GAPI: A G-Lab Application-to-Network Interface

G-Lab Special Interest Group Functional Composition
Group leader: Florian Liers (florian.liers@tu-ilmenau.de)

I. Introduction

Future Internet research yields an increasing number of frameworks for new protocol stacks. Each of them defines its own API in order to reduce the networking know-how an application needs for communicating with others. Thus, networking functionality, today done by the applications themselves, is pushed down below the API. This comprises in particular name-to-address resolution and the selection of protocols. Within the G-Lab project[1], we aim at developing a common API for demo applications suitable for the frameworks emerging from [1]–[3].

Our interface outlined in the next section aims at achieving the following goals: (1) separating application and networking concerns, and (2) being suitable for any current and future networking technology. The proposed API will be implemented for the aforementioned frameworks, and its functionality will be shown in common demo applications.

II. Application Interface

Our Future Internet provides three different interfaces to applications.

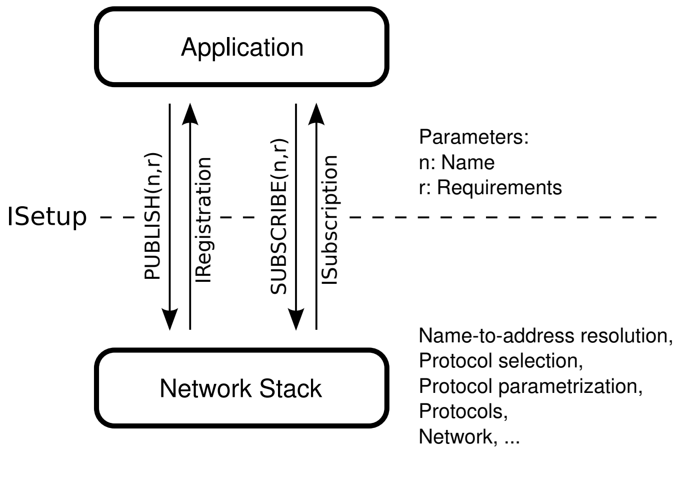

Figure 1. Overview of our G-Lab Interface ISetup

The first interface ISetup is used to start the interaction with a network stack. On the one hand, it provides methods for announcing services, which should be available for others (PUBLISH). On the other hand, others can use the interface to connect to these announced services (SUBSCRIBE). In both cases, the methods are used to get references to instances supporting one of the following interfaces.

The second interface IRegistration is used to change requirements for the announced service and to cancel it. In addition, it provides information about incoming subscriptions.

The third interface ISubscription is used to communicate with the service and the peer using it, respectively. It allows sending and receiving of data either in stream or datagram mode.

In the following, the focus will be on the ISetup interface, since it is the most important one for the interaction between network stacks and applications.

A. Methods

As mentioned before, the ISetup interface provides methods, which are used by an application to start interaction with a network stack. An overview of them is given in Figure 1. This interface provides the following two methods:

- IRegistration PUBLISH(Name, RequirementDescription): announces an application service to a network stack, by specifying a name and requirements of the service for the network. The network stack is made aware of the service and should provide other members of the network access to this service. The name is basically treated as a label. Its role will be discussed in more details in Section II-C. The announcement is represented by an instance supporting IRegistration (or a handle in a procedural programming language). The requirements have to be satisfied by the network for each peer, willing to use the service. Requirements are discussed in detail in Section II-D.
- ISubscription SUBSCRIBE(Name, RequirementDescription): establishes a communication association to a service, which was published before. The name parameter defines the service to talk with.

[1]http://www.german-lab.de/

The name must match with the name handed over to the PUBLISH method before. The requirements defines the characteristics of the communication relationship the network has to provide. In contrast to the requirements in the PUBLISH method, they are only valid for a single connection.

B. Call Sequence

A sequence of method calls for a typical client server scenario starts with the announcement of the service. The server application calls PUBLISH in order to make its service available under a name chosen by itself. This step is somehow comparable to BIND calls in today's IP networks, but binds a service to a name not an address and port. An association is started by a client application with SUBSCRIBE. The client has to use the same name as the server application. This step is comparable to today's CONNECT calls with a different name semantic and additional requirements the network has to fulfill. As a result, SUBSCRIBE will return a reference to a ISubscription object. At the server side, the network will inform the server about the new communication association. The server will get an ISubscription reference for the communication association via its IRegistration interface as well. Now, both sides have references to the communication association and can start to transfer data. On the client side, the reference can be used to send its request to the server, which uses its reference to receive it. The server will answer by sending data by using its ISubscription reference and the client will use its reference to receive it. Finally, either the client or the server can cancel its ISubscription. The cancellation will be propagated to its communication partner and invalidate its communication association, too.

C. Naming

The most important goal of the API is to hide network and network stack specific issues from the applications. In particular, that refers to the mapping from names to addresses and the selection of an address space (like using IPv4 or IPv6 addresses). Both parts should not be anticipated by applications. Consequently, applications must specify names in their own domain and based on their own name space. We propose a globally unique naming scheme based on *Uniform Resource Identifiers* (URIs). In order to allow different name spaces, we will use the scheme given in URIs to distinguish between them (like mailto:// or file://).

D. Requirements

An application states its requirements explicitly in order to specify the characteristics of a communication association. A requirement consists of an *Effect* linked by an *Operator* to an *Attribute*.

Effects describe the visible outcome of an operation of a building block or the network. Effect is a neutral term: Encryption is an effect as well as delay. Through further specification it becomes clear whether something is being provided or wished to be avoided. For example, an application can request a Packet Loss of 0, whereas a certain network connection could provide a Packet Loss of 5% average.

Attributes quantify or qualify effects and can be divided into two distinct parts: Inherent and qualitative. For inherent attributes, it must be clear to decide whether a certain property can fulfill a requirement or not (e.g., 200ms). An inherent attribute will normally be expressed by a number, however, functional requirements via boolean values are also possible, e.g. VirusScan == true. Qualitative attributes can be used for optimization or to describe attributes that cannot be objectively quantified, but where optimization can be applied by expressing the quality the effect provided by one algorithm in relation to the same effect provided by another. For example, the encryption quality of Rot-13 is inferior to the El Gamal Public Key algorithm.

Operators link Effects to Attributes. They are used for comparisons. Typical operators are equal, lower, etc.

Acknowledgment

This work was carried out within the research project G-Lab (funded by the German Ministry of Education and Research, BMBF) in the context of the SIG Functional Composition. It contains contributions from Florian Liers, Thomas Volkert (TU Ilmenau), Denis Martin, Helge Backhaus, Hans Wippel (KIT), Eric MSP Veith, Abbas Ali Siddiqui, and Rahamatullah Khondoker (University of Kaiserslautern). In addition, we would like to thank Christian Henke (TU Berlin) and Markus Bauer (Alcatel-Lucent) for valuable discussions and feedback.

References

[1] B. Reuther and D. Henrici, "A model for service-oriented communication systems", *Journal of Systems Architecture*, vol. 54, no. 6, pp. 594–606, 2008.

[2] F. Liers, T. Volkert, and A. Mitschele-Thiel, "Forwarding on Gates: A clean-slate Future Internet Approach within the G-Lab project", in *EuroView2009*, WÃrzburg, Jul. 2009.

[3] D. Martin, L. Völker, and M. Zitterbart, "A Flexible Framework for Future Internet Design, Assessment, and Operation", *Computer Networks*, vol. 55, no. 4, pp. 910–918, Mar. 2011.