Realization of Service-Orientation Paradigm in Network Architectures

Rahamatullah Khondoker[1], Abbas Siddiqui[2], Paul Müller[2] and Kpatcha Bayarou[1]

[1]*Fraunhofer SIT, Rheinstr. 75, Darmstadt, Germany*
[2]*Integrated Communication Systems, University of Kaiserslautern, Germany*

Received: 24 May, 2013; Accepted: 8 October, 2013

Abstract

The implementation of communication protocols in the current Internet architecture is tightly-coupled which hinders the evolution of the Internet. This article describes how the principles of Service Oriented Architecture (SOA) can be employed to develop a flexible network architecture. The prototype of the concept has been developed and demonstrated in the EuroView 2012 workshop. We showed that the SOA paradigm can be applied to networks by utilizing the concepts of self-contained building blocks, dynamic protocol graphs (PGs) and functional composition (FC) methods. We demonstrated that both short-term flexibility (i.e., networks are adapted based on application requirements) and long-term flexibility (i.e., networks can be evolved) can be achieved by using the architecture.

Keywords: SOA, Network Architecture, Selection & Composition, Template, AHP.

1 Introduction

The Internet today faces many challenges in terms of security, addressing, and mobility [9]. The problems originate from the architectural [23] issues of network functionality, their relationship, and the lack of design and evolution principles.

Journal of ICT, Vol. 3 & 4, 329–346.
doi: 10.13052/jicts2245-800X.134
© 2014 *River Publishers. All rights reserved.*

The Internet architecture is a layered system like TCP/IP stack based on OSI model. According to the OSI reference model, it should be possible to modify or even exchange the implementation of a layer without the need to adapt to the adjacent layers [19].

In today's practice, there are layer violations in the Internet, because of dependencies among protocols of different layers. For example, the addressing protocol in layer 3 (IPv6) requires an updated transport protocol (TCP) in layer 4. Thus, the evolution of the Internet "depends on rough consensus about technical proposals, and on running code" [2]. The Internet has become a complex system where it is hard to predict how the modification of one protocol affects the overall system. Many issues considered by the IETF IPv6 working group reflect this complexity [4].

As major changes in the Internet seem to be impossible, especially within a short time-frame, as a result new disruptive features are deployed in overlay networks. Overlays are usually designed only for a specific purpose such as filesharing, telephony, video-broadcasting, and are typically not open for arbitrary extensions or reuse. Hence, overlays are not a suitable alternative for a generic network infrastructure like the Internet.

Thus, there is a need of rethinking about network architectures [1]. This article presents the basic concepts of network architectures based on the service orientation paradigm from layered to layerless as shown in Fig. 1. The main goal is to develop a flexible network architecture which can be adapted to application requirements and network capabilities as well as to be able to integrate new functionalities easily.

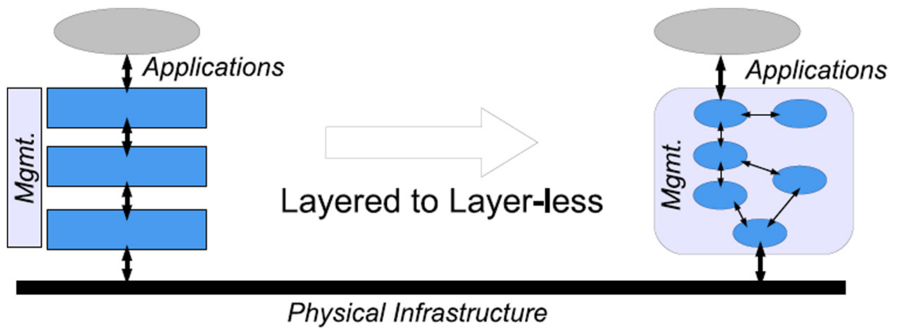

Figure 1 Layered to LayerLess Architecture

2 Service Oriented Network Architecture

Now the question is: how to apply the SOA paradigm to constitute a network architecture? The main element of SOA is a service. A service reflects the effects of an activity, i.e., a service represents a higher abstraction level since different algorithms may implement the same service. A building block is an implementation of a service. A Micro-protocol (MP) is an example of a building block such as a retransmission MP, a data encryption MP, and a Monitoring MP. Each building block can generate one or more effects, for instance, a retransmission BB has an effect of reliable data transmission, a data encryption BB has an effect of confidential data transmission. But there are also effects like increasing the end-to-end delay or reducing the maximum payload size. The interfaces of a building block should reflect the provided service and hide the implementation details. Building blocks should also use generic interfaces (i.e. as used in WSDL) so that interaction between building blocks does not require extra adapters.

