Multimed Tools Appl
DOI 10.1007/s11042-014-2072-5

Requirement driven prospects for realizing user-centric network orchestration

Thomas Zinner · Tobias Hoßfeld · Markus Fiedler · Florian Liers · Thomas Volkert · Rahamatullah Khondoker · Raimund Schatz

Received: 22 April 2013 / Revised: 17 April 2014 / Accepted: 30 April 2014

Abstract The Internet's infrastructure shows severe limitations when an optimal end user experience for multimedia applications should be achieved in a resource-efficiently way. In order to realize truly user-centric networking, an information exchange between applications and networks is required. To this end, network-application interfaces need to be deployed that enable a better mediation of application data through the Internet. For smart multimedia applications and services, the application and the network should directly

T. Zinner (✉) · T. Hoßfeld
Institute of Computer Science, University of Würzburg, Würzburg, Germany
e-mail: zinner@informatik.uni-wuerzburg.de

T. Hoßfeld
e-mail: hossfeld@informatik.uni-wuerzburg.de

M. Fiedler
Department of Communication Systems, Blekinge Institute of Technology, Karlskrona, Sweden
e-mail: markus.fiedler@bth.se

F. Liers · T. Volkert
Technical University of Ilmenau, Ilmenau, Germany

F. Liers
e-mail: florian.liers@tu-ilmenau.de

T. Volkert
e-mail: thomas.volkert@tu-ilmenau.de

R. Khondoker
University of Kaiserslautern, Kaiserslautern, Germany
e-mail: khondoker@informatik.uni-kl.de

R. Schatz
Telecommunications Research Center Vienna – FTW, Vienna, Austria
e-mail: schatz@ftw.at

Published online: 15 June 2014

communicate with each other and exchange information in order to ensure an optimal Quality of Experience (QoE). In this article, we follow a use-case driven approach towards user-centric network orchestration. We derive user, application, and network requirements for three complementary use cases: HD live TV streaming, video-on-demand streaming and user authentication with high security and privacy demands, as typically required for payed multimedia services. We provide practical guidelines for achieving an optimal QoE efficiently in the context of these use cases. Based on these results, we demonstrate how to overcome one of the main limitations of today's Internet by introducing the major steps required for user-centric network orchestration. Finally, we show conceptual prospects for realizing these steps by discussing a possible implementation with an inter-network architecture based on functional blocks.

Keywords Quality of experience · User-centric network orchestration · Application requirements · Network-application interface · Service composition

1 Introduction

The user satisfaction with multimedia application performance in communication networks has attracted increased attention during the recent years. Parts of this growth of interest in Quality of Experience (QoE) issues can be explained by the increasing competition amongst providers and operators, which increases the risk that users churn if they become dissatisfied. Operators can adopt a user-centric perspective on Internet application delivery in order to improve the QoE and, thus, counter that risk. The user-centric perspective requires a cooperation between applications and networks, in which both consider end user requirements in terms of QoE metrics. Thus, applications and networks have to provide QoE-related information to each other.

In the current Internet, however, it is not possible to exchange these information directly between applications and transport and network layers. There are no network-application interfaces deployed that support a smart mediation of application data through the Internet. In this context, content mediation includes resource management as well as traffic management.

To overcome these limitations, several different approaches exist that can mainly be classified as network-aware applications or application-aware networks. Application-aware networking is often realized on behalf of additional monitoring systems or network elements. For example, packets are classified within the ISP's network and processed according to the application requirements as specified in generic policies [20]. Other options are more specialized entities like media-aware network elements [9]. Network-aware applications measure the network performance and react on these measurements. The Skype VoIP application, for example, measures the current network conditions and reacts appropriately at the application layer to overcome network problems, e.g., by adapting/changing the voice codec or performing re-routing on application layer [11]. For smart applications and services in the Internet, however, the application and the network should directly communicate with each other and exchange information adequately, so that the end user experiences good quality.

The main goal of this paper is to investigate prospects for realizing user-centric network orchestration. The conceptual prospects considered here include different service composition approaches like "Forwarding on Gates" [21] or "SONATE" [17]. These architectural

concepts dynamically react to user and application requirements. To this end, we follow a *use-case driven approach*. In particular, we consider Quality of Experience (QoE) for three relevant Internet services from the users point of view. These are A) streaming services like high-definition Internet TV, B) Video-on-Demand (VoD) streaming with high video quality playback demands and C) user authentication for payed multimedia services with high security and privacy demands.

To these three use cases, the following methodology is applied. First, the user perceived quality of each service is quantified by means of subjective user tests and full-reference video quality metrics. This allows to derive end user requirements based on the resulting QoE model, which can be translated into certain application-specific requirements. These application-specific requirements are then further mapped to network-specific requirements. Second, these detailed (functional and non-functional) requirements need to be passed to the network stack, which is not possible with today's APIs. Therefore, new application-network interfaces providing the information needed by composition and forwarding of services have to be designed. Finally, the network stack must be able to handle these requirements dynamically and select optimal entities and services by matching the application and user requirements with the offered resources and capabilities of the network.

The contribution of this paper is twofold. First, user, application and network requirements for the three use cases are derived. Practical guidelines, which translate QoE requirements to network requirements, are provided. Second, these translated requirements are handed over to the network, which selects the appropriate network resources and functions to satisfy these requirements. Different prospects for realizing this user-centric network orchestration are revisited from a user-centric network orchestration point of view. One of these prospects – an implementation with "Forwarding on Gates" – is reviewed in detail.

The remainder of this article is structured as follows. Section 2 quantifies the requirements for the TV (Section 2.1), VoD (Section 2.2), and AUT (Section 2.3) use cases. As a result, Section 3 shows the implications for the design for user-centric network orchestration. In particular, the specification of the requirements for a Communication Service Description Language (CSDL) is depicted in Section 3.1, the selection of optimal entities is discussed in Section 3.2, and a possible interface between application and network is reviewed in Section 3.3. Conceptual prospects for future network architectures and related work are discussed in Section 4. First implementation insights are described in Section 5. Section 6 concludes this work with an outlook on future work.

2 Use cases and their requirements

In this article we follow a use case driven approach in order to derive the requirements for user centric network orchestration. In general, the user perceived QoE drives the application requirements which then drive the network requirements. As a result, guidelines for orchestrating the use cases are developed in this section.

In particular, we consider high-definition TV streaming, video-on-demand, and user authentication, e.g., for a payed multimedia streaming like the ones mentioned before. These use cases are fairly orthogonal in terms of (i) user requirements (e.g., high resolution, smooth video playout without interruptions, high security and privacy), (ii) application requirements (e.g., forward error correction, prebuffering, authentication), and (iii) corresponding network requirements (e.g., low packet loss, reliable transmission, small delays). We differentiate between functional requirements – when a particular functionality

is requested– and non-functional requirements – when the modality of providing a special functionality is requested explicitly, e.g., a throughput of 420 kbps. A summary of these requirements is given in Table 1. The user-centric column describes the requirements for the specific use case from the user's point of view. The network orchestration column summarizes the requirements from the application and the network perspective. It is split into application and network requirements due to possibly different involved control entities.

2.1 Use case 'TV': scalable live HD video streaming

2.1.1 User requirements

The TV use case considers live streaming services. The user expects high video quality and a high resolution for watching, e.g., a soccer match in HD resolution at home. Furthermore, the contents are to be delivered in real-time to guarantee the live experience during watching. Based on these user requirements, the network and application requirements are derived.

