EURASIP Journal on Wireless Communications and Networking 2005:3, 390–400

Multimode Communication Protocols Enabling Reconfigurable Radios

Lars Berlemann
Chair of Communication Networks, RWTH Aachen University, Kopernikusstraße 16, D-52074 Aachen, Germany
Email: ber@comnets.rwth-aachen.de

Ralf Pabst
Chair of Communication Networks, RWTH Aachen University, Kopernikusstraße 16, D-52074 Aachen, Germany
Email: pab@comnets.rwth-aachen.de

Bernhard Walke
Chair of Communication Networks, RWTH Aachen University, Kopernikusstraße 16, D-52074 Aachen, Germany
Email: walke@comnets.rwth-aachen.de

Received 24 September 2004; Revised 21 February 2005

This paper focuses on the realization and application of a generic protocol stack for reconfigurable wireless communication systems. This focus extends the field of software-defined radios which usually concentrates on the physical layer. The generic protocol stack comprises common protocol functionality and behavior which are extended through specific parts of the targeted radio access technology. This paper considers parameterizable modules of basic protocol functions residing in the data link layer of the ISO/OSI model. System-specific functionality of the protocol software is realized through adequate parameterization and composition of the generic modules. The generic protocol stack allows an efficient realization of reconfigurable protocol software and enables a completely reconfigurable wireless communication system. It is a first step from side-by-side realized, preinstalled modes in a terminal towards a dynamic reconfigurable anymode terminal. The presented modules of the generic protocol stack can also be regarded as a toolbox for the accelerated and cost-efficient development of future communication protocols.

Keywords and phrases: generic protocol stack, link layer functions, modular layer composition, reconfigurability, software-defined radio.

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, 2]. The addressed future wireless network will have to combine several *radio access technologies* (RATs). Consequently, multimode capable terminals and base stations are required to enable the seamless interworking between these RATs. Multimode architectures can already be found in existing systems, like IEEE 802.16 [3] with different modes of the *physical layer* (PHY).

Software-defined radios (SDRs) [4, 5] are a promising approach towards these multimode devices. The recent technological progress allows an extension of the key issues in research of SDRs from the signal processing of the physical layer on the complete communication chain used for wireless communication. The current research efforts are targeting at the realization of cognitive radios [4, 6, 7]: self-aware, frequency-agile radio systems that are able to identify unused radio spectrum. These cognitive radios require protocol reconfigurability to unfold their advantage of dynamic spectrum usage. Therefore, this paper extends the focus of SDRs on the protocol software used for reliable communication over the air. Especially, the *data link layer* (DLL) and network layer corresponding to the ISO/OSI reference model [8] are considered. This work supplements the research of the integrated project E2R [9] dealing with end-to-end reconfigurability. The approach taken in this work aims at maximizing flexibility by providing a framework that is both general enough to accommodate a wide range of protocols, yet efficient enough to ensure competitive performance.

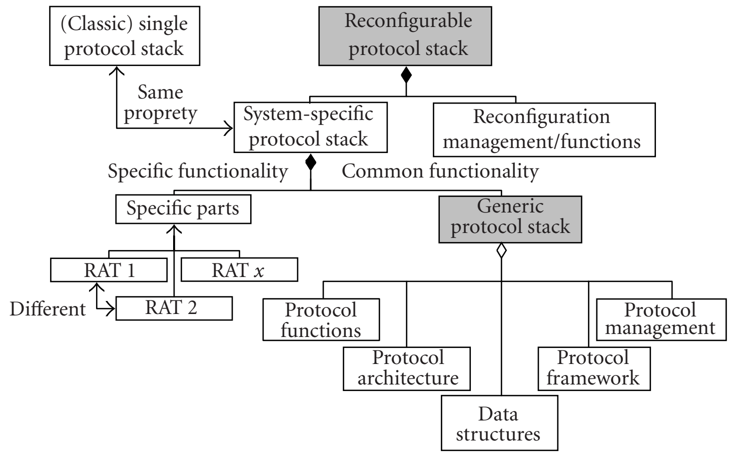


Figure 1: UML diagram of the generic protocol stack in the context of protocol reconfigurability.

Similar goals have been formulated and followed in the software engineering domain. The x-Kernel [10] architecture composes a protocol graph of protocol components together into a system, but the approach does not permit dynamic reconfiguration of the protocol graph and does not specifically target wireless communications. The DARPA active networking program [11] tries to answer the key question of location (and nature) of programmability with the perspective to build a flexible distributed computing system, again with a focus on fixed networks.

This paper consolidates previous publications [12, 13] and deduces summarizing conclusions. After introducing the idea of a generic protocol stack in Section 2, its application for protocol reconfigurability in the context of a multimode capable protocol architecture is outlined thereafter. The realization of a generic protocol stack, based on 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. Section 5 introduces the composition of system-specific layers at the example of the UMTS *radio link control* (RLC) layer, the *transmission control protocol* (TCP), and the IEEE 802.11 *medium access control* (MAC) layer, which differ in their development history as well as in their layer classification corresponding to the ISO/OSI reference model. A segmentation/reassembly module and an *automatic repeat request* (ARQ) protocol module are validated and evaluated analytically and through simulations in Section 4. These modules are used with different parameterization for composing the above-mentioned three specific protocol layers.

