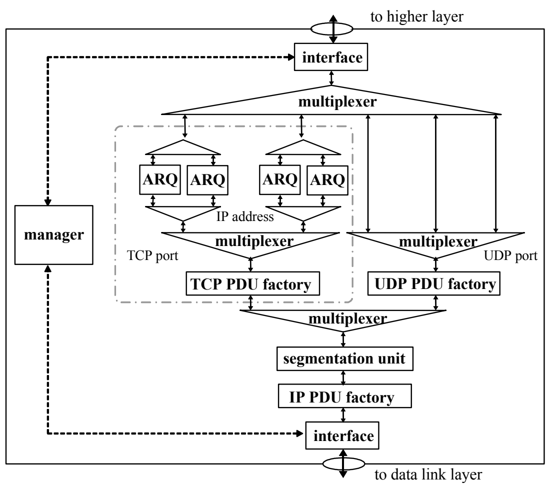


Fig. 5: TCP, IP and UDP layer based on the functional modules of the generic protocol stack. The TCP layer (gray dash-dotted line) is considered here.

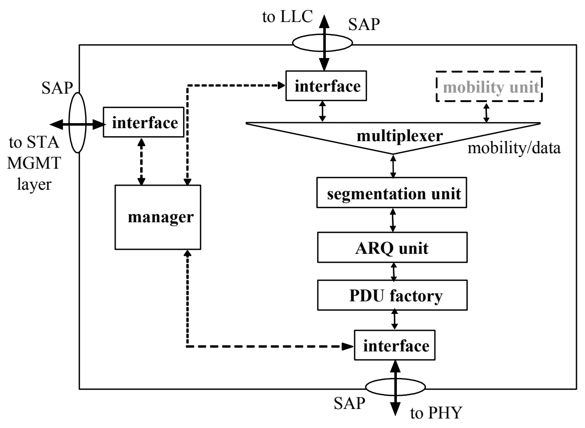


Fig. 6: 802.11 MAC layer based on the functional modules of the generic protocol stack.

- ARQ protocol characteristic, for instance Go-Back-N ARQ or Selective-Reject ARQ
- Transmitter or/and receiver role
- Receive and transmission window size
- Fixed, variable (TCP) window length or open/shut mechanism (LLC)
- Timer value, after a packet is assumed to be lost

- Connection Service: inexistent (UMTS RLC), separated for each direction (802.11 - CSMA/CA with RTS/CTS), 2-way handshake (GSM LLC) or 3-way handshake (TCP)
- Use of Negative ACKnowledgments (NACKs)

III. 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. The consideration of Fig. 5 is limited in the following 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*.

IV. Simulative Evaluation and Validation of the Functional Modules

The parameterizable 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, an erroneous channel model and statistical evaluation methods. This section introduces the modular approach to protocol functions with a focus on the ARQ module at the example of TCP and 802.11.

A. Transmission Control Protocol Layer

As introduced above in Fig. 5 a TCP layer can be composed out of the functional modules of the link layer as being part of the generic protocol stack. To validate the Go-Back-N mechanisms of the TCP layer's ARQ module we measure the protocol overhead in dependency on the payload packet size in the case of an erroneous channel. The focus is thereby on the influence of two effects: The bit error ratio *ber* of the radio channel, i.e. the wireless medium, and the size of the send and receive window *w*.

With a packet length of $l_{packet} = l_{header} + l_{payload}$, where TCP has fixed header length of $l_{header} = 40\ Bytes$, can the packet error ratio *per* be calculated to

$$per = 1 - (1 - ber)^{l_{packet} \cdot 8}\,. \qquad (1)$$

Based hereon the overhead to payload quotient for the Go-Back-N ARQ can be derived and approximated [10] to

$$\frac{overhead}{payload} = \frac{l_{header}}{l_{payload}} + \frac{\sum_{i=1}^{w/2} (\frac{w}{2} - i + 1) \cdot (1 - per)^{i-1} \cdot per}{\frac{w}{2}} \cdot \frac{l_{packet}}{l_{payload}}, \qquad (2)$$

where w is the length of the transmission window leading to the analytical results as depicted in Fig. 7. This figure illustrates the overhead to payload ratio in dependency of the frame length of the payload data for on the one hand (a) a bit error ratio of 10^{-5} and 10^{-6} and on the other hand (b) window length of *8* and *64*. Fig. 7 (a) shows the expected performance corresponding to the Go-Back-N ARQ: The overhead to payload ratio increases with increasing bit error ratio and an optimal frame length for the payload data to minimize said ratio can be determined. From the cross protocol optimization perspective, this frame length may be used as dimensioning rule for segmentation. The same stands for Fig. 7 (b): There the overhead to payload ratio increases with increasing window length, as the amount of data which has to be retransmitted, in the case of an error corresponding to the Go-Back-N ARQ, increases. In summary, the ARQ module of the generic protocol stack fulfills adequately its intended purpose.

