A Framework for Protocol Reconfiguration

E. Patouni, N. Alonistioti, P. Magdalinos

Communication Networks Laboratory,
Department of Informatics and Telecommunications, National and Kapodistrian University of Athens,
Ilisia Campus 157 84 Athens, Greece

Contact Details:
Eleni Patouni
Communication Networks Laboratory,
Department of Informatics and Telecommunications,
National and Kapodistrian University of Athens,
Ilisia Campus 157 84 Athens, Greece

Tel:+30 6945 665693
E-mail: elenip@di.uoa.gr

Abstract
Over the last decade the mobile industry spawned numerous wireless technologies and access systems with different capabilities that coexist forming a new era in the area of wireless technologies. The current trend is the facilitation and the combination of the capabilities of the different systems, and is introduced within the reconfigurability concept. The latter lies in the dynamic adaptation of the equipment in order to meet the evolving application requirements and other needs. This concept can be realized with the introduction of flexible protocols. In this analysis, a proposed framework which describes the necessary mechanisms for the dynamic binding and reconfiguration of protocol components is presented.

***Keywords*— protocol component, binding, reconfiguration**

A Framework for Protocol Reconfiguration

E. Patouni, N. Alonistioti, P. Magdalinos

Communication Networks Laboratory,
Department of Informatics and Telecommunications, National and Kapodistrian University of Athens,
Athens, Greece
email: { elenip, nancy, panagis }@di.uoa.gr

Abstract— Over the last decade the mobile industry spawned numerous wireless technologies and access systems with different capabilities that coexist forming a new era in the area of wireless technologies. The current trend is the facilitation and the combination of the capabilities of the different systems, and is introduced within the reconfigurability concept. The latter lies in the dynamic adaptation of the equipment in order to meet the evolving application requirements and other needs. This concept can be realized with the introduction of flexible protocols. In this analysis, a proposed framework which describes the necessary mechanisms for the dynamic binding and reconfiguration of protocol components is presented.

***Keywords*— protocol component, binding, reconfiguration**

I. INTRODUCTION

In the next decade, the evolution of 3G mobile systems and the introduction of 4G in mobile and wireless communications are expected to play a central role in all aspects of mobile users' activities. The technology will substantially expand on the current concept of "anywhere, anytime" to a new paradigm summarized in the notion of "advanced user experience" of mobile services.

The expectation for the evolution of current mobile and wireless communications is the support of multiple access environment based on heterogeneous networks. In such a complex environment, reconfigurable systems in all network entities will be essential to support flexibility. This concept sketches the existence of multi-mode, reconfigurable equipment, which will support the dynamic and seamless adaptation to the different wireless networks. The realization of the above concept requires the introduction of similar flexibility to the protocol layers that form the equipment's protocol stack.

An important aspect for protocol reconfiguration is the decomposition of protocol functionality into components able to be re-assembled dynamically, thus providing the full protocol functionality. Major research issue is the introduction of a component based framework which enables the dynamic synthesis of protocol layer functionality (protocol component dynamic binding) and protocol reconfiguration (i.e. dynamic replacement of a protocol component with another). In this paper we present such a framework and mechanisms for protocol reconfiguration.

The rest of the paper is organized as follows. In Section II, related work is presented. Section III clarifies the necessary mechanisms that enable the protocol reconfiguration process. In Section IV, the definition of the protocol component is presented. Section V analyzes the steps for the protocol reconfiguration process. Section VI describes the process of validating the proposed framework, while section VII presents the performance assessment of this system. Finally, some conclusion remarks and directions for future research are drawn.

II. RELATED WORK

The idea of composing protocols out of components was the basic concept of many frameworks in the past. One of these frameworks is the X-Kernel [1]. In this standard runtime framework, composite protocols are composed of microprotocol objects with respect to the application requirements using protocol graphs. The Cactus system extended the above concept of the X-kernel framework, offering a hierarchical composition mechanism for composite protocols [2], [3]. A different mechanism is being implemented in the Appia Framework [2]. In this system, a protocol is composed of one or more modules, placed in layers. The communication between the individual modules is achieved using event routing between static communication channels. All the above-mentioned frameworks are characterized by a serious disadvantage: the correctness of composition cannot be completely assured and formally verified.

Ensemble is a group communications system which supports the communication between the protocol modules using the linear stacking composition mechanism and exploiting the packet concept [3]. This approach has the disadvantage that it allows every component to communicate directly only with its two adjacent components.

