Application of Functional Unit Networks to Next Generation Radio Networks

Marc Schinnenburg, Fabian Debus, Ralf Pabst
Communication Networks
RWTH Aachen University
Aachen, Germany
{msg | fds | pab}@comnets.rwth-aachen.de

Abstract— **This paper presents a framework for building re-configurable protocol stacks. A high degree of re-configurability is achieved through composing complex behavior of a communication system using Functional Units, forming Functional Unit Networks. The feasibility of Functional Unit Networks and its application to next generation radio networks will be discussed. The applicability of Functional Unit Networks to wireless communication systems is exemplarily shown in the context of a current research project regarding next generation radio networks (WINNER).**

Reconfigurability, Flexible Protocol Stack, Multi-Mode Architecture

I. Introduction

Ubiquitous radio access at high data rates and low delays is the customer's expectation at next generation communication systems. To meet this expectation the protocols of future communication systems need to efficiently exploit the available spectrum in a dynamic way. The need to achieve optimal performance in a variety of different environments (e.g., indoor/outdoor) will force devices and their protocols to adapt themselves to the current situation.

The Wireless World Initiative New Radio (WINNER) is a European research project funded under the 6th Framework research funding Program (FP6) of the Commission of the European Union addressing the design of a next generation radio network. Among the requirements for WINNER are the ubiquitous, spectrally efficient radio access at high data rates and low delays, embedded into a unified radio access technology. An efficient adaptation of the system to different environments and scenarios (such as short-range vs. wide-area, LOS- vs. NLOS propagation etc.) is therefore inevitable. Such an adaptation in many cases will take place in terms of switching between algorithms of certain air interface functions or changing between (sets of) parameters. In some cases, the adaptation may even involve changing the behavior of the air interface, e.g., by switching to another duplex scheme. Different duplex schemes are in the WINNER terminology referred to as so-called different "physical layer modes" of operation. These modes represent the highest degree of adaptation and optimization to different scenarios.

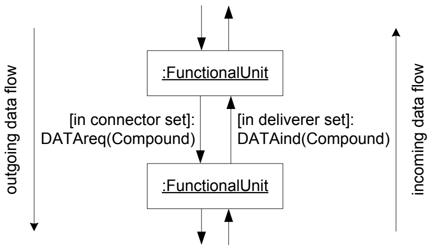

Figure 1. FU connections for compound handling.

The required degree of adaptivity and the need for reconfigurability puts high demands on the design and implementation of future communication systems and protocols. The inherent complexity can lead to costly and time-consuming standardization and implementation processes.

To cope with the rising complexity of wireless communication systems the authors propose the composition of protocols using Functional Unit Networks [2]. Functional Units serve as basic building blocks for the aggregation of protocol functionality and enable the protocol designer to efficiently design flexible protocol stacks. This structure is also considered beneficial in systems that are not based on different Physical Layer Modes, because multiple modes of operation are typically also employed in higher protocol layers, e.g., as a consequence of different service-types and/or Quality of Service (QoS)-requirements.

This paper gives a brief overview of Functional Unit Networks along with the application of Functional Unit Networks in the context of WINNER.

II. Functional Unit Networks

As discussed in [3] Data Link Layers (DLLs) of protocol stacks of wireless communication systems in general comprise among others the following set of functions: Automatic Repeat request (ARQ), Segmentation and Reassembly (SAR), scheduling, multiplexing and buffering.

In [2] a framework for implementing functions of a DLL as FUs and creating complete protocol layers by interconnecting FUs is presented.

A Functional Unit Network (FUN) is built by connecting FUs. FUs within a FUN mainly communicate by propagating compounds. A compound is an arbitrary chunk of data together with a pool of commands, where a command denotes the control information provided by every FU.

The interface of FUs has been identified to consist of five different aspects:

1. **Compound Handler**
   Implement the handling of compounds of an FU including intra FUN flow control. Handling of compounds includes mutation, dropping, injection and forwarding.

