Evaluating a Future Internet Cross-Layer Composition Prototype

Julius Mueller[1], Abbas Siddiqui[2], Martin Becke[3], Michael Kleis[4]

[1] Technical University Berlin, Germany
julius.mueller@tu-berlin.de
[2] Technical University of Kaiserslautern, Germany
siddiqui@informatik.uni-kl.de
[3] University of Duisburg-Essen, Germany
martin.becke@uni-due.de
[4] Fraunhofer FOKUS, Germany
michael.kleis@fokus.fraunhofer.de

Abstract. The Internet was initially designed to enable information exchange between research institutes using the best-effort packet transport paradigm. Since then, more and more areas of our daily live are reached by the Internet including business critical areas, causes through its big success, fast acceptance and emerged possibilities. A side effect of the underlying best-effort transport principle is that for instance Quality-of-Service and security sensitive services are difficult to realize. As a consequence one focus of Future Internet (FI) research is on re-design of the current Internet architecure. The so-called clean slate approach addressed today's demands and requirements that could not be solved by the current Internet architecture. In this paper we present a clean slate cross-layer architecture approach in which functionalities at the network layer are supported by the service layer, in order to optimize underlying connections. In the second practical part we validate the presented architecture based on a prototype implementation. An evaluation section discusses measurements done with the prototype. An outlook on our future work concludes the paper.

Key words: Cross-Layer, Future Internet, Service Composition, Functional Composition

1 Introduction

The Internet was initially designed at CERN to enable information exchange between research institutes in a distributed manner. The protocols and the network architecture were once designed in the way to enable http and mail data transfer using best effort packet transport paradigm. The big success of the Internet reaches other sectors fast and growth steadily until these days and even further on. Initially used in the research domain, its acceptance covers nowadays areas like traveling, finance, health, community, government, education and the business sector with mobile as well as in fixed connections. These new sectors

brought new requirements and demands like Quality-of-Service (QoS), security, reliability, etc, which were not included in the original early design.

The origins of the Internet reach back to the 1960s with both private and United States military research into robust, fault-tolerant, and distributed computer network, on which several optimizations have been applied to the initial architecture. This evolutionary way of modifying the Internet from its early stages to the Internet we are using now resulted in partial network security and QoS solutions not supported by every network operator and far from full coverage.

The term Future Internet (FI) is linked to four main research directions. The network of the future virtualizes the physical network and provides virtual network overlays within the physical network. The Internet of content re-arranges the network organization through locator/identifier splitting. The Internet of things is characterized by machine-to-machine (M2M) communication. The Internet of services includes Functional Composition (FC), in which the functionalities of the classic layered ISO/OSI network stack are decoupled to be exposed and combined in an optimized composition for individual services. [1]

On key idea is to re-design the Internet from scratch based on todays requirements, demands and use cases, which are not possible or difficult to realize within the current Internet. This concept is known as clean slate approach of the Future Internet, which takes the revolutionary way and effects all layers of the ISO/OSI reference network stack. One aspect of FC includes cross-layer composition, in which the strict transparency of service and network layer is relaxed, by making the services network aware and enabling service awareness to networks at the same time. Using this idea, services are able to state requirements to the network and the network is able to provide feedback using e.g. a subscription/notification mechanism regarding individual connections.

This paper presents a clean slate approach using a cross-layer design approach in which functionalities of the network layer are supported directly by the service layer. A service composition orchestrates services to serve the demands of intent statements send by the User Equipment (UE). Security policies and network monitoring secure the individual connection from an operator point of view. A FC framework combines network functionalities demanded by the service layer and establishes a data path between to communicating devices.

A theoretical design part presents a clean slate cross-layer design approach in which functionalities of the network layer are addressed directly by the service layer, in order to optimize underlying connection. The second practical part validates our presented architecture with a prototypical implementation. Finally an evaluation discusses measurements. An outlook on our future work concludes the paper.

2 Design of a QoS aware Cross-Layer Service Delivery Concept

This section presents the design of our generalized G-Lab DEEP [15] architecture for cross-layer functional composition service delivery architecture [2, 3, 4] depicted in figure 1. The key components of this architecture are presented and their main tasks are discussed. Finally the interfaces and protocols are highlighted.

The figure 1 includes three types of different interfaces. The dashed lines are used to exchange control information, the solid lines indicates data paths and the large arrow connecting UE and the broker depicts the intent statement that triggers the composition.