2. THE GENERIC PROTOCOL STACK

The rationale for approaching a generic protocol stack is that all communication protocols have common functions. These commonalities can be exploited to build an efficient multimode capable wireless system. The aim is to gather these common parts in a single generic stack and specialize this generic part. Thereby, the particular requirements of the targeted RAT are considered, as depicted in Figure 1. The targeted advantages of this concept are runtime reconfigurability and maintainability, code/resource sharing, protocol development acceleration through reusability, and continuous performance evaluation in the context of *quality-of-service* (QoS) dimensioning.

The initial step towards a generic stack is a detailed, layer-by-layer analysis of communication protocols to identify their similarities. The realization of generic parts is crucial for the success of the proposed concept in the face of a tradeoff between genericity, that is, general usability, and implementation effort. As depicted in Figure 1, the generic protocol stack comprises

(i) fundamental protocol functions;
(ii) a protocol architecture;
(iii) data structures;
(iv) a protocol framework;
(v) protocol management.

They form, together with RAT specific parts, a system-specific protocol stack. An efficient multimode capable reconfigurable 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 Section 3.

2.1. *Two complementing approaches for realization*

There are in general two possibilities from the software engineering perspective 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 considered in [14, 15]. Additionally, [16] 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 DLL. However, the link layer protocols are not the only protocols that have to be considered in a multimode capable network but the complete protocol stack. This implies, for instance, higher layer functions as the control and management of the radio resources as well as mobility.

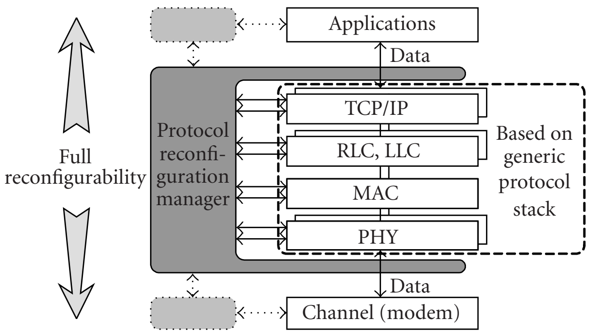


Figure 2: A reconfigurable protocol stack based on generic functional modules in the context of a completely reconfigurable terminal.

2.2. 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, developers have fallen back on well-proven functions and mechanisms which are adopted to the specific requirements of their application. The approach towards an efficient multimode 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 introduced above as one basis for a generic protocol stack, contrary to [14, 15] 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. The functions of segmentation/reassembly or an ARQ protocol used for error correction are an example for this. They are located in the RLC as well as in the transport layer, namely in the TCP.

3. ENABLING RECONFIGURABILITY

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

3.1. Protocol reconfiguration manager

The protocol reconfiguration manager has thereby the following tasks:

(i) management of the (permanently or temporally) parallel existing protocols and protocol stacks;
(ii) creation, destruction and/or reconfiguration of a single protocol or complete protocol stack;
(iii) administration of the user data flow during the reconfiguration process, as for instance the redirection of the user data from the old to the reconfigured protocol stack;
(iv) cross-layer optimization through protocol convergence, as introduced in the next section. This implies for example the transfer of protocol or user data from the old stack to the new one;
(v) support and enabling of reconfiguration functions of the network, as for instance the support of a network-initiated reconfiguration or an update of the network information about the status of the terminal.

The reconfiguration of the protocol stack, administrated through the protocol reconfiguration 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 reconfiguration of the existing protocol stack in exchanging the parameterization of the corresponding modules.

3.2. Protocol convergence in future wireless networks

The generic protocol stack enables the protocol convergence in future wireless networks. The convergence of such multimode 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.

Figure 3 depicts the transition between two protocol stacks of different-air interface modes of a wireless network. The different PHY options of IEEE 802.16 could serve as an example, see [3]. Presently, these PHY options are not envisaged to coexist in terminal or access equipment, although they share a common MAC protocol with very little option-specific extensions. 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 PHY, 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 have certain requirements to the

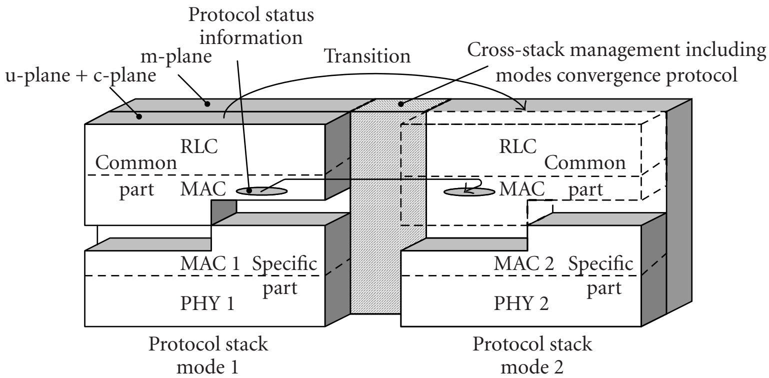


Figure 3: A multimode capable protocol architecture under consideration of an optimized protocol convergence. Protocol information is transferred between two modes, here exemplary depicted on the level of the MAC layer. This reflects, for instance, the case of IEEE 802.16 [3].

