Design of Generic and Adaptive Protocol Software (DGAPS)

Matthias Siebert, Bernhard Walke

Communication Networks
RWTH Aachen University of Technology, Germany
E-Mail: {mst|walke}@comnets.rwth-aachen.de
WWW: http://www.comnets.rwth-aachen.de/~{mst|walke}

ABSTRACT

Within the evolution of mobile communication systems from second over third to fourth generation, existing air-interfaces will be improved and new ones will be added. Based on the experiences from today's mobile networks, it appears appropriate to extract and re-use the approved features for future systems. This approach is valid especially for protocol software. This paper introduces a new design technique for enhancing existing and developing future protocol stacks. It is proposed to identify commonalities of signalling, user data transfer and management protocols of e.g. ISDN BRI, GSM, DECT, 3G, Bluetooth and HiperLan/2. The goal is to develop a *generic protocol stack* to be complemented to become a dedicated air-interface protocol stack by adding the standard specific parts as supplements. This approach offers the potential to design highly adaptive and reusable protocol software for Software Defined Radios.

***Keywords* – Software (Defined) Radio, Generic Protocol Stack, Reconfigurable Protocol Software, Network Mobility, SW Reusability**

Introduction

The different existing air-interfaces together with the different services they support still require the use of multiple physical handsets. Cordless telephone systems like DECT and cellular systems like GSM are examples, where dedicated mobiles are being used. Similar situations apply if the mobile is to be used abroad. First solutions to this problem offer multi-band, multi-mode portables that mainly represent a number of dedicated devices integrated into a single cover. This way of solution means that a certain overhead has to be accepted, due to the multiple protocol software having basically the same structure and functionality. Furthermore, updating of software in a terminal is difficult or even impossible. The introduction of new services like GPRS, HSCSD or EDGE reveals the disadvantage of the aforementioned solution: Though only a part of the air-interface related software has to be changed, newly developed devices will be necessary.

Software (Defined) Radios (SDRs) [1][2] maybe a more efficient solution. An SDR is based on processors and wide band front-ends and all kind of functionality is achieved by means of dedicated control software. Thus it is possible to design a multi-mode radio capable of supporting multiple air-interfaces and protocol stacks.

1. Requirements on SDRs

As indicated in Figure 1, SDRs will be able to meet various demands. The support of several air-interface standards provides the ability to roam across access networks with one single handset, called network mobility. Since all modules of an SDR are under software control, a change of the functionality is easier to realize. Dedicated interfaces will allow to add new services without the need of hardware modifications.

Generation bridging will be eased, an important aspect since current and new standards like e.g. GSM and UMTS will have to coexist for a transitional period.

Another goal is reconfigurability: By modifying a radio's configuration software, its use with different access networks and the adaptation to different air-interfaces can be achieved. Since the protocols running at the base station (BS) and mobile station (MS) are basically symmetrically, reconfigurability is applicable to both network elements. Thereby an operator can built up an infrastructure that easily can be reconfigured to support a new standard in addition when necessary. This is interesting for both, developed countries that are in the migration from 2G to 3G mobile radio networks and for developing countries that are planning to introduce mobile technology but do not want to decide for a standard now due to the unpredictable acceptance of the systems under construction.

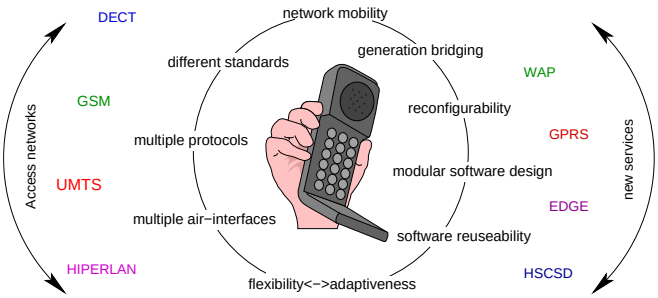

Figure 1: Requirements on Software Defined Radios

Reusability of software is a big concern. New air-interface standards typically rely on well-understood protocol stacks of predecessor systems. Cost intensive re-engineering of software can be avoided and software can be re-used if designed in a suitable way. Modular software design entails several advantages. On the one hand, portability of dedicated functionality is supported. Using well defined interfaces, the same module can operate within different systems. On the other hand software-upgrades are easily facilitated, as the respective modules can be changed individually.

