A Demonstrator for Cross-Layer Composition

Martin Becke*, Konrad Campowsky†, Christian Henke†, Dirk Hoffstadt *, Julius Müller †,
Carsten Schmoll ‡, Abbas Ali Siddiqui §, Irfan Simsek *, Thomas Magedanz †,
Paul Müller §, Erwin Rathgeb * and Tanja Zseby ‡
*Universität Duisburg - Essen
Ellernstr. 29, D-45326 Essen
Email: martin.becke@uni-due.de, dirk.hoffstadt@iem.uni-due.de, irfan.simsek@iem.uni-due.de, erwin.rathgeb@iem.uni-due.de
†Technical University Berlin
Strasse des 17. Juni 135, 10623 Berlin
Email: konrad.campowsky@tu-berlin.de, c.henke@tu-berlin.de, julius.mueller@tu-berlin.de, tm@cs.tu-berlin.de
‡Fraunhofer FOKUS
Kaiserin-Augusta-Allee 31, 10589 Berlin
Email: carsten.schmoll@fokus.fraunhofer.de, tanja.zseby@fokus.fraunhofer.de
§TU Kaiserslautern
Fachbereich Informatik, Postfach 3049, 67653 Kaiserslautern
siddiqui@informatik.uni-kl.de, pmueller@informatik.uni-kl.de

I. Introduction

Services of all-IP networks today are expecting the best-effort packet transport paradigm of the underlying access network to which the mobile and fixed devices are connected to. Thereby applications are not able to indicate their requirements on the connection for the delivery of their data flows. The project G-Lab DEEP presents a Future Internet architecture based on the cross-layer composition concept in which the network is composed according to application specific requirements. Functional composition is used to selects and composes functionalities on the network data path, suitable for the demanded application level requirements.

II. G-Lab DEEP Architecture

This section covers the main components briefly and presents their key functionalities.

- MyMONSTER: The IMS client framework MyMONSTER [1] is equipped with an add-on providing functionalities to state an intent which is either a standard or emergency call. This intent is signaled to the Service Broker.
- IMS: The IP Multimedia Subsystem (IMS) hosts the user database (HSS), which is used for accounting, charging, provisioning of users. Additionally it enables Authentication, Authorization and Accounting (AAA).
- Service Broker: The Service Broker [2] translates the intent in first place, determines the information of the demanded action and formulates a request. The Service Broker identifies the required services, which are necessary to satisfy the user's request. Such a service may consist of a single service or several services, which might be combined. The Service Broker is connected to an IMS and performs AAA through the IMS based on this request. After selecting the services, the application level requirements are derived. These requirements are signaled on to the Mediator.
- Cross-Layer API: The API needed to exchange the information (e.g. applications' requirements, offered functionalities at network layer) between service and network layer. A mapping of requirements is done before 'a best solution' is evaluated e.g. through a cost function.
- 3PI: The Third-Party-Initiation (3PI) is an extension of the existing Third-Party-Call-Service (3PCS). Both have in common to allow an operator service create a multimedia session that connects two participants. The difference between 3PI and 3PCS is the initiation process. A third party triggers the communication setup in 3PCS (e.g. click2dial), whereas the caller initiates the connection setup in 3PI, what later invokes a 3PCS.
- SONATE: In order to manipulate, manage, and deliver the services, the SONATE framework [3] has been developed which consists of different components. For the scope of the text, only crucial components of the framework will be discussed. The components such as building blocks, a building block Repository, and a workflow engine. In our approach building blocks are the implemenation of the services whic are composable. Nevertheless, they do have some constraints and dependencies. The repository holds available building blocks. A local Repository will be updated every time any building block is updated, removed or appended within the local repository. Building blocks are independent but they can communicate with each other by exchanging information. Work flow engine is responsible to execute building blocks according to a given work flow

besides it forwards incoming data to a responsible building block, where data will be processed further. Building blocks describe themselves by holding information such as covered services, efficiency, Quality-of-service, and requirements plus constraints for the execution of a building block. This description of building blocks will be used in a service selection and composition process.

