Application and Network Services Composition with the Help of Mediation

Abbas Siddiqui*, Michael Kleis†, Julius Mueller‡, Paul Müller*, Thomas Magedanz‡
* University of Kaiserslautern, Postbox 3049 67653 Kaiserslautern, Germany
Email: {siddiqui,pmueller}@informatik.uni-kl.de
†Fraunhofer FOKUS, Kaiserin-Augusta-Allee 31, 10589 Berlin, Germany
Email: {michael.kleis}@fokus.fraunhofer.de.de
‡Technical University Berlin, Straße des 17. Juni 135, 10623 Berlin, Germany
Email: {julius.mueller, thomas.magedanz}@tu-berlin.de

I. Introduction

In todays Internet the network stack is divided into distinct layers which can be implemented by different protocols. Each layer offers a service to directly adjacent layers. Although this crisp and robust design has proved its advantages (e.g. functionality scoping, stability) there are also disadvantages of this architecture. Protocols on different layers implement the same functionality (e.g. IP and TCP Checksum), the physical layer is not aware of the application and cannot adapt error correction or coding (e.g. for multimedia over wireless). Besides this, there are also issues like cyberspace tussles [8], the increasing mobility of the end-hosts and the ossification of the Internet due to the increasing complexity of the protocol interdependencies that lead to some new architecture proposals for a Future Internet. One proposal is Functional Composition (FC) which decomposes the functionalities of the network stack in different functional blocks. These functional blocks are loosely coupled and provide means to exchange information between functionalities of different levels. Many projects (e.g. ANA [3], RBA [2], 4WARD [1], Net-Silo [5], RNA [6], Network Service Architecture [4], and SONATE [9]) have addressed this approach from different perspective to find a best solution for a flexible future Internet architecture which can cope with the requirements of futuristic trends.

The G-Lab DEEP [11] cross-layer FC architecture leads to a two-layer functional composition architecture. Services (e.g. web services, encoding service) are composed at service layer and network services at network layer. Scope of services at network and service layer is not limited as any kind of service could be implemented at any of both layers but it is important to take in to consideration where specific service would be most optimized and efficient. This separation is still valuable because an application designer should not know and compose the network functional blocks by himself but explicitly state the abstract requirements of an application, e.g. encryption and QoS (maximum delay, maximum loss). Nevertheless, there should be a feedback of the network if requirements can be met or not, thus service level can react by realizing e.g. encryption on service level, using another media encoding, or by selecting a different content source. For this purpose we propose in this paper a cross layer mediator that negotiates and exchange information between the two layers. We will explain the main concept of the mediator in the following sections.

II. Cross-Layer Mediation

In the G-Lab DEEP project [11], we consider FC on Application and Network Level. The reason for this is based on the fact that e.g. functional blocks for real-time media processing may not be instantiated as network components because of their comparable high computing demands.

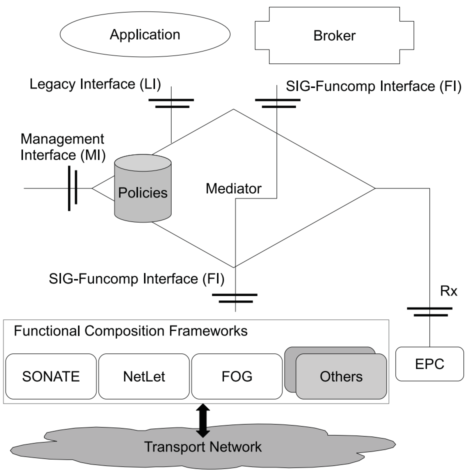

Fig. 1. *Mediator and Interfaces*

To be able to instantiate such functional blocks on application level a Broker component is used. In case a client demands a service, which cannot be resolved by a single service instance, the Broker is capable to combine several service blocks into a workflow. Additionally the broker derives the different service block requirements and signals them to the mediator.

The Mediator component ,we propose, is comparable to an intelligent middle ware allowing the application to abstract from the used FC framework or more general transport network. In fig. 1, the conceptional placement of components is shown such as mediator, functional composition frameworks and relative position of APIs. The actual mediation is not about selection of a FC framework but services presented at service level and network level which may be provided by different FC frameworks. Nevertheless in G-Lab DEEP context, we are focus on the SONATE FC framework [9]. For the mediation process Policies are used to resolve the conflicts and to derive a mediation decision. The used policies are considered to be domain (e.g. telephony, multimedia, file transfer) specific. The interfaces provided by the Mediator are:

1) Legacy Interface (**LI**): The LI interface is based on BSD Sockets and can be used by legacy applications to access a FC based network. The Mediator performs all required tasks to establish network connectivity for the legacy application.
2) Management Interface (**MI**): The MI Interface is used for mediator to mediator communication or can be utilized by network operators to inject policies to be used for the actual mediation process.
3) Functional Composition Interface (**FI**): The FI interface is based on the abstraction library developed by the Special Interest Group on FC [10]. Based on calls to the FI library Network connectivity and network FC based on specified requirements can be triggered for all FC frameworks developed inside G-Lab Projects. The FI API offers an URI based communication paradigm comparable to current content based addressing schemes.
4) EPC conform **Rx** Interface: This interface can be used to interact with the 3GPP EPC framework.