There are a number of research issues that need to be addressed, like

- the definition of elementary commonalities of the various mobile communication systems
- resource-sharing within telecommunication software
- modular software design and interfaces
- reusability of software
- interworking of different systems
- structure of a generic protocol stack
- nature of software extensions to an existent system
- composition of an adaptive protocol stack architecture

2. Design of a Software Defined Protocol Stack

In this section we describe how to design generic and adaptive protocol software (DGAPS).

Since the configuration software and an implemented protocol stack represent one important part of a device, their structural composition, implementation and realization is of fundamental research interest. Having the ISO/OSI reference model in mind a high degree of similarity can be found for different air-interface standards. Concerning the control software many features can be implemented as shared resources. Applying DGAPS results in a generic protocol stack, that provides a common basis for a number of different systems. Specialization by introducing standard-specific functions to the generic stack stepwise results in a specific realization towards a specific protocol stack.

2.1 A Software Defined Protocol Stack

All the software specifications are being done formally with the help of SDL (Specification and Description Language)[3][4]. SDL is an object orientated programming language that supports features like inheritance and information hiding. Because of its graphical (besides an equivalent phrase-) representation and an easy to understand finite state machine basis, SDL has been accepted worldwide for the specification of communication protocols.

To obtain the portable source code from an SDL specification, an automatic translator, called *SDL2SPEETCL* [5], is used. It takes the phrase representation (SDL/PR), generated by the SDT-Analyzer [6] (thus syntactical and semantical errors are excluded), as input and maps the system behaviour to classes of SPEETCL (SDL Performance Evaluation Tool Class Library) [7]. SPEETCL is a C++ class library developed for the integration of protocols specified in SDL into an event-driven simulation environment.

The proposed DGAPS approach to develop a software defined air-interface protocol stack is as follows:

1) Identification of commonalities In a first step (step 1) different systems, say System I and System II, are analyzed layer by layer to identify their commonalities. A more detailed description of the analysis process (exemplarily for DECT and GSM) together with a reference implementation is described in [8]. The number of different systems to be considered may be two or larger. The result will be an SDL specification of a common subset of the access protocol stacks for the systems, see Figure 2. Since this stack provides the common characteristics of the considered air-interface standards it is called a *generic protocol stack*.

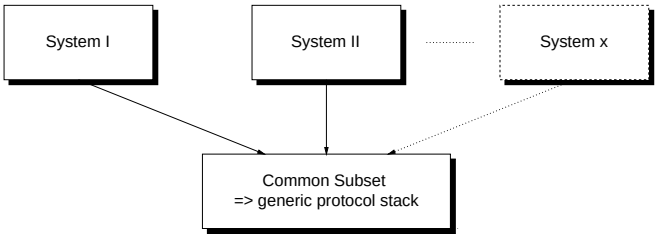

Figure 2: Generic protocol stack for various system types

2) Development of standard-specific supplements The next step (step 2), cf. Figure 3, is to develop SDL-specifications specific to given air-interface standards, say for System I or System II. These include functions that are specific to respective standards and thus represent the individual behaviour of a system. Different approaches can be taken to achieve that goal. In order to make use of the object-oriented properties of SDL together with inheritance, it is suggested to implement these parts as subclasses derived from base classes implemented within the generic stack. This is of special advantage, if more than two systems are considered; procedures that are common to most but not necessarily to all standards still will be implemented within the generic stack. The standard-specific supplements than will have to redefine the respective procedures and the behaviour required is achieved then.

3) Integration of a dedicated air-interface standard To end in a dedicated air-interface standard, the generic protocol stack and the standard-specific supplement, have to be merged (step 3). This is done by means of inheritance.

