MULTIRADIO SCHEDULING AND RESOURCE SHARING ON A SOFTWARE DEFINED RADIO COMPUTING PLATFORM

Ari Ahtiainen (Nokia Research Center; ari.p.ahtiainen@nokia.com)
Kees van Berkel (ST-NXP Wireless Research; kees.van.berkel@stnwireless.com)
David van Kampen (ST-NXP Wireless Research; david.van.kampen@stnwireless.com)
Orlando Moreira (ST-NXP Wireless Research; orlando.moreira@stnwireless.com)
Antti Piipponen (Nokia Research Center; antti.piipponen@nokia.com)
Tommi Zetterman (Nokia Research Center; tommi.zetterman@nokia.com)

ABSTRACT

Architecting Software Defined Radio (SDR) for handheld multimedia devices has become a major challenge in bringing mobile internet services to consumer markets. This paper describes a multiradio computer driven approach to software radio design. A model for a unified radio system is presented as the key concept to bring different radio applications under a common multiradio resource management scheme. This architecture provides basis for designing a multiradio operating system to guarantee that simultaneously executing radios can meet their real-time behavior requirements while sharing the computing, communication and hardware resources of the radio computer.

1. INTRODUCTION

The emerging need for high data rate wireless services and the ever-growing number of wireless standards leads to an urgent demand for an SDR architecture that is optimized for multiradio control and run-time resource management. Major issues are related not only to new technology driven radio access methods but also to the coexistence with earlier standards having a solid global installation base, like GSM, WCDMA and WLAN. The optimized radio access for each application strongly depends on technology parameters and market situations. This has led to an ever growing number of radio standards embedded in consumer devices such as multimedia computers, internet tablets, enterprise products and mobile phones.

To satisfy these needs an architecture for a multitasking *radio computer* is presented in Section 2. Together with the *radio operating system*, which provides multiradio resource management, multiple individual radios can be executed simultaneously on top of shared computing, communication and hardware resources. Note that a radio computer also differs from conventional ones and has several unique characteristics. The behavior of radio applications is strictly regulated by their frequency and time domain behaviors, which has led to the *unified radio system model* described in Section 3.

All functionalities of the multiradio computer are specified in a model-driven and service-oriented manner with well defined interfaces between all service and system components. This architecture is then further decomposed in order to separate SDR control functions from the set of radio system applications. These SDR control functions, which are common to all radio applications, are provided at the Universal Radio System Interface. This interface makes all radios subject to a common multiradio resource management framework as described in Sections 4 and 5.

This architecture and the related resource model are essentially platform neutral in the sense that they allow widely different implementation choices. As an embodiment, we outline in Section 6 a heterogeneous multiprocessor platform which supports the multiradio resource management framework and allows for fine-grained resource sharing, while still allowing each radio to be designed according to common paradigm and executed in virtual isolation from other radios.

2. FUNCTIONAL ARCHITECTURE OF A RADIO COMPUTER

2.1. Overview of the SDR functional architecture

The functional architecture of a radio computer device is illustrated in **Figure 1**. All services of the radio computer are provided at the Multiradio Access Interface. The services may include connectivity and data transfer, but also positioning and broadcasting services. User applications access the radio computer via a networking stack and mobility policy manager, which maintains user preference policies for selecting radios. Additional services for installing new radios into radio computer are available to the administrator user.

The Multiradio Access Interface is supported by a common SDR control framework, which consists of the Configuration Manager, Radio Connection Manager, Flow

Controller, Multiradio Controller and Resource Manager system components. Their functionalities are common to all radio systems and become part of the radio operating system. This SDR control framework is responsible for installing, loading and activating different radios and maintaining user data flows, which can also be switched from one radio system to another. The radio operating system also does multiradio control and resource management for sharing of available spectrum and radio computer resources among simultaneously executing radio systems.