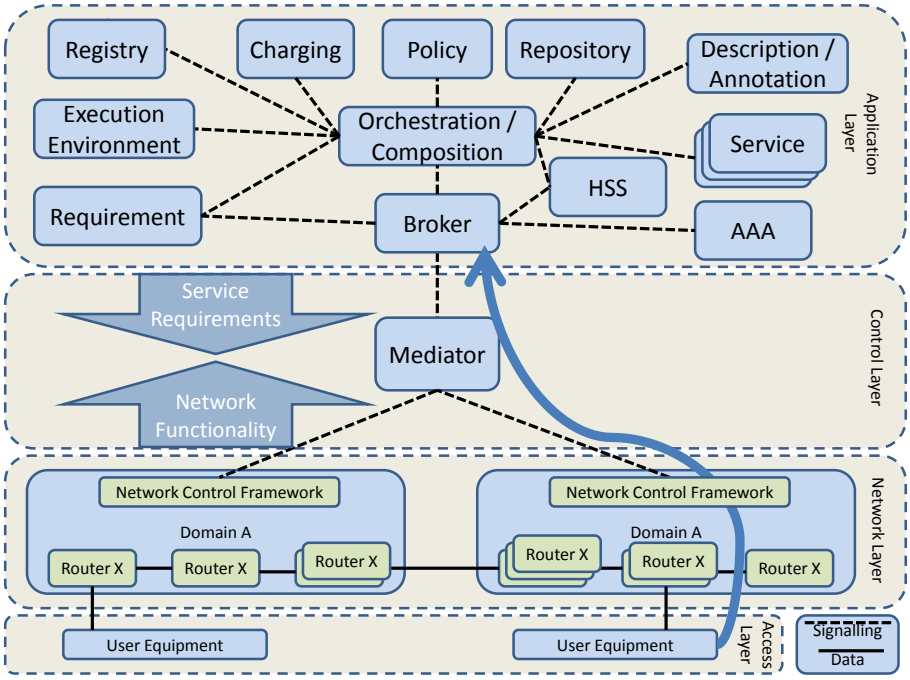

Fig. 1. General Cross-Layer Architecture

The architecture is divided into four main layers namely the application, control, network and access layer from the top to the bottom. The application layer manages services to expose functionality to users. The functionality resides inside applications which are maybe hosted in distributed environment. A service is either single/atomic or consists out of several services as a complex combined service.

Available services from (3rd party) service providers are registered in the registry, which is a database containing references to all registered services. Each service has an individual description to ensure the valid access, which is also mapped by the Registry to the specific service. Services may be hosted in the service provider domain in a repository and managed in a service execution environment.

The process chain to execute a user demanded service and is initialized through an intent statement by the User Equipment (UE). Such an intent consist of a service request and parameters required by this particular service and is stated from the UE towards the broker component.

The broker [5] transforms such an intent into a request by deriving abstract services out of the request and determining an abstract service workflow to solve the requested functionality. Based on the created service workflow, security related functionalities like Authentication, Authorization and Accounting (AAA) of the user request are validated as first step before proceeding with the next steps.

The selection of services to assemble the workflow is done through the orchestration and composition engine. Such an orchestration or composition engine selects services based on predefined policies given through the network operator or service provider. A centralized database provides user profiles, which are used as base to derive policy decisions on in order to grant or revoke access to specific services.

The process of selecting services for a workflow also identifies service level requirements on the underlying network. These service requirements are related to bandwidth, prioritization, etc.

As composition is executed at two different layers, service and network layer, there can be scenarios where exact requested functionality can not be provided or conflict may occur among offered functionality at service layer and requested functionality at network layer. Mediator can help to negotiate between two layers to resolve the conflict. Mediator is a part and main component of a control layer.

To resolve a conflict the mediator [6] needs various information (as shown in fig 2) such as requirements, offerings, policies and dependencies. Requested and offered functionality knowledge helps mediation to decide for appropriate functionality with respect to presented constraints, dependencies and policies. Dependencies provide knowledge about existing conflict between services, while policies defines the rules to resolve those conflicts. There are various dependencies possible where exclusion or inclusion of a functionality makes more sense such as it is not reasonable to have compression of encrypted data or, to use answering-machine-functionality and call-forwarding-functionality at the same time. Examples of policies could be where certain functionality not allowed to use in particular environment because of hardware, software or network constraints.