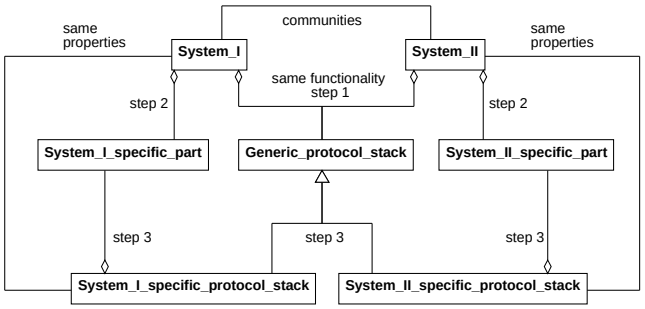

Figure 3: Interaction of components for an SDR protocol stack

Figure 3 shows the correlations and dependencies of the aforementioned parts in the notation of UML. In order to distinguish between a specific protocol stack that is designed either conformant with the above presented approach or not, the notation *System_X* (non-conformant) and *System_X_specific_protocol_stack* (conformant) is used.

4) Optimization of the SDL-Specifications To result in a run-time efficient code after translation it is necessary to undertake some code optimization. Investigations have shown that the following rules will result in a substantial speed-up:

- use of pointers in SDL instead of parameter lists
- decrease the number of process switching
- reduce the number of events and timers
- replace SDL data types by C-code constructs

5) Further Optimizations and Validation In a final step those parts of the code are to identify that are most frequently used during runtime and the respective functions have to be considered to be implemented in hardware, say in FPGAs etc.

The optimized SDL specification can be guaranteed to be conformant to the original SDL specification, since errors would appear when run against each other and conformance will be reached after removal of these errors in the informally implemented code.

2.2 Reusability of software modules

An attracting property of the DGAPS is the reusability of software modules. Assume an existing air-interface standard, say System I, is available, designed according to the proposed technique (System_I_specific_protocol_stack). The following steps would be necessary to end with a protocol stack for a new system, referred to as System III, see Figures 4 and 5.

6) Adaptation of the generic protocol stack Firstly, the genericity property of the generic protocol stack part of System I (Generic_protocol_stack_S_I) need to be checked with respect to genericity for System III (step A). If applicable, those parts that are not generic have to be removed so that a new protocol stack generic to Systems I and III is achieved (step B).

7) Design of a system specific part To re-use as much code as possible the design of the specific part of System III starts by taking the system specific part of System I as an input (step C). The specific parts of the System I stack that are formulated as blocks with external signal paths have to be identified. Analysis of the usability of these blocks with respect to their applicability to cover System III specific functions has to be done next. Implementing modifications required together with introducing new supplements finally results in a System III intrinsic part (System_III_specific_part). Afterwards new functions not contained in the System I air-interface have to be identified. They are assigned either to existing or to blocks newly to be developed.

In order to end with the System III air-interface protocol stack, the two parts, generic protocol stack and System III specific supplement, have to be brought together (step D). Having regarded the strategy presented above, the further proceeding is similar to the steps described in subsection 2.1.3 and thereafter.

3. Example of applying DGAPS

In the following we show how to apply DGAPS. Instead of two abstract systems I and II, the systems DECT and GSM are considered.

1) Example: Identification of commonalities According to subsection 2.1 the first step is to analyze the systems layer by layer to identify their commonalities. We here concentrate on the Call Control (CC) entity within the network layer, a central service instance in both systems dealing with

