A COMPONENT-BASED ARCHITECTURE FOR PROTOCOL DESIGN AND DEVELOPMENT IN SDR FRAMEWORKS

Maurizio Colizza, Marco Faccio, Claudia Rinaldi, Fortunato Santucci
University of L'Aquila, Center of Excellence DEWS
L'Aquila, Italy
e-mail: colizza@westaquila.com, marco.faccio@univaq.it, claudia.rinaldi@univaq.it, fortunato.santucci@univaq.it

ABSTRACT

The increasing interest in software defined radio (SDR) as enabling technology for defining and developing advanced wireless systems, e.g. mobile ad-hoc networks (MANET) with high degree of adaptivity and ability to (re)configure in application scenarios, motivates research efforts in developing methods and tools for supporting a complete and sound design flow, that encompasses i) waveform/protocol specification, ii) thorough validation through accurate simulations and early stage testing, and then iii) rapid code development on selected target platform. This paper proposes a SW architecture which derives from the application of a new methodology fors synthesis, design and analysis, called Tissue Methodology. The proposed architecture aims at reducing the development time through the use of reconfigurable SW components and the application of automatic code generation techniques.

Key words : Architecture, Methodology, Automatic code generation.

1. INTRODUCTION

The most recent evolutions concerning telecommunication systems have presented the problem of an efficient use of spectral resources. This has pushed the research and industrial communities into the investigation of algorithms for resource dynamical access[7]. The modelling and design of a SDR requires the ability of dynamically configuring a communication system as a function of radio services offered by the environment. Furthermore, the reconfiguration covers all layers of the protocols stack. This high degree of configurability opens new possibilities in terns of services dynamical access from one side, while introducing new problems on the other side due to the dynamicity introduced in the protocol stack.

Despite those needs of high degree of configurability , it can be observed that state-of-the-art approaches offer too complex and heavy architectures resulting in underutilization of their potentialities and still lack significantly in several components, with critical drawbacks in those environments where performance estimation and assessment are particularly challenging (e.g. MANETs). A (non exhaustive) list of current weaknesses in the available tools (e.g. NSx,OPNET, OMNET) can be provided as follows,

1. simulators are not typically conceived to offer the opportunity to reuse the developed code for subsequent implementation on the target device;
2. merging of measurement code and business code is not typically addressed;
3. tools for sound and easy support of tracking projects requirements into the developed code are not available;
4. tools for supporting automatic generation of reports that rely on qualitative and quantitative performance assessment are not satisfactory;
5. high level cross verification for performance analysis (e.g. logic trigger) is not addressed.

With the motivation of bringing improvements in the depicted technical framework, our research group is involved in several projects, e.g. ARTEMIS PRESTO [8] and FP7 NoE HYCON2 [9] both co-funded by the EC. Although the main research problems are different, both projects are conceived with the problem of overcoming limitations due to the current in use technologies. Specifically, the PRESTO project is mainly focused on: 1. the improvement of test-based embedded software development and validation procedure, while considering the constraints of industrial development processes; 2. the definition of functional and performance analysis with platform optimization at an early stage of the development process. The project also intends to explicitly consider some industrial development constraints: simplified use of tools, smooth integration in current design processes, framework of tools that is flexible enough to adapt to different process methodologies, design languages and integration test, platform modeling for early comparison of results with real scenarios and fast prototyping. Through the WP6, HYCON2 also pursues research advances in developing methods and tools for analysis and design in the broad range of complex and networked embedded systems.

The present paper is intended to report on our research activity, that is focused on defining and developing a set of tools (suite) to support the sound design, appropriate verification/test and development of embedded software for SDR systems. Specifically, we are defining a workflow whose qualifying features are as follows: 1. the design of a system or subsystem in a network/protocol stack is model-based; 2. the amount of manually written code (firstly for simulation) is minimized, while the code is usually obtained from the model through a set of procedures for automatic code generation; 3. the probes for measurement may be placed in the model; they can be automatically switched off when the model is used to produce code for target devices; the model holds true independently of the target device; 4. when a probe for measure is selected, the generated track can be automatically added to a technical report. The suite is intended to provide the designer with the abilities of: 1. developing a protocol model through the composition of library components; 2. generating code for simulation and test in a network simulator starting from the model; 3. generating code for a target device, starting from the model; 4. integrating protocol models with application related models, e.g. those encountered in the context of networked control systems. The paper will report on already achieved results in terms of developed models and simulation environments.
The paper is structured as follows: in section 2 we describe the architecture proposed, in section 3 we present the requirements arising from the tissue ide (Integrated Development Environment) assumed, while section 4 is related to the application of tissue pattern for protocol designed, a case-study for IEEE 802.15.4 physical layer is investigated in section 5 and finally section 6 concludes the paper and discusses future works.