   The methods provided are:

   - `DATAind(Compound)`
     Receive a compound in the incoming flow (figure 1).
   - `DATAreq(Compound)`
     Receive a compound in the outgoing flow (figure 1).
   - `wakeup()`
     Try to forward compounds as part of flow control.
   - `isAccepting(Compound)`→`Boolean`
     Give permission to FUs to propagate the given compound as part of flow control.
2. **Command Type Specifier**
   Define the type of command provided by the FU. This type will be used to create an initial command pool and to verify unit dependencies as will be discussed in section 3.
3. **Connector**
   Hold the set of FUs that compounds will be delivered to in the outgoing direction. Define a strategy to select the appropriate FU for a given compound
4. **Receptor**
   Hold the set of FUs in which the FU itself is in the connector set. Define a strategy to wake up FUs.
5. **Deliverer**
   Hold the set of FUs that compounds will be delivered to in the incoming direction. Define a strategy to select the appropriate FU for a given compound.

A. Flow Control

As discussed in [2], there is a need for intra FUN flow control. To implement flow control between FUs, two methods are necessary:

- `isAccepting(Compound)` → `Boolean`
- `wakeup()`

Before an FU is allowed to deliver a compound to another FU using `DATAreq`, it has to ask for permission using the `isAccepting` method. If the response is negative, it may not send a compound to the questioned unit.

When an FU can not deliver further compounds, it cannot proceed and thus ceases operation until it is triggered again. Such triggers can come from new compounds being delivered, timers expiring, but it may as well happen that an FU in its connector set changes its state to accept compounds again.

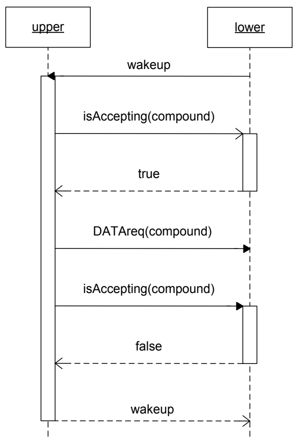

Figure 2. Intra node flow cotntrol.

The method used for informing other FUs that they might succeed in sending a compound is `wakeup`. A set of FUs that have to be notified when an FU is willing to accept new compounds is called receptor set. The receptor set of an FU "A" contains exactly those FUs that have FU "A" in their connector set.

Figure 2 shows an example of two FUs transmitting compounds with respect to intra node flow control.

B. Flow Separation

FUs as described have a cohesive responsibility within a FUN. In terms of object-oriented design a FU represents a class, whereas a FUN is composed of FU instances. Thus, every FU comprises state and behavior. An SAR unit for example needs to store segments of compounds to be able to apply segmentation and reassembly.

A FUN being part of a protocol stack that is supposed to handle multiple connections needs to hold separate states for each of the connections. One option is to let each FU maintain the states for different flows and to select the appropriate state for the processed compounds. This approach has a drawback: It complicates the implementation of FUs, since it burdens the FU with the maintenance of different, flow specific states. Another option is to implement a specialist in separating flows

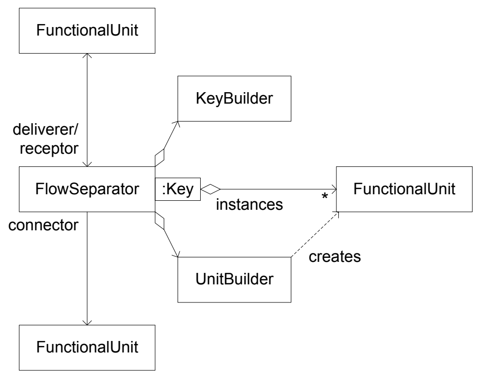

Figure 3. FlowSeparator in a FUN.

and maintaining states. Thus, keeping FUs simple, leaving them unaware of flows. Every FU implements functionality and holds state for exactly one flow. The specialized flow separator itself again is a FU, separating flows using a configurable separation strategy and delegating requests to the according FU instance.

Figure 3 shows the structure of flow separation. The `FlowSeparator` itself is a FU. Like any other FU it is connected to other FUs using the deliverer, receptor and connector sets. It is configured using a `KeyBuilder` and a `UnitBuilder`. The `KeyBuilder` is a strategy to extract information from a compound that is sufficient to distinguish between compounds of different flows. The keys generated by the `KeyBuilder` serve as key to an instance container. For every compound, the flow separator creates a new key, inspects its instance container and delegates the request to the according instance. In case of receiving a compound that has no available instance, yet, the flow separator creates a new instance using the `UnitBuilder`.