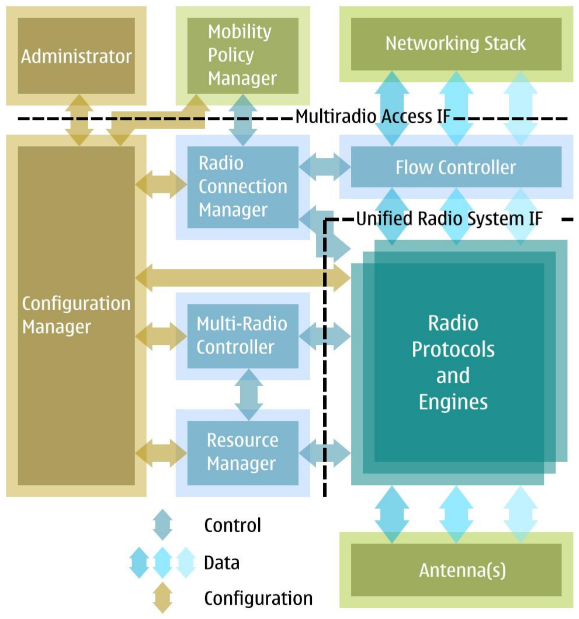


Figure 1 – SDR Functional Architecture

The services of the SDR control framework are available to the radio systems at the Unified Radio System Interface (URSI). With this interface every radio system can be integrated into radio computer in a unified manner by making them subject to common multiradio resource management.

The Unified Radio System Interface can be used as reference, when checking the compatibility of individually developed radio system software implementations. It can also be regarded as a harmonization interface for developing future radio systems to this architecture. More detailed description of this interface is given in Section 3.

2.2. Architecture modeling

The functional architecture described above has been modeled by using model-driven architecting method [1], which is based on rigorous service-oriented approach for specification of communicating distributed systems [2]. By being able to execute the radio computer functions already at this architectural specification phase the new approach to SDR system design can be visualized and communicated. It also provides basis for specification and implementation of testware for checking compatibily against the two major architectural interfaces.

3. UNIFIED RADIO SYSTEM MODEL

3.1. Life cycle of a Radio System

Following the radio computer paradigm, the SDR control framework allows bringing in and removing radio applications during run-time. A set of radio applications may be pre-installed into the radio computer, and new ones may be brought in by using the administrator services of the Multiradio Access Interface. We describe the life cycle of a radio application inside the radio computer by using four distinct administrative states, which differ by their use of the shared platform resources.

A *not installed* radio application is unknown by the radio computer. In the *installed* state, the radio computer has a copy of the radio system package (including resource budgets and executables); it may be stored in mass storage in compressed format for minimal memory footprint. A *loaded* radio application is available for the end user, but is not yet in execution. Once an instance of the radio application is in execution, it is considered to be in the *active* state, and is using various hardware codecs and radio frequency circuitry in addition to the computation, memory and communication resources.

3.2. Radio System Operational State

From the resource sharing point of view, the *active* administrative state is the most interesting. Operational states are defined as sub-states of the *active* administrative state, and are used to describe different resource requirements.

For example, an 802.11 WLAN station that is in the power-saving mode only needs to process beacon frames sent by the access point, and has no need for any transmitter resources, leaving them for the use of other radio systems. If the access point indicates buffered frames for the station, or the station itself has frames to transmit, transition to normal communication operational state occurs.

The radio system designer may divide the application to various operational states, in order to facilitate more efficient resource sharing. Inside an operational state, the radio application is allowed to operate freely within the given limits. Transitions between operational states originate from the user or an external entity (e.g. radio network), and are requested from the resource manager, which is part of the SDR control framework. No real-time guarantees are given for serving these requests, and the radio system must accept also denied transition requests due to resource limitations. In those cases, the application may propagate the deny information to the Multiradio Access Interface so that the higher-level control elements can take the necessary actions (e.g. deactivate lower priority radios, thereby freeing resources). The SDR control framework guarantees resources for the granted operational states.

3.3. Decomposition of a Radio System

The platform neutral SDR functional architecture decomposes the unified radio system into two parts, *(radio) protocols* and *(radio) engines.* The *protocols* part controls the time behavior of the radio system, and the *engines* part does the radio communication operations at a specific time and with a specific configuration.