Table 1 Functional ('F') and non-functional ('N') user, application, and network requirements for the use cases and in general

Use cases	User-centric		Network orchestration			
	User requirements		Application requirements		Network requirements	
TV	F	high video quality, no artifacts	F	support of scalable video streaming for quality adaptation	N	low jitter and delays (real-time)
	N	high resolution			N	sufficient bandwidth, low packet loss
	N	live, real time	F	forward error correction (app/net)	F	in-order transmission of frames
VoD	N	small initial delays	F	prebuffering of video frames according to network conditions	N	sufficient bandwidth
	N	no stalling			F	reliable transmission to avoid video quality degradation
AUT	N	small waiting times	N	small response times	N	small delays
	F	secure action, no hacking, privacy	F	authentication	F	reliable transmission
			F	encryption (net/app)	F	end-to-end tunnel
general	F	high availability, high reliability	F	service description language: requirements resources, capabilities		
	N	high QoE, high comfort	F	service composition, selection of optimal entities		
	N	scales with number of users & services	F	application-network interface		

2.1.2 Application requirements

An effective way to reduce the required bandwidth and adjust the video transmission is provided by the scalable SVC extension of the video codec H.264/AVC [16]. This extension enables the encoding of a video file at different qualities within the same layered bit-stream. Besides of different resolutions, this also includes different frequencies (frames displayed per second) and image qualities with respect to Signal-to-Noise Ratio (SNR). These three dimensions are denoted to as spatial, temporal and quality scalability. If the available bandwidth is not sufficient to stream the full video quality, the question arises which layers are more important than others. At least a minimum resolution, quality and frame rate, which also depends on actual usage context, has to be provided. Otherwise the user will not accept the video service. Higher quality-related layers will increase the QoE further, but also require a higher bandwidth.

Current research on the impact of packet loss on user perceived quality indicates a strong impairment of the video already at small packet loss rates of less than 2 % [1]. This motivates mechanisms able to protect a video stream against packet loss, such as Forward Error Correction (FEC) methods. These techniques provide means to correct corrupted or lost packets and thus allow an exchange of bandwidth for packet loss protection.

2.1.3 Network requirements

Due to the real-time nature of TV, only a small video buffer is available at the application which is in the order of a few seconds only. Hence, the network has to deliver the data with low delay and also low jitter that the video buffer can cope with. If data packets in the network get lost, retransmissions are often not possible without violating the real-time constraint. Typically, connectionless transport protocols are used therefore. However, data packets still should arrive in order at the video client that the video player on application layer does not need to take care about the order of frames. Thus, in-order transmission and reception of frames is a network requirement.

As the user demands high video quality and high resolution, the network has to be capable of providing sufficient bandwidth to carry the video contents. Since video streaming is prone to packet loss and small packet loss rates already lead to a strong impairment of QoE [12], the packet loss ratio should be fairly low. Such insufficient network resources, e.g., too little bandwidth or packet loss, result in a strong impairment of the video service in form of video decoding errors and frame drops. Controlled quality adaptation, e.g., by reducing image quality/resolution, allows for considerable bandwidth savings while having only minor impact on user perceived quality [10, 12].

2.1.4 Guidelines

In order to optimize the Quality of Experience (QoE) for live video streaming by real-time adaptation of video bitrate and thus video quality to the current network situation (i.e., available bandwidth or unreliable links with packet losses), it is necessary to know the bandwidth requirements of the different layers and their impact on QoE. Therefore, we conducted an intensive measurement study with different network conditions. For evaluating the user perceived quality, we use a full-reference video quality metric SSIM [4] and map it to subjective mean opinion scores (MOS) [6]. We use the scalable video codec H.264/SVC and adapt the spatial scalability. For that we generate a SVC encoded file with the three resolutions 1920×1080, 960×540, and 480×270 pixels. The smaller resolutions were scaled up

to 1920×1080 using bilinear interpolation and then compared with the highest resolution. Please note that depending on the specific SVC encoding also quality or temporal scalability can be adjusted. For this study we implemented the forward error correction (FEC) on behalf of Luby codes [24].

The results of our study are depicted in Fig. 1 with the user perceived quality in terms of MOS on the y-axis (5: very good quality, 1: bad quality). The x-axis denotes the goodput, i.e., the application-perceived throughput, which is equal to one if the maximum quality can be streamed. For streaming a video in 1920×1080 resolution with no FEC, QoE decreases rapidly and gets minimal for packet loss larger than 2 %, i.e., a goodput of 98 %. In case of controlled quality degradation (lower resolutions) but no packet loss, QoE decreases much slower and keeps on a tolerable value even for small resolutions at 480×270 pixels. In that case, only 15 % of the bandwidth with maximum resolution is required. Please note that the results for 640×360 and 960×540 pixels are interpolated. Hence, in case of bandwidth shortage, SVC is a powerful mechanism for reducing the QoS requirements of the application without compromising QoE too much.

The other cases in Fig. 1 denote controlled service degradation with FEC mechanisms. As an example, streaming a video with a resolution of 960×540 with FEC is considered. Switching to this resolution reduces the required goodput by 75 %. We add a FEC mechanism which adds an overhead of 20 %, resulting in a relative goodput of 30 %. That impact of the FEC is illustrated by the red arrow. However, by transmitting more symbols it is now possible to cope with packet loss rates up to 70 % and still ensuring a very good quality of the video stream and not allocating more bandwidth than before. For lower video stream qualities, more bandwidth can be used to protect the video stream.

The results in Fig. 1 provide clear guidelines how to optimize QoE for live TV streaming according to the current network situation.

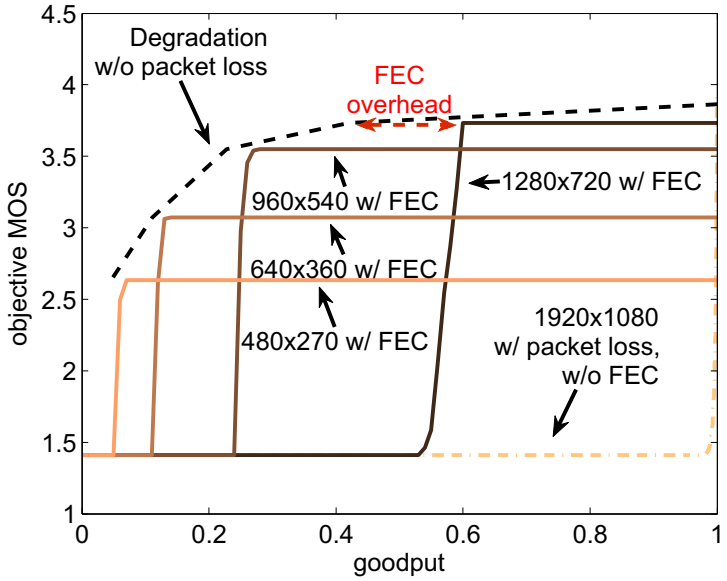

Fig. 1 Guideline for selecting the resolution of the SVC video stream depending on the available network capacity. In case of packet loss due to unreliable transmission, forward error correction (FEC) is suggested to achieve high QoE, which increases the need for goodput

2.2 Use case 'VoD': video-on-demand streaming

2.2.1 User requirements

The VoD use case is motivated by the high popularity of the YouTube video streaming service and the large share of Internet traffic caused by it. Since the videos are requested on demand without hard real-time requirements (in contrast to the TV use case), a reliable transport protocol can be used to guarantee the data delivery. The corresponding QoE models are different from traditional UDP-based video streaming, as only the video playback itself is delayed while the transmitted audiovisual content remains unaltered. If the available network data rate is lower than the video bit rate, video transmission becomes too slow, gradually emptying the playback buffer until underruns occur. Then, the user notices interrupted video playback, commonly referred to as stalling. From a QoE perspective, novel models for quantifying the impact of temporal impairments to the user-perceived quality have to be developed. In [14], we developed a QoE model depending on the number of stalling events and the length of individual stalling events by means of subjective user studies. Even very short stalling events of a few seconds already decrease user perceived quality significantly. Furthermore, we quantified the impact of initial delays for filling the video buffer on QoE in [14]. The results show that initial delays up to 20 seconds are perceptible, but not annoying. Indeed, from the user's perspective, no stalling must occur and the initial delays should be minimal.

2.2.2 Application requirements

According to the user requirements, the application needs to prebuffer as much video frames as necessary in order to avoid stalling in any case. This will lead to the optimal QoE according to the current network situation, as resource limitations are overcome by prebuffering. However, [13] shows that the computation of the optimal initial delay requires complex models taking into account diverse information of the video contents, reflecting e.g., autocorrelation of frame sizes or scene changes within the video. In this context, the smallest initial delay for a particular video and a given network data rate which avoids stalling is referred to as optimal initial delay. To realize the information exchange between network and application and to compute optimal initial delays before video playout, several solution approaches exist. For example, the video frame structure is sent as metadata before the transmission of the video data to the application. Another option is, that the application (or some other network entity) signals the video server the currently available network capacity, such that the video server (having the entire video structure information) computes the optimal initial delay and sends it back to the client. However, such detailed requirements cannot be passed to the network stack with today's APIs. Therefore, new application-network interfaces are required, see Section 3.3.