Network functional composition will be performed at network layer where functionality of network stack is decomposed into building blocks which could be composed dynamically at request. Composition can be performed at the run-

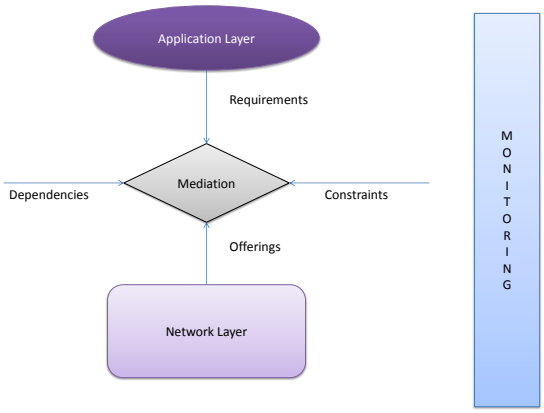

Fig. 2. Mediation between Network and Service Layer

time or at the design-time, both kinds of composition have its own pros and cons. Design-time composition is faster to use as it has been composed beforehand, but it can not cope with change of constraints and/or parameters of network and doesn't have ability to adapt to the environment. On the other hand run-time composition is slower as it requires time to compose but it can adapt to dynamically changing requirements and constraints. Nevertheless as being part of cross-layer composition, run-time composition is an appropriate method unless some pre-composed template(s) can be used.

The UE is located in the access layer. The UE is able to state an intent statement using a RESTful protocol to an operator predefined first point of contact.

3 Use Cases and Demo Scenarios for a Cross-Layer Concept

The demonstrator addresses the following Use Cases in the Cross-Layer architecture using Functional Composition to enable traffic differentiation in a voice call scenario. A voice call is established under different circumstances within this scenario. Demonstrated circumstances for a call are:

1. Successful normal voice call under normal network conditions
2. Attacker overloads the network (i.e. Denial of service)
3. Normal call fails due to the utilized network
4. Successful emergency call in an overloaded network, which uses prioritization to ensure QoS

A main benefit of the new network architectural approach should be determined by the ability of assuring needed services on demand, in best case in a dynamic way. The base functionality consists in assuring basic services, such as

voice calls for example. In addition to it, further challenges are in the focus of this approach such as security, high availability and additional feature sets. In order to describe the advantages of this concept and architecture, four scenarios are considered in figure 3.

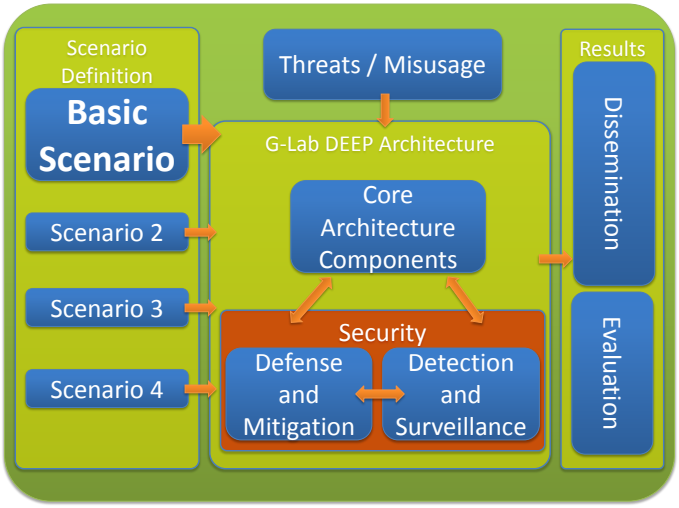

Fig. 3. G-Lab DEEP Demo and Architecture

Scenario I allows to verify the basic functionalities and shows the base lines. For the special case of SIP Calls, the data transfer between the caller and the callee can be divided into two steps, first for signaling purposes and second for data transfer. The caller does not have to be aware of this differentiations, it just sends a request to the application interface which forwards it to the broker. Here the first main task will be done; the intent of the caller will be transformed in a more technical request, which allows the Broker to identify general service classes. Selecting the available service instances out of a common service pool is the second step inside the Broker. As a result of this procedure, the user intent is turned over into a composition workflow. In this Scenario a wrapper for a Third Party Service is part of this workflow. It represents an interface to the well-known functionality of SIP, which itself requires a composition workflow of an UDP-like transport network service.

This and other requirements, such as codec for example, are derived out of the workflow and will be signaled in a third step to the Mediator. The mediator creates a workflow for the network, which is a kind of a network requirements catalog. It is used to transfer the requirements to the involved instances of the network to make it able to support the workflow by utilizing the functional composition framework SONATE [7]. All this steps performed for the signaling steps are then repeated for the data transfer with another workflow. After finishing the

call, all network connections are terminated and all blocked network resources are freed.

In the end we described here the common well-known business case of a VoIP call in a new architecture. Keep in mind that the purpose of this work is not to establish a new or better VoIP service but to build up an architecture which is able to support this service. For this purpose, the scenario II came into consideration.