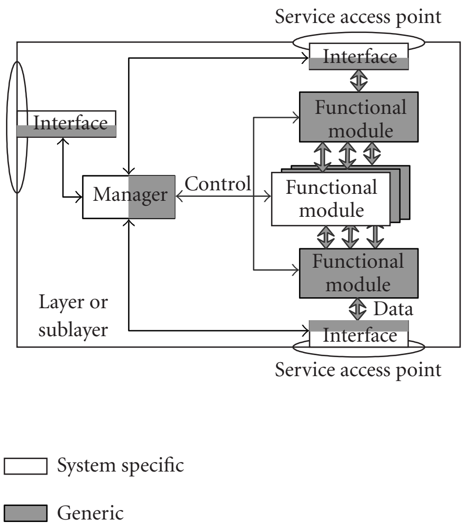


Figure 4: Composition of a protocol specific layer or sublayer on the basis of generic and system-specific functional modules.

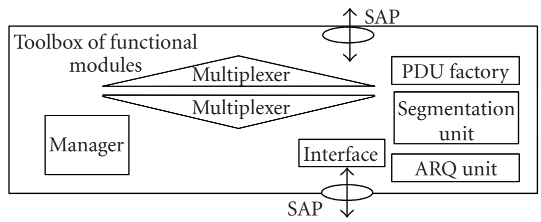


Figure 5: Toolbox of functional modules as part of the generic protocol stack.

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. 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 [17].

4. MODULAR APPROACH—THE GENERIC PROTOCOL STACK AS TOOLBOX OF PROTOCOL FUNCTIONS

Again, the generic protocol stack is the realization of the common parts, as illustrated in Figure 1, and implements its common functions on the basis of 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 Figure 4. The communication inside the layer is performed with the help of generic data structures, that is, generic service primitives and generic *protocol data units* (PDUs), which are also considered as being a part of the generic stack, see again Figure 1. The functional modules form a toolbox of protocol functions as illustrated in Figure 5.

A unique manager as well as interfaces for the *service access points* (SAPs) to the adjacent layers complete the fully functional protocol layer as depicted in Figure 4. In detail, the mentioned components have the following tasks.

(i) *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.

(ii) *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's internal communication, as for instance the connection of the layer's modules through generic service primitives. It is the layer's counterpart

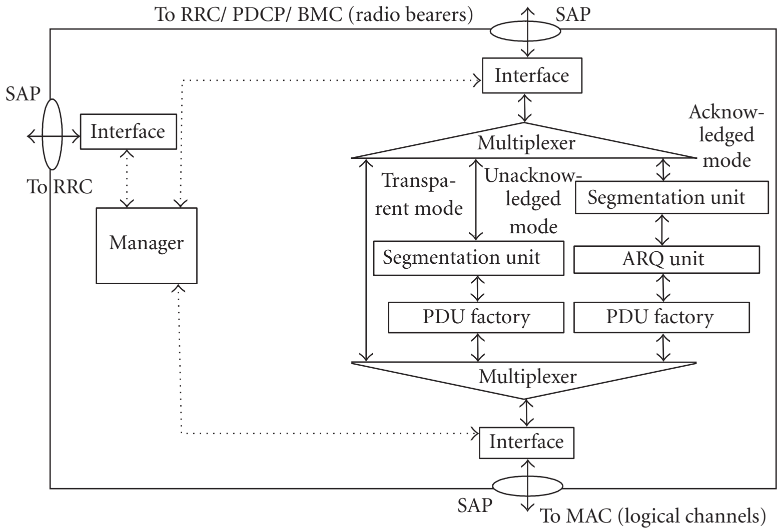


Figure 6: UMTS RLC layer based on the functional modules of the generic protocol stack.

of the protocol reconfiguration manager as introduced above in Figure 2 and it realizes the reconfigurability of the layer.

(iii) *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.

(iv) *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 the 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.

(v) *PDU factory* (as functional module, later depicted in Figures 6, 7, and 8): 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 later.

4.1. Generic protocol functions of the data link layer

Taking Section 2.2 into account, the following functions of the DLL are considered as being part of the generic protocol stack:

(i) error handling with the help of forward error correction (FEC) or ARQ protocols as for instance Send-and-Wait ARQ, Go-back-N ARQ*, or Selective-Reject ARQ;
(ii) flow control*;
(iii) segmentation, concatenation, and padding of protocol data units (PDUs)*;
(iv) discarding of several-times received segments*;
(v) reordering of PDUs*;
(vi) multiplexing/demultiplexing of the data flow, as for instance the mapping of different channels*;
(vii) dynamic scheduling;
(viii) ciphering;
(ix) header compression.