2. ARCHITECTURE DEFINITION

In order to introduce improvements with respect to limitations enumerated in the previous section, this section proposes an implementation of a new methodological approach, namely Tissue Methodology[1].
The methodology proposed in [1] emphasizes the following modeling paradigms:

1. modular programming [3], [4], [5];
2. patterns programming ;
3. events oriented programming [6];
4. fractal programming [2].

The design pattern used in this methodology was named as Tissue Pattern. The Tissue pattern has the aim of enabling the design by using a basic module which is able to :

1. receive and generate events (H);
2. process events (P);
3. storing a state space or other information (S);
4. increase their “skills” through interaction with other units or through a reconfiguration;

The growth of tissues is achieved through the repetition of basic units, as well as the fractal structures; the link between H, S and P is represented by functional calls, an access to remote resources, or any communication protocol. The basic units can be used to build macro structures that can be in turn used for growing a tissue. Following this approach, a protocol stack can be rethought as show in Figure 1,.
It show a model made up of a basic tissue pattern; each of the H modules receives events from the other layers and, at the same time, it generates events towards the others layers. Each event, in each layer, is processed from the P module. The data exchanged between two layers, or the data needed to a P module, are stored in the S module.

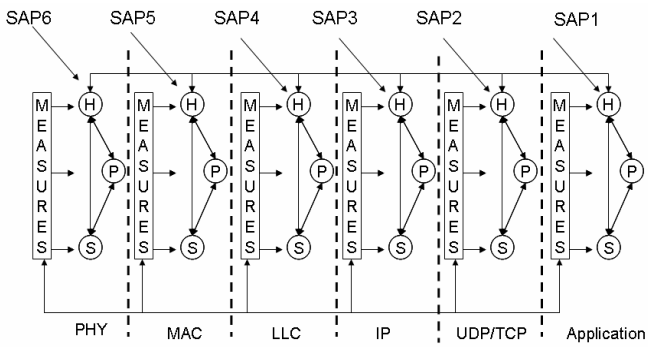

Figure 1 This figure show an example of how to use the basic tissue pattern to model a protocol stack. Specifically, the SAPs (Service Access Points) between two layers is modelled through the H module.

ID	handle
ID1	handle1
IDN	handleN

Figure 2 Data set.

It is worthwhile mentioning that each S module represents a system to store data (e.g. a bank of memories, a remote data source). The P module, or the H module, may retrieve a specific data set, through an identification code. Through this code, the module, which needs to use the data set, receives an handle; this handle enables the use of the data set in read/write mode, Figure 2. The data set could be composed by basic types (e.g. boolean, int, double) or structured data type. If the designer defines all data types needed by the project, the S module may be implemented through automatic code generation. Moreover, using standardized data structures, or rather, data structures which were obtained from predefined data structures, the measure code may be generated in automatic mode too and may be automatically switched off when the model is used to produce code for the target device. Along this vision, it is important to define :

1. a technique to exchange information between different layers, or rather, between different modules of type H;
2. a technique to index of each data set;
3. a technique to manage events generation and the processing.

In the next sections we will suggest a SW implementation for these techniques.

3. REQUIREMENTS OF THE TISSUE IDE

This section deals with the definition of requirements needed to implement a system through the exploitation of the Tissue Methodology. The first step consists in isolating the main characteristics of each paradigm that is at the basis of the Tissue Methodology. The modular programming is the first concept to be taken into account.
In order to be able to build a system through input/output functionalities and memory partitioning, the environment has to provide a support for the creation of the module, that has to be supplied with input/output ports for receiving and generating events (Req.1). Moreover, each module needs to provide an handle through which it is possible to interact with it (Req.2).
In order to exchange events, a communication protocol (Req.3) is required. One of the possible protocols to be used is the Message Passing Interface (MPI); this protocol is particularly interesting because of the existence of a version for real time systems, named Real Time MPI. This is very important for our purposes because the environment we want propose has to allow the designer to simulate the architecture that is going to be implemented on the target device (Req.4). With this design choice, starting from architectural models, it is possible to automates, through automatic code generation, the implementation of the model for simulation activities, or for prototyping activities, in order to avoid any porting activity, any redesign activity, any new testing activity (Req.5).
Basing on previously described requirements, the events simulator OMNET++ has been chosen.
The next section is devoted to describe the Sequence diagram of the design pattern which is used in Figure 1.