In this case, an attacker is part of the scenario. The attacker is used to initiate denial of service attacks by sending SIP requests from distributed malicious clients in order to drain the available bandwidth between caller and Callee or to overbook the callee. For the application, it is easy to be aware of good or bad traffic. For the network, it is more difficult. In fact, the application has to handle every message what could increase the non-availability of the callee and/or decrease the call quality. An Emergency call, for example has to be guaranteed.

The last two scenarios are made to demonstrate the benefits of the new architecture. For both of them, the functional composition framework is extended with a quality of service functionality which is able to assure priority. In scenario III we have the same behavior like in scenario I, with exact the same workflow. In this case the new functional block does not belong to the workflow, also not for an emergency call, because it is not needed, and more, it would decrease the network performance.

In contradiction to scenario III, in case of scenario IV, a congestion caused on an attack is detected and signaled from the application to the mediator. In this case the mediator changes the workflow for the functional composition framework. This makes it possible to show the dynamic behavior of the cross-layer composition by initiating an emergency call. An emergency call intent has high prioritization requirements. It is transferred as workflow for this connection to the network and the resources are allocated for this kind of call. The functional composition framework differentiates on demand between classes of data flows, which are distinguished by their prioritization.

These are in the end simple scenarios which easily show how it could be possible to use a Composed Framework for Knowledge transfer to support Requested Security in case of defense and Mitigation.

4 Implementation of a Prototypical Demonstrator

After presenting the design of a Future Internet Cross-Layer Composition concept, this section outlines the demonstrator setup and presents the technical details of our implementation. This prototypical implementation depicted in figure 4 focuses on a voice call scenario, in which cross-layer composition creates a QoS aware data connection differentiating the divers requirements between a normal and an emergency call.

The service composition framework [7] provides a mechanism for composing and orchestrating registered services in order to achieve a more complex and

personalized functionality. A composed service is defined through a workflow of services invocations. The workflow consists of operations to invoke the specific services (which can be also another composed service) and to pass the corresponding values as input parameters for the services. Such a workflow of services can be executed sequentially or in parallel, depending on the flow definition thus providing enriched functionality according to specific conditions. The composed services differentiate by no means regarding description and invocation from other atomic services in the platform.

A first prototype uses the Fraunhofer FOKUS service broker because of the enriched functionality provided by this component. The service broker represents the main entity which ensures the service management in a service delivery platform. It offers a service registry for registered services, an environment for execution of services and an access control and personalization entity. The FOKUS service broker is implemented making use of the advantages of the OSGI which provides a dynamic environment for services management. In OSGI there are two types of entities: bundles and services. A bundle consists of the same elements as a jar file (e.g. classes, manifest file) excepting the manifest content which describes the exported and imported packages. Thus the creator of the bundle can easily control which packages to be exported and which not. The exported packages can be imported by other bundles in the environment. Beside the exposed functionality, as library, the bundles can register also specific services defined by an interface and implementation. The OSGI platform contains a local registry were the services information (e.g. name, description, configuration properties) is stored and respectively discovered by other bundles.

The service broker is based on the Equinox [8] implementation of the OSGI specification. In order to store more information about services like interface name, implementation class or description an XDMS (XML Document Management System) based registry was integrated into the platform.

For composing functionality, the FOKUS Service Broker integrated and extended the Apache SCXML (State Chart XML) engine [9]. Using SCXML one can easily define a flow execution of states. The engine was extended so that each state execution consists of a service invocation of which result will define the state transition.

A policy engine extended the functionality of the service broker with access control and personalization. The policy engine evaluates operator given policies, which result in a specific behavior through enforcing the rules by the framework. The policies have different priorities based on the identities which defined it. Thus the highest priority represent the policies defined by the system, followed by the policies defined by the service providers and later by the service customer.

Policies are identified as a logical set of rules while each rule consists of conditions and actions. During the evaluation, specific policies are selected (based on the service name, operation name, originator of the invocation respectively target of the invocation) and their conditions are evaluated against specific parameters (e.g. platform parameter, service parameter etc). As a consequence of the conditions evaluation success the associated actions are further executed.

Based on the action configuration, the enforcement of the actions is not only performed by the policy engine but by also delegated resources.

Our scenario points out QoS on a voice data connection, which differentiates prioritized emergency calls in contrast to not prioritized normal voice calls. In case of an emergency call stated in the intent statement by the UE, the policy engine activates a rule, which includes the service level requirement prioritization in the composition workflow created by the composition engine.