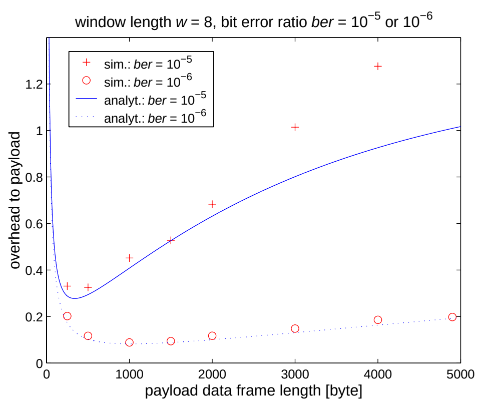
B. IEEE 802.11 Medium Access Control Layer

In this section the modular composition of an IEEE 802.11 MAC layer as illustrated in Fig. 6 is validated and evaluated. Therefore, the average throughput of the Carrier Sensing Multiple Access with Collision Avoidance (CSMA/CA)-based decentralized medium access by the DCF with and without Request To Send/Clear To Send (RTS/CTS) is analyzed and simulated.

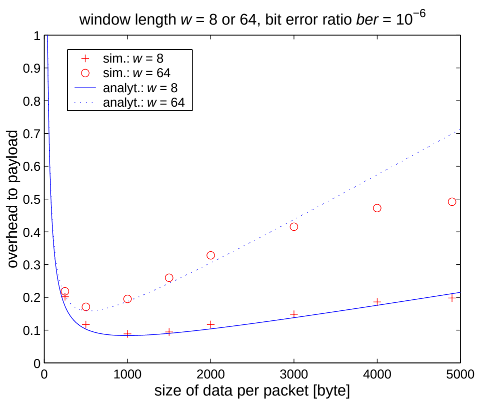
The channel capacity is mainly wasted by two effects: MAC header sending and collisions. One way to an analytical approach for determination of the throughput is to calculate the collision probability *p* and the access probability τ with the help of a two-dimensional Markov chain for the modeling of the backoff window of the DCF [11] and [12] resulting into

$$p = 1 - (1 - \tau)^{n-1}\ ,\quad \tau = 2 \cdot \left(1 + W_0 + pW_0 \frac{1-(2p)^8}{1-2p}\right)^{-1}, \qquad (3)$$

where *n* is the number of stations and W_0 the minimum backoff window size, here we chose $W_0 = 8$. With the help of the average time slot length $T_{average}$ on the basis of Tab. 1 can the average total system throughput $t_{saturation}$ be calculated to

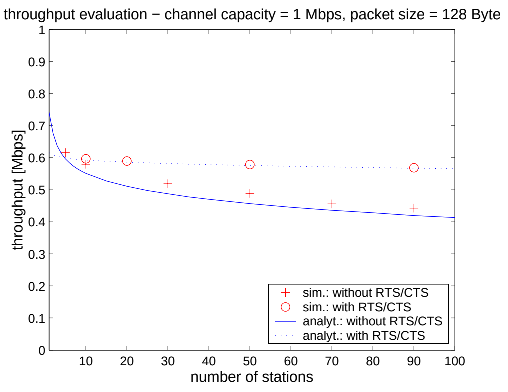


(a) varied bit error ratio *ber*

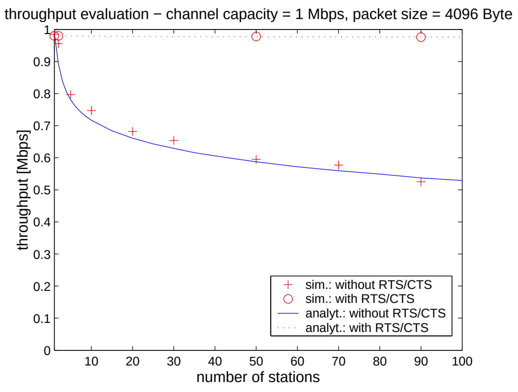


(b) varied window length *w*

Fig. 7: TCP layer Go-back-N ARQ validation and evaluation. The protocol overhead to payload ratio in dependency on the frame length of the payload data is depicted. The lines are analytic results corresponding to (2), while the markers indicate simulative evaluation.