The asterisk $*$ marked functions are considered in this paper while the other functions are target of the authors'ongoing work.

4.2. Parameterization of functional modules

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. Thus, the parameterization of functional modules may mean (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:

(i) use of concatenation;
(ii) use of padding, that is, filling up of the PDU to reach a certain size;
(iii) transmitter or/and receiver role;
(iv) buffer size for SDUs concatenated in a single PDU;
(v) size of PDU after handling;
(vi) behavior in case of error, that is, interworking with ARQ module.

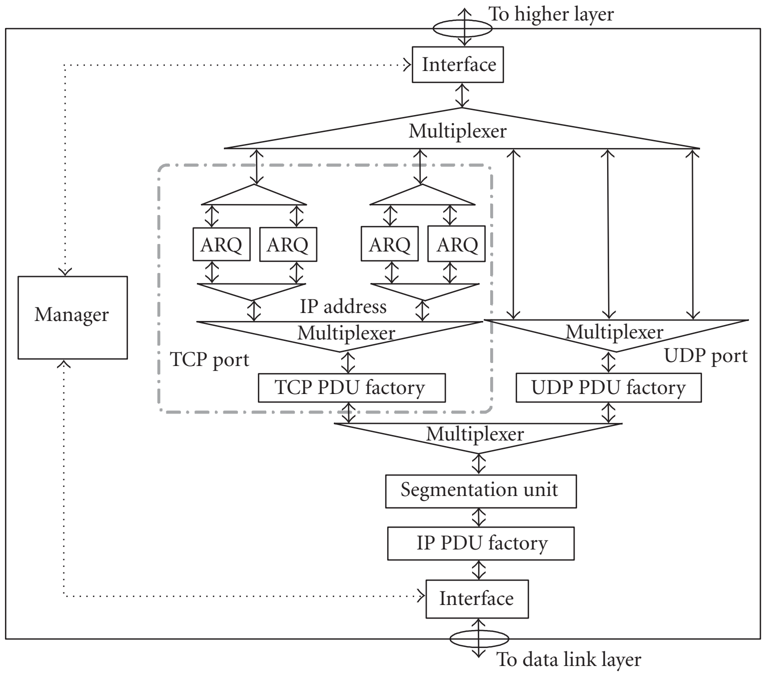


Figure 7: TCP, IP, and UDP layers based on the functional modules of the generic protocol stack. The TCP layer (gray dash-dotted line) is considered here.

At the example of the ARQ module, the parameterization means the specification of

(i) ARQ protocol characteristic, for instance Go-Back-N ARQ or Selective-Reject ARQ;
(ii) transmitter or/and receiver role;
(iii) receive and transmission window sizes;
(iv) fixed, variable (TCP) window length or open/shut mechanism (LLC);
(v) timer value, after a packet is assumed to be lost;
(vi) 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);
(vii) use of *negative acknowledgments* (NACKs).

5. 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 based on the toolbox of protocol functions of Figure 5, 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 Figure 6, (2) a TCP, IP, and UDP layers in Figure 7, and (3) an IEEE 802.11 MAC layer in Figure 8. The consideration of Figure 7 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. The performance of these three layers is evaluated in the subsequent section.

6. 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, a model of an erroneous transmission channel and statistical evaluation methods. This section introduces the modular approach to protocol functions in focusing on the segmentation in UMTS and the error correction through ARQ protocols in TCP and 802.11. The adequateness of the modules for being used in a multimode capable protocol stack is shown. The modules are parameterized corresponding to a specific protocol layer and their simulative behavior is compared to analytical models known from the literature.

6.1. UMTS radio link control layer

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. It may also be motivated through the capacity of a transport channel offered by the physical layer, using the notation of UMTS.

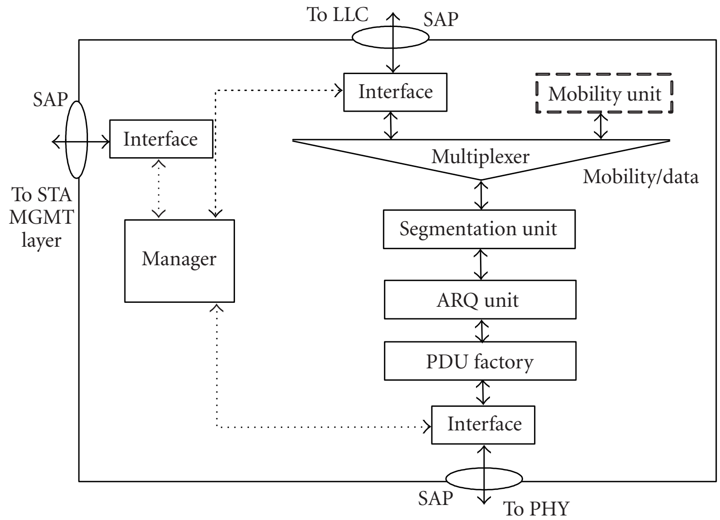


Figure 8: 802.11 MAC layer based on the functional modules of the generic protocol stack.