The first stage of the mediator passes requirements from the service down to the network layer without negotiating and optimizing the connection parameters because of simplicity. Therefore flow information is extracted by the mediator and the appropriate functional composition framework is selected afterwards, which appropriately enforces the requirements.

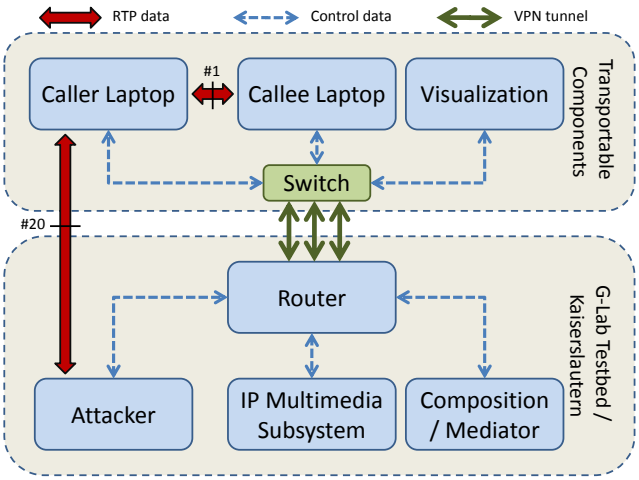

Fig. 4. Prototype Testbed Implementation Architecture

The service layer is secured by using the IP Multimedia Subsystem. We use the MONSTER IMS client from Fraunhofer FOKUS [10] as a basis for our UE. An extension of the MONSTER client adds functionality to send the intent statement to the broker using a SOAP message. Location information of the user is stated using Google Maps API [11]. The location information is signaled to a location service using SOAP. The UE registers at the IMS and authenticates his UE and authorizes his requests. The Multimedia Open InterNet Services and Telecommunication EnviRonment (MONSTER) is an extendible plug-and-play framework developed by Fraunhofer FOKUS. This toolkit enables the creation of rich terminal applications compliant to NGN, IPTV and WEB standards.

The interfaces between the service and network layer and mediator are using a subscribe/notify mechanism. Given the implementation of the service broker and the functional composition framework was done in JAVA, we made use of

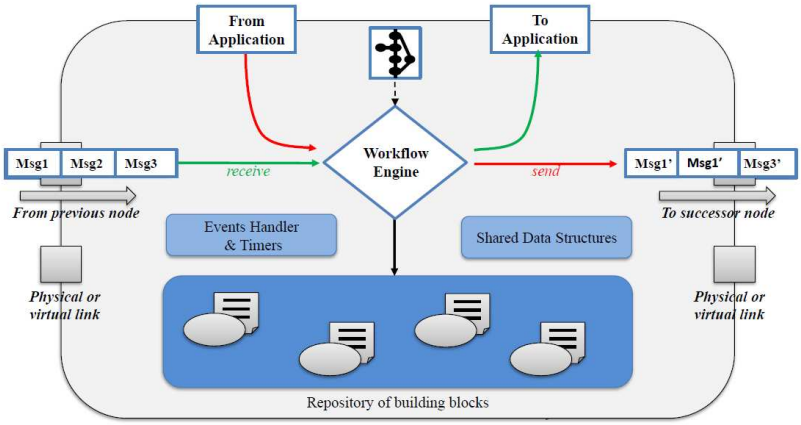

Fig. 5. SONATE Functional Composition Framework

the Java Messaging System (JMS) [12] features which also provides the necessary subscribe/notify features.

In a research concept [7] a Functional Composition Framework (SONATE) (as shown in fig: 5) has been developed to manage, execute and deliver the requested services. Functional Composition Framework consists of various components such as building blocks, a building block repository, a workflow engine. Building blocks are the implementation of the services and building can be composed. To hold those building blocks one repository has been created which has knowledge about all existing building blocks in the framework. Building blocks are independent but they have well-defined interfaces which is used to communicate with each other and the framework. Every building block has an attached description which holds information such as covered services, QoS parameters, requirements and constraints related to the building block. The workflow engine executes building blocks with given sequence which is defined by a workflow. realize the demonstration some required building blocks for the scenarios have been implemented in addition to basic functionality (e.g. transport, addressing). In presented scenarios, there have been requirements of traffic identification and prioritization of traffic. Different types of traffic (e.g. normal-call, emergency-call, Dos attack) have been identified in the demonstration which further have been prioritized with respect to given priority from the service layer. Location based authentication has been part of service layer which further demonstrates cross-layer composition and information exchange.