The component concept is also applied in the THINK-FRACTAL frameworks. FRACTAL introduces a generic

recursive component model which allows sharing of sub-components between components [4]. However, this model cannot be easily applied to protocol layers. THINK is another component-based framework that targets on building flexible operating system kernels [5].

Furthermore, another category of frameworks were developed, which considered a whole protocol layer as a component. This design is applied in DiPS/CuPS, which allow dynamic adaptation of protocol stacks, in order to come up to the applications requirements or the network optimization issues [6], [7].

III. MECHANISMS FOR PROTOCOL RECONFIGURATION

The deployment of the protocol reconfiguration concept lies in the capability to realize the specified reconfiguration during runtime interaction between the protocol components. In this context, it should be possible to dynamically compose the protocol components so as to realize the functionality of the specified protocol. Another issue is the capability to replace protocol functionality, specified in protocol components, thus guaranteeing that the new component will be configured in a proper way to achieve seamless replacement [8].

The above requirements introduce great complexity, which should not affect the reliable operation of the protocol stack. Therefore, it is necessary to introduce mechanisms which should control the protocol reconfiguration process. Our concept lies in the introduction of the following mechanisms:

- a Manager entity within each protocol, and
- a semantic layer of information for each protocol component; its metadata [9].

This concept is illustrated in Fig.1, which presents a reconfigurable protocol stack; each protocol layer comprises several components and one Manager.

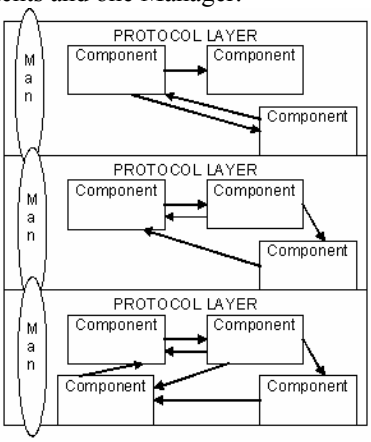

Fig.1. A Reconfigurable Protocol stack exploiting the Protocol Component Concept

The Manager is responsible for verifying and realizing the dynamic composition of the protocol components. In addition, the Manager has the overall control of the process of the dynamic replacement of protocol components. The Manager performs the above-mentioned functions based on the component's metadata.

Therefore the components metadata should include all the necessary information that will allow the Manager to identify the binding between the components, verify the correctness of the proposed binding and perform this binding (realize the communication between the protocol components that should be bound together). In this framework, the component's metadata include a unique identifier (i.e, the name of the component), the version of the component and the component's composition information.

Both the unique identifier and version enable the identification of each protocol component. In addition, the versioning information is used in order to prevent the binding of incompatible components. The component composition information indicates which components are composed with the specified component. In particular, the component composition information is included within two array structures; an array for the input components (the components that send data to the specified component) and an array for the output components (the components that receive data from the specified component). The Manager based on the input and output components arrays identifies the components that should be bound to the specified component. The above process is illustrated with an example. Let us suppose that componentX's metadata include the following input and output arrays: inputArray = [componentA], outputArray = [componentB].

The Manager should first check the input array for the componentX; this array includes componentA. Then it should verify the composition between componentX and componentA by checking that the output array in the componentA's metadata includes componentX. Thereafter, the Manager should check the output array in the componentX metadata and verify the composition between componentX and componentB in the same way.

At this point, the implementation issue of the component metadata will be discussed. Up to now, this semantic layer of information is inserted in an ASCII file for each component. Another approach is the representation of the metadata with the use of XML or RDF(S), since the latter provide great interoperability, extensibility and are commonly comprehensive [10]. In both cases, the Metadata should accompany a protocol component; every time a protocol component is being downloaded in the reconfigurable

equipment, its metadata will be downloaded too.

IV. Protocol Component Definition

The basic step for enabling the reconfigurability concept within the protocol layers is the definition of the protocol component. The latter should be done taking into account that the defined protocol components should realize the functionality of the specified protocol when they are integrated. In addition, it should be possible to compose a specified protocol by using combinations of different protocol components; however, the specified protocol functionality should remain the same.