In case of multiple users sharing a common channel, the segmentation can increase the channels efficiency in having a multiplexing gain. Concatenation increases thereby the channel utilization as it is outlined in Figure 9. The channel utilization as quotient of user *payload* and available *channel capacity* is given through

$$\frac{\text{payload}}{\text{channel capacity}} = \frac{l_{\text{payload}}}{\lceil l_{\text{payload}}/\text{packet size}\rceil \cdot \text{packet size}} \qquad (1)$$

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, that is, the effectively used capacity compared to the amount of provided capacity, is decreased and follows a "zigzag" behavior when an additional physical channel is used as illustrated by the solid line in Figure 9.

The introduced example of Figure 9 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 bytes. The simulative results with and without concatenation are given by the markers. The payload packet size, that is, the user data, is increased up to 500 bytes and the channel utilization as introduced above 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 Figure 9 for $l_{\text{payload}} = 384$ bytes in the case of no concatenation. The same stands for the usage of concatenation, as the observed channel utilization does not match 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 multimode capable protocol stack.

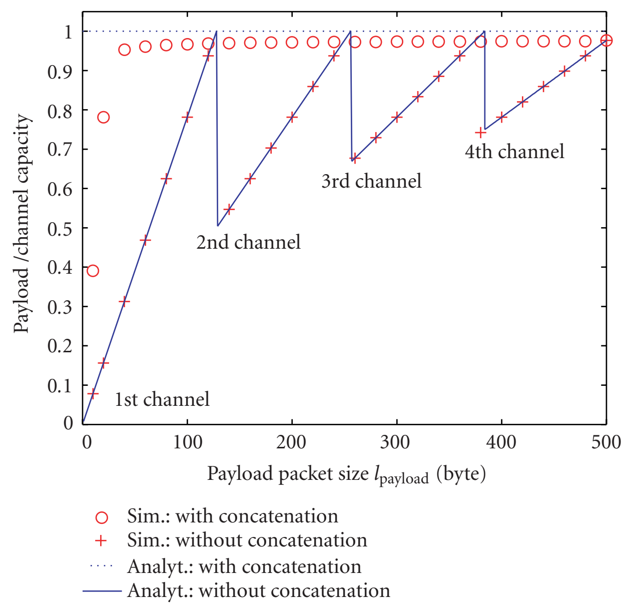


Figure 9: Utilization of channels with a fixed packet segmentation size of 128 bytes.

6.2. *Transmission control protocol layer*

As introduced above in Figure 7, a TCP layer can be composed out of the functional modules of the DLL as being

