Prospects for Realizing User-Centric Network Orchestration: FEC-protected SVC Streaming

Thomas Zinner[1], Florian Liers[3], Tobias Hoßfeld[1], Dirk Rauscher[1], Bernd Reuther[2], Daniel Günther[2], Thomas Volkert[3], Markus Fiedler[4]

[1]University of Würzburg, Institute of Computer Science,Würzburg, Germany, [zinner;hossfeld;rauscher]@informatik.uni-wuerzburg.de

[2]Technical University of Kaiserslautern, Kaiserslautern, Germany, [reuther;guenther]@informatik.uni-kl.de

[3]Technical University of Ilmenau, Ilmenau, Germany, [florian.liers;thomas.volkert]@tu-ilmenau.de

[4]Blekinge Tekniska Högskola, Karlskrona, Sweden, markus.fiedler@bth.se

I. Introduction

To overcome the current limitations of the Internet, several different approaches are existing which can be mainly distinguished into network-aware applications and application-aware networks. Bringing together both approaches, network-application interfaces are to be deployed to realize **smart applications** as well as **smart mediation** of application data through the Internet. Via this interface, the application and the mediation network can directly communicate with each other and exchange information adequately, such that the end user experiences a good quality. While smart mediation includes resource management as well as traffic management to optimally utilize network resources and to deliver the contents according to the application requirements, smart applications react dynamically on the current network situation for optimal Quality of Experience (QoE), e.g., by reducing application requirements. In this work, we consider **live streaming services** based on connectionless transport protocols like video conferencing and IPTV with high video quality playback demands.

The main goal of this work is to optimize the QoE for live video streaming by real-time adaptation of the required video bitrate and thus the video quality to the current network situation (i.e., available bandwidth or unreliable links with packet losses). Insufficient network resources, e.g., less available bandwidth than required or packet loss, result in a strong impairment of the video service [4]. In our case, this results in video decoding errors and frame drops. A smart way to reduce the required bandwidth and adjust the video transmission is provided by the scalable extension of the video codec H.264/AVC [1]. A Scalable Video Codec (SVC) stream can be adjusted by the service provider, the network provider or at the customer's device. While the service provider can save storage compared to single streams of different qualities, the network provider can adjust the required bandwidth and avoid congestion, and the end device may select the playback quality according to its decoding capabilities in terms of computational power.

Current research on the impact of packet loss on the user perceived quality indicates a strong impairment of the video already at small packet loss rates of less than 2% [4]. This motivates mechanisms, like Forward Error Correction (FEC) methods, which can protect a video stream against packet loss. These techniques provide means to correct corrupted or lost packets at the cost of additional overhead resulting in a trade-off between bandwidth and packet loss protection. In order to optimize the user perceived quality of a scalable video stream for the current network conditions, it is necessary to know the bandwidth requirements of the different video quality layers and their impact on the user perceived quality. Further, the influence of the current network situation (e.g., packet loss and bandwidth) has to be measured and taken into account for deciding a) which SVC layers to transport and b) which FEC method with which parameters to use. Based on this information, it is possible to provide an improved QoE for the current network conditions. However, it is not clear where to place this measurement and decision functionality (within the network or within the application). There are different prospects for placing this functionality and for which information is exchanged via the network-application interface. Functionality on application level adds complexity to the application and requires information from the network, functionality in the network stack requires information from the application. In order to find an appropriate placement of the functionality and to define which parameters have to be exchanged, the impact of the different prospects on the user perceived quality and the corresponding costs has to be evaluated carefully.

This work provides a first step towards an evaluation of the different prospects for a video streaming system using scalable video coding and forward error correction mechanisms. The paper is structured as follows. Section 2 briefly discusses the scalable video codec H.264/SVC and forward error correction with Luby codes [2]. Further we discuss the benefits of a combination of both approaches. The required functional blocks and their interaction, as well as an implementation concept are discussed in Section 3. A brief discussion on future work and enhancements is given in Section 4.