Our prototype includes IMS to provide a secure network and service environment as well as client authentication. Therefore services are also located in the IMS running on SIP Application Server (AS). We use a location service to store location information like GPS coordinates, address information, etc. We use a SIP based 3rd party call service (3PCS) to connect to call parties after retriev-

ing a user intent requesting a voice call to another instance. A modification in the 3PCS extracts IP addresses and ports during the call establishing process of sending to SIP INVITES of the two communicating parties. These information are signaled on to the mediator to assist in establishing a data path through the functional composition framework.

The attacker are using an Asterisk SIP server to create voice calls between two parties. The attacker establishes each call and plays an audio-file afterwards on both sides of the connection to emulate a voice call to generate network traffic.

A technical view on the architecture depicted in figure 4 divides the testbed into two physical divided parts. One fixed part of the components is located in the G-Lab testbed of University Würzburg, Germany running in virtual machines and the other transportable part is running on Laptops which are connected with the testbed in Würzburg through a VPN connection.

We implemented a work around support legacy applications with SONATE. We need to insert the RTP data into the SONATE framework to apply functionalities on the network traffic. A TUN/TAP emulates a virtual network interface. The TUN/TAP interface tunnels the UDP/IP RTP traffic in an extra IP packet, in order to perform routing in the functional composition framework and apply QoS on specific flows.

The laptops run Ubuntu 10, Java JDK 6 Update 22 and are connected over Ethernet to Deutsches Forschungsnetz [13]. The virtual server in the G-Lab testbed run VMWare Ubuntu 10 with 1,5 GB RAM. An VPN tunnel using OpenVPN interconnects the components.

5 Measurements and Evaluation of the Prototype

This section presents measurements based on the prototypical implementation running a demo scenario and concludes with an evaluation of the presented work. The first part is focusing on the evaluation of measurements, while the second part discusses the advantages and disadvantages of our Functional Composition and Cross-Layer Composition Architecture.

5.1 Testbed Measurements using the Demonstrator

We focus on traffic prioritization of an emergency call before attack traffic is injected into the network. The packet loss ratio of the prioritized stream is expected to be significant smaller then the un-prioritized one.

We capture the network traffic with the network protocol analyzer Wireshark [14] at the caller and the callee. Our VoIP scenario with attacker background network traffic was repeated five times with a different amount of attackers. We started with 20 attackers in the first round and increase the amount up to 25 attackers in the last round. An aggregation of all measurements regarding the loss of specific flows was computed afterwards and is depicted in figure 6, which points out the minimum, maximum and average loss ratios per flow. We ordered

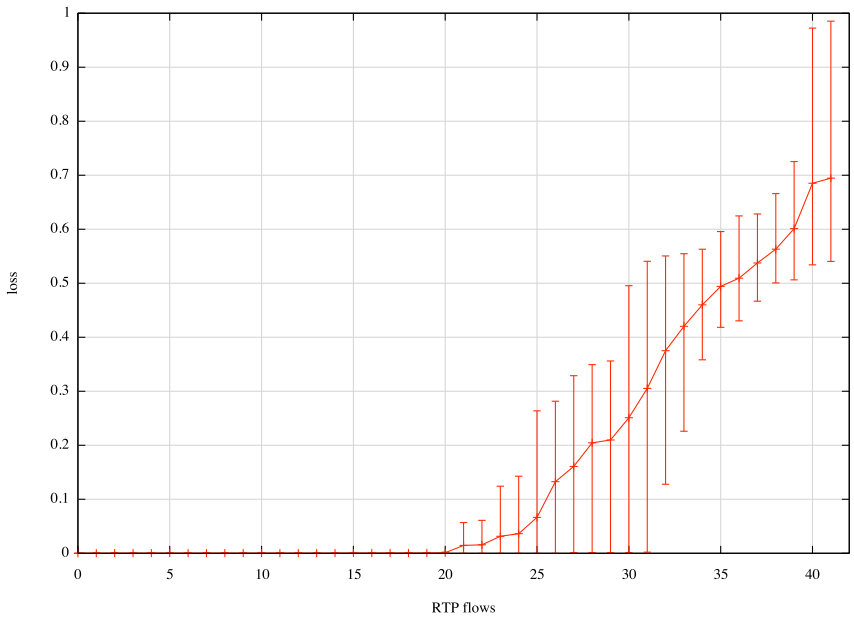

Fig. 6. Packet Loss Ratio per Flow

the flows in regard to their loss ratio from low at the left side to higher loss ratios on the right side.