A network architecture should be flexible in two ways. Firstly, networks should be able to adapt to specific customer or application needs and changing environmental conditions. Secondly, networks should be able to evolve, i.e. to add, change and even remove the functionality.

The flexibility is achieved by composing several (smaller) services to a more complex and customized service. In today's networks, complex protocols are organized in layers, building a static protocol stack, sometimes called Protocol Graph (PG) ([15]). Service oriented network architectures aim at supporting dynamic composition of services, i.e. dynamic PGs[1]. Without being dependent on a static PG it is easier to make use of new protocols (i.e., building blocks) and to reuse a functionality on different levels. Having dynamic PGs implies that there is no static placement of a functionality as defined by the layers of the OSI reference model. In this sense, such networks will be layerless so that compression or encryption can be used for application payload only or also for some protocol headers. Furthermore, it is not necessary that protocols are processed in a sequence, for example there might be different branches in a PG to handle different but related data types within a flow, e.g., signaling and media streaming. In order to enable dynamic PGs the interaction between the building blocks should not be defined by an executable code, but by the description which can be easily changed.

[1]corresponds to a workflow in SOA terminology

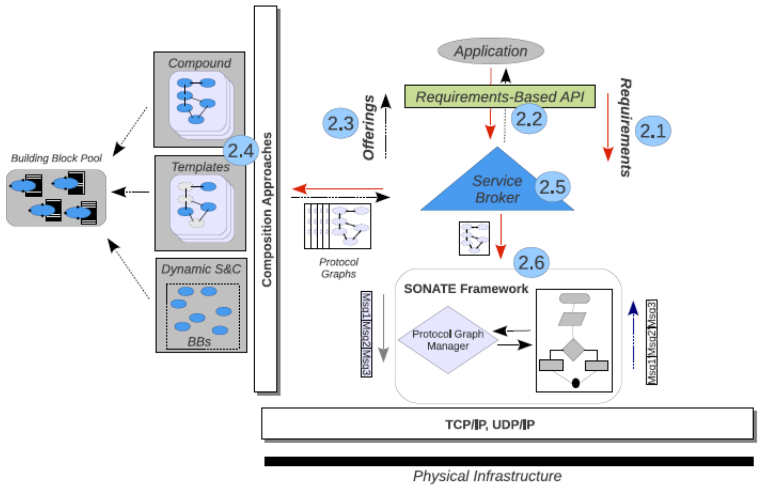

Figure 2 Proposed Network Architecture (the number in the blue circle indicates the subsection where the component is described in the paper)

Fig. 2 depicts the proposed approach. An application sends its requirements via the requirements-based API which is received by the service broker. Service broker forwards the requirements to the composition approaches where the PGs are generated. The composition process may generate more than one PGs so that the most suitable one can be selected by the selection process running inside the service broker. After selection of the most suitable PG, it will be directed to the SONATE framework for the execution and to initiate the communication between applications.

The following sub-sections describe the components of the architecture in a sequence: 2.1. application / user requirements, 2.2. requirements-based API, 2.3. network offerings, 2.4. a functional composition approach based on templates, 2.5. AHP-based service selection method in the Service Broker, and 2.6. the SONATE execution framework.

Subsection 2.7 briefly explains the prototype that has been demonstrated in the EuroView 2012 workshop. Standardization possibilities of the concept is discussed in Section 3. Section 4 concludes the paper.