2.2.3 Network requirements

For a smooth video playout without stalling, the average network data rate must be sufficiently larger than the video bit rate –which will be discussed in the guidelines of this use case– and the variance of the network data rate must be small enough so that no buffer underrun occurs. Since the frames have to be shown in order and the player does not want to sort them by itself, they must be delivered by the network stack in order, too. Despite today's usage of error and loss free transmission via TCP, the video codec might be able to

deal with some loss or bit errors. The maximum amount of bit errors or lost packets depends on the video codec and on the required QoE level. Nevertheless, for the use case VoD, reliable data transmission is required. On the one hand, the stack must be able to buffer data locally, in order to sort them and to reduce the variance of the data rate. On the other hand, the network must be able to reserve data rates and to quickly retransmit lost or corrupted packets. Both must be done in a scalable way in order to support the large amount of users for successful VoD services.

2.2.4 Guidelines

For reducing the complexity and the overhead for computing optimal initial delays, a practical guideline is desired, which maps the initial delay as experienced by the end user to a certain network data rate. Hence, there is a trade-off between bandwidth provisioning in the network and optimal initial delays on application layer to fill the video buffer.

However, since the optimal initial delay depends on the actual video contents, we conducted an extensive measurement series taking place from July to August 2011 during which more than 37,000 YouTube videos were requested and about 35 GByte of data traffic was captured. More details concerning the measurements can be found in [13].

Figure 2 shows the results of the analysis. On the x-axis, the optimal initial delay normalized by the video duration is depicted. By normalizing the duration of the video clip, user expectations can be taken into account, since the accepted waiting time is correlated with the duration of the video clip [19]. On the y-axis, the bandwidth provisioning factor β is plotted, which is the ratio between the network data rate B and the video bit rate V, i.e., $\beta = \frac{B}{V}$. The different curves in the figure show the P-quantile over the set of video from the measurements. Hence, the P-quantile denotes that with a probability of P there will be no stalling for the corresponding initial delay and bandwidth provisioning factor for any video. The measurement curves can additionally be fitted to get a compact representation which may be easily exchanged between application and network.

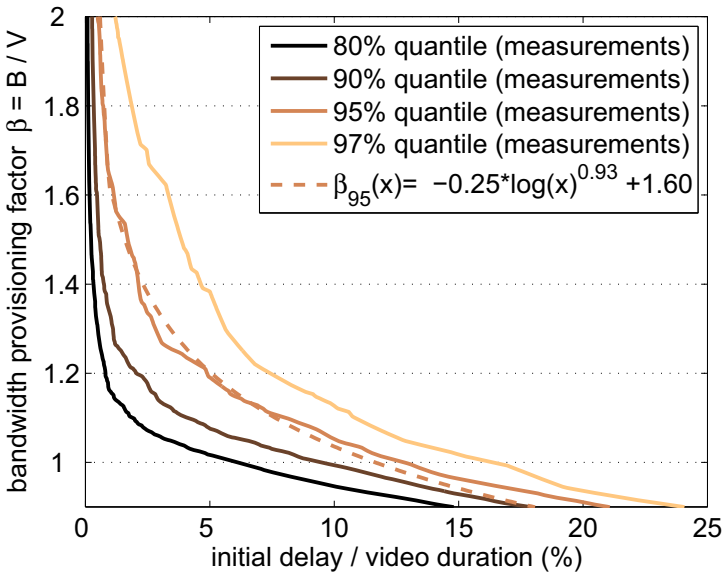

Fig. 2 Trade-off between bandwidth provisioning in the network and initial delay experienced by the user for VoD streaming

As one example, the fitting function $\beta_{95}(x)$ for the 95 % quantile is depicted in Fig. 2. As a rough guideline, an optimal initial delay of 5 % of the video duration requires a network data rate which is 120 % of the video bit rate.

2.3 Use case authentication for multimedia services

2.3.1 User requirements

While the previous two use cases are dealing with streaming media services, we consider now the authentication for a service. In particular, we focus on the authentication for paid multimedia services like Sky Go or Netflix. For such services, the service provider demands for high security and privacy to ensure that content is only delivered to users paying for it. However, at the same time, users do not want to wait and demand small waiting times [5].

For the OSN use case, we utilize the results of previous studies in [22, 23]. These experiments on user perceived QoE for web-based login times were tested using a laptop with a browser. The web page of the social network in this experiment used a remote OpenID server for authenticating users. A traffic shaper ensured fixed pre-determined response times for the authentication procedure when the user logged in. After encountering a certain response time for login, the participant rated their subjective experience. In particular, users were asked how they experienced the login with regard to the response time.

2.3.2 Application requirements

Figure 3 illustrates how the user requirements are mapped to application requirements which then guide the network requirements. In particular, the use case OSN is considered as an example. The authentication process requires N messages between the client device and the

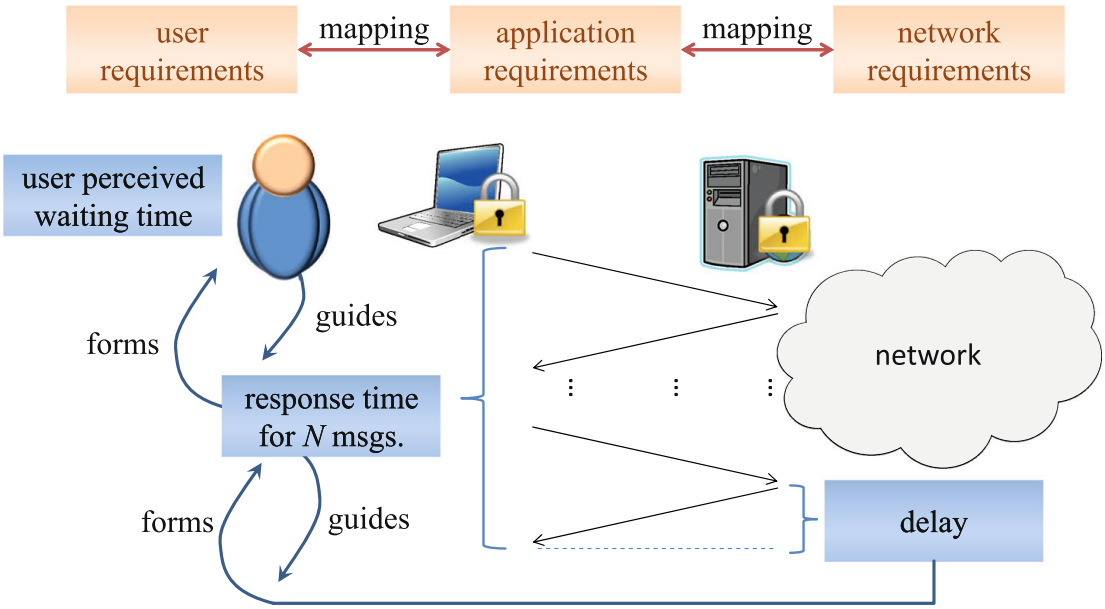

Fig. 3 Use case OSN. In general, user requirements guide application requirements which in turn guide network requirements. On the other hand, network performance influences application performance and thus QoE

authentication server. The corresponding response time on application layer forms accordingly the waiting time and thus the QoE. [23] shows that the QoE can be estimated on the response time T_R in the following way, $QoE = 4.7e^{-0.1T_R/\mathrm{s}}$. On the other hand, the QoE model guides the application which response times are required for a certain QoE level. As further application requirements, the authentication mechanism itself as well as the encryption of the data delivery are necessary.