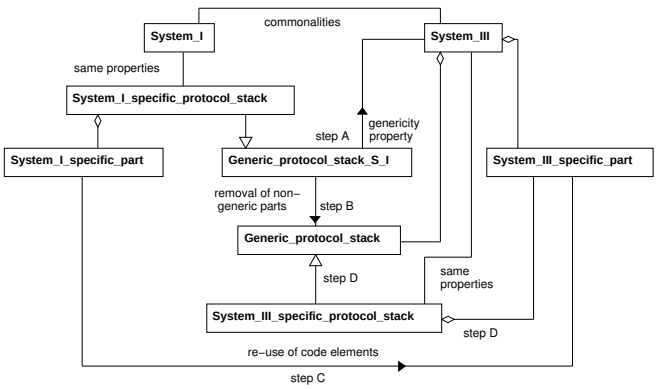

Figure 4: Design of a new protocol stack by re-using existing software modules

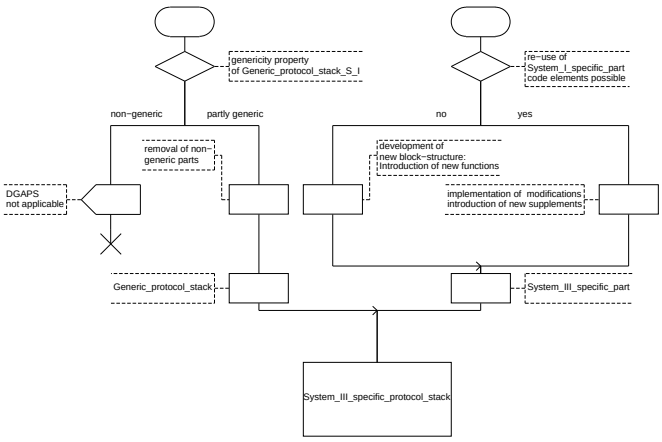

Figure 5: Design of a new Protocol Stack with DGAPS

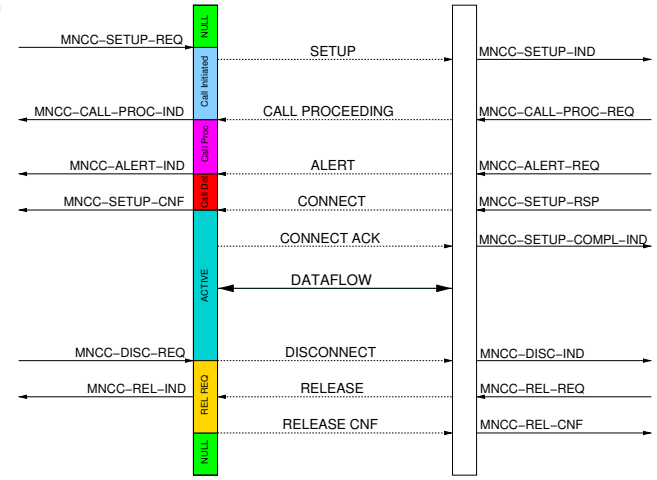

Figure 7: Generic MSC with respective CC states

call establishment -maintenance and -release. The logical steps that have to be taken can be modeled with the help of Extended Finite State Machines (EFSMs) notation. Each step within the three phases can be represented then by a well defined state with related state transitions. The latter are specified to signals incoming and outgoing to/from the states or to expiration of timers whereby locally kept variables can be manipulated. The signals can either have a peer-entity as addressee and thus are transmitted across the air-interface, or belong to the internal (vertical) communication between two adjacent layers.

The two different EFSMs of the CC entities of the systems show a high similarity. This is because both are derived from the Digital Subscriber System No. 1 (DSS1), specifying the ISDN User-Network interface on layer 3 for basic call control, see [9] and [10].

A generic CC entity can be designed, as a result of step 1 for the *generic protocol stack*.

Figure 6 illustrates the EFSM graph of the generic CC entity of the mobile station. The states on the left hand side belong to a mobile originated call, the states on the right hand side belong to a mobile terminated call, respectively.

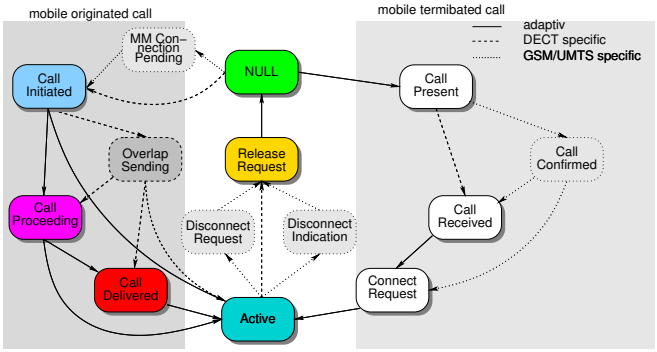

Figure 6: EFSM of a generic CC entity

In the same way one can also determine the relationships between the two systems DECT and GSM for the signals to be exchanged. Thus it is possible to draw a generic Message Sequence Chart (MSC) as indicated in Figure 7. The generic states of the mobile station during CC are in the same greyscales as shown in Figure 6.