2.1 Requirements

Requirements from a user or an application are represented by effects, operators, attributes, and weights as shown in Fig. 3 [13]. An effect is just the name, attributes quantify or qualify the effects and operators link effects

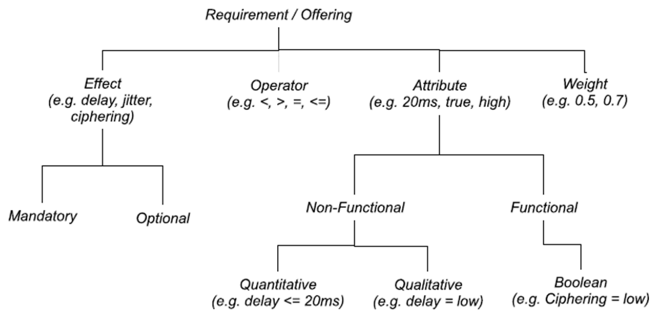
Requirement / Offering
Effect (e.g. delay, jitter, ciphering)
Operator (e.g. <, >, =, <=)
Attribute (e.g. 20ms, true, high)
Weight (e.g. 0.5, 0.7)
Mandatory
Optional
Non-Functional
Functional
Quantitative (e.g. delay <= 20ms)
Qualitative (e.g. delay = low)
Boolean (e.g. Ciphering = low)

Figure 3 Requirements/Offerings

to attributes, typical operators are <, >, =, <=, >=, etc. Attributes can be represented in different ways by giving an exact quantity (e.g. *delay < 20ms),* Boolean values (e.g. *Packet ordering = true)* or in a qualitative way (e.g. *delay = low),* though qualitative parameters need extra mapping or definition, it may vary with respect to the context. The weight of an effect expresses the application's or user's priority over other effects.

2.2 Requirements-Based API

The existing API was made with the assumption that the Internet supports a limited number of protocols and relies on applications to specify the exact protocol to use. The current API hinders the deployment of new transport protocols such as SCTP [24], DCCP [6] and new addressing schemes as an application is obliged to specify an address type (IPv4 [17] or IPv6 [3]). Stipulation of protocol by applications fosters tightly bound coupling, which forces the network stack to use that exact protocol rather than an improved version of a protocol or one that is more suitable with respect to the network conditions. In order to deploy a new protocol in the Internet architecture, it is not enough just to change the application but it also requires modifying the API or using another API so that a user can unveil its intention to use the newly deployed protocol.

An application needs to be modified if it wants to use TCP [18] or UDP [16]. Peer addressing and address-resolution are part of an application, which makes an application address type dependent. Different addressing types require different socket so that there is a difference between the IPv4 TCP socket structure and the IPv6 TCP socket structure and same is true for UDP.

The call of "setsockopt" is an example of another dependency where protocol specific options such as TCP_NODELAY can be turned on or off, as options are specific to a protocol so that it is a must for an application to know details about a protocol.

Currently, there are multiple existing APIs each developed for different transport protocols. If an application needs to switch a transport protocol, it is not enough just to adjust socket options or to change addressing family but it is also required to use a particular API given for a particular transport protocol.

Abstraction is used for hiding the complexity and to encourage the flexibility. In our approach, we propose an API by which an application can send its requirements in an abstract form to the underlying network such that applications do not need to rely on specific protocols; the process of selection would rather be handled by the network architecture. This triggers the requirement for the network architecture to be able to handle those abstract requirements from applications. Abstract requirements from applications also help to create an unified API so that a single API can be used for multiple transport protocols.

Current applications are tightly coupled with the given protocols, though they only care about whether its connectivity demands are fulfilled. A requirements based API [22] will alleviate the developer from choosing a protocol or even a protocol specification. Instead, requirements will be communicated to the underlying network architecture. Using these requirements, the explicit connection characteristics are requested for the new communication relationship. Requirements are specified in terms of effects / capabilities, an effect is a visible outcome of a functionality such as flow control functionality provides effect of transmission rate adaptation between two parties.

2.3 Network Offerings

To constitute a PG based on the application requirements, the offered services from the building blocks needs to be described so that the most suitable BB can be selected and composed. Moreover, the services of the PGs should also be described so that the best PG based on application requirements can be selected and used for communication. For describing these services, a communication service description language was developed. The language consists of a taxonomy of vocabularies and a grammar [13]. The details of the description language is beyond the scope of this paper.

2.4 Functional Composition

Functional Composition (FC) is the process of selecting and binding of the building blocks (BBs). The following sub-section describes the template-based functional composition approach which is developed and demonstrated in the EuroView 2012 workshop [8] under the umbrella of a German-based Future Internet (FI) research and experimentation project named G-Lab [7].