The aggregated figure depicts only the emergency call flow plus 20 attacker calls. Additional attackers with the numbers 21 to 25 are excluded from the computation in order to generate the min, max and average graph. Each bidirectional call is an RTP session, which consists out of two data flows. Therefore we are analyzing 21 calls with 42 flows that are aligned on the X-axis. The Y-axis indicates the packet loss in percent.

The RTP flows are identified through their individual 32 bits Synchronization Source (SSRC) and the packet loss ratio is measured through analyzing the continuous sequence number of packets of the same RTP session.

A zero packet loss ratio was measured on the prioritized emergency call flow between the two clients through the functional composition framework SONATE. The prioritization mechanism of SONATE handles traffic flow specific. An internal buffer re-orders the incoming packets with regard to their prioritization. New incoming prioritized packets are send out earlier than un-prioritized packets, which are already in the buffer. The attacker calls 11 to 20 begin to have a loss ratio of a few percent up to an average loss ratio of 70 % of attacker 20.

The successful prioritization is proven through the zero packet loss of our measurements which proves and validates the idea behind the presented cross-layer functional composition framework. An intent statement of the UE triggers a service composition, which determines requirements for the network, which are

in turn executed on the network level through activating functional blocks in the FC framework.

5.2 Evaluation of the Cross-layer Concept

The implementation and the done observations concerning the established service on the new network architecture approach show four functional scenarios with the expected behavior. Based on the discussed results, an integration of the current used legacy application support could be expected. Using bridges between the current and the new architecture should allow a soft revolution in the existing network.

However, network architectures with similar service levels, should in the end only be distinguishable by the provided interface between service and network, whereby the new approach supports a service oriented notation. Even there are different challenges on the design of a service oriented approach; it should be a benefit for the development with respect on requirements of service oriented application development.

Beside the co-existing usage, the new approach will provide new dynamic abilities on the endpoints of a communication path, also on the network itself. The shown security example could be mentioned. The approach is utilized by detecting attacks on the application level and by mitigating their effect on the network in a dynamic way. There is also a thinkable scenario where the network is able to adapt the routing functionalities for special kind of calls.

Furthermore, using a network with dynamic workflows on endpoints and network elements allows the carriers of the networks to provide customized workflows for special requirements, which could be offered as unique feature.

The providers are free to setup the same workflow in different granularities and qualities, this can be shown for example in the demonstrated scenario. With a business view over the network, new charging schemes could be build up (e.g., charging by workflow). A Service Delivery Platform (SDP) is possible that assigns flows a certain QoS level in order to provide a Quality of Experience (QoE) at the user side, which matches the user profile and the individual account credentials. Data traffic of premium users might be prioritized from standard users traffic.

However, these alternative architecture approaches could have some disadvantages such as unknown security vulnerabilities concerning the dynamic composition or possible difficulties related with network interaction functionalities such as congestion control and fairness. Also one big issue is to define fine granularity (atomic) functional blocks; here the approach is just on a starting point. This approach needs standardization; with more focus on the classes of functions and the framework which have to interpret them than on the functionality itself.

It is true that this approach could exist beside the current layer based architecture, by using the current service level as subset. However, the complete potential will only be available in case of exchange of the routers and all middle boxes. This would be, in other words, a clean slate approach.

In case of establishing this architecture as alternative just on endpoints, with a pool of functionalities representing the current Internet, this will give the chance to overwork the application interface in a service oriented way and setup this new approach in silent revolution.

The following list summarizes the main advantages and disadvantages of a general cross-layer architecture using a service and functional composition:

- Supporting QoS (prioritization, bandwidth, etc.) for specific data flows.
- Flexible service delivery: New operator and service provider business models arise.
- Enabling security on multiple layers that benefits from interleaving network and service layer security.
- New charging schemes occur.
- Flexible integration and deployment of new (third party) services.

- Access network heterogeneity requires standardization of FC Frameworks, Service Level Agreements (SLA) and federation of different operator networks.
- Open security issues.
- Future Internet Applications need to state their requirements for a specific service or connection. Dynamic user selected requirements may be send through an intent statement or static operator policies may be applied.
- Clean slate requires costly changes in the physical network operator deployment is necessary to introduce a FC framework on existing routers.
- The Service Broker, Mediator and Functional Composition framework are designed as single instances what results in missing scalability.
- Limitation of the service spectrum to communication services that enable 3rd party initialization.

6 Future Work on the Cross-Layer Architecture

The implementation of a cross-layer composition approach, as demonstrated in this paper, works in a straight forward way. It demonstrates a way how it is possible to overcome the limitations of the 7-layer OSI reference model.