2) Example: Development of standard-specific supplements As shown in Figure 6, the generic CC entity does not contain all states, needed for a specification conformant to the respective standards DECT and GSM. An example of this is the GSM specific state MMConnectionPending that is part of the establishment of a mobile originated call. In the DECT reference model [11] the MM service entity is arranged in the same layer as the CC entity [12]. Thus the operation of CC and MM does not directly depend on each other in DECT. In GSM, however, the setup of a CC link requires an MM connection to be established in advance [13].

Consequently the MMConnectionPending state has to be implemented in a GSM-standard-specific supplement. As said in section 2 this supplement has to be realized within a GSM CC entity that is inherited from the generic CC entity within the generic protocol stack. The implementation is done in such a way, that the original transition from the state NULL to the state CallInitiated is split up and the system specific state MMConnectionPending is inserted.

Figure 8 shows a simplified SDL realization of this approach. On the left hand side we can see a part of the SDL flow chart of the generic CC entity (<<System Type stGeneric/Block Type btNetwork>> ptCCEntity) created in step 1. Being defined as virtual (Virtual Process Type) offers the possibility to make use of inheritance which means that a redefinition or an add-on of supplements in derived implementations is possible. The latter is done with the code presented on the right hand side (Redefined Process Type). The topic (<<System Type stGSMMS/Block Type btNetwork>> ptCCEntity) shows, that this flow chart is

valid for GSM mobile terminals only. As it is derived from the implementation of the generic CC entity (Inherits stGeneric/btNetwork/ptCCEntity) the complete generic code implicitly is contained within this implementation as well and in addition the changes/supplements shown.

On the left hand side, some of the generic states presented in the EFSM and the MSC within the previous section can be recognized. Running a simulation with this SDL code, the CC entity (of either system) will start with the state NULL. On receiving the signal sMNCC_Setup_req that is also generic since it belongs two both systems a certain procedure is called and a new state is archived. This is the point where DECT and GSM differ, thus the system-specific supplement has to take over and to perform the aforementioned 'split-up' of the state-transition. As sMNCC_Setup_req was defined to be virtual/redefined in the respective code, a GSM mobile will continue by working out the code on the right hand side, calling the procedure pdMM_Connect_req that ends in the GSM specific state MMConnectionPending, whereas a DECT handset will continue executing the generic code, calling the procedure pdCC_CallInit_MS. Finally both implementations end up in the generic state CallInitiated.

3) Example: Integration of a dedicated air-interface standard The two preceding sections have shown the approach of applying DGAPS on the example of a generic and system-specific CC entity of the network layer. Similar considerations have to be taken for remaining other service entities, e.g. Mobility Management (MM), of layer three and the other layers, respectively. Additionally this concept has to be applied on signals, management services (like Lower Layer Management Entity, LLME) and the whole structural composition of protocol stacks. The result will be a generic protocol stack that inherits its features to a system-specific stack making use of the system-specific supplements and changes, compare Figure 3, step 3. In such a way, a highly