2.4.1 Template-based functional composition

In order to create a requirements-based PG out of given functionalities, it is necessary to define data and control flow of selected functionalities. The control flow in a PG is defined by the placement of functionalities while the data flow is defined by the connections among functionalities. The template-based composition is a partial-runtime approach where ordering of functionalities and their connections are defined at the design-time.

The basic idea of the approach is to split the functional composition process among different time-phases (i.e. design-time, deployment-time, and run-time) so that relatively inefficient activity in terms of time is performed at design time and potentially less time consuming activities are performed at run-time. In this case, time consuming activities are the selection and the placement of functionalities but not the actual building blocks (i.e. selection of encryption and compression functionalities but not their implementations/BBs) and placing them in an appropriate order (i.e., encryption is placed on the top of compression) in addition to connect them so that they can interact with each other. To utilize the less time critical epoch (e.g. design-time) and yet to provide flexibility, the template based composition approach utilizes the devised abstraction of place-holder instead of using actual functionality as shown in Fig. 4.

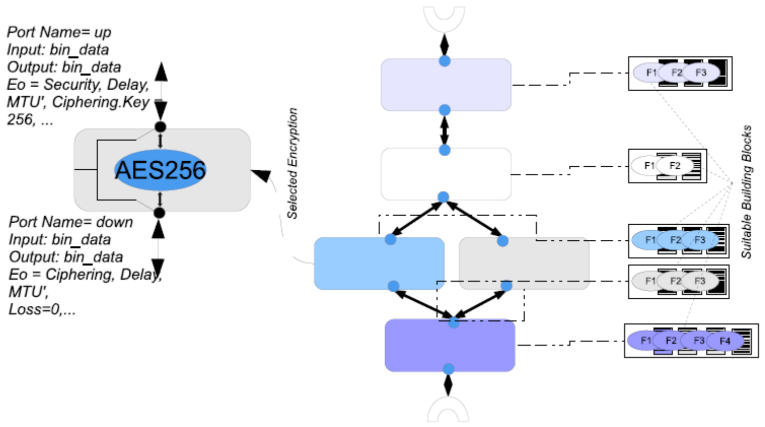

Figure 4 Template and Placeholder

The Place-holder is one of the major entities in this approach. The Place-holder provides named endpoints so called ports. The ports are well defined in terms of effects or capabilities that must be provided by a place-holder. Effects can be differentiated by the offered (i.e. provided by the port) and the required (i.e. accepted by the port) effects as shown in Fig. 4. An example effect of an encryption functionality is ciphering and this effect is covered by various mechanisms such as AES, DES, Blowfish, etc. so that it should be possible to change the mechanism without changing the effect. Aforementioned mechanisms can have many implementations and may differ in terms of defined ports (i.e. it implies same covered effects on those ports). Any implementation which also has same ports as described in a place-holder can be a suitable match to fill that place-holder.

code.1 Placeholders Example

```
<Placeholders>
        <PlaceHolder Name=“Encryption” ID=“1”>
            <ToggleEnable isEnable=“true”/>
            <Port PortID=“up”>
                <OfferedEffect Effect=“Ciphering.key” Operator=“=” ↩
                      Attribute=“256”/>
                <OfferedEffect Effect=“Delay” Operator=“=” Attributed ↩
                      =“1ms”/>
            </Port>
            <Port PortID=“down”>
                <OfferedEffect Effect=“Ciphering” Operator=“=” ↩
                      Attribute=“true”X/Of f eredEffect >
                <OfferedEffect Effect=“Loss” Operator=“=” Attributes↩
                      =“0”> </OfferedEffect >
            </Port>
            </PlaceHolder>
</Placeholders>
```

The Place-holder also contains ports for general functionalities such as management, administration and monitoring. But those ports are active only when a selected BB also provides that data, hence these ports are optional and not considered in the BB selection process.

Template Description Language: The template description is split in four main parts so called Domains, Placeholders, Connections and CoveredEffects. Where, Domains section describes the types of domains that are covered by a template, examples of domains are telephony, video streaming, file transmission, etc. The Placeholders section describes the covered functionalities in a template, which is further sub-divided into individual placeholder and its ports. Connections section of the language deals with the ordering and the connections of the place-holders.

Example of an encryption placeholder is shown in the code.1, which provides two offered effects (i.e. Ciphering.key and Delay) through the port "up" and two offered effects (i.e. Ciphering and Loss) through the port "down". In this example no required effects are described as this functionality accepts binary data as an input and gives binary data as an output.

