Requirements Based Automatic Service Composition: A Demonstration

Dennis Schwerdel, Rahamatullah Khondoker, Eric MSP Veith, Bernd Reuther, Paul Mueller
Integrated Communication Systems Lab, University of Kaiserslautern, Germany
{schwerdel, khondoker, veith, reuther, pmueller}@informatik.uni-kl.de

I. Introduction

The current Internet architecture was designed decades ago. Back then the main goals of the architecture were stability, performance and of course its functionality. Current trends, e.g. mobile devices, cloud computing, energy efficiency pose new requirements that the current Internet architecture cannot fulfill. Rather than building new functionality on top of the Internet or adding more complexity to it by introducing even more conditional functionality the Future Internet should have a smarter way to deal with diverse requirements and environmental constraints. Future network architectures must be flexible both long-term and a short-term in order to evolve and adapt to changing application requirements and new transport technologies (fixed and/or mobile) with different capabilities, and should enable evolutionary changes of the network itself. Thereby *long-term flexibility* can be seen as the capability of a system to evolve with updated protocols and network capabilities. *Short-term flexibility* is understood as the capability of a system to adapt itself and react to network conditions and an application requirements [1].

The Service-Oriented Network Architecture [2] (SONATE) tries to handle these requirements and constraints by using concepts of modularity and service-orientation. As such, *services* are central elements of SONATE: A service reflects the *effects* of an activity rather than the algorithms and data structures that implement it. Thus, a service can be provided using different algorithms. A *building block* is a functional block that implements one distinct networking functionality. Examples for building blocks are retransmission mechanisms, data encryption algorithms, and monitoring functionality. Each building block usually has several effects like increasing the end-to-end delay or reducing the maximum payload size in addition to its main function. All the effects of a building block form the service which the building block provides. Since a service does not define the mechanism that is needed to provide it, several building blocks can provide the same service and are thus exchangeable. The services of multiple building blocks might be needed to provide the service which the application requested.

II. Selection & Composition

Building blocks implement a minimal networking functionality that can be used to build up so called *protocol graphs* with complex network functionality that is similar to the functionality of today's network stack. The process of combining multiple building blocks into a protocol graph is called *composition*.

To enable this combination, building blocks in SONATE have interaction points called *ports*. These ports can be connected to allow the building blocks to communicate. Two special building blocks represent the network and the application, so that the protocol graph can establish a communication between the application and the network. Since the network building blocks can communicate via the network, this also enables communication between applications.

Determining the service that a protocol graph provides is a complicated task since the combination of building blocks can add additional effects that none of the parts had, but it can also remove effects that one part already had. To calculate the combined service, a recursive approach is used. Each building block offers a description that describes the service that the building block provides on each of its ports. These descriptions can refer to the services that other building blocks provide on ports that are connected. Thus the building block description can express an MTU[1] reduction of 10 bytes (because a header of that size is being added) rather than having to give an absolute value.

Although SONATE tries to offer generic interfaces for building block interaction, still some restrictions on which building blocks can be combined arise. The most obvious restriction for port compatibility is that the data types used to communicate on these ports must be compatible. Another restriction are requirements that building blocks have for the service provided by connected ports. Since building blocks implement only minimal functionality, they require some effects which they do not provide themselves. For example, stream compression building blocks need the effects of order preservation and losslessness.

The two special building blocks application and network serve as placeholders for the application and the network, provide a service describing them, and contain their requirements and constraints. This way, an application developer or the system administrator can express restrictions and requirements on the combined service of the protocol graph.

The task of both finding and combining a set of building blocks to form a protocol graph that can fulfill all these restrictions and requirements is called *Selection and Composition (S&C)*. If multiple protocol graphs match the restriction and