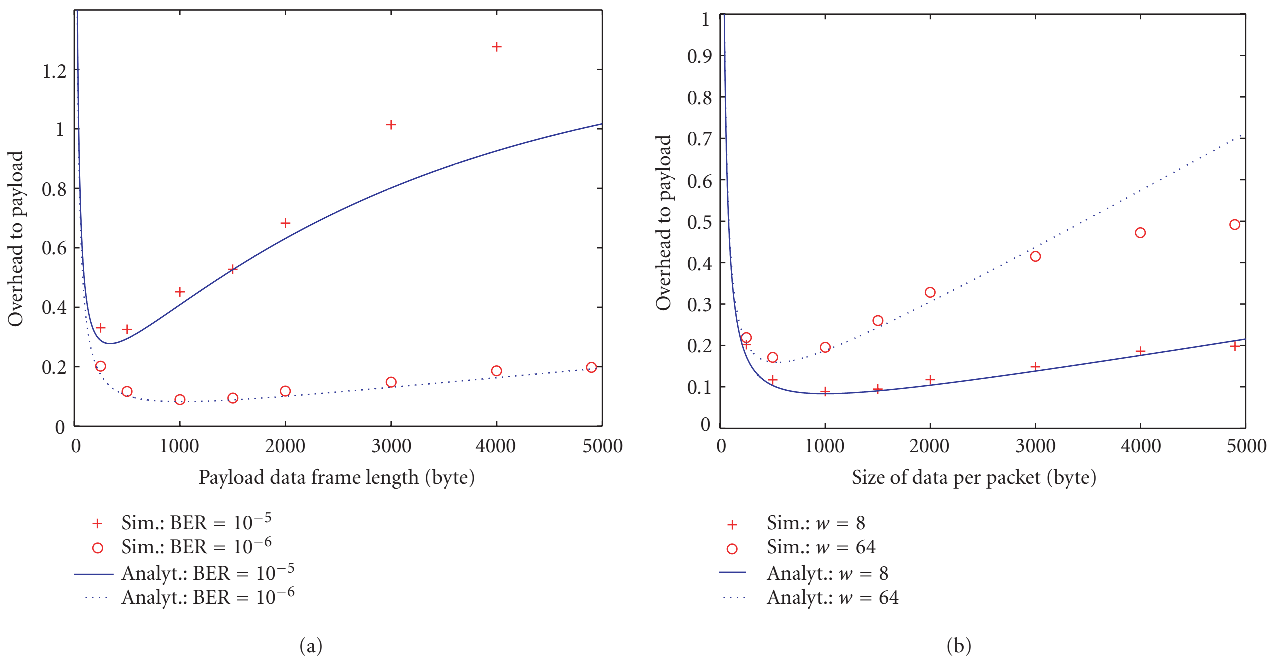


FIGURE 10: TCP layer Go-back-N ARQ validation and evaluation. The protocol overhead to payload ration in dependency on the frame length of the payload data is depicted. The lines are analytic results corresponding to (2), while the markers indicate simulative evaluation. (a) Varied BER $= 10^{-5}$ or 10^{-6} and window length $w = 8$. (b) Varied window length $w = 8$ or 64 and BER $= 10^{-6}$.

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 erroneous transmissions. The focus is thereby on the influence of two effects: the bit error ratio (BER) of the radio channel, that is, the wireless medium, and the size of the send and receive window w.

With a packet length of $l_{\text{packet}} = l_{\text{header}} + l_{\text{payload}}$, where TCP has fixed header length of $l_{\text{header}} = 40$ bytes, the packet error ratio (PER) can be calculated to

$$\text{PER} = 1 - (1 - \text{BER})^{l_{\text{packet}} \cdot 8}. \tag{2}$$

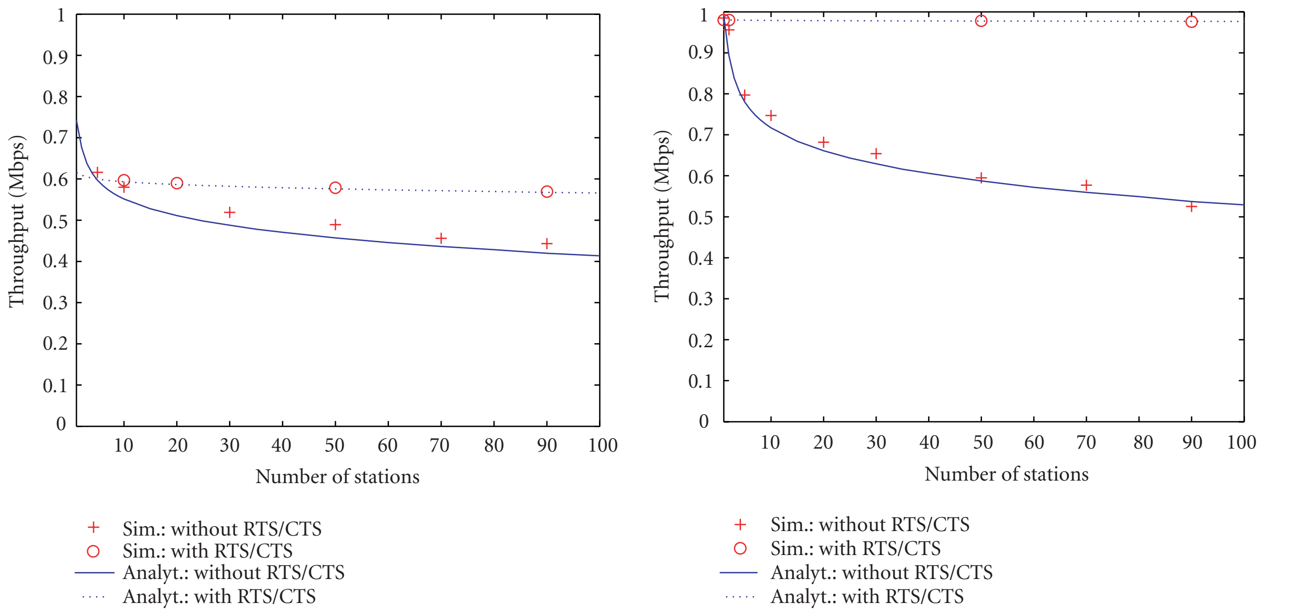
Based hereon the overhead to payload quotient for the Go-Back-N ARQ can be derived and approximated [12] to

$$\frac{\text{overhead}}{\text{payload}} = \frac{l_{\text{header}}}{l_{\text{payload}}} + \frac{\sum_{i=1}^{w/2} \left((w/2) - i + 1\right) \cdot (1 - \text{PER})^{i-1} \cdot \text{PER}}{w/2} \cdot \frac{l_{\text{packet}}}{l_{\text{payload}}}, \tag{3}$$

where w is the length of the transmission window leading to the analytical results as depicted in Figure 10. This figure illustrates the overhead to payload ratio in dependency on the frame length of the payload data for (a) a BER of 10^{-5} and 10^{-6} on the one hand and (b) window length of 8 and 64 on the other hand. Figure 10a 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 the said ratio can be determined. From the cross-protocol optimization perspective, this frame length may be used as a dimensioning rule for segmentation. The same stands for Figure 10b: 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. The difference between analysis and simulation for large payload data frames in Figures 10a and 10b is reasoned by the send/receive buffers of the implemented TCP layers. Data packets in these buffers have to be discarded when an error is detected and are neglected in the approximation (3). In summary, the ARQ module of the generic protocol stack fulfils adequately its intended purpose.