Template Selection: For the selection of a template two simple matching algorithms have been implemented. After receiving the application requirements, the domain will be extracted and it will be checked against stored domain policies at the system. The domain policies are required in order to have the special constraints/requirements which are not provided by an application such as compression in file transfer (i.e. an application does not care if data will be compressed or not as long as a file is efficiently transferred to the communicating partner). After retrieving the domain policies, it will be merged with the application requirements to find a matching template.

The first matching process goes through all of the merged requirements and check against covered effects in a template. This selection process works separately for a single template. The advantage of this approach is that multiple processes can run in parallel without being dependent on each other.

In the second algorithm, the selection process reads first the requirement and checks against the available templates at the system and separates the one which fulfills the requirement. After that it reads the second requirement out of the requirements list and examines against sorted out templates from the previous step. And this process goes on until one or more than one templates have been sorted out which fulfill the given requirements. This method is useful for the devices with limited cache as memory switching in threading can be costly. When more than one templates satisfy the requirements then multi-criteria decision analysis algorithms can be used to select the most optimal template.

Selection of BBs for Filling a Template: There can be more than one BBs which fulfill the application requirements and network and other constraints so called "suitable" building blocks. If an optimal choice has to be made then all possible workflows are generated. Based on the qualitative parameters, the best PG is selected. The Analytic Hierarchy Process (AHP) is used to select the best PG which is further described in the following section.

2.5 Service Broker

The Service Broker is responsible for selecting protocol graphs (PGs) with respect to the application requirements. Selecting the best PG using a single selection criterion is trivial. For example, if there are two PGs where one offers 100ms end-to-end delay and another offers 200ms, then we should obviously select the one with less delay.

However, the communication services provided by PGs have multiple selection criteria such as delay, throughput, loss ratio, jitter and cost. That is why, selecting the best PG is a Multi-Criteria Decision Making problem (MCDM). For solving such a problem, several Multi-Criteria Decision Analysis (MCDA) approaches are used in managerial science like Analytic Hierarchy Process (AHP), ELECTREIII, Evamix, Multiple Attribute Utility Theory (MAUT), Multi - Objective - Programming (MOP), Goal Programming (GP), NAIADE and Regime [5].

We used AHP to select the best service for two reasons, firstly, it uses an absolute scale to derive priorities that also belong to the relative absolute scale (like probabilities) that can be combined like the real number system. Secondly, there is a way to check the consistency of the evaluation measures.

2.5.1 Adaptation of Analytic Hierarchy Process (AHP) for service selection

The Analytic Hierarchy Process (AHP) needs to be adapted for selecting the best communication service automatically.

AHP is a process designed for assisting human decision making which is used in many application areas like social, personal, education, manufacturing, political, engineering, industry and government [20]. Basically, AHP is used for determining priorities of different alternatives. The details of the AHP process is beyond the scope of this text.

To use AHP in PG selection, the following steps are performed

1. Define the goal and the selection criteria for achieving the goal
2. Priority assignment of the selection criteria as an application requirement
3. Priority assignment of the criteria for the offered services

The first step is to define the goal, which is to select the best communication service, and the selection criteria to achieve that goal. The selection criteria are actually a set of required effects. Examples of the selection criteria are delay, throughput, loss rate, jitter, MTU and cost. Both functional and non-functional criteria can be selected.

After determining the selection criteria, the next step is to assign pairwise priority between the selection criteria. One of the reasons of pairwise priority assignment is that, it is easier for a person to take two criteria and to assign a priority one over the other. It is initially difficult for a new application developer to assign a pairwise priority. But, the efficiency of the priority assignment process can be improved with the experience of the application developer.

The third step of the process is to assign pairwise priority between the offered services based on those selection criteria. However, as pairwise priority assignment is a time-consuming task, and as offerings are decoupled from the applications, the pairwise priority assignment of the offered services based on those selection criteria needs to automated.

This requires a mapping mechanism to map the measured/calculated values of the offered services to the pairwise priority assignment scale which will be discussed in the next section.

The priority vector coming from the application side is then multiplied by the priority vector from the offering side. The result is then called the overall priority vector. The service with the highest priority value in the overall priority vector is the best service.