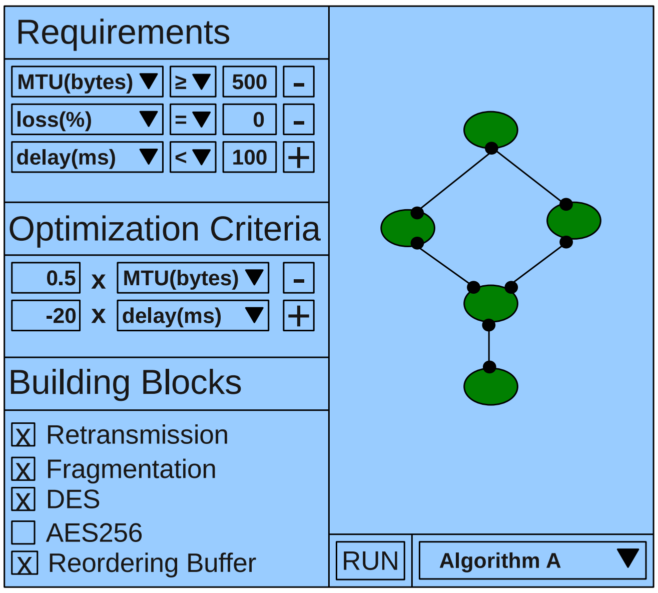
[1]Free space in the packet for payload

requirements an objective function provided by the application is used to rate the services of the protocol graphs and to select the best protocol graph to be instantiated and used for communication.

Finding the optimal solution to the selection and composition problem is not trivial and might be NP complete. For practical needs, the optimal solution is not needed and a protocol graph that fulfills all requirements and restrictions and has a good score in the objective function is enough. Several approaches for selection and composition are possible, ranging from manually defined protocol graphs to automatic composition at runtime.

III. The Demonstration

The demonstration presents automatic selection and composition of building blocks for creating a protocol graph based on the requirements provided by the application. The graphical user interface for the demonstration is shown in figure 1.


Requirements

MTU(bytes) ≥ 500 -

loss(%) = 0 -

delay(ms) < 100 +

Optimization Criteria

0.5 x MTU(bytes) -

-20 x delay(ms) +

Building Blocks

[✓ Retransmission]

[✓ Fragmentation]

[✓ DES]

AES256

[✓ Reordering Buffer]

Figure 1. The graphical user interface for an automatic service composition

The graphical user interface contains requirements, optimization criteria and available building blocks on the left side; the resulting protocol graph is displayed on the right side of the GUI.

In the requirements part the service requirements of the application can be configured. Each line is an inequation considering one effect of the service of the protocol graph. A valid protocol graph must fulfill all of these requirements. The objective function that is used to select the best valid protocol graph can be specified with optimization criteria. Each line consists of an effect and a weighting factor. The resulting objective function is the sum of all weighted effects configured in the optimization criteria. Building blocks that are available for selection can be configured on the lower left of the GUI. The selection and composition algorithm can only use building blocks that are selected but can also choose not to use them. Different algorithms can be used to create a protocol graph such as evolutionary algorithm [3].

Using the demonstrator the following results of selection and composition in SONATE can be shown:

- a protocol graph being created according to given requirements and optimization criteria
- changes in the requirements result in the creation of different protocol graphs to fulfill them
- an improved building block becoming available is automatically being used instead of the old one

IV. Future Work

The used selection and composition algorithm will be further improved and also other algorithms will be developed to improve the quality and speed of the selection and composition. The selection and composition shown in the demonstrator will be integrated with the SONATE framework that can run the resulting protocol graphs. An API for the application to express its requirements and objective function for the selection and composition and to use the resulting connection to communicate is being developed in cooperation with the partners in the special interest group “functional composition” of the German Lab project. Together with these partners also a language to describe building blocks that can be used in the process of selection and composition is being developed. The selection and composition algorithm improvement, the framework integration, the application API, and the building block description language will complete the chain from the application requirements to a connection that can be used for communication.

V. Acknowledgements

This work has been done as part of the German Lab research project, funded by the German Federal Ministry of Education and Research (BMBF).

References

[1] F. Group, “Fundamental limitations of current internet and the path to future internet,” *Draft Ver: 0.9*. [Online]. Available: http://www.europa.eu/

[2] P. Mueller, “Position paper: Towards a service-oriented internetworking architecture,” *Presented in the workshop on Future Internet Architecture: Internet Design Principles, Brussels*, May 2011.

[3] D. Schwerdel, B. Reuther, and P. Mueller, “On using evolutionary algorithms for solving the functional composition problem,” *In the proceedings of 10th Wuerzburg Workshop on IP: Joint ITG, ITC, and Euro-NF Workshop on ’Visions of Future Generation Networks’ EuroView 2010.*