(a) packet size = 128 Byte



(b) packet size = 4096 Byte

Fig. 8: 802.11 MAC layer evaluation of the DCF-based medium access under utilization of the ARQ module. The total system throughput depending on the number of transmitting stations is depicted. The lines are analytic results corresponding to (4), while the markers indicate the simulative evaluation.

$$t_{saturation} = \frac{P_s L_{payload}}{T_{average}}, T_{average} = P_e T_e + P_s T_s + P_c T_c . \qquad (4)$$

We assume a channel rate of *1 Mbps*, slot length, SIFS and DIFS are *1*, *6* and *10 μs* resulting into the analytical as well as simulative results of Fig. 8. There, the overall system throughput, with and without RTS/CTS, in dependency on the number of stations is depicted. For small packets, $L_{payload}$ = *128 Bytes* Fig. 8 (a), the headers are the main cause for an inefficient use of the medium. For larger frames, $L_{payload}$ = *4096 Bytes* Fig. 8 (b), a collision wastes more time, as a transmitting station is only able to notice an interfered frame after its ending. Therefore, the RTS/CTS mechanism is introduced, to have just a small RTS frame lost in case of a collision. The simulation agrees mainly with the analytic determination of the throughput of (4) and illustrates the superiority of the RTS/CTS based solution. As the ARQ module of the generic protocol stack reflects the expected behavior of RTS/CTS mechanism [12], this module can be legitimately used in an 802.11 MAC layer.

Tab. 1: Time slot durations in *μs* and probabilities that the medium is empty (e), successfully (s) allocated or a collision (c) occurs [11] and [12]. The fixed values result from the time length of a RTS/CTS sequence.

probability	duration with RTS/CTS	duration without RTS/CTS
$P_e = (1-\tau)^n$	$T_e = 1$	$T_e = 1$
$P_s = n\tau(1-\tau)^{n-1}$	$T_s = 636 + 8l_{payload}$	$T_e = 636 + 8l_{payload}$
$P_c = 1 - P_e - P_s$	$T_e = 170$	$T_s = 234 + 8l_{payload}$

V. Conclusion

The generic protocol stack, as a collection of modular protocol functions, takes up the usual advance of software engineering in the field of protocol development and evaluation: It is fallen back on well-proven and known protocol functions and behavior from the portfolio of the engineers' experience. A generic realization of these functions in the form of independent modules results in a library of protocol functions as construction kit for protocol development. In taking the tradeoff of genericity into account these thoughtful realized modules stimulate efficiency through reusability and maintainability as well as accelerate the development process itself. The efficiency of protocol re-configurability benefits from the introduced generic approach and implies a clearly limited additional effort of protocol management. Thus the introduced approach is a first step to an efficient end-to-end re-configurable wireless system.

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] `http://www.sdrforum.org/`.

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

[3] W. Tuttlebee, "Software Defined Radio: Enabling Technologies," *Wiley Series in Software Radio*, ISBN 0470843187, May 2002.

[4] End-to-End Re-configurability (E²R), IST-2003-507995 E²R, `http://www.e2r.motlabs.com`.

[5] ITU. "Information Technology - Open Systems Interconnection - Basic Reference Model: The Basic Model," ITU-T Recommendation X.200, International Telecommuncation Union (ITU), Geneva, 1994.

[6] M. Siebert, B. Walke, "Design of Generic and Adaptive Protocol Software (DGAPS)," in Proc. of *Third Generation Wireless and Beyond (3Gwireless '01)*, San Francisco USA, June 2001, `http://www.comnets.rwth-aachen.de`.

[7] M. Siebert, "Design of a Generic Protocol Stack for an Adaptive Terminal," in Proc. of *1st Karlsruhe Workshop on Software Radios*, pp. 31-34, Karlsruhe Germany, March 2000, `http://www.comnets.rwth-aachen.de`.

[8] L. Berlemann, M. Siebert, 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, November 2003, `http://www.comnets.rwth-aachen.de`.

[9] L. Berlemann, A. Cassaigne, B. Walke, "Modular Link Layer Functions of a Generic Protocol Stack for Future Wireless Networks", to appear in Proc. of *Software Defined Radio Technical Conference*, Phoenix USA, November 2004.

[10] A. Cassaigne, "Design and Evaluation of Protocol Functions for the Data Link Layer of a Generic Protocol Stack," *Diploma Thesis*, ComNets, Aachen University (RWTH), 2004.

[11] G. Bianchi, "Throughput Evaluation of the IEEE 802.11 Distributed Coordination Function," in Proc. of *5th Workshop on Mobile Multimedia Communications*, Berlin Germany, October 1998.

[12] A. Hettich, „Leistungsbewertung der Standards HIPERLAN/2 und IEEE 802.11 für drahtlose lokale Netze," *Ph. D. Dissertation*, Aachen University (RWTH), ABMT 23, ISBN:3-86073-824-0, `http://www.comnets.rwth-aachen.de`.