2.3.3 Network requirements

To enable secure service delivery, an end-to-end tunnel is required and messages between client and server need to be exchanged reliably. In order to realize small waiting times for the end user, i.e., small response times on application layer, the network requires small delays on the end-to-end path. In particular, [23] reports on the following relationships which nicely demonstrate the mapping between user, application, and network requirements. The application response time T_R for authentication is found as $T_R = 2Nd + T_i$ depending on the network delay d, the number N of messages, and the internal time T_i determined by the actual software and hardware, respectively.

2.3.4 Guidelines

Bringing network delay and QoE together, we arrive at $QoE = 3.9e^{-2.2d/\mathrm{s}}$ for $N = 11$ messages and $T_i = 1.8\,s$ as practical guideline for orchestrating this service. If the end-to-end delay is larger than a certain value, then a less secure, but faster authentication scheme may be used according to the user's and service provider's preferences.

3 Requirements for the system design

In general, users demand for a high QoE, usability and convenience, while at the same time the service has to be available and reliable independent of the overall number of other users and services. However, applications and networks have to take available resources, costs, and fairness across applications and users into account. This includes scalability issues regarding the number of autonomous systems, number of connections and number of network functions. Moreover, networks have to flexible to react on the requirements of the users and to adapt to them in an efficient way.

Such flexibility can be achieved by using *functional blocks*. Functional blocks are instances of functions that provide some arbitrary service to packets. Their specific implementation differs between different future network approaches, more details are given in Section 4.

Systems that manage functional blocks and react on requests for additional functions typically use a so called *service description language*. Such a language is used to describe available resources or capabilities. If the same language is used for describing functional and non-functional requirements of a transmission request, it can be used as input for the *selection and composition process* as well. This process is responsible to compose suitable functional blocks in a particular order to a function *chain*. A chain has to include all functions required to fulfill the requirements from the transmission request. The result of the selection and composition process is also called service composition or functional composition. The composed functional blocks are fed with the application data and execute their internal function on the application data.

However, traditional APIs like the Berkeley Socket API do not provide the possibility to define the previously listed set of requirements of a transmission request. Hence, a new class of API is needed, which support more flexibility for defining additional transmission requirements and support a mechanism for getting feedback from the network. This feedback should provide information about the result of the transmission request, e.g., if all requirements could be fulfilled or if an error occurred.

The resulting system design for the user-centric network orchestration is shown in Fig. 4. It can be separated into the following parts: requirements, resources, capabilities, selection and composition process, and the service composition in the network. The selection and composition process takes the requirements from users, applications and networks, and derives required functions and their order. The service composition or functional composition, which is depicted at the bottom of Fig. 4, composes functional blocks to chains. It uses the result from the selection and composition process as guideline and executes its decision.

Requirements can be injected to the selection and composition process by networks and applications. They can be categorized as functional and non-functional. A functional requirement is a request for a particular functionality. An example requirement is “secured transmission”. It prompts the network to enabled security features along the route to provide end-to-end secured transmission of application data. A network can also have non-functional requirements such as maximum acceptable delay, loss, and jitter. A video receiving application may request a particular resolution, because a user has selected it or it fits the graphical user interface. Users may influence the requirements in multiple ways. For example, one user watches a movie with high resolution while another user wants to watch a video stream of a video conference in real-time.

Independently from the source of a requirement – if it was defined explicitly by a user, implicitly by an application or automatically by a network (operator) – the system must be able to process it and react in form of a parameterization of the transmission path. For this purpose, a mapping between the different types of requirements types must exist, as described in Section 2.3.

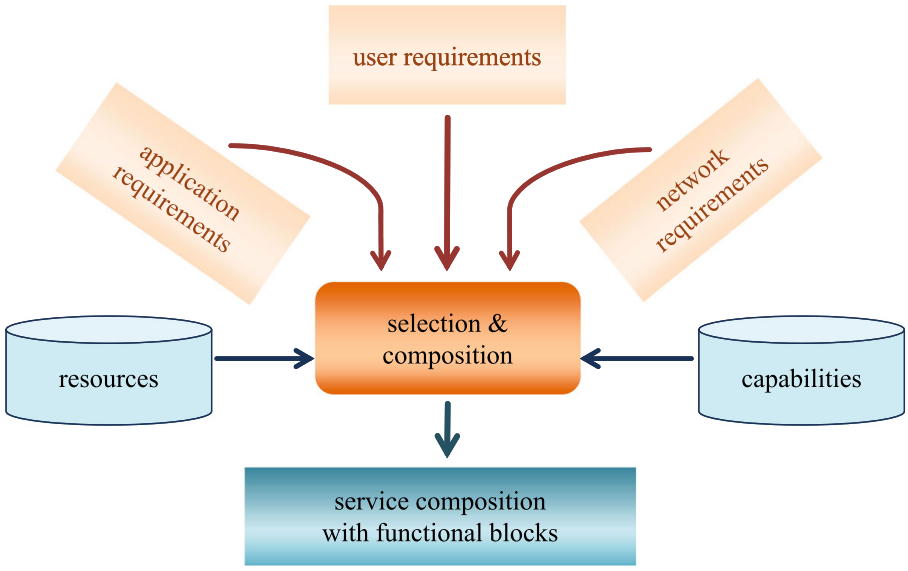

Fig. 4 Required system design for user-centric network orchestration using service or functional composition

The available resources and capabilities influence the selection and composition process as well. For example, functions such as encryption, authentication, and compression might only be instantiated in some networks or are available on end hosts only. Furthermore, non-functional capabilities like delay and loss rate differ along possible routes from the source to the destination. Therefore, the system has to support a requirements-aware routing to guide application data along paths providing suitable capabilities to satisfy the requirements. This routing is part of the selection and composition process.

The different capabilities of nodes lead to a distributed orchestration of functional blocks. Some functional blocks at end hosts in combination with blocks at relay nodes within a network make up the end-to-end chain. Since not all nodes know all functions supported within a network, the process of constructing a chain will most probably be distributed. Various nodes along a path will contribute with small, orchestrated parts of a chain.

3.1 Specification of the requirements for using communication service description language (CSDL)

All of the application, network and user requirements, available capabilities and resouces can be specified by using a CSDL. This special type of service description language has to have several components: effects, operators, units, interfaces, datatypes, influence, dependencies and aggregators [18]. These components are necessary for selection and composition processes.

Each implementation of a protocol, algorithm or mechanism is named as functional block. For example, an implementation of CRC-32 can be seen as a functional block. A functional block has a minimum of two interfaces: *up* (towards the application) and *down* (towards the network). Each interface is associated with a particular datatype. The type of the interface determines whether the interface can receive a connection from a particular interface. Compatibility of the connections between the interfaces is checked by using commonly known datatypes.

The component *influence* determines whether a functional block affects the header, the payload, the complete packet, or it does not affect at all. For example, a CRC-32 functional block adds a checksum into the header. In contrast, a forwarding functional block, which is used to forward the packets to the next interface, does not change a single bit of the packet.

During selection and composition dependencies between the functional blocks, services, or a functional block and a service might be considered which can be achieved by the specification of dependencies. The more dependencies a functional block has, the less easy it is to reuse. That is why the dependency description of a functional block is optional.

However, when the selection and composition process is completed, which produces a composed service of functional blocks, it can be described by a simple construct as follows {*effect operator attribute*}.

An *effect* is the requested or offered capabilities to/from a fine-grained functional block (for example, implementation of a retransmission algorithm) or a (virtual) network, which can be seen as a coarse-grained functional block. An attribute is the value of the effect and an operator connects an effect to an attribute.

For example, the user requirements for our first use case can be expressed as $\{live = true\}$, $\{videoquality = high\}$, $\{artifacts = false\}$ and $\{resolution = high\}$.

The application requirements for the same use case can be described as $\{SVC = true\}$, $\{bandwidth = medium\}$, $\{packetloss <= 2\%\}$ and $\{FEC = true\}$.

The networks requirements can be written as $\{jitter = low\}$, $\{delay = low\}$ and $\{in\text{-}order\text{-}delivery = true\}$.

This simple construct has several advantages: firstly, suitable functional blocks can be chosen just by matching the requirements with the offers, secondly, this supports evolution of the network because as soon as new requirements and building blocks emerge, they can be described by using this construct. This allows to adopt and extend the language whenever new requirements or building blocks are available.