6.3. IEEE 802.11 medium access control layer

In this section, the modular composition of an IEEE 802.11 MAC layer as illustrated in Figure 8 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.



(a) (b)

Figure 11: 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 (5), while the markers indicate the simulative evaluation. Throughput evaluation-channel capacity=1Mbps; (a) packet size=128 bytes, (b) packet size=4096 bytes.

Table 1: Time slot durations in microseconds and probabilities that the medium is empty (e), successfully (s) allocated, or a collision (c) occurs [18, 19]. The fixed values result from the time length of an RTS/CTS sequence.

Probability	Duration with RTS/CTS	Duration without RTS/CTS
$p_e = (1-\tau)^n$	$T_e = 1$	$T_e = 1$
$p_s = n\tau(1-\tau)^{n-1}$	$T_s = 636 + 8l_{\text{payload}}$	$T_e = 636 + 8l_{\text{payload}}$
$p_c = 1 - p_e - p_s$	$T_e = 170$	$T_s = 234 + 8l_{\text{payload}}$

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 [18, 19] resulting into

$$p = 1 - (1-\tau)^{n-1}, \quad \tau = 2 \cdot \left(1 + W_0 + pW_0 \frac{1-(2p)^8}{1-2p}\right)^{-1}, \tag{4}$$

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 Table 1, the average total system throughput $t_{\text{saturation}}$ can be calculated to

$$t_{\text{saturation}} = \frac{P_s L_{\text{payload}}}{T_{\text{average}}}, \quad T_{\text{average}} = P_e T_e + P_s T_s + P_c T_c. \tag{5}$$

We assume a channel rate of 1 Mbps. The slot length, *short interframe space* (SIFS) and *distributed coordination function IFS* (DIFS) are 1, 6, and 10 microseconds resulting into the analytical as well as simulative results of Figure 11. There, the overall system throughput, with and without RTS/CTS, in dependency on the number of stations is depicted. For small packets, $L_{\text{payload}} = 128$ bytes, Figure 11a, the headers are the main cause for an inefficient use of the medium. For larger frames, $L_{\text{payload}} = 4096$ bytes, Figure 11b, 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 (5) 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 [19], this module can be legitimately used in an 802.11 MAC layer.

7. CONCLUSION

The introduced concept of a generic protocol stack enables protocol software for future multimode capable systems under the consideration of protocol reconfiguration. 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 has 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 toolbox of protocol functions as a construction kit for protocol development. Dimensioning rules for an adequate support of QoS in wireless communication from the perspective of the protocols can be derived. 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. 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 advantage of this approach. The efficiency of protocol reconfigurability benefits from the introduced generic approach and implies a clearly identified effort of protocol management. Thus, the introduced approach is a first step to an end-to-end reconfigurable wireless system.

ACKNOWLEDGMENTS

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). This work has been continously funded from third parties' grants.

REFERENCES

[1] B. Walke, R. Pabst, and D. Schultz, "A mobile broadband system based on fixed wireless routers," in *Proc. International Conference on Communication Technology (ICCT '03)*, vol. 2, pp. 1310–1317, Beijing, China, April 2003.

[2] R. Pabst, B. Walke, D. Schultz, et al., "Relay-based deployment concepts for wireless and mobile broadband radio," *IEEE Commun. Mag.*, vol. 42, no. 9, pp. 80–89, 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," *IEEE Commun. Mag.*, vol. 33, no. 5, pp. 26–38, 1995.

[5] W. Tuttlebee, *Software Defined Radio: Enabling Technologies*, Wiley Series in Software Radio, John Wiley & Sons, New York, NY, USA, 2002.

[6] J. Mitola, *Cognitive Radio*, Ph.D. thesis, KTH, Stockholm, Sweden, 2000, pp. 45–90.

[7] P. Mähönen, "Cognitive trends in making: future of networks," in *Proc. IEEE 15th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC '04)*, Barcelona, Spain, September 2004, invited paper.

[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] N. C. Hutchinson and L. L. Peterson, "The X-Kernel: An architecture for implementing network protocols," *IEEE Trans. Software Eng.*, vol. 17, no. 1, pp. 64–76, 1991.

[11] J. M. Smith and S. M. Nettles, "Active networking: one view of the past, present, and future," *IEEE Trans. Syst., Man, Cybern. C*, vol. 34, no. 1, pp. 4–18, 2004.

[12] 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. 7th International Symposium on Wireless Personal Multimedia Communications (WPMC '04)*, pp. 5–5, Abano Terme, Italy, September 2004.