More specifically, the *protocols* part takes care of interaction with the user and the external communication partners. It knows the current operational state and detects the need to transition to another, if the resource demand changes. The *engines* part implements the signal processing and radio frequency functions, as instructed by the protocols part, and contains the accurate time of the radio system.

It is worth mentioning that the split between *protocols* and *engines* is not according to the OSI data link layer and physical layer, or real-time properties, or according to platform components. The split is purely functional and allows the unified interface to be used for all radio systems. Other division criteria would lead to non-uniform interface description because radio systems differ in the OSI layer radio functionality distribution and real-time properties.

3.4. Services at the Unified Radio System Interface

The purpose of URSI is to harmonize the way of accessing dissimilar radio systems, as well as their behavior on the SDR platform. To this end, all radio systems must provide a set of services as specified in the URSI, to be compatible with the SDR functional architecture. They gain access to the shared platform resources and the radio spectrum only by using the SDR control framework services.

The services provided by unified radio systems relate to activation and deactivation, neighbor device discovery, and establishing communication and user data flows. Even though mapping of specific radio system functionality to the URSI services may not always be straightforward, the benefit is that all radio systems can be used in the same manner.

The SDR control framework services are used by the radio systems to set up the baseband signal processing and radio frequency resources, and subsequently to access the radio spectrum.

4. MULTIRADIO CONTROL AND SCHEDULING

4.1. Multiradio scheduling concepts

Like a conventional computer, the radio computer has to be able to execute multiple concurrent applications. Radio spectrum is perhaps the most critical scarce resource all active radios need to access. The SDR functional architecture contains a special entity to dynamically schedule access to spectral resources, called *multiradio controller* (MRC). Its main responsibility is to detect in advance the interoperability problems between simultaneously active radios and solve them. Typical sources of interoperability problems are 1) wide band noise from frequency synthesizer and power amplifier, 2) harmonics, 3) intermodulation results of two or more transmitters, 4) RF blocking (desensitization) and 5) clock leakage. The typical way to solve interoperability problem is to forbid one or more of the simultaneously active radios to operate. Such a decision can be based on for example different priorities associated to radios.

Radios connect to MRC via URSI, which makes it possible to add new radios to the multiradio control scheme. The MRC scheduling concept is a fundamental paradigm change compared to the traditional way where there are separate pairwise coexistence schemes between radios. For example, the described scheme allows radios loaded at run time to cooperate cleanly with existing radios.

To enable this scenario, radios executed in the SDR platform need to have two additional functions: 1) The radio has to tell MRC in advance its temporal requirements and parameters of transmission and reception and 2) it has to be able to provide its internal time and synchronization information to MRC. Optionally, a radio can be built to utilize scheduling results, for example it may start to prepare retransmission when MRC denies operation or go to sleep mode at the end of granted radio access.

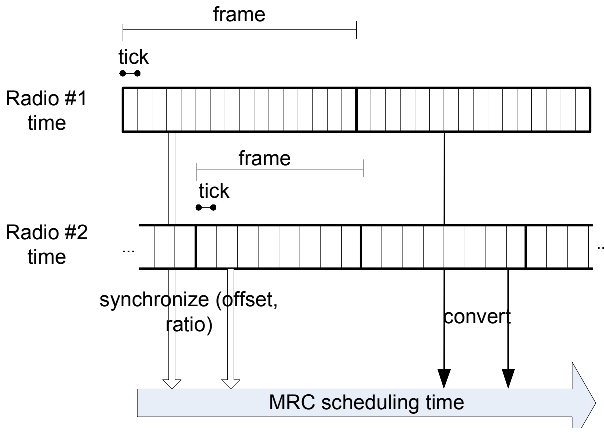
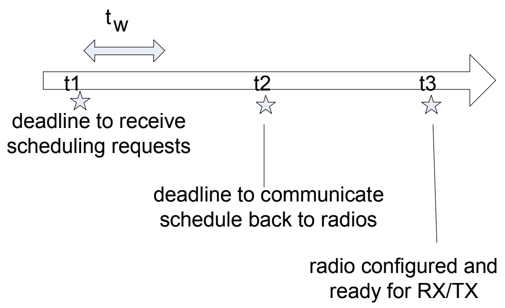


Figure 2 - Time concept in MRC scheduling

Figure 2 presents the time concept used for multiradio scheduling. Each individual radio has its own native time. In the SDR control framework, this is called “radio time” and it is generalized to consist of a smaller time unit called *tick* which represent the finest time granularity, and a larger time unit called *frame* which represents a repetitive structure used to refer time inside the radio. One frame consists of multiple ticks. Both the length of the tick and the number of ticks in one frame can be freely selected for each radio.