2.5.2 Automated priority assignment for the offerings

Different PGs can have different effects. The value (or attribute) of these effects can be assigned beforehand based on benchmarks or can be obtained dynamically by using a sensing software. Whichever way the attributes are obtained, the offered effects need to be automatically prioritized as the offerings are decoupled/hidden from the application. Therefore, an automatic The mapping should have certain properties. First, the mapping must be generic, i.e. not specific to effects or units of measured values. Second, the mapping must be monotonic.

An approach for mapping has been proposed which uses a monotonic interpolation/extrapolation scheme [12] as shown in Fig. 5. In this case, the application requirements provide value points for interpolation/extrapolation (must be monotonic) of measured values to the priority scale. A monotonic interpolation/extrapolation of these points is used to define a mapping. In addition, the specific measured values of the offerings are then mapped to these priorities. Assuming that $f()$ is a function used to define a mapping. As

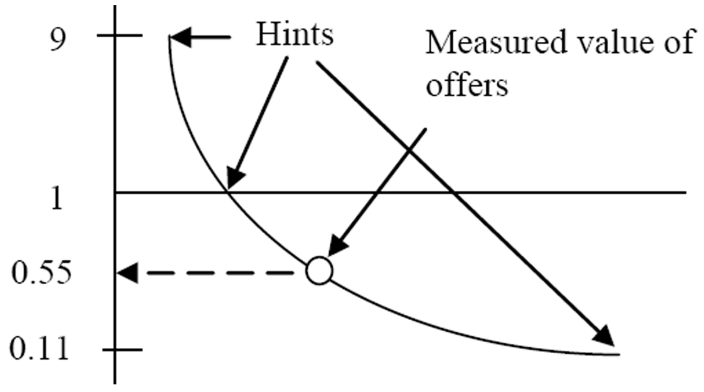

Figure 5 Mapping mechanism

an example, considering interpolation, the requirements must contain at least the following two points

- x_0, *where* $f(x_0) = 0.11$
- x_n, *where* $f(x_n) = 9$

If there are measurement values, y, not within the interval [x_0, x_n], we can extrapolate

- *if* $y < x_0$, *then* $f(y) = 0.11$
- *if* $y > x_n$, *then* $f(y) = 9$

To use inter-/extrapolation, an application developer must specify two points but can have as many parameters as he wants to be more precise.

The aforementioned mapping mechanism is used to assign a priority of one service over another for every selection criteria (effect).

2.6 SONATE Execution Framework

The framework for service oriented network architecture (SONATE) executes the selected PG. The flow consists of operations to invoke the specific BB and to pass the corresponding values as input parameters to the connected BBs.

The PG is described in an Extensible Markup Language (XML). The SONATE framework has the ability to process the XML based PG. The processing of PG involves retrieval of BBs from the repository, connection of BBs as specified in the PG and execution of BBs in the given order.

The SONATE framework has been implemented using the JAVA programming language. The "Port concept" [21] has been used for the interaction among building blocks. A crucial point in maintaining flexibility is loose coupling. The building blocks that are tied to specific implementation details of other building blocks, reduce the flexibility to freely combine building blocks. To enforce loose coupling, the BB interaction model hides all BB's implementation details from each other.

In the runtime composition approach, BB instances need to communicate with each others to make a PG, the concept of named communication endpoints called "ports" is introduced. Building blocks can use the ports to distinguish different kinds of communication partners or different operation modes.

2.7 Prototype Demonstrated

We demonstrated the concept of service oriented network architectures in the EuroView 2012 workshop [8]. More specifically, we showed how networks react to user's requirements, network/administrator constraints by dynamically selecting and composing a customized protocol graph.

In the demonstration, the Firefox browser retrieves different types of data (Voice, Image) with different user requirements, network constraints and domain policies.

Two servers with different network conditions such as bandwidth and jitter were configured. We have extended a Firefox plugin, to interconnect Firefox with the Requirement-based API to communicate with the components in the SONATE Framework. Firefox sends the requirements via API to the service broker, which is responsible for selecting the best PG at the end. The service broker forwards requirements further to the template based composition engine, which creates and returns multiple PGs so that the most suitable is selected by the service broker using AHP. We encoded all application requirements, network constraints and administrative policies in the URL so that they can be uniquely identified. The selected PGs are displayed by a browser-based visualizer.