C. *Configurability*

The high degree of configurability of protocol stacks using FUNs is achieved by allowing configuration at several levels. The levels of configurability in order of increasing abstraction are:

Parameterization level: The lowest level of configuration includes the parameterization of concrete FUs: What is the window size of the selective repeat ARQ unit? What is the Maximum Transfer Unit (MTU) of the SAR unit?

Concretion level: The next higher level focuses on the selection of concrete FUs to fill the respective places in a FUN. Concrete implementations have to be chosen for intended protocol functions.

Layout level: The highest level of configuration comprises the placement of protocol functions in a stack: the scaffolding of a protocol stack, including the interconnections of FUs and their intended functions. The order in which certain processing is applied to compounds as well as the overall set of supported messages is determined at this level of abstraction.

III. FUNS IN THE CONTEXT OF WINNER

The basic protocol architecture as proposed in [6] aims at providing a framework which enables all three levels of adaptation mentioned above. The goal is to allow a flexible configuration of the WINNER protocols and the efficient integration of multiple potential WINNER modes in a complementary way, thereby allowing to take maximum benefit of the commonalities between the modes (see [1]). As a consequence, such a reference model requires protocols that conform to the structure given in figure 4. In order to exploit commonalities between different modes of operation, the software of these protocols would (i) ideally follow a modular approach to allow a high degree of reusability and (ii) provide suitable structures and interfaces for the flexible composition of the individual modules.

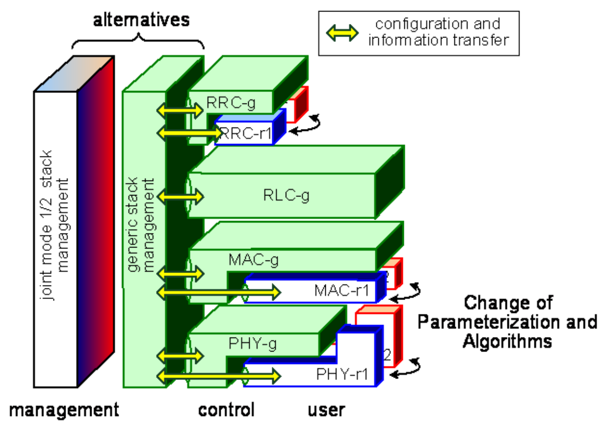
alternatives
configuration and information transfer
joint mode 1/2 stack management
generic stack management
RRC-g
RRC-r1
RLC-g
MAC-g
MAC-r1
PHY-g
PHY-r1
2
Change of Parameterization and Algorithms
management
control
user

Figure 4. Overview of Layered Protocol Architecture and Management Plane Interaction

Another main requirement towards the structure of the WINNER Reference Protocol Architecture is to match the layered service architecture as proposed in [7] and [8].

To match the requirements about modular composition of the functionality of the protocol layers which result from the reference model in figure 4, the set of functions identified above have been decomposed into a set of FUs. We further describe the identified FUs and how they are connected to form the FUN.

Figure 5 and figure 6 exemplarily show how the intended functionality of the Data Link Layer-User Plane - as it is currently discussed within WINNER - can be composed out of a set of mode independent FUs and a small number of mode-specific FUs. The used units can be further subdivided into three different classes:

1. The common, system-independent functions, shown as light green boxes, these can be taken from a toolbox of generic protocol functions that can also be used to implement protocols for other, non-WINNER radio systems, examples are:
   - ARQ: figure 5 shows an upper ARQ for securing packets end-to-end over multiple radio hops and a lower ARQ that operates on a per-hop basis.
   - Buffers
   - Segmentation and Reassembly
2. The mode-independent, but WINNER-specific functions, shown as light gray boxes:
   - IP Convergence Layer
   - Service Classification: among others dealing with flow handling and addressing
   - Relay Inject Buffers: for PDU handling at the Relay Nodes
   - Service Level Controller (SLC): in the WINNER terminology, this is how the mode independent QoS scheduling and per-flow buffering is referred to. This unit also deals with flow addressing.