To be able to detect spectral conflicts, MRC converts radio access start and stop times into its own scheduling time domain. The relation between different radio times and common MRC scheduling time has to be known in advance. Each active radio executed in the SDR platform is required to provide MRC its time synchronization information when the radio is initially started, and maintain that during the time the radio is active.

4.2. Time behavior model



t_w
t1
t2
t3
deadline to receive scheduling requests
deadline to communicate schedule back to radios
radio configured and ready for RX/TX

Figure 3 - Multiradio scheduling timeline

Figure 3 illustrates the MRC time domain behavior:

- Scheduling window t_w defines the time window of scheduling requests solved during one scheduling round
- Time *t1* defines the deadline when MRC must have the information about the behavior of radios to be able to calculate a schedule.
- Time *t2* defines when scheduling decisions have to be communicated back to radios to have enough time to configure radio HW and SW for radio operation.
- Time *t3* defines when radio HW and SW configuration has to be ready. It may be earlier than the actual starting time of radio operation.

Defining the values for these time parameter depends both on the performance of the SDR platform and the ability of radios to tell their estimated schedule in advance.

4.3. Multiradio scheduling service

MRC provides scheduling service for radios via URSI. This service can be used in three different ways:

1. *Rigid request* is used when a single radio operation has fixed start and end times. The requested time slot is either accepted as a whole, or rejected. Because the request needs to be solved during one scheduling round, it cannot be longer than t_w.
2. *Continuous request* is used when a single radio operation takes longer than t_w. Continuous request is scheduled in multiple parts and communicated back to the radios one part at a time
3. *Flexible request* is used when MRC is allowed to select one of multiple alternatives or modify the timing of radio operation in case of conflict. For example, MRC may either advance or delay the granted time slot.

To be able to accurately detect and solve interoperability problems, MRC needs to know the characteristics of requested radio operations. Besides timing parameters, the following parameters can be specified for each scheduling request: 1) priority of request, 2) transmitter power, 3) receiver signal quality information (RSS) 4) channel bandwidth, 5) carrier frequency and 6) crest factor.

5. RESOURCE MANAGEMENT FRAMEWORK

5.1 Multiradio resource management concepts

Radios are Real-Time (RT) applications. This means that the validity of the computational results depends not only on values, but also on the time at which these are produced. The RT behavior of an application is dependent on the amount of resources available to it during execution. Uncertainty in resource provision may cause unpredictable temporal behavior.

In a multiradio system, many radios may be active at the same time. Furthermore, each radio can be in one of several different operational states. Each operational state has different resource requirements.

In order to allow maximum flexibility at the lowest cost, radios must share computation, memory and communication resources, as well as hardware resources like RF transceivers and antennas. This poses a difficult problem for any RT system: as satisfaction of the temporal constraints depends on resource availability, resource sharing can make the temporal behavior of each radio depend on the behavior of all other radios in the system, which is difficult to predict in a case such as this, where radio combinations are dynamic.

Our approach to this problem is to give radios a degree of independence from the rest of the system by using a strong resource management policy. Conceptually, it is designed to isolate each radio in such a way that it only sees a fraction of the platform resources, and these are exclusively reserved for it. It is thus a form of virtualization. We do this by associating a resource budget with each operational state of a radio. The resource budget lists all resources of the platform that the radio will require once it is in that operational state to meet its RT requirements.

The resource management policy should ensure two things: *a) admission control* - a radio is only allowed to change operational state if the system can allocate upon request the resource budget it requires for that operational state; and *b) guaranteed resource provisions* - the access of a radio to its allocated resources cannot be denied by any other radio. Notice that a change of operational state may be rejected by lack of resources on the platform. Because of this, no RT guarantees can be given across operational state changes.