We showed that both long and short term flexibility are achieved by the proposed architecture.

Short Term Flexibility: In this scenario, different image compression mechanisms have been provided which are selected based on the application requirements and the network conditions. Application requirements provide trade-off between expected quality and transmission speed. If an application requires a better quality then no image compression or a very low compression mechanism is deployed. But, if transmission speed is the priority of a user then the best compression mechanism (i.e., high compression ratio) is selected. The scenario is complicated by the given administrative policies with respect to network conditions. If the given bandwidth is less than or equal to certain threshold like 1 Mbps then compression mechanism is deployed, otherwise, in the case of faster networks, no compression mechanism is deployed in the PG.

Long Term Flexibility: In this scenario, a better implementation of an encryption mechanism has been deployed in the system which provides better security in terms of key-strength. As soon as a user demands for higher security, the newly deployed mechanism is used in the automatically generated PG. The scenario shows how quickly a newly introduced functionality can be deployed in the presented architecture.

3 Standardization Candidates

The components of the architecture including communication service description language, requirement-based API, and template-based functional composition can be seen as the potential candidates for standardization. These items can be included in the ITU-T study group 13 (ITU-T SG 13) where the potential future networks technologies like cloud computing, mobile, and next-generation networks are discussed [11]. Moreover, the API can be examined within the Name-Based Sockets Architecture community of IETF [10]. In addition, the language can be considered in the Operations and Management Area of IETF [14].

4 Conclusion and Future Work

The implementation of communication protocols in the current Internet architecture is not loosely-coupled which hinders the evolution of the Internet. This article describes how the principles of Service Oriented Architecture (SOA) can be employed to develop a flexible network architecture. The SOA paradigm can be applied to networks by utilizing the concepts of self-contained building blocks, dynamic protocol graphs (PGs) and functional composition (FC) methods. It is shown that both short-term flexibility (i.e., networks are adapted based on application requirements) and long-term flexibility (i.e., networks can be evolved) can be achieved by using the architecture.

Heterogeneity, availability of diverse functionalities in different network elements, is an issue which needs to be tackled in the future. The proposed approach expects to have a controlled environment where every node has the same functionalities available. However, existing mechanisms like negotiation can be used to deal with heterogeneity. Moreover, new functionalities can be deployed from a trusted domain.

References

1. R. Braden, D. D. Clark, S. Shenker, and J. Wroclawski. Developing a next-generation internet architecture. 2000.
2. B. Carpenter. Architectural Principles of the Internet. RFC 1958 (Informational), June 1996. Updated by RFC 3439.
3. S. Deering and R. Hinden. Internet protocol, version 6 (ipv6). *RFC2460*, Dec 1998.
4. Ralph Droms and Jari Arkko. Ipv6 status pages, March 2008.
5. Matthias Ehrgott and Xavier Gandibleux. Multiple criteria optimization state of the art annotated bibliographic surveys. *Kluwer Academic Publishers*, 2003.
6. S. Floyd and E. Kohler. Profile for datagram congestion control protocol (dccp) congestion control id 2: Tcp-like congestion control. *RFC4341*, March 2006.
7. German lab (g-lab). http://www.german-lab.de/. Online; accessed 11-October-2012.
8. Daniel Günther, Dennis Schwerdel, Abbas Siddiqui, Rahamatullah Khondoker, Bernd Reuther, and Paul Müller. Selecting and composing requirement aware protocol graphs with sonate. *In 12th Würzburg Workshop on IP: Joint ITG, ITC, and EuroNF Workshop on 'Visions of Future Generation Networks' EuroView 2012*, July 2012.
9. Mark Handley. Why the internet only just works. *BT Technology Journal*, 24(3), 2006.
10. Name-based sockets architecture draft-ubillos-name-based-sockets-03. http://tools.ietf.org/html/draft-ubillos-name-based-sockets-03. Online; accessed 05-May-2013.
11. Itu-t sg13: Future networks including cloud computing, mobile and next-generation networks. http://www.itu.int/en/ITU-T/studygroups/2013–2016/13/Pages/ default.aspx. Online; accessed 05-May-2013.
12. Rahamatullah Khondoker, Bernd Reuther, Dennis Schwerdel, Abbas Siddiqui, and Paul Müller. Describing and selecting communication services in a service oriented network architecture. *In the proceedings of the 2010 ITU-T Kleidoscope event, Beyond the Internet? Innovations for future networks and services, Pune, India*, December 2010.