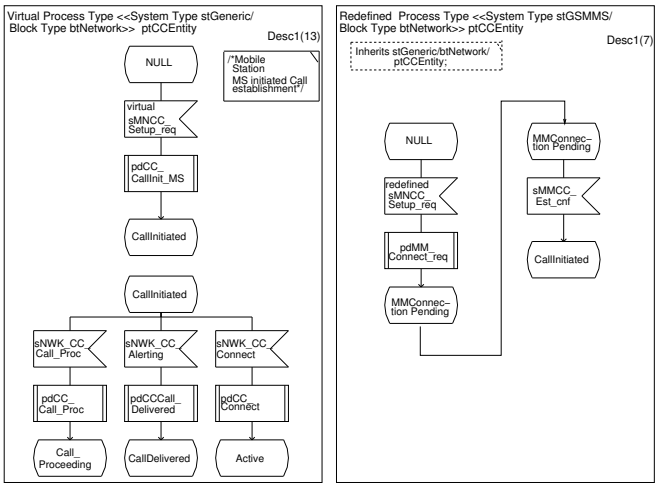

Figure 8: Generic CC entity and GSM supplement

modular protocol stack based on a generic skeleton (and thus reusable for other standards) is achieved. Additionally, it is possible to support different standards by one single handset that needs less memory since the generic parts only have to be allocated once.

4. Conclusions

Software Defined Radios promise to be an efficient solution within the further development of mobile radio systems since different existing and future air-interfaces can be supported just by software reconfiguration. In order to follow the rapid development from 2G to 3/4G systems, protocol software also has to be evolutionary. Within this paper the requirements on protocol software were shown and a design technique called DGAPS (Design of Generic and Adaptive Protocol Software) was introduced. Thereby basic features of various mobile radio systems are provided by a generic kernel (generic protocol stack) and dedicated system behaviour is achieved by adding specific supplements. Following DGAPS allows the realization of highly structured and modular designed code that can be used to support different air-interfaces (adaptive) by minimizing the overhead. A further attracting property of the here mentioned technique is the reusability of the code for future air-interface designs to avoid cost-intensive re-engineering.

I. References

[1] "http://www.sdrforum.org." Homepage SDR Forum Web Site.

[2] J. Mitola, "Technical challenges in the globalization of software radio," in *IEEE Communications Magazine, February 1999*, pp. 84–89, February 1999.

[3] ITU-T, "Specification and description language (sdl)." ITU-T Recommendation Z.100, Nov. 1999.

[4] A. Olsen, O. Færgemand, Møller-Pedersen, R. Reed, and J. Smith, *Systems Engineering Using SDL-92*. ELSEVIER SCIENCE B.V., 1994.

[5] M. Steppler, *SDL2SPEETCL — An SDL to C++ code generator, Rel. 4.1.0*. AixCom GmbH (www.aixcom.com), Dec. 2000.

[6] Telelogic, Malmö, Sweden, *SDL Design Tool (SDT) 4.1 Reference Manual*, 2000.

[7] M. Steppler, *SPEETCL — SDL Performance Evaluation Tool Class Library, Rel. 3.2.0*. AixCom GmbH (www.aixcom.com), Dec. 2000.

[8] M. Siebert, "Design of a Generic Protocol Stack for an Adaptive Terminal," (Proc. of the 1st Karlsruhe Workshop on Software Radios, Institut für Nachrichtentechnik Karlsruhe, Germany), pp. 31–34, March 2000.

[9] ITU-T Recommendation Q.930, *Digital Subscriber Signalling System No. 1 (DSS 1) - ISDN User-Network interface layer 3 - general aspects*. ITU, 1994.

[10] ITU-T Recommendation Q.931, *Digital Subscriber Signalling System No. 1 (DSS 1) - ISDN User-Network interface layer 3 specification for basic call control*. ITU, 1994.

[11] ETSI, “Digital european cordless telecommunications (dect) common interface part 1: Overview.” European Telecommunications Standards Institute, October 1992. ETSI European Telecommunication Standard ETS 300 175-1.

[12] ETSI, “Digital european cordless telecommunications (dect) common interface part 5: Network layer.” European Telecommunications Standards Institute, October 1992. ESTI European Telecommunication Standard ETS 300 175-5.

[13] E. T.-S. 3, “Digital cellular telecommunications system (phase 2+); mobile radio interface layer 3 specification (gsm 04.08),” Technical Specification 5.3.0, European Telecommunications Standards Institute, Sophia Antipolis, Frankreich, July 1996.

[14] B. Walke, *Mobile Radio Networks*. Chichester, UK: Wiley, 2nd ed., 2001.