A resource budget, may contain for example, per task, a list of resource requirements like processor type, required amount of instruction memory, data memory and scheduler settings (e.g. size of time slice on a TDMA scheduler). Furthermore, the channels for inter-task communication have bandwidth, and buffer memory requirements.

The resource budgets are pre-computed offline per operational state of the radio. The RT components of a radio can be expressed as Synchronous Data Flow (SDF) graphs [3]. This allows for the temporal analysis of execution on a specific platform and calculation of worst-case resource requirements to guarantee compliance to RT demands. For this to be possible, the streaming platform must allow tight bounds on resource usage to be inferred. Resource budget calculation from SDF graphs is described in [4], where 802.11a WLAN and TD-SCDMA are given as examples.

Algorithms have been proposed [5] that do the admission control by mapping a vector of such requirements to a vector of provided resources on the platform, which amounts to solving an extension of the vector bin-packing problem.

5.3 Hierarchical resource management services

Different components of a radio application have different RT requirements (in terms of strictness and time scale). Baseband (BB) and RF are essentially hard RT, as a failure in meeting deadlines may cause the output to be worthless (e.g. decoding a WLAN packet must be done within the SIFS deadline), while the higher level control protocols can be laxer in meeting their RT constraints.

They are also different in terms of the type of computation: the RT BB processing mostly performs iterative algebraic and bit-manipulation operations over a periodic incoming stream of data, whereas radio protocols are typically communicating finite state machines, exhibiting complex control flow and much more irregular activation patterns.

Because of these different requirements, and also due to vendor specialization, one can expect a radio platform to be divided into several subsystems, or platform components, each tailored to a specific part of the radio functionality and related to a type of RT requirement. Also, the different type of temporal requirements per platform component means that there may be diversity in the way that resource management is applied to each sub-system. This means that, for instances, the strict timings of the baseband stage may require strict budgeting based on the dataflow approach referred to above, while for protocols it may be enough to provide mere estimations of resource requirements and also less strict scheduling algorithms may be used.

Due to these considerations, the resource management functionality in our SDR radio computer platform is distributed in a hierarchical manner. The central resource manager services operational state change requests. It oversees the platform component resource managers, each of which provides admission control and resource reservation services for its platform component. The platform component resource managers communicate with other entities (e.g. local processor schedulers, memory managers) that actually own the resources and do the fine-grained scheduling.

In the current framework, resources are reserved and configured in the following steps. First, an admission check is done, without reservation, for all the platform component resource managers. This is to avoid roll back of reservation, if admission on one of the platform components is denied. The platform component resource managers retrieve the operational state specific resource budgets from the radio system description package, if needed. Once admission has been granted on all platform components, the central resource manager proceeds with resource reservation. Resource reservation includes configuration of the resources. Local dynamic loaders are instructed to request executables from the radio system package in the central repository and store them in a specific position in memory.

In the central resource manager, the operational state change requests for the different radio jobs are queued. The admission check and resource reservation with configuration constitute one single atomic operation.

6. RADIO COMPUTER PLATFORM

The resource management framework of a multiradio system must provide to each radio, per operational state, a virtual platform which is a subset of the resources of the actual radio computer platform. It is part of the radio operating system on the radio computer platform (Figure 4).

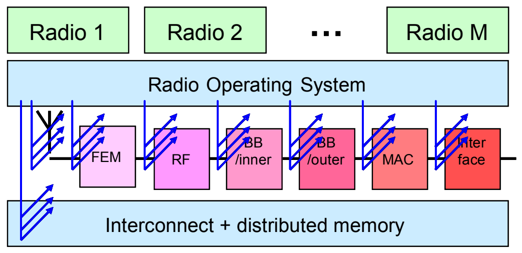


Figure 4 - Radio Computer Platform

The radio computer platform is made of seven stages, each covering both receive and transmit functions. From left to right (Figure 4) these stages comprise respectively one or multiple:

1. antennas;
2. front-end modules (filters, power amps, etc);
3. RF transceivers;
4. baseband processors for (de)modulation;
5. baseband processors for (de-)coding;
6. control processors for protocol stacks;
7. application interface units;

Notice that while this split in stages may be reflected in the decomposition of the radio system in platform components, this is not mandatory, as the decomposition is also dependent on the other factors that were stated in Section 5.