4. TISSUE PATTERN FOR PROTOCOLS DESIGN

The issue of designing a protocol layer using tissue methodology, can be solved by the use of the tissue pattern shown in Figure 3.
This pattern represents the basic configuration to build a system where the tasks to deal with are shared among different entities, that can be classified in logical layers. However, if a cross layer interaction is needed, the architecture depicted in Figure 1 inherently supports this interaction. This concept can be explained with the help of Figure 4.

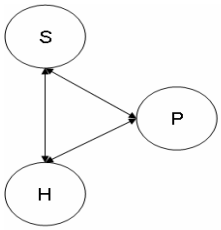

Figure 3 Basic Tissue Pattern.

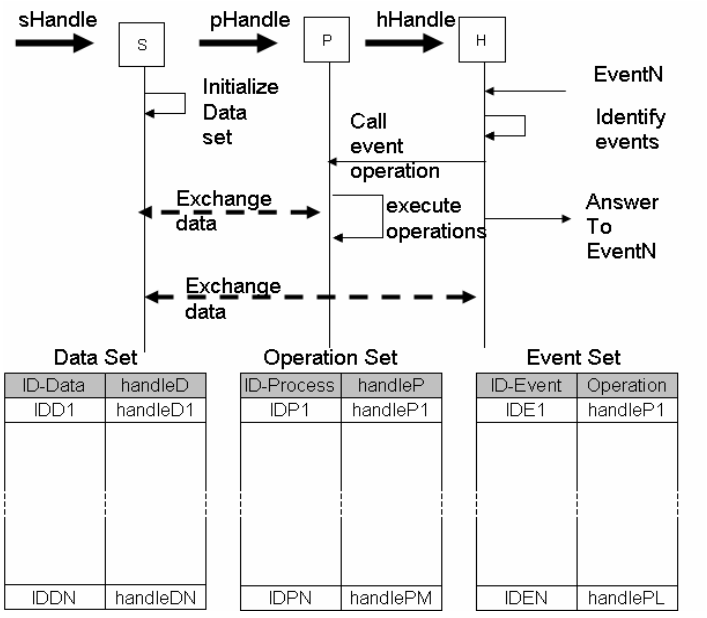

Figure 4 Sequence Diagram for a Basic Tissue Pattern.

Referring to Figure 4, it has to be observed that in addition to the Data Set there are: the operation set,containing the handles for each functionality, and the event set, containing the handles to the events. Each data set is characterized by two coordinates, the handle of the reference storage, sHandle, and the handle of the specific data, handleDk, with k referring to indices data types in range [1,N]. for the data set, [1,M] for the operation set, [1,L] for the event set.
This choice brings to the following advantages :

1. all the data set can be moved from one storage module to another;
2. any data set can be handled from any P module or H module; indeed it is possible to use the pair (sHandle,handleDk) from any module;

Moreover, the exploitation of previous advantages facilitates measurement operations; this way another requirement of our IDE (Integrated Development Environment) is fullfilled.
Before going into the development of techniques to automate the generation of the code through which implementing the tissue pattern, and hence the protocol stack, it is necessary to verify the feasibility of these techniques. Next section proposes an application of the tissue pattern to a 802.15.4 physical layer.

5. APPLICATION OF THE TISSUE PATTERN TO IMPLEMENT 802.15.4 PHYSICAL LAYER

In order to verify the feasibility of this methodology and to spotlight the differences with a traditional design, like OOP (Object Oriented Programming), we started from an existing project and we redesigned this project through the reusing the existing code in order to produce a Tissue Methodology compliant implementation. The project chosen is part of a framework developed to simulate MANET networks through the use of the OMNET++. The framework is named INETMANET; we focused on the physical layer 802.15.4 implementation. The process followed to do this conversion includes the following steps:

1. definition of data types to cover all the data managed into the phy layer;
2. association of a unique identification code to each data type;
3. association of a unique handle to each data type;

Figure 5 shows the identification codes, while Figure 6 shows the implementation of the handles for each data type, basing on the “map” data structure of C++ STL (Standard Library). The following methods have been implemented to manage data types:

1. virtual void* select802154Data(const char* data,int* typeData,wrapper_t tW): it returns the handle to specified through the typeData ID;.
2. virtual void set802154Data(const char* data,int* typeData,wrapper_t tW,void* dataMP): it adds a new data structure

in order to retrieve the handle of the storage module, the needed methods are :