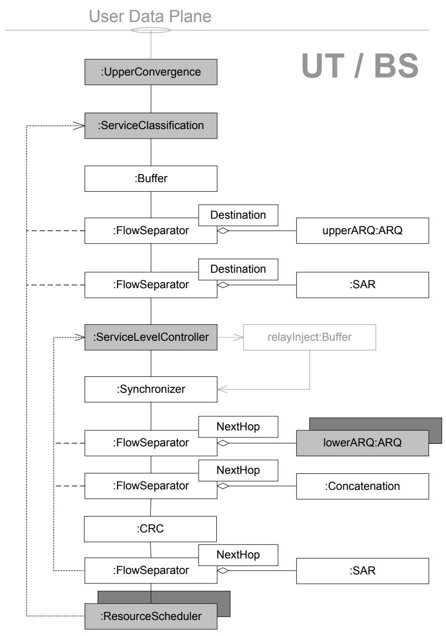

Figure 5. Functional Units in the user data plane of User Terminals / Base Stations

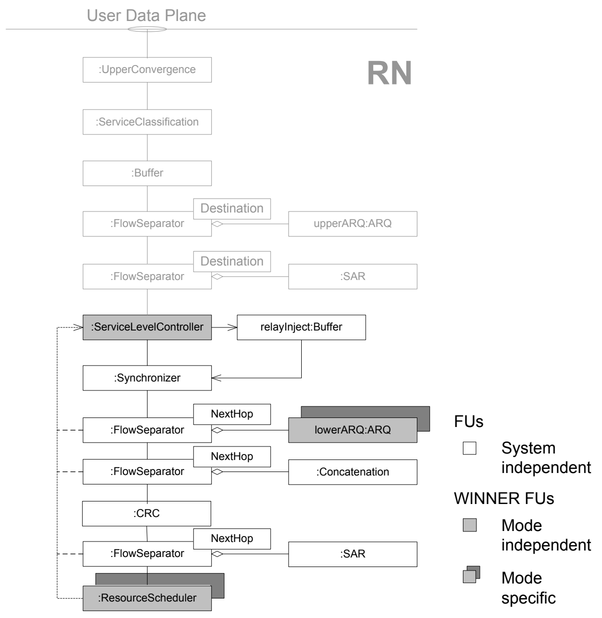

Figure 6. Functional Units in the user data plane of Relay Nodes

3. The WINNER-mode-specific functional units, which are shown as gray / dark gray boxes. Example for such functions are:
   - Resource Scheduler: performs the actual mapping of data flows onto physical resources. It is therefore a unit that is specific to a certain physical layer mode being employed.
   - ARQ: figure 5 shows a lower ARQ which is likely to be a Hybrid ARQ and thus also closely linked to the physical layer mode being used.

Note that the combination of Resource Scheduler and lower ARQ is referred to as MAC-r ('r' stands for radio-specific) in the reference model in figure 4. This also illustrates that common and mode-specific functionality can be arbitrarily located inside the protocol layer, since the FUs referred to as MAC-r do not necessarily have to be directly connected to each other.

The mode-specific parts are the only FUs that may be affected by layout-level adaptation to different WINNER physical layer modes. The mode-independent boxes may undergo a change of their parameter set, but their essential functionality remains the same. The change of the parameters would be the responsibility of the entity that manages the respective protocol layer. Figure 5 and figure 6 show that different subsets of FUs can be active in different FUNs. The grayed-out boxes denote the FUs that are unused at the BS/UT or RN respectively. A FUN implementing a RN requires an additional buffer ("relayInject") when forwarding Compounds. This buffer is unused in the FUN of a BS/UT. On the other hand the SLC of a RN will always forward Compounds and never deliver them to higher FUs, rendering all FUs above the SLC in every RN unused.

IV. Realization of Funictional Unit Networks

To investigate the feasibility of the presented concept of FUNs a reference implementation is developed as part of the Wireless Network Simulator (WNS) at ComNets. Besides the herein presented work on the WINNER project another simulator module is based on this reference implementation: IEEE 802.16 (WiMAX).

The experiences gathered so far have shown that the decomposition of protocols into FUs with distinct functionality simplifies and accelerates the implementation of protocols compared to monolithic realizations. The uniform interfaces and the mechanisms to avoid tight coupling between FUs directly leads to increased testability of protocol implementations: During the work on the reference implementation a number of patterns for testing FUs emerged. The uniform interfaces of FUs clearly contribute to the identification of such patterns. The current set of FUs is covered by extensive unit tests what led to a noticeably lower defect rate.