- TUN/TAP: Unlike other hardware interfaces TUN/TAP are software-only interfaces. A code could be attached to TUN/TAP interfaces to read and write data which pass through these interfaces. TUN-interface is used for accessing raw IP packets while, TAP-interface is used for accessing full ethernet frames. In our approach TUN has been seen as a component in an approach towards supporting a legacy application, as our framework (i.e. SONATE) does not have inheritance support for a legacy application. While using TUN one can capture raw IP packets, in our approach we will use these raw IP packets to append some additional functionality via SONATE framework. To add functionality, into raw IP packets, need additional information which will be sent in terms of requirements from an application to the SONATE framework. After functionality has been added it will be sent back to the TUN-interface to dispatch to a physical network (i.e. ethernet).
- Visualization: A visualization of all important steps of the demo workflow is presented graphically. It consists the user intent, service composition, requirements identification, offered network functionality, mediated functionality, connection, data path, etc.

III. G-Lab DEEP Demo description

This section presents the demo scenario applied to our first prototypical implementation. The demo consists of three scenarios demonstrating our cross-layer composition concept, which shows the cross-layer composition concept realized in the context of G-Lab DEEP using a voice communication service. The three scenarios are cascaded. A following scenario uses the setup of the previous scenario and is equipped with additional functionalities.

A. Scenario: Standard voice call

A first step of the scenario demonstrates a cross-layer composition using a voice call between two endpoints (MyMONSTER). The underlying network is not utilized and the call will be established successfully. Our scenario is divided into signaling and data transfer, which are presented in this order. The signaling starts at client side by formulating the client's action in an intent (e.g. 'make emergency call'), which is send to the Broker. In a first step the Broker located in the IMS transforms such an intent into a request and identifies general service classes of the required services for answering the user's intent. The Broker selects and composes available service instances out of a pool of multiple services into a composition workflow in a second step. The Third-Party-Initiation is one of the selected services. This service established a SIP session between two clients and establishes a communication channel, by determining connection specific parameters like ports, codec's, etc. Requirements are derived out of this workflow, which are signaled on to the Mediator to the network selection and composition component. Offered functionalities from the network and demanded requirements from the application layer are exchanged via cross-layer API, which is the basis for producing a workflow at the network layer. Such a workflow identifies the required network functionalities and is signaled to each node (running the SONATE functional composition framework) on the envisaged path between the two clients. The signaling phase ends and the data transfer is started through the initiating client by sending RTP packets over the new established network path. Packets are addressed to a TUN/TAP network interface. After terminating the call, the network path is terminated, too and the blocked network resources are freed.

Conclusion: An interworking of all components from different partners as well as a functional composition was shown.

B. Scenario: Utilized network

In a second scenario the network is fully utilized with traffic from other clients. A new connection establishment like the first scenario is not possible and fails.

Conclusion: An attacker component is used to emulate traffic in a network.

C. Scenario: Prioritization of an emergency call in a utilized network

In a third scenario the network is fully utilized with unclassified traffic, too, but an emergency call with a high prioritization is able to allocate resources successfully. The importance of the emergency call is stated by the prioritization requirement explicitly.

Conclusion: The functional composition framework differentiates between classes of data flows, which are distinguished by their prioritization.

D. Visualization

The scenarios described above will be visually during our demonstration by an appropriate GUI. The visualization presents decisions taken by the system, such as the initial intent statement, the transformed request, the service composition workflow, the derived requirements, and selected network functionalities.

Acknowledgment

The project G-Lab DEEP is funded by the Federal Ministry of Education and Research (BMBF).

References

[1] http://www.monster-the-client.org/, Fraunhofer FOKUS, 2009.

[2] N. Blum, I. Boldea, T. Magedanz, U. Staiger, H. Stein, 'A Service Broker providing Real-time Telecommunications Services for 3rd Party Services', Proc. of 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC), Seattle, July 2009, ISBN 978-0-7695-3726-9, DOI 10.1109/COMPSAC.2009.202

[3] Reuther, Bernd and Mueller, Paul, Future Internet Architecture - A Service Oriented Approach, Oldenbourg Verlag, Mnchen, 2008