3.2 Selection of functional blocks

Based on the user, application and network requirements, suitable functional blocks should be selected and used. For example, when security is requested, a functional block, which implements an encryption algorithm, should be selected.

If more than one functional block provide the same functionality with different quality parameters, the selection has to take the parameters into account. For example, security functionality can be provided with different strength using different length of the keys (AES 128, AES 192, AES 256). If a user requested the "best" security, the best encryption algorithm (AES 256) should be selected and used.

3.3 Application-network interface

With a CSDL an application can specify its requirements. However, the application requires an application programming interface (API) in order to inform the network about them. Such an API resides between applications and the network stack of a host. It provides all functions the application requires to organize communication via a network. Therefore, the API has to provide functions to announce an application as public service – as server application – or initiate a communication association – as client application – to a server application. For user-centric network orchestration both parts of the API must include requirements. Requirements stated for server announcements influence all connections, which are established later on. By defining such overall requirements a server can for example enforce encryption for all connections. As counterpart to the server side, clients are allowed to define requirements for initiating a communication association to a server application. But those are only valid for this single communication request. In addition to these application triggered API functions, the network stack must be able to give feedback to the application via additional API functions. On one hand, this feedback must include the requirements, which were fulfilled. On the other hand, the feedback is required to inform about failures during the lifetime of a communication association or a server application. Especially, changing link conditions in wireless scenarios are in need for such a dynamic feedback system.

Traditional APIs like the Berkeley Socket API, do not provide the possibility to define requirements. Furthermore, there are different functions for different services (datagram oriented API parts for UDP, stream oriented API parts for TCP). However, newer APIs generalize these functions and provide support for requirements and further support a feedback system [8].

4 Conceptual prospects for future network architectures

In this section we discuss the underlying functionalities for a future network architecture. After that, we briefly summarize candidate approaches that can be used to implement the presented use cases.

4.1 Discussion on architecture elements

The use cases presented in Section 2 show the need for an adaptation of application and network behavior in order to optimize the user perceived quality. That requires additional information of the network and of the application quality. Together with the application capabilities, a decision unit can enforce an appropriate adaptation strategy. Figure 5 depicts an generic example setup of functional building blocks for a network application.

The individual blocks are described in the following.

Decision Unit: The decision unit knows the capabilities of an application and influences the application behavior and network transport based on monitored data on network and application level. For the live streaming use case, this unit tries to maximize the transmitted video quality without causing congestion and packet loss. However, if packet loss appears although the quality has been reduced, it can try to cope with this problem by adding packet loss protection with FEC.

Application Adaptation: According to the application capabilities an adaptation of the application quality can be performed to minimize the impact of network problems, e.g., congestion. In case of live video streaming this can be the reduction of video quality, for Video-on-Demand streaming this can be achieved by initial buffering.

Network/Transport Encoding: Additional functionality on data transport or within the network may improve the user perceived quality. In case of a lossy link functions like packet retransmissions or FEC may prevent a bad application quality. However, this comes at the costs of retransmission delays and additional overhead. Further, this building block also comprises adaptations to the network like prioritization of the video flow if possible.

Network Transport Decoder: If packets are encoded, e.g., by FEC mechanisms, additional functionality is required to decode them.

Network Observer: Detailed information of the current network situation may be necessary to tune the data transport mechanisms. For instance, the additional FEC overhead may be adapted tightly focused to the packet loss to reduce the network overhead. In addition, the application quality can be adapted continuously with respect to the available network resources.

Application Observer: Application monitoring can complement network monitoring or be used as indicator of the current network conditions, if no specific network monitoring is available. For instance, the filling level of the VoD playback buffer can be used as such

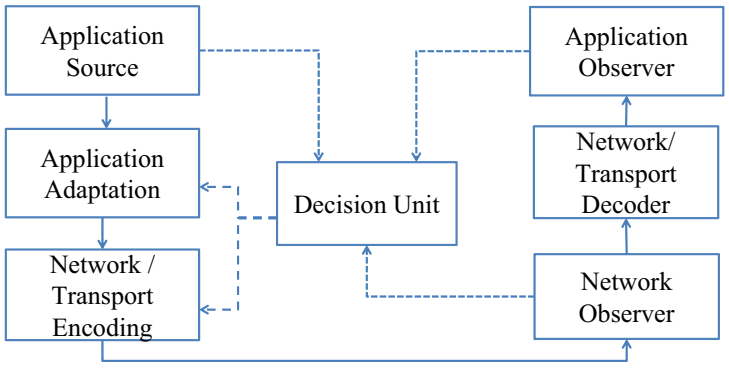

Fig. 5 Generic functional building blocks for application and network adaptation

an indicator. If it falls below a certain threshold, the application or network has to react in order to avoid video stalling.

In current networks, the functional blocks are mainly placed at end hosts. For example, Skype monitors network and application states from the end hosts. It reacts by adapting its video and voice quality and uses FEC mechanisms to protect the transmission. This complex mechanism introduces implementation overhead for the application. Although it could theoretically be reused, other applications interested in it would have to implement it on their own. In order to decrease the implementation effort for all applications, a common library for such mechanisms would be beneficial. From the architectural point of view, moving this functionality into the stack would be even more beneficial. It allows a network to place it not only in the stack of an end host but also on relay nodes within the network. Thus, the functionality can be used where it is needed without the overhead for the complete end-to-end transmission. For example, FEC and retransmission functionalities can be placed at lossy links within networks in order to reduce end-to-end error handling. Consequently, a user-centric network has to determine a place for its functionality. Its architecture has to take this aspect into account be allowing different placements.

We evaluate different architectures with the aspects mentioned above in mind. Due to the limitations of IP, we focus on future network architectures, that provide the possibility to implement the discussed use cases. Several suitable future network approaches are described in the following. We evaluate them with respect, e.g., to network overhead and the user perceived application quality.

4.2 Candidate future network approaches

In the following, different approaches for future networks are briefly revisited regarding the implementation of user-centric network orchestration and its requirements. These approaches include Service Oriented Network Architectures (SONATE), Forwarding on Gates (FoG), Autonomic Network Architecture (ANA), Forwarding directive, Association, and Rendezvous Architecture (FARA), and NEtlet-based Node Architecture (NENA).

4.2.1 Service oriented network architectures (SONATE)

User centric network orchestration can be realized using the Service-Oriented Network Architecture (SONATE) approach [17] where the suitable functional blocks are composed to constitute a protocol graph based on the applications, users, networks and, other requirements and constraints. Nodes communicate with each other using the constructed protocol graph. As the name indicates, SONATE is based on services which can be seen as a set of visible effects of an implementation of a protocol or mechanism, which is called a functional block (originally called building block). For example, error correction is a service of an implementation of a hamming code functional block. Each functional block is self-contained having a set of well-defined interfaces. Functional blocks provide services and communicate with each other through these interfaces. All of the visible effects, application requirements and, network and other constraints, are described using the communication service description language [18]. Selection of both services provided by the functional blocks and protocol graphs are done by using matching process, and the selection of the best service is done by employing a Multi-Criteria Decision Analysis (MCDA) method named Analytic Hierarchy Process (AHP).

4.2.2 Forwarding on gates (FoG)

FoG provides flexible and scalable support for networks composed dynamically by functional blocks. Its architecture separates forwarding from routing. The forwarding is based on an index-based forwarding concept, which supports functional blocks. In general, index-based forwarding uses a stack of indices to describe a route between two nodes in a network. This stack is processed index by index, while each index defines the next link between two hops of the network. The indices have to be unique per hop scope and not for the entire network, since they refer only to local interfaces. Such forwarding approaches were already proven to reduce the amount of required entries in the routing information base in inter-network scenarios [7].

As required for user-centric network orchestration, FoG's forwarding and routing operate on functional blocks. It distinguishes between two types of functional blocks:

- *Gates* represent arbitrary functionality such as encryption, FEC, and retransmission. They take packets as input for the function. Gates have just one input and one output.
- *Forwarding nodes* are limited to multiplexing and can have multiple outputs. Each output forwards a packet to the next gate. Forwarding nodes implement the index-based forwarding by using *gate numbers*, which – in the simplest case – name the output directly. Thus, gate numbers can represent an index for the next functional block to which a packet has to be transferred.