[13] L. Berlemann, A. Cassaigne, R. Pabst, and B. Walke, "Modular link layer functions of a generic protocol stack for future wireless networks," in *Proc. Software Defined Radio Technical Conference (SDR '04)*, pp. 6–6, Phoenix, Ariz, USA, November 2004.

[14] M. Siebert and B. Walke, "Design of generic and adaptive protocol software (DGAPS)," in *Proc. International Conference on 3rd Generation Wireless and Beyond (3Gwireless '01)*, San Francisco, Calif, USA, June 2001.

[15] L. Berlemann, M. Siebert, and B. Walke, "Software defined protocols based on generic protocol functions for wired and wireless networks," in *Proc. Software Defined Radio Technical Conference (SDR '03)*, Orlando, Fla, USA, November 2003.

[16] J. Sachs, "A generic link layer for future generation wireless networking," in *Proc. IEEE International Conference on Communications (ICC '03)*, Anchorage, Alaska, USA, May 2003.

[17] L. Berlemann, R. Pabst, M. Schinnenburg, and B. Walke, "A reconfigurable multi-mode protocol reference model facilitating modes convergence," in *Proc. 11th European Wireless Conference (EW '05)*, Nicosia, Cyprus, April 2005.

[18] G. Bianchi, "Throughput evaluation of the IEEE 802.11 distributed coordination function," in *Proc. 5th Workshop on Mobile Multimedia Communications (MoMuc '98)*, Berlin, Germany, October 1998.

[19] A. Hettich, "Leistungsbewertung der Standards HIPERLAN/2 und IEEE 802.11 für drahtlose lokale Netze," Ph.D. dissertation, RWTH Aachen University, Aachen, Germany, 2001, ABMT 23, http://www.comnets.rwth-aachen.de.

Lars Berlemann was born in Aachen, Germany, in 1977. After having absolved a student internship at Infineon Technologies, San Jose, he received his Diploma degrees in electrical engineering and economics from Aachen University in 2002 and 2003, respectively. Since 2002, he served as a Research Assistant at the Chair of Communication Networks (with Professor B. Walke) of RWTH Aachen University, where he is working towards his Ph.D. degree. He is a scholarship holder of the Research College (Graduiertenkolleg) "Software for Mobile Communication Systems" of the German Research Foundation (DFG) and received the "Siemens Mobile Academic Award" in 2003. He is responsible for the organization of the scientific program of European Wireless Conference 2005 and PIMRC 2005. His research work has so far been taken up by various national and European Union funded research projects, such as the Academic Network for Wireless Internet Research in Europe (ANWIRE), Wireless World Initiative New Radio (WINNER), and CoCoNet. His research interests include cognitive radios, distributed QoS support in spectrum sharing scenarios, spectrum etiquette, and flexible protocol stacks. He is the author/coauthor of 18 reviewed scientific publications and he is a Student Member of the IEEE ComSoc and ITG/VDE.

Ralf Pabst was born in Cologne, Germany, in 1974. After having absolved a student internships at SIEMENS ICN and Vodafone Germany, he received his Diploma in electrical engineering from Aachen University in 2001 and served since then as a Research Assistant at the Chair of Communication Networks (with Professor B. Walke) of Aachen University, where he is working towards his Ph.D. degree. He has been working on various national and European Union funded research projects, such as the German National Project Multifunk, and the EU FP5 IST projects; Wireless Strategic Initiative (WSI), Wireless World Research Initiative (WWRI), Network of Excellence in Wireless Applications and Technology (NEXWAY), and Academic Network for Wireless Internet Research in Europe (ANWIRE). He has been actively involved in the organization of the WG4 in the Wireless World Research Forum (WWRF). His current research activities are focused on the Wireless World Initiative New Radio (WINNER) Project under the 6th Framework Program (FP6) of the European Union. His research interests include the performance evaluation of relay-based deployment concepts for next-generation wireless networks, associated resource management protocols, and spectral coexistence of wireless standards. He is the author/coauthor of 19 scientific publications, including 2 journal articles and 2 textbook chapters.

Bernhard Walke for the last 13 years is running the Chair for Communication Networks, RWTH Aachen University, Germany, where about 35 researchers work under his guidance on topics like air-interface design, formal specification of protocols, fixed network planning, development of tools for stochastic event-driven simulation, and analytical performance evaluation of services and protocols of XG wireless systems. During that time, he has supervised more than 650 M.S. theses and 43 Ph.D. theses covering most aspects of fixed and mobile communication networks. He has published more than 120 reviewed conference papers, 25 journal papers, and seven textbooks on architecture, traffic performance evaluation, and design of future communication systems. Prior to joining academia, Professor Walke worked in various industry positions for AEG-Telefunken (now part of EADS AG). Professor Walke holds Diploma and Doctorate degrees in electrical engineering, both from the University of Stuttgart, Germany. He is the Cochair of the PIMRC 2005, Berlin, Steering Board Member of the annual European Wireless Conference, Senior Member of IEEE ComSoc, and Member of ITG/VDE and GI.