The fulfillment of the above requirements raised the need for the definition of a generic reconfigurable component interface (ReconfigurableComponentInterface) and the introduction of the ReconfigurableComponent, which realizes this interface (Fig.2.). Each protocol component should be implemented as a subclass of the ReconfigurableComponent (Fig.2). This way the functionality of the protocol component is different for each protocol component and it is introduced in the function method.

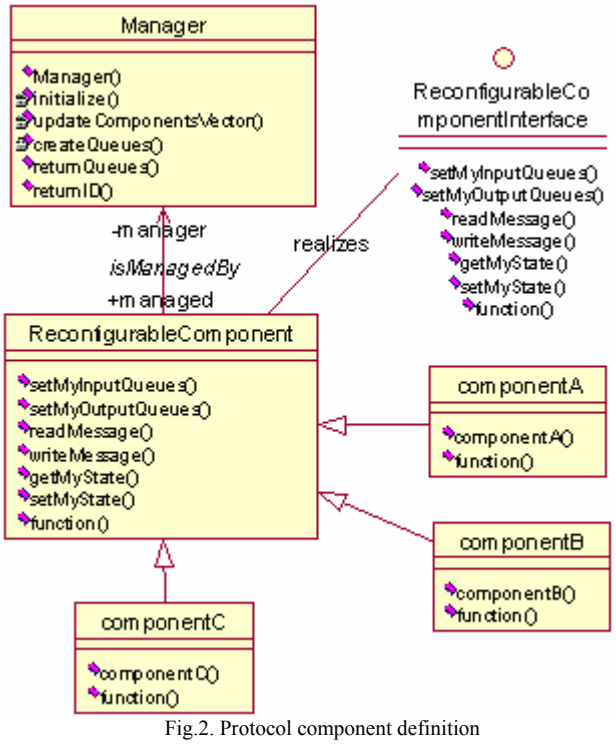

Fig.2. Protocol component definition

At the same time, all the protocol components have some common mechanisms regarding the control and deployment of their reconfiguration. The latter are defined in the following methods, which will be further analyzed later in this paper:

- the writeMessage/ readMessage methods. These methods define the behavior of the component during the communication with other protocol components.
- the setMyOutputQueues/setMyInputQueues methods. These methods are used for the composition of the different components.
- the getMyState/setMyState methods. These methods were introduced for the achievement of seamless protocol component reconfiguration

V. Protocol Component Reconfiguration

Up to this point, the mechanisms that enable the establishment of the protocol reconfiguration process were analyzed. In this section, the interactions between the Manager and the protocol components during the process of their dynamic binding and replacement are identified.

Regarding the dynamic binding of protocol components, once the Manager identifies that two protocol components should be composed, it realizes their composition; the latter is implemented with the use of First-In First-Out (FIFO) queues. In particular, the Manager creates a FIFO queue for each pair of components that should communicate with each other. Thereafter it passes the queue handler to both of the components. The sending component places its packets in the FIFO queue and the receiving component extracts them.

Supposing that a protocol component communicates with N components, then this component will realize this communication with the use of N FIFO queues the Manager creates. These N components, when composed, create a graph; each edge of this graph is implemented with a FIFO queue. Furthermore, the queue handlers corresponding to the FIFO queues are stored in a two dimensional array in order to enable their retrieval when necessary.

At this point a reconfiguration scenario which requires the downloading of a new protocol composed of two components, componentA and componentB, will be described. The first phase of this reconfiguration scenario is the downloading procedure of the components and their metadata, which is being realized by a downloading entity. Thereafter, the Manager is triggered to realize the protocol component binding (Fig.3). Initially, the Manager retrieves the components Metadata. Then it checks the composition of the two components. Supposing that the composition realizes a directional communication between componentA and componentB (componentA sends data to componentB), then the Manager should create only one FIFO queue. Thereafter it passes the queue handler to both the protocol components.

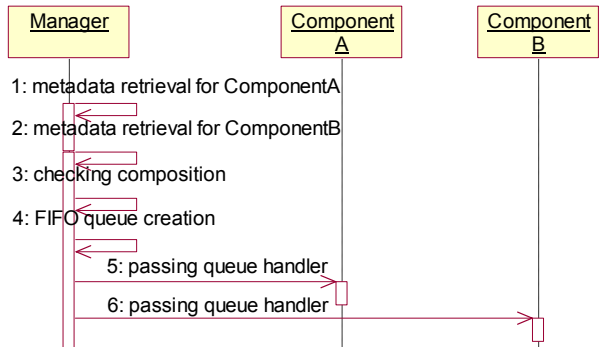

Fig.3. Sequence diagram illustrating the composition process of two protocol components.