FoG virtualizes each physical link between two hosts by at least one unicast gate, which represents a unicast transmission from one host to the next one. Multiple gates can coexist at the same physical link. These gates may differ in their internal functionalities and transmission attributes, e.g., QoS guarantees.

Communication paths between two interacting applications are implemented as chains of gates in a FoG network. Such a chain defines the functions and the order, in which they are executed. They are calculated by FoGs routing, which takes over the tasks of the selection and composition process. These tasks are implemented in a distributed fashion by the *incremental routing process* of FoG. It combines the orchestrated functional blocks from multiple nodes (relay nodes and end hosts) in an "end-to-end" chain. The choreography between these individual parts is organized with a FoG-specific signaling protocol. FoGs forwarding contains the functional blocks and implements the decisions of the routing. For QoS supporting routing, FoG can utilize a hierarchical approach called "Hierarchical Routing Management" [26].

4.2.3 Autonomic network architecture (ANA) framework

The ANA framework [2] uses functional blocks, called "bricks", which implement the requirements of an application. A route in the ANA framework consists of several concatenated bricks. This approach is very similar to the FoG concept. However, ANA lacks the support for QoS and does not provide the flexibility for describing application requirements like FoG does. Furthermore, ANA doesn't support the instantiation of functions in the network; it focuses on the network stack on the end hosts.

4.2.4 Forwarding directive, association, and rendezvous architecture (FARA)

The concept of the NewArch project is an additional alternative approach. The proposed solution uses a model, which is called FARA [3]. In FARA an association focuses on

unicast transmissions, only. FARA does not support the variety of transmission requirements, which are needed for the system design proposed in this work. Especially, QoS attributes are not considered in the FARA concept.

4.2.5 NEtlet-based node architecture (NENA)

Orchestration approaches differ according to when the composition is performed: design time, deployment time, or runtime. In this respect, the NEtlet-based Node Architecture (NENA) [25] can be understood as a design time composition approach, where the composition of functionalities is performed during design time by the application developer. A software can assist this composition process as different protocol graphs are constructed for different platforms. However, the selection of a suitable protocol graph based on quality parameters is performed during runtime.

5 Demonstrating the video streaming use case with FoG

In order to show the applicability of our approach we implemented the VoD and the HD Live Streaming use-cases in the FoG architecture. In this section, we present details on the implementation of the use cases and summarize the lessons learned by our implementation.

5.1 Implementation

As mentioned in Section 4.2.2, FoG is using functional blocks, called gates, which represent the functionality of a network. The FoG emulator [15] has been extended by gates for the video use cases and by a video viewer application. The application uses the Communication Service Description Language (CSDL) to define its requirements. The FoG network stack is informed about the requirements via the network-application interface GAPI. Furthermore, the FoG stack uses the GAPI to inform the application about events in the network such as failures or incoming connections. FoG reacts on the requirements from applications by selecting and composing appropriate gates. This process is influenced by the capability and the policy of the network. The configuration of the gates is derived from all these influencing factors. Depending on these factors, FoG has several options for combining gates. First, FoG can reserve capacity in the network to satisfy the requirements. Second, FoG can use a best-effort route and adapt dynamically the gates on the end hosts to the situation in the network. And third, FoG can combine both. Which gates are chosen is described for each use case in the next sections.

The overall process of deriving an end-to-end chain of functional blocks is done in a distributed way. Multiple nodes along the path are contributing small parts of the chain. On the one hand, this avoids a central point of failure and enables functions that are only supported by a small number of nodes. On the other hand, nodes contributing to the chain (not all nodes have to) have to have a larger overview over the network in order to handle routing issues. If a node detects conflicting requirements or requirements, which cannot be fulfilled with the available resources, it reports an error to the requesting application. If the error is detected at a node, which is not an end hosts, the error is signaled to the end hosts, which, in turn, reports it to the application via the GAPI. For the error handling, for the signaling of the requirements, and the organization of the orchestration, a FoG-specific signaling protocol is used. It combines elements known from traditional resource reservation protocols with the new possibilities of the FoG route definition.

5.1.1 Video-on-demand scenario

The first use-case focuses on the on-demand distribution of videos with a defined length, like YouTube videos. The application requests a video from a source with a defined quality. The user wants to see the video in exactly this quality and neither the video application nor the network should reduce it. Consequently, FoG has to transport a fixed amount of video data through its network. As long as the bandwidth is large enough for transferring the video with a higher data rate than it is required for the video, the video player can start immediately the video play out.

However, if there is not enough bandwidth available, the video will stall. Such a stall has a significant negative impact on the QoE, since it degrades the QoE more quickly than an initial start delay of the play out as discussed in Section 2.2. FoG reacts on this by adding a gate, which buffers the video. The buffering delay is influenced by the available average bandwidth and the required data rate of the video. Figure 6 is showing the setup on the right hand side. The green (or darker) boxes are forwarding nodes and the arrows between them are gates. The white box marks the connection end point at the video viewer. The property view on the lower part of the figure shows the configuration parameter of the video decoder gate. In the demo, the user can select the video quality and start the transmission. The network will start buffering until the probability, that stalling will occur, falls below a threshold. Then the play out of the video starts and will continue without stalling until the end of the video. The user has the ability to cancel buffering at any time and start the play out immediately. However, the implementation shows that after such a cancellation the buffer runs empty and the video stalls.

5.1.2 Live video streaming

In contrast to the videos in the on-demand use case, live videos do not have a defined duration. Therefore, buffering to avoid stalling is of limited use. If the available bandwidth is not sufficient to transmit the video stream, the quality of the stream has to be adjusted.

In this use case, the video application requests the video stream with best-effort requirements. However, the network should optimize QoE for the user. FoG reacts on this by transferring the video stream in a best-effort manner. To increase the QoE, it creates gates taking advantages of the SVC coding of the video to deal with varying bandwidth. One SVC-splitter gate is created at the video source host and a measurement gate is introduced at the end host. FoGs routing ensures that the video stream is forwarded through both gates. The measurement gate is reporting its measurements to the SVC-splitter gate, which reacts by dropping some of the SVC stream content depending on the received bandwidth. In contrast to random packet drop, this ensures a controlled QoE degradation at the source. However, packets might get dropped within the network caused by bit failures or full queues. As recommended in Section 2, FoG introduces additional gates adding forward error correction (FEC) information to a video stream to cope with such drops. At the sender, one gate is created doing FEC encoding. At the receiver, one gate reconstructing the original packets via FEC decoding is added. As for the SVC gate and the measurement gate before, FoG forwards the video stream through these gates. Figure 7 shows the setup of the gates for a live stream. Since there are more gates for the direction towards the white connection end point, they are arranged with one additional forwarding node situated in the upper left part of the gate structure.