A. *How Mediation Works*

To perform a mediation, it requires input from different resources as shown in fig. 2 e.g. application requirements, services from network and service layer, policies.

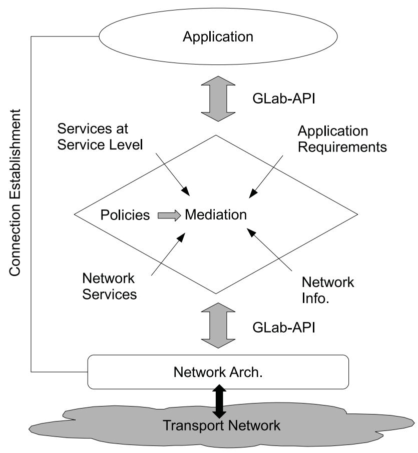

Fig. 2. *Required input for mediation*

An application will send requirements to the broker, it is a task of the broker to look for existing services at the service layer with respect to the application requirements and inform the mediator about existing services and the application requirements. As soon as the mediator received a request from the broker, it will look for possible services from the network layer and if possible then check network constraints (e.g. bandwidth, wireless or wired network, etc). Policies play one of the major roles in the mediation process. Policies are simple rules which are related to a particular domain (e.g. telephony, file transfer). The mediator component uses given policies to infer the cross-layer composition. Possible workflows are also part of policies but those workflows are not filled up with any particular implementation (i.e building blocks) of a service . After selecting a suitable workflow with respect to QoS parameters, mediator will delegate the task of execution of services to service and network layer. An FI API call is triggered to set-up a connection. The resulting connection instance will be given back from the network architecture, in case of a successful execution of a workflow, which will be further forwarded to the application via broker so that a connection will be established.

III. Conclusion

In this paper a mediation process has been proposed which provides more flexibility in a cross-layer FC architecture. Instead of only following a top down approach where the application tells the network its requirements, the network and application can interact to find a suitable solution. In a FC approach, certain application level services are likely to move down to the network level. Mediation helps to determine where functionalties should be executed in an optimal manner. In the poster presentation related to this abstract we provide a more technical description of the described use case.

IV. Acknowledgment

This work is funded by the German Federal Ministry of Education and Research within the scope of the G-LAB DEEP project [11] as part of the G-Lab project.

References

[1] 4WARD EU Project. http://www.4ward-project.eu/
[2] Robert Braden, Ted Faber, and Mark Handley. From protocol stack to protocol heap: Role-based architecture (2003).
[3] Autonomic Network Architecture (ANA). http://www.ana-project.org
[4] Sivakumar Ganapathy and Tilman Wolf. Design of a network service architecture, in Proc. of Sixteenth IEEE International Conference on Computer Communications and Networks (ICCCN), Honolulu, HI, (Aug. 2007).
[5] R. Dutta, G.N. Rouskas, I. Baldine, A. Bragg, and D. Stevenson. The Silo architecture for services integration, control, and optimization for the future internet Communications, 2007. ICC '07. IEEE International Conference, (June 2007).
[6] Venkata Pingali, Joseph D. Touch, Yu-Shun Wang. A recursive network architecture (2006).
[7] Howard Foster, Arun Mukhija, David S. Rosenblum and Sebatian Uchitel. A Model-Driven Approach to Dynamic and Adaptive Service Brokering using Modes.
[8] David D. Clark, Karen R. Sollins, John Wroclawski, Robert Braden. Tussle in Cyberspace: Defining Tomorrowds Internet, In Proc. ACM SIGCOMM (2002)
[9] Paul Mueller, Bernd Reuther. Future Internet Architecture - A Service Oriented Approach, it - Information Technology, Jahrgang 50 (2008)
[10] G-Lab Special Interest Group Functional Composition. GAP: A G-Lab Application-to-Network Interface, Euroview2011, Würzbug, Germany
[11] BMBF Funded Project, G-Lab DEEP, http://www.g-lab-deep.de/
[12] 3GPP TS 36.300, Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN).
[13] 3GPP TS 23.203, V10.0.0 (2010-06), Policy and charging control architecture (Release 10).