However, simple four scenarios do not represent the complexity of the communication in the current used networks (e.g., the Internet). It also lacks of a full model of (atomic) functional blocks, which represents the service of the current used networks. However, this is essential in order to support a soft revolution. Such an approach, which tries to impose a monolithic structure which should allow complete diversity, presupposes that models will be setup which shows the interaction of the (atomic) functional blocks and the limits in practical applications. In order to address the problems adequately, the first requirement is the knowledge of the extent to which the goals being pursued are realistic, and of possible side-effects.

In addition to QoS, many different features will be addressed, such as reliable transport, congestion control, buffer management or connection maintenance.

However a starting point of selecting and composing functional blocks will be the transport layer of the OSI Model. The transport protocols offer huge variety of functions, but have no hardware impact and represent in the end the current "de-facto" interface to the application level. Furthermore current efforts on the Multi-Path extensions of TCP and SCTP will also demonstrate the benefits of the shown approach.

Next steps on the framework development itself are to simplify the interfaces and the interaction of the components. Furthermore the components utilized for composing are currently designed in a centralized manner, which will be classified as a drawback. One goal is to work on decentralization in order to support a P2P-like exchange of selecting and composing functional blocks.

Another aspect of the future work on the cross-layer architecture concerns the QoS aware Service Broker and monitoring aspects. Monitoring of network and service layer components brings awareness of utilization on both layers. On one side, this monitored data is used to identify security anomalies and on the other side QoS is supported. Therefore the new gathered information will be aggregated and analyzed for anomaly detection. Deep packet inspection classifies network traffic and services collect service executions statistics, which are aggregated to identify and mitigate attacks on both layers as soon as possible. A distributed firewall will react on potential security threats, by applying IP and port filters close to the identified network component.

One aspect of our future work hosts services in the cloud. The service execution and utilization statistics will be used to provide elasticity on the service performance by providing scalability in the service delivery process through service load balancing.

Finally a QoS aware service broker will select services based on their current utilization, performance and costs. The service composition and orchestration process is interleaved with monitoring, security and service scalability.

7 Acknowledgments

This work is funded by the German Federal Ministry of Education and Research within the scope of the G-LAB Deep project [15] as part of the G-Lab project. G-LAB offers an experimental facility that enables researchers to conduct experiments in a distributed, real-world environment. In the G-LAB Deep project we are especially concerned with requirement driven composition between services on network and service layer.

References

1. Future Internet, The Cross-ETP Vision Document, Version 1.0, 2009
2. Irina Boldea, Konrad Campowsky, Christian Henke, Julius Mueller, "QoS aware Cross-Layer Service Delivery Platform", 3rd GI/ITG KuVS Fachgesprch on NG SDPs "Towards SDPs for the Future Internet", October 2010, Berlin.

3. Julius Mueller, Abbas Siddiqui and Dirk Hoffstadt: "Cross-Layer Security Demonstrator for Future Internet", "Security in NGNs and the Future Internet", Part of Future Internet Symposium FIS 2010 September 20, 2010, Berlin, Germany
4. M.Becke, D.Hoffstadt, E.Rathgeb (University of Duisburg-Essen, Germany), K.Campowsky, C.Henke, J.Mueller, T.Magedanz (TU Berlin, Germany), C.Schmoll, T.Zseby (Fraunhofer FOKUS, Berlin, Germany), A.Siddiqui, P.Müller (University of Kaiserslautern, Germany), "Addressing Security in a Cross-Layer Composition Architecture", EuroView 2010, Würzburg, Germany
5. N. Blum, I. Boldea, T. Magedanz, U. Staiger, H. Stein, "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
6. A.Siddiqui, D.Guenther, P.Muller, C.Henke, T.Magedanz, Mediation between Service and Network Composition, EuroView 2010, Wuerzburg, Germany
7. Paul Mueller, Bernd Reuther. Future Internet Architecture - A Service Oriented Approach, it - Information Technology, Jahrgang 50 (2008)
8. Equinox project, http://www.eclipse.org/equinox/
9. State Chart XML (SCXML), http://commons.apache.org/scxml/
10. A.Bachmann, M.Alice, T.Magedanz,"Requirements for an extendible IMS client framework", MOBILWARE, MOBILe Wireless MiddleWARE, Operating Systems, and Applications, 2007, Innsbruck, Austria
11. Google Maps API, http://code.google.com/intl/de-DE/apis/maps/
12. Java Message Service (JMS), http://java.sun.com/products/jms/
13. Deutsches Forschungsnetz, http://www.dfn.de
14. Wireshark project, www.wireshark.org/
15. G-Lab DEEP Project, http://www.g-lab-deep.de/