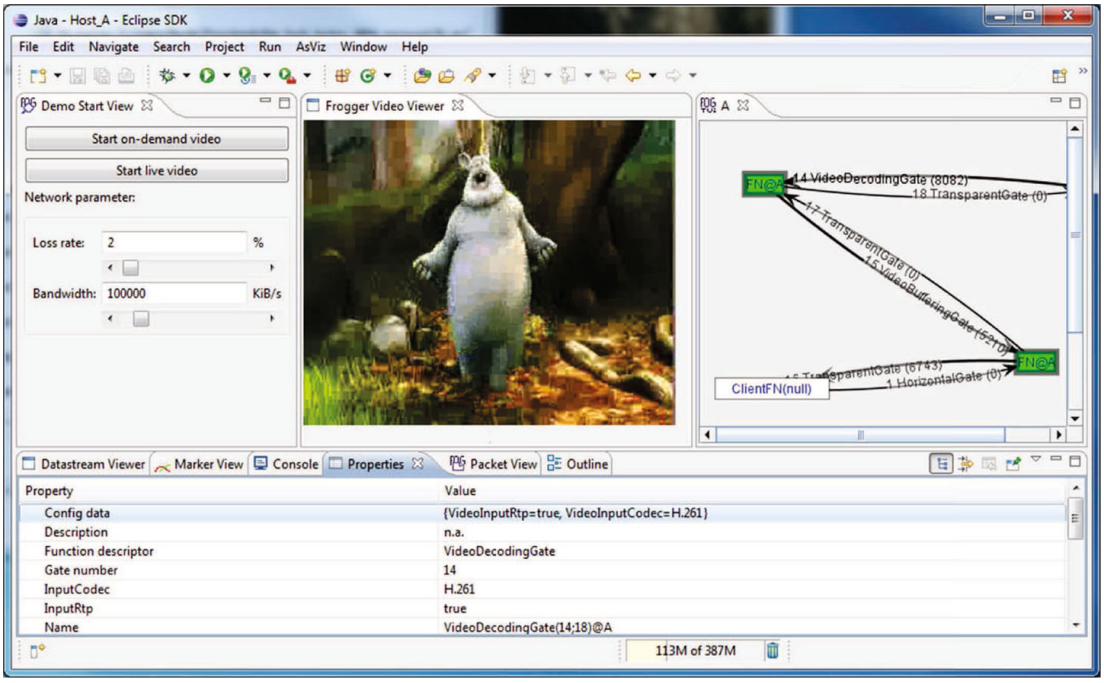

Fig. 6 Screenshot of the implementation of the VoD use-case with FoG

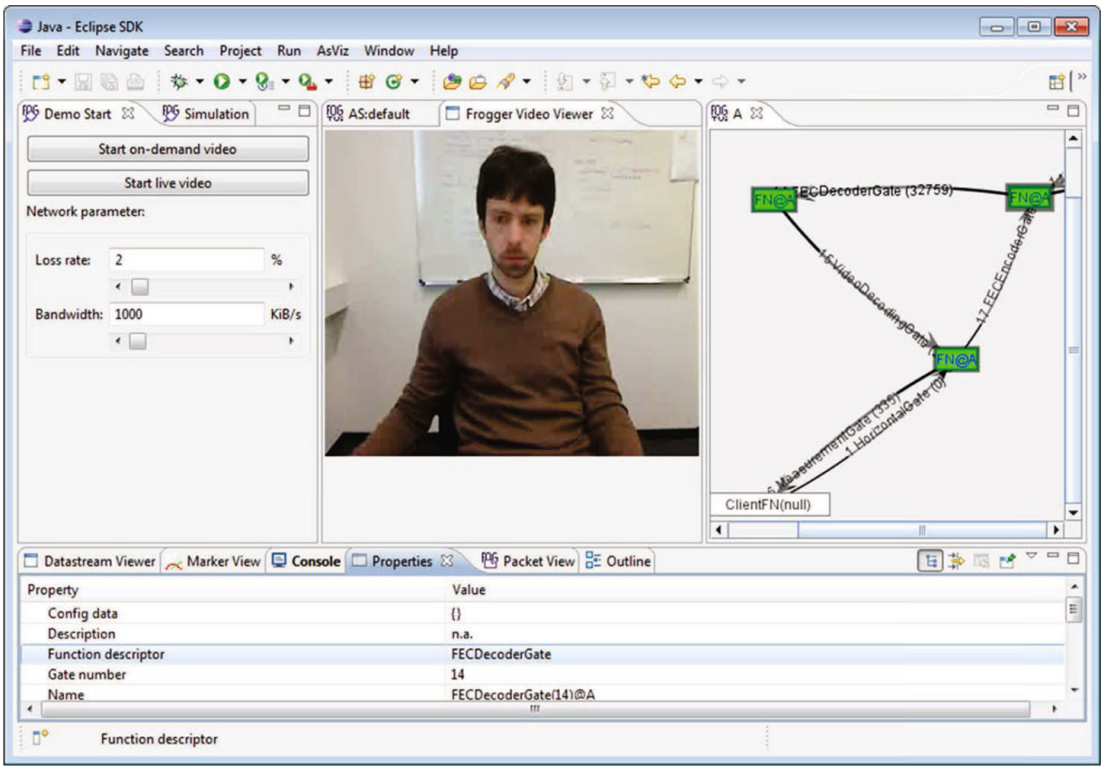

Fig. 7 Screenshot of the implementation of the live streaming use-case with FoG

5.2 Lessons learned

Our implementation shows the orchestration of functional blocks, called gates, depending on requirements for transmission defined by applications. The goal of the orchestration is the maximization of the QoE for the user, while taking into account the available resources and functionalities in the network. The demo focuses on two scenarios dealing with video transmission. First, on-demand videos are transmitted with an average bandwidth. To avoid stalling, a gate for buffering is introduced. Second, live video streaming uses gates for SVC-splitting to deal with variable bandwidth and FEC gates to deal with packet loss. Both scenarios show the high impact of user-centric orchestration on the achieved QoE. Furthermore, we demonstrate that it is feasible to implement such a system with Future Internet approaches using functional blocks. In special, the implementation shows the feasibility of FoG to handle the requirements of a user-centric network orchestration.

However, the implemented logic is highly related to the use cases. A generic method for defining the relationship between functional blocks, their parameters and requirements could not be implemented. It is up to future works to define such a generic method.

6 Conclusions and future work

This article has discussed network orchestration based on requirements and parameters communicated between users, applications and networks. To illustrate the importance and impact of such parameters, three use cases that relate user, application and network requirements to each other were discussed. For each use case, we developed guidelines for a successful network orchestration. Our main focus are multimedia services, since they are seen as crucial applications in the future. The use cases ranged from scalable and on-demand streaming to user authentication as used for payed multimedia services like Sky Go. The related wide range of services demands for a flexible network orchestration approach. We reviewed the properties of such approaches in terms of general service composition issues, such as requirement description, selection of functional blocks, and application-network interfaces, followed by an in-depth description of several future network approaches like the Service-Oriented Network Architecture (SONATE) and Forwarding on Gates (FoG). Both approaches are future network stacks, which can be run on top of (and thus compatible with) existing networks. We implemented the discussed video streaming use cases in the FoG architecture in order to show the feasibility of our investigations.

Future work will be based on the implementation of the presented concepts and demonstrate their feasibility for additional use cases. In addition, a comprehensive evaluation of different future network architectures and specific implementations has to be performed with respect to user- and network-centric evaluation metrics. For that, focus has to be put also onto dynamic composition based on a close, timely interaction between the functional blocks involved in the network orchestration.

Acknowledgments This work was funded by the Federal Ministry of Education and Research of the Federal Republic of Germany (support code 01BK0917 and 01BK0935, G-Lab) and by the European FP7 Network of Excellence “Euro-NF” through the Specific Joint Research Project “PRUNO”. The authors alone are responsible for the content of the paper.