At this point, the necessary steps for the replacement of a protocol component will be presented (Fig.4.). In this scenario, we consider the dynamic replacement of componentA with componentC. In addition, it is assumed that componentA is binded to the same components that componentC does. At first, the Manager retrieves the metadata for the new component (componentC) and checks which components it is composed with. Then it pauses the function of the replaceable component (componentA). To the next step, the Manager retrieves the state of componentA and sets the state of componentC to the same state with componentA. This is necessary since the replacement of a component which is in a specific state, with a component which is in an idle or initial state, does not guarantee the reliable operation of the protocol.

Finally, the Manager should establish the composition between componentC and the components which it communicates with. This is achieved using a similar way to the component binding mechanism. At first the Manager retrieves the queue handlers that correspond to the components that componentA communicates with (from the two dimensional array they are stored). Then it simply passes this set of queue handlers to componentC.

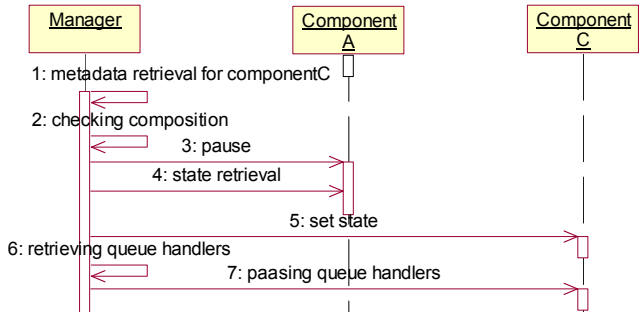

Fig. 4. A sequence diagram illustrating the process of replacing componentA with componentC

In case that the new component communicates with more components than the old one, the Manager should not only retrieve the queue handlers that correspond to the components the old component communicates with, but it should also create the necessary queue handlers that correspond to the other components the new component communicates with.

VI. VALIDATION OF THE PROPOSED FRAMEWORK

In order to validate the proposed framework architecture and functionality we have implemented a proof-of-concept prototype. Initially, a test protocol was implemented and used for verification of the mechanisms; this protocol was decomposed into three protocol components. The function of these components was the forwarding of the received information or both the modifying and forwarding the received information.

In terms of applicability of the proposed framework also to other protocols, currently, successful tests and respective validation have been performed for the FTP protocol functionality. The FTP protocol was decomposed into two components. The first FTP component includes the necessary functionality for establishing the FTP connection plus handling the process of uploading a file. The second FTP component realizes the downloading process. Their communication is unidirectional; the first FTP component sends data to the second FTP component. In particular, the data exchange is related to the request for downloading a file and the necessary parameters for the downloading phase. The process of dynamic component binding and reconfiguration was successfully validated under the current prototype implementation. Both the proposed framework and the protocols were implemented in Java. The prototype implementation has proven that the protocol components, when reassembled, have resulted in a robust protocol functionality based on the initial protocol specification.

VII. PERFORMANCE ASSESSMENT

In addition, preliminary performance tests of the proposed framework have been executed within the components of the FTP protocol. This performance assessment targeted to the calculation of the delay that is created by the insertion of the FIFO queue for the communication between the FTP protocol components. More specifically, we considered the user request to download a file. Firstly the overall time T for processing this request until entering the downloading phase was counted for the simple FTP protocol. To the next step, we counted the delay that is created within the component based FTP protocol, by the requirement for the first FTP component

to place its packet into the FIFO queue and the second FTP component to extract it. Moreover the percentage (%) of this binding delay to the time T was calculated (binding delay/T) for a set of 100 samples. The mean value of the binding delay to the time T for this set of samples was calculated to be 4.72%. The minimum of this value was 1.10%, while the maximum was 11.08%. Fig.5 illustrates the percentage of the binding delay divided to the time T for each sample, considering the capabilities of the underlying hardware (Pentium III 800MHz PC with 512MB RAM).