This multi-stage arrangement exploits structural similarities among a large variety of different radios. Each stage has to offer the *appropriate* amount of reconfigurability or programmability [6]: sufficient to cover a specified set of radio standards and performance levels, but no more than can be afforded. Accordingly, the *baseband* part of an SDR platform for handheld devices may comprise:

- a number of programmable digital signal processors (e.g. EVP [6]);
- a set of (reconfigurable) accelerators (e.g. a multi-standard Turbo decoder or descrambler);
- a number of general purpose processors (e.g. ARM);

In order to support the huge memory bandwidths required for DSP processing, these processors have to be supplemented with carefully designed distributed memory architecture and a flexible interconnect.

We have opted for a form of FIFO-based asynchronous intertask communication. This allows for distributed, data-driven synchronization of tasks, leading to a scalable solution with a low and bounded control overhead.

Dealing with the strict RT requirements of a multiradio use case on such a heterogeneous multiprocessor is challenging. Furthermore, radios need be designed independently of one another. This calls for a composable system [8]. Accordingly, cooperative scheduling cannot be used among different radio instances, leading to distributed preemptive scheduling. For example, meeting the so-called SIFS deadline (16 μs) of WLAN involves multiple processors. When a DVB-H radio occupies these processors, each of them must be preempted, and the SIFS-related tasks must be scheduled timely. Such tight RT guarantees can only be honored when (tight) upper bounds can be given for the timing of all involved transactions, including inter-processor traffic and memory accesses. This poses some further constraints to the choice of arbiters and schedulers in the system. In [7], scheduler requirements for predictable dataflow execution are discussed. The TDM scheduler used in [5] fits these requirements.

Given these upper bounds, worst-case execution times can be computed for all the tasks. This allows for offline computation of scheduler settings per operational state of the radio, as referred in Section 5, and described in detail in [5].

The radio operating system must also virtualize the distributed memories, to allow for run-time mapping of multiple instances of each radio to the multiprocessor platform by the resource manager, and provide memory protection.

7. CONCLUSION

To meet the reconfigurability and resource sharing requirements of handheld software radio devices a functional architecture for a radio computer has been presented. This architecture provides functional interfaces for installing, loading and activating new radio systems at run-time. All active radio systems become subject to common multiradio scheduling and resource management framework provided by the radio operating system. Mapping of such architecture onto a programmable radio computing platform has been presented and is currently being experimented.

8. REFERENCES

[1] A.Ahtiainen et al, “Architecting Software Radio”, in *Proceedings of the SDR Forum 2007”*, 2007

[2] S.Leppänen, *Rigorous Service-Oriented Development of Communicating Distributed Systems,* PhD Thesis, Tampere University of Technology, 2008.

[3] E.Lee and D.Messerschmitt, “Synchronous Data Flow”, in *Proceedings of the IEEE,* 1987

[4] O.Moreira, F.Valente and M.Bekooij, “Scheduling Multiple Independent Hard Real-Time Jobs in a Heterogeneous Multiprocessor”, in *Proceedings of the ACM/IEEE EMSOFT 2007,*

[5] O.Moreira, J.D. Mol, and M. Bekooij, “Multiprocessor Resource Allocation for Hard-Real-Time Streaming with a Dynamic Job Mix”, *in Proceedings of the IEEE RTAS 2005*

[6] K. van Berkel et al, “Vector Processing as an Enabler for Software-Defined Radio in Handheld Devices”, "EURASIP J. on Applied Signal Processing," Vol. 2005, 16, pp. 2613-25

[7] M. Bekooij et al, “Predictable and Composable Multiprocessor System Design: A Constructive Approach”, *Proceedings of the Bits & Chips Symposium on Embedded Systems and Software*, 2007

[8] Kopetz, H. “Real-Time Systems: Design Principles Embedded Applications”, Kluwer Academic Publishers, 1997