References

1. Abboud O, Hohlfeld O, Hoßfeld T, Tran-Gia P, Zinner T (2010) Towards QoE management for scalable video streaming. In: 21th ITC specialist seminar on multimedia applications - traffic, Performance and QoE. Miyazaki
2. Bouabene G, Jelger CS, Keller A, May M, Schmid S, Tschudin CF (2012) The autonomic network architecture. doi:10.1109/JSAC.2010.100102
3. Braden R, Clark D, Chiappa N, Faber T, Falk A, Katabi D, Handley M, Kulik J, Pingali V, Sollins K, Wroclawski J, Yang X (2003) Newarch: Future generation internet architecture. In: Technical report
4. Brunet D, Vrscay ER, Wang Z (2011) A class of image metrics based on the structural similarity quality index. In: Kamel M, Campilho AC (eds) ICIAR (1) of Lecture notes in computer science, vol 6753. Springer, pp 100–110
5. Egger S, Hoßfeld T, Hoßfeld T, Schatz R (2012) Time is bandwidth? narrowing the gap between subjective time perception and quality of experience. In: Proceedings of the IEEE international conference on communications (ICC), Ottawa, p 2012
6. Engelke U, Kusuma M, Zepernick H-J, Caldera M (2009) Reduced-reference metric design for objective perceptual quality assessment in wireless imaging. Sig Processing-Image Commun, 525–547
7. Ganichev I, Godfey PB, Shenker S, Stoica I (2009) Pathlet routing. In: Proceedings of SIGCOMM 2009, Barcelona
8. G-Lab Special Interest Group Functional Composition (2011) GAPI:A g-lab application-to-network interface. In: Proceedings of 11th Wuerzburg Workshop on IP: Joint ITG and Euro-NF Workshop EuroView2011. Wuerzburg
9. Hellwagner H, Kofler I, Kuschnig R, Ransburg M (2008) Design options and comparison of in-network h.264/svc adaptation. J Vis Commun Image Represent 19(8):529–542. doi:10.1016/j.jvcir.2008.07.004
10. Hayashi T, Yamagishi K (2008) Parametric packet-layer model for monitoring video quality of iptv services. In: Proceedings of the IEEE international conference communication (ICC 2008), pp 110–114
11. Hoßfeld T, Binzenhöfer A (2008) Analysis of skype voIP traffic in UMTS: End-to-end qos and urements. Comput Netw 52(3):650–666. doi:10.1016/j.comnet.2007.10.008
12. Hoßfeld T, Fiedler M, Zinner T (2011) The QoE provisioning-delivery-hysteresis and its importance for service provisioning in the future internet. In: Proceedings of the 7th conference on next generation internet networks (NGI). Kaiserslautern, Germany
13. Hoßfeld T, Liers F, Schatz R, Staehle B, Staehle D, Volkert T, Wamser F Quality of experience management for youTube: clouds, foG and the aquareYoum, PIK - Praxis der informationverarbeitung und -kommunikation (PIK)
14. Hoßfeld T, Plissonneau L, Biersack E, Schatz R (2012) Internet video delivery in YouTube: From traffic measurements to quality of experience. In: Ernst Biersack MM, Callegari C (eds) Data traffic monitoring and analysis: from measurement, classification and anomaly detection to quality of experience springers computer communications and networks series
15. Homepage FoGSiEm on GitHub (April 2013). https://github.com/ICS-TU-Ilmenau/fog/wiki
16. ITU-T Recommendation (2010) H.264: advanced video coding for generic audiovisual services
17. Khondoker R, Mueller P, Reuther B, Siddiqui A, Schwerdel D (2010) Describing and selecting communication services in a service oriented network architecture. In: Proceedings of the 2010 ITU-T Kleidoscope: beyond the internet?-innovations for future networks and services. India, pp 1–8
18. Khondoker R, Mueller P, Veith EM (2011) A description language for communication services of future network architectures In: Proceedings of the international conference on the network of the future, pp 68–75
19. Krishnan SS, Sitaraman RK (2012) Video stream quality impacts viewer behavior: Inferring causality using quasi-experimental designs. In Proceedings of the 2012 ACM conference on internet measurement conference. ACM, pp 211–224
20. Lakshman TV, Sabnani K, Woo T Softrouter:An open extensible platform for tomorrows internet services, Bell Labs, Alcatel-Lucent.
21. Liers F, Volkert T, Mitschele-Thiel A (2012) The forwarding on gates architecture: Merging intserv and diffserv. In: Proceedings of international conference on advances in future internet (AFIN), p 2012
22. Lorentzen C, Fiedler M, Johnson H, Jorstad I, Shaikh J (2010) On user perception of web login - a study on qoe in the context of security. In: Proceedings of Australasian telecommunication networks and applications conference, Auckland, pp 84–89
23. Lorentzen C, Fiedler M, Lorentzen P (2011) Decisive factors for quality of experience of openid authentication using eap, Advances in electronics and telecommunications. Spec issue Recent Adv Teletraffic 2(3):79–87
24. Luby M (2002) LT codes. In: 43rd annual IEEE symposium on foundations of computer science

25. Martin D, Völker L, Zitterbart M (2011) A flexible framework for future internet design, assessment and operation. Comput Netw 55(4):910–918
26. Volkert T, Osdoba M, Becke M, Mitschele-Thiel A (2013) Multipath video streaming based on hierarchical routing management. In: Proceedings of 27th IEEE international conference on advanced information networking and applications (AINA)IEEE. Barcelona

Thomas Zinner studied computer science at the University of Wrzburg, Germany. He finished his PhD on performance modelling of QoE-aware multipath video transmission in the future Internet in 2012. He is heading now the NGN research group Next Generation Networks at the Chair of Communication Networks in Wrzburg. His main research interests cover video streaming techniques, implementation of QoE-awareness within networks, software-defined Networking (SDN) and network virtualization, network function virtualization and the benefits of cloudification, as well as the performance assessment of these technologies and architectures.

Tobias Hossfeld is heading the FIA research group "Future Internet Applications & Overlays" at the Chair of Communication Networks in Wrzburg. He finished his PhD in 2009 and his professorial thesis (habilitation) "Modeling and Analysis of Internet Applications and Services" in 2013. He has been visiting senior researcher at FTW in Vienna with a focus on Quality of Experience research. His main research interests cover social networks, crowdsourcing platforms, content distribution networks and clouds, as well as investigations on Quality of Experience for Internet applications like Skype, YouTube, Web Browsing or cloud applications in general.

Markus Fiedler received his doctoral degree in electrical engineering/ICT from Universitt des Saarlandes, Saarbrcken, Germany, in 1998. Since then he has been with Blekinge Institute of Technology, Karlskrona, Sweden, holding a Docent degree in telecommunication systems since 2006. Within the School of Computing, he performs and supervises research on quality of experience, seamless communications, network virtualization, service chains, and networks of the future (NF). He is leading and participating in several national and European projects. He is serving on the Steering Board of the European Network of Excellence Euro-NF and coordinating its specific joint research projects.

Florian Liers studied computer science at the Technical University of Ilmenau with a focus on optimization and mobile communication networks. He finished his Diploma about optimized mobility support in 2005 and continued his research at the Integrated Communication Systems group. He worked on resource optimization in UMTS and distributed systems for embedded devices. Since 2009 he is leading the research project Forwarding on Gates. His current interests are networks architectures enabling broader optimization possibilities, scalability in inter-networks, recursive layers, flexible network protocols and networks based on functional blocks.

Thomas Volkert studied computer science at the technical university of Ilmenau. He got his diploma degree in 2007 and continued his research as member of the "Integrated Communication Systems Group" at the Technical University of Ilmenau. Within the group, he worked for the projects: "Empowered Network Management" (ENERGy), "Mobile Satellite Communication in the Ka-band" (MoSaKa) and "Forwarding on Gates" (FoG). His research focus is on distributed routing management, QoS provisioning and multimedia streaming. Furthermore, he is developer of the open source video conferencing tool "Homer Conferencing".

Rahamatullah Khondoker completed his first M.Sc. from the Department of Information and Communication Engineering, Islamic University, Kushtia, Bangladesh in 2006, and his second M.Sc. from the department of Computer Science, University of Bremen, Germany, in 2009. From January 2010, he has been working towards his PhD degree on "Description and Selection of Communication Services for Service Oriented Network Architectures" at the University of Kaiserslautern in Germany. He was the winner of IMS Service Composition from Ericsson, Germany in the year 2008. Moreover, he was awarded from the FIA Research Roadmap group in the Future Internet Event held at Poznan in Poland (October 2011) by providing the most thought-provoking presentation of the research topic that was missed in the previous version of the research roadmap and was included in the next version. Currently, he is affiliated with the department of Mobile Systems at the Fraunhofer Institute for Secure Information Technology located in Darmstadt, Germany. His current research interests are the security and trust aspects of Future Internet Architectures.

Raimund Schatz is a Senior Researcher at FTW, where he currently manages the research projects (ACE, ACE 2.0 and ACE 3) on Quality-of-Experience (QoE) assessment and modelling for mobile and wireline broadband services. Furthermore, he is actively involved in the CELTIC project QuEEN, COST IC1003 Qualinet and in the FP7 STREP project Optiband. Dr. Schatz holds an Msc. in Telematics (Graz University of Technology), a PhD in Informatics (Vienna University of Technology) as well as an MBA and an Msc. in International Finance and Management (both Open University, UK). Being an active member of ACM and IEEE, Dr. Schatz has more than 80 publications in the fields of HCI, Pervasive Computing and Quality of Experience.