1. cModule*hs802154PHY=(getParentModule()->getSubmodule("sphy"));
2. ::S802154PHY*hS802154PHY=check_and_cast<S802154PHY *>(hs802154PHY);

This is a way to satisfy the requirement Req.2; in order to meet Req.3, the basic functionalities of the H module are listed below :

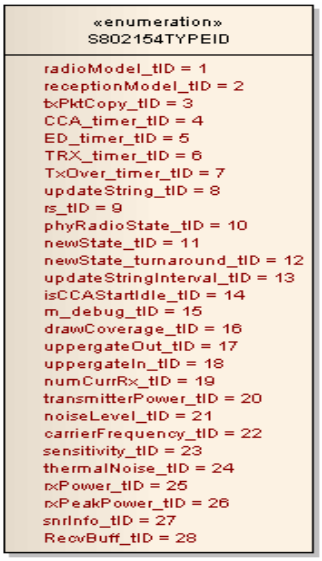

Figure 5 Data ID list.

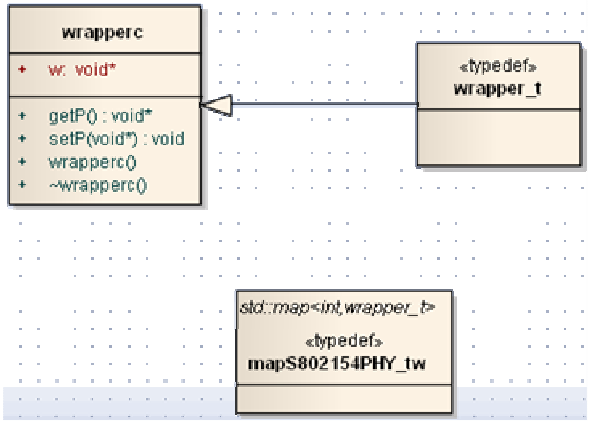

Figure 6 Class Diagram for the generic set.

1. virtual void fCSend(cMessage* msg,int idGate,int sel,simtime_t t); it is needed to control the generation of events in the H module;
2. virtual void fCSelfMsg(simtime_t t,cMessage* msg); it is needed to set internal events (e.g. Timer);
3. virtual void fCancEvent(cMessage* msg,int sel); it is needed to cancel an event which is expired or that was processed;
4. virtual void deleteSelfMsg(cMessage* msg); it is needed to cancel an internal event which is expired or that was processed;

These methods are necessary since the H module has to generate events; however, the P module, as a result of a processing, could need to generate an event.

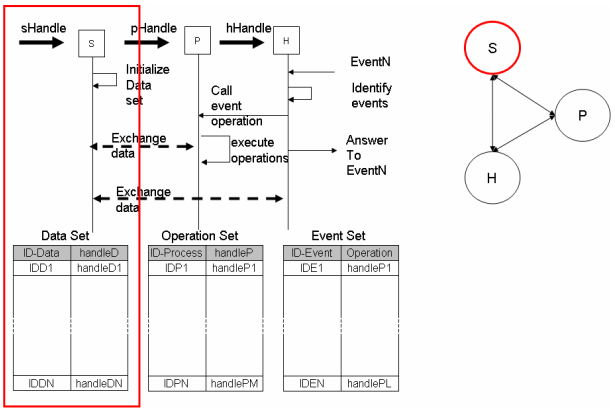

Figure 7

To do this, the P module has to be able to let the event to the H module, in order for the H module to generate the event towards the destination.
When an event is received on the H interface, the H module ask the P module for the execution of one of the following operations :

1. updateDisplayString(*drawCoverage,*sensitivity,*transmitterPower,updateString,*updateStringInterval);
2. handlePrimitive(msg->getKind(), msg) : it is useful to manage the primitives exchange between the 802.15.4 physical layer and 802.15.4 mac layer;
3. handleUpperMsg(airframe) : it is useful to manage the messages originated from the mac layer;
4. handleSelfMsg(msg) : it is useful to manage the internal messages;
5. handleLowerMsgStart(airframe);
6. bufferMsg(airframe) : it is useful to manage the queue of the air frames Protocol Data Units;

All these methods are placed in the P module. This closes the implementation of the basic tissue pattern of Figure 4. Now, basing on Software Defined Radio paradigms, the reprogramming of a device is to be taken into account. Using tissue architecture, the reprogramming could be implemented through the execution of the following steps :