Today's protocol designs show the need for interfaces between protocol layers not directly connected to each other (often referred to as "cross-layer design") [9]. On the one hand FUNs facilitate the implementation of more distributed in contrast to strictly hierarchical protocol stacks, thus fulfilling the need for more direct information exchange between protocol entities. On the other hand FUNs strongly emphasize dependencies between protocol functions, since in they require any dependency to be made explicit. The direct information exchange of one FU with one or more other FUs is desirable in terms of protocol optimization. At the same time it introduces a number of drawbacks:

1. **Less felixibility:** FUs can only be part of a FUN if all their dependencies are met
2. **Increased complexity:** FUs behavior not only results from their own state and their limited interfaces but from a complex interworking with other FUs
3. **Worse testability:** Directly results from 1.) and 2.), since the test scenarios have to meet the dependencies and cover a larger state space

Besides the aforementioned work on the simulator modules, further investigations are needed for the reconfigurability of FUNs. In principle, FUNs can be modified at run-time, allowing for reconfigurability at all levels: Parameterization level, concretion level and layout level (see II.C).

The efforts required at the different levels of reconfiguration are quite different. Reconfiguration at parameterization level has already been looked into for the implementation of control plane functionality which is required to modify behavior of the user data plane.

Reconfiguration at the remaining levels is expected to require functionality residing in a management plane to handle the configuration of a protocol layer or even a whole protocol stack. This management functionality goes beyond the scope of this work and will be subject for future studies.

V. Conclusions

The presented protocol architecture achieves the high level of adaptability required from the WINNER system concept through composition of functionalities from FUs with cohesive responsibility. The main benefits of this concept for the system design are:

1. **Flexible design:** possibility to easily investigate different protocol options and logical ordering of functionalities and faster performance evaluation of concurrent design proposals
2. **Efficient design:** through increased re-use of FUs
3. **Reduced complexity of the design:** making dependencies between different functional units explicit helps to (i) understand (and question) their necessity and (ii) maintain the modular design approach
4. **Reliable design:** better testability of protocol software

The presented concept also opens up potential for an abstract description and with this the possibility of external configuration of the protocol stack and -layers via a formal description language. As indicated above, the layer and stack management tasks will be among the next research issues.

Acknowledgements

The authors would like to thank Prof. Dr.-Ing. B. Walke for the fruitful discussions and comments to the content of this paper. The presented work has partly been funded by the European Commission in the FP6 IST-Project WINNER (IST-2003-507581).

References

[1] Berlemann, L., Pabst, R., Schinnenburg, M. and Walke, B. H., "A Flexible Protocol Stack for Multi-Mode Convergence in a Relay-base Wireless Network Architecture", in 16th IEEE Conference on Personal, Indoor and Mobile Radio Communications, PIMRC 2005, Berlin Germany, September 2005

[2] Schinnenburg, M., Debus, F., Otyakmaz, A., Berlemann, L., Pabst, R., "A Framework for Re-configurable Functions of a Multi-Mode Protocol Layer", SDRforum'05, Orange County, USA, Nov. 2005

[3] Berlemann, L., Pabst, R., Walke, B. H., "Multimode Communications Protocols Enabling Reconfigurable Radios", in EURASIP Journal on Wireless Communications and Networking 2005:3, pp. 390-400

[4] Pabst, R. et al., "Relay-Based Deployment Concepts for Wireless and Mobile Broadband Radio," in IEEE Communications Magazine, vol. 42, no. 9, Sept. 2004, pp. 80-89.

[5] Gamma, E. et al, "Design Patterns", Addison-Wesley, 1995

[6] IST-2003-507581 WINNER WP3 D3.2: "Description of deployment concepts for future radio scenarios integrating different relaying technologies in a cellular infrastructure including definition, assessment and performance comparison of RAN protocols for relay based systems", February 2005.

[7] IST-2003-507581 WINNER WP2 D2.10: "Final report on identified key radio interface technologies, system concepts and their assessment"

[8] IST-2003-507581 WINNER WP7 D7.6: "WINNER System Concept Description"

[9] Srivastava, V., Motani, M., "Cross-Layer Design: A Survey and the Road Ahead", IEEE Communications Magazine, Dec. 2005