13. Rahamatullah Khondoker, Eric MSP Veith, and Paul Müller. A description language for communication services of future network architectures. *In Proceedings of the 2011 International Conference on the Network of the Future*, pages 69–76, 2011.
14. Operations and management area. https://datatracker.ietf.org/wg/.Online; accessed 05-May-2013.
15. Sean W. O'Malley and Larry L. Peterson. A dynamic network architecture. *ACM Transactions on Computer Systems*, 10:110–143, 1992.
16. J. Postel. User datagram protocol. *RFC768*, Aug 1980.
17. Jon Postel. Darpa internet program. *RFC791*, Sept 1981.
18. Jon Postel. Transmission control protocol. *RFC793*, Sept 1981.
19. Recommendation. Recommendation x.200 (07/94), x.200 information technology, open systems interconnection, basic reference model the basic model. 1994.
20. Thomas L. Saaty. Decision making with the analytic hierarchy process. *Int. J. Services Sciences*, 1(1):83–98, 2008.
21. Dennis Schwerdel, Danile Günther, and Rahamatullah Khondoker. A building block interaction model for flexible future internet architectures. *7th EURO-NF CONFERENCE ON NEXT GENERATION INTERNET, 2011*.
22. Abbas Siddiqui and Paul Müller. A requirement-based socket api for a transition to future internet architectures. *Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS-2012)*, 2012.
23. IEEE Std. Ieee std., 1471-2000 ieee recommended practice for architectural description of software-intensive systems-description. 2000.
24. R. Stewart, Q. Xie, and et al. Stream control transmission protocol. *RFC2960*, Oct 2000.

Biographies

Abbas Siddiqui is a Ph.D. candidate with the topic of Flexibility in the Network Architectures, where he proposed a partial-runtime composition approach to create customized network-stacks. The name of his approach is “Template-Based Composition”. After completion of his master in “Electrical & Communication Engineering” with focus on Mobile & Internet Engineering from University of Kassel, Germany, he started to work as a software engineer in the industry, where he gathered several years of experience as a software developer before, leaving the industry for the sake of research. His current research revolves around Service Architectures, Smart Living, e-Health, and Sensors Technology.

Paul Müller is a Computer Science (CS) professor and director of the regional computing center at the University of Kaiserslautern in Germany. His current research interests are mainly focused on distributed systems, Future Internet (FI), and service-oriented architectures. His research group Integrated Communications Systems(ICSY) is aiming at the development of services to implement integrated communication within heterogeneous environments especially in the context of the emerging discussion about FI. This is achieved by using concepts from service-oriented architectures (SOA), Grid technology, and communication middleware within a variety of application scenarios ranging from personal communication (multimedia) to ubiquitous computing.

Dr. Kpatcha Bayarou received his Diploma in electrical engineering/automation engineering in 1989, a Diploma in computer science in 1997, and his Doctoral degree in computer science in 2001, all from the University of Bremen in Germany. He joined the Fraunhofer Institute for Secure Information Technology (Fraunhofer SIT) in 2001. He is the head of the "Mobile Networks" department that focuses on Cyber Physical Systems and Future Internet including vehicular communication. Dr. Bayarou managed several EU and nationally funded projects and published several conference papers related to security engineering of mobile communication systems, mobile network technology, and NGN (Next Generation Networks).

Since 2010, Rahamatullah Khondoker has been working towards his PhD degree on "Description and Selection of Communication Services for Service Oriented Network Architectures (SONATE)" at the University of Kaiserslautern in Germany. He was awarded from Ericsson, Germany in the year 2008 and from the FIA Research Roadmap group in October 2011. Currently, he is affiliated with the Fraunhofer SIT located in Darmstadt, Germany. He worked with the DFG project (PoSSuM), BMBF projects (G-Lab, G-Lab DEEP, Future-IN), and EU projects (PROMISE, EuroNF). Currently, he is focusing on the security of Future Internet Architectures, Software-Defined Networking (SDN), and Network Function Virtualization (NFV).