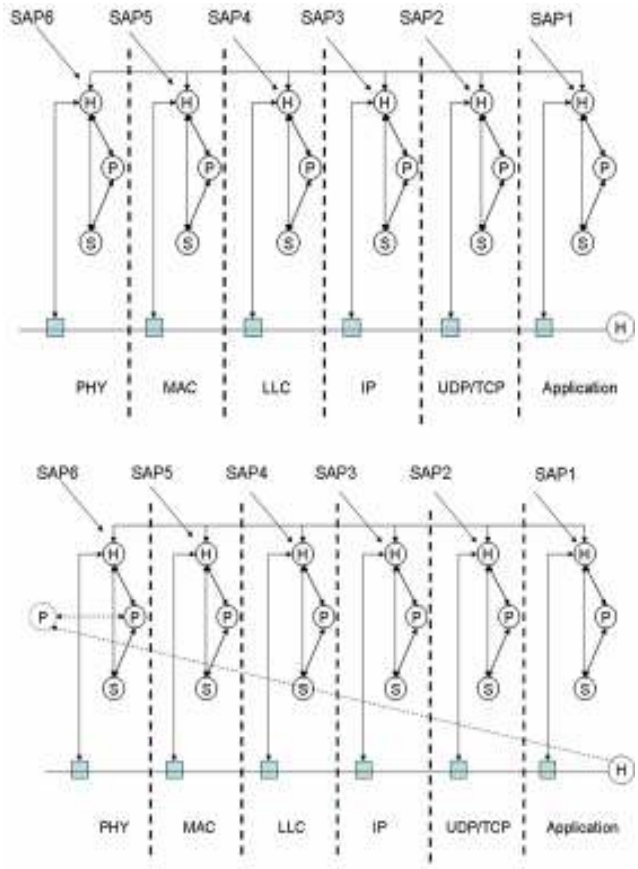

Figure 8 Example of dynamic tissue pattern reconfiguration.

1. if the generic H module is not able to identify an operation request for its P module, it sends a notification of not recognized event to another H module, devoted to recognize this type of events. This specific module, is trained to process new events through the use of an xml file, which contains the information about the type of event and the type of P module (part of a collection of P modules) which has to be instantiated.

2. The H module that has to manage unrecognized messages can instantiate a new P module, specifically designed for the new types of event, and link the new P module with the old P Module, as depicted in Figure 8. Moreover, the old H module is updated;

6. CONCLUSIONS AND FUTURE WORK

In this paper we analyzed the implementation requirements needed of a new methodology, named Tissue Methodology; moreover, an environment which satisfies these requirements was selected, specifically OMNET++. In the context of this environment, a basic tissue pattern was developed and exploited to study the applicability of the tissue methodology to implement an IEEE 802.15.4 phy layer model.

The future works, already in progress, regards the development of techniques to automate the generation of the code for S, H and P modules for the basic tissue pattern used for IEEE 802.15.4 PHY layer, and to extend the basic tissue pattern to others layers of the protocols stack, specifically 802.15.4 MAC layer.

7. REFERENCES

[1] M. Colizza, M. Faccio, C. Rinaldi, F. Santucci, " A METHODOLOGY TO DESIGN AN ADVANCED FRAMEWORK FOR EFFICIENT MODELLING AND TESTING OF MANETS", in Proc. of Wireless Telecommunication Synmposium, IEEE April 2012, to appear

[2] http://fractal.ow2.org/documentation.html

[3] K.K. Lau and Z. Wang. Software Component Models. IEEE Trans. Software Eng., vol. 33 n° 10 , October 2007.

[4] S. Sicard, F. Boyer, and N. De Palma. Using Components for Architecture-Based Management: The Self-Repair Case. in Proc. of 30th International Conference on Software Engineering (ICSE 2008). ACM, 2008, ISBN: 978-1-60558-079-1

[5] P.C. David, M. L´eger, H. Grall, T. Ledoux, and T. Coupaye. A Multi-stage Approach for Reliable Dynamic Reconfigurations of Component-Based Systems. In 8th IFIP Int. Conf. Distributed Applications and Interoperable Systems, DAIS 2008, volume 5053 of LNCS, 2008.

[6] K. Mani Chandy, Michel. Charpentier, Agostino Capponi, Towards a Theory of Events DEBS '07, June 20–22, 2007 Toronto, Ontario, Canada

[7] III Mitola, J. and Jr. Maguire, G.Q., "Cognitive radio: making software radios more personal," Personal Communications, IEEE, vol. 6, no. 4, pp. 13 –18, aug 1999.

[8] http://www.presto-embedded.eu/

[9] http://www.hycon2.eu/