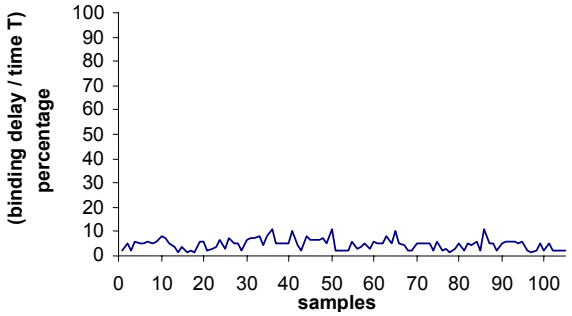

Fig. 5. The percentage of (binding delay / T) for a set of 100 samples

At this point, we should note that the proposed framework inserts a very low value of delay in the functionality of the FTP protocol. This fact is caused by the design of the analyzed framework and the design approach for the FTP protocol components. The definition of protocol components within this protocol was carefully done so as to fulfill the following two requirements.

- to allow the defined components to have robust functionality enabling the execution of meaningful reconfiguration scenarios (i.e., a reconfiguration scenario for the replacement of the FTP component which implements the downloading procedure with another one which includes more capabilities),
- not to introduce overhead for the communication between the composed protocol components; this is achieved when the information that should be exchanged between the protocol components is not actual protocol data, but configuration information or request for a function of this protocol.

VIII. CONCLUSION

The mechanisms that were presented in this framework introduce an innovative system for dynamic protocol component binding and reconfiguration. This approach does not limit the number or the functionality of the protocol components, as it provides the possibility that different combinations of protocol components may compose a protocol. Moreover, the correction of the composition is verified by the functionality implemented in the protocol Manager.

Further research lies in the deployment and simulation of the proposed mechanisms in a set of protocols. In addition, simulations concerning the queue length should be executed, in order to specify whether the introduction of a light flow control mechanism for queue overflowing is necessary.

Another issue is the introduction of the Manager within the protocol. At first, the introduction of a common Manager entity for the whole protocol stack should be considered. To the next step, the possibility of removing the Manager and distributing its functionality in the protocol components should be taken into account.

ACKNOWLEDGMENT

This work has been performed in the framework of the EU funded project E2R. The authors would like to acknowledge the contributions of their colleagues from E2R consortium.

REFERENCES

[1] Norman C. Hutchinson and Larry L.Peterson: The x-kernel: An architecture for implementing network protocols. IEEE Transactions on Software Engineering, 17(1):64-76, January 1991.

[2] Sergio Mena, Xavier Cuvellier, Christophe Gregoire, Andre Schiper: Appia vs. Cactus: Comparing Protocol Composition Frameworks. 22nd International Symposium on Reliable Distributed Systems (SRDS'03), Florence, Italy, October 2003.

[3] Magesh Kannan, Ed Komp, Gary Minden, Joseph Evans: Design and Implementation of Composite Protocols. Technical Report ITTC-FY2003-TR-19740-05, Feb 2003

[4] E. Bruneton, T. Coupaye, and J.B. Stefani. Recursive and Dynamic Software Composition with Sharing. Seventh International Workshop on Component-Oriented Programming (WCOP02), Monday, June 10, 2002 - At ECOOP 2002, Malaga, Spain (June 10-14, 2002)

[5] Jean-Philippe Fassino, Jean-Bernard Stefani, Julia Lawall and Gilles Muller. THINK: A Software Framework for Component-based Operating System Kernels. In Proceedings of the USENIX Annual Technical Conference, 2002.

[6] N. Janssens, S. Michiels and P. Verbaeten: DiPS/CuPS: a Framework for Runtime Customizable Protocol Stacks. CW Technical Report 328, Dept. of Comp. Science, K.U.Leuven. November 2001.

[7] S. Michiels, T. Mahieu, F. Matthijs and P.Verbaeten: Dynamic Protocol Stack Composition: Protocol independent Addressing. In 4th ECOOP Workshop on Object-Orientation and Operating Systems (ECOOPOOOSWS' 2001), June 2001.

[8] G. Minden, E. Komp, S. Ganje, M. Kannan, S. Subramaniam, S. Tan, S. Vallabhaneni, J.Evans: Composite Protocols for Innovative Active Services. DARPA Active Networks Conference and Exposition (DANCE'02), San Francisco, CA, May 2002.

[9] Vagelis Gazis, Nancy Alonistioti and Lazaros Merakos: Metadata Design for Introspection-Capable Reconfigurable Systems. Third IFIP-TC6 Networking Conference (Networking 2004), Athens, May 2004.

[10] http://www.w3.org/TR/rdf-schema, (Resource Description Framework (RDF) Schema Specification 1.0a)