II. FEC-protected SVC Streaming

This section details scalable video coding, forward error correction and how to combine both approaches.

A. Scalable Videostreaming

The SVC extension enables the encoding of a video file at different qualities within the same layered bit stream. This includes besides different resolutions also different frequencies (frames displayed per second) and different image qualities w.r.t. Signal-to-Noise Ratio (SNR).

These three dimensions are denoted to as spatial, temporal and quality scalability. Figure 1 gives an example of different possible scalabilities for a video file. The left "subcube" at the bottom is the base layer, which is necessary to play the video file, here with CIF resolution, 15 Hz frame-rate, and quality Q0. Based on this layer, different additional enhancement layers permit a better video experience with a higher resolution, better SNR or higher frame rate, respectively. If the available bandwidth is not enough 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 the user's context, has to be provided as indicated by Figure 2. Otherwise the user will not accept the video service. Higher layers will increase the QoE further, but also require a higher bandwidth. We implemented a SVC splitting unit which allows adjusting the quality of a SVC stream and thus reducing the required bandwidth.

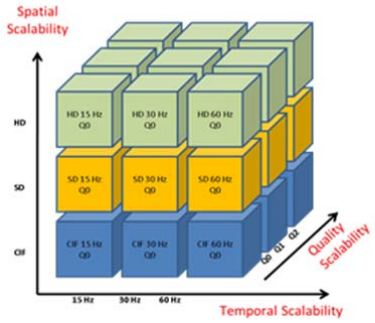

Figure 1: SVC cube

Figure 2: QoE management for SVC

B. Rateless FEC with Luby Codes

Luby codes are the first practical implementation of Digital Fountain codes and were proposed by Michael Luby. They provide the basis for Raptor Codes, which are widely used in commercial applications such as Digital Video Broadcasting [3].

The encoding mechanism creates linear combinations using XOR operations out of k source symbols. The amount of combined symbols and their selection is chosen randomly for each encoding symbol. Thus this enables the production of a quasi-unlimited amount of encoding symbols from a finite set of source symbols. The decoder requires any number $k+\varepsilon$ encoded of encoded symbols to decode the source symbols, whereas ε is the required overhead. Thus, in order to protect the video stream against a packet loss rate pl, the sender has to transmit more than $(k+\varepsilon)/(1-pl)$ encoding symbols. Thus, enough encoding symbols are available to decode the message. In case of higher packet loss rates, this number has to be increased accordingly. The overhead and also the decoding complexity depend on the decoding algorithm, the number of source symbols and the parameters of the random distribution.

We implemented the basic encoding mechanisms and two decoding mechanisms, namely the Ripple and the Gaussian elimination decoding algorithm. Whereas the Ripple algorithm requires a larger decoding overhead than the Gaussian algorithm, it outperforms the Gaussian in terms of complexity and thus in decoding time. Generally, for small message sizes of, e.g., 100 and 500 symbols, the Gaussian Elimination algorithm is used, whereas for larger message sizes of 1000 symbols and more, while for large messages of 1000 symbols, the Ripple decoder is preferred, due to the differences in processing time.

C. QoE Provisioning-Delivery Hysteresis

With the described implementations we extended the work on the QoE Provisioning-Delivery Hysteresis (PDH) presented in [5]. The PDH is observed for different scenarios like VoIP, web surfing and video streaming and implies that a controlled reduction of the application layer throughput by e.g., speech codecs with lower quality, affects the QoE to a much lesser extent than the uncontrolled reduction through information loss. By combining SVC and FEC capabilities, controlled quality reduction can be combined with error protection. Thus, a good streaming quality can also be assured in case of packet loss whereas the network is balanced simultaneously.

The results of our study are depicted in Figure 3. The x-axis denotes the goodput, which is the application perceived throughput and is 1 if the maximum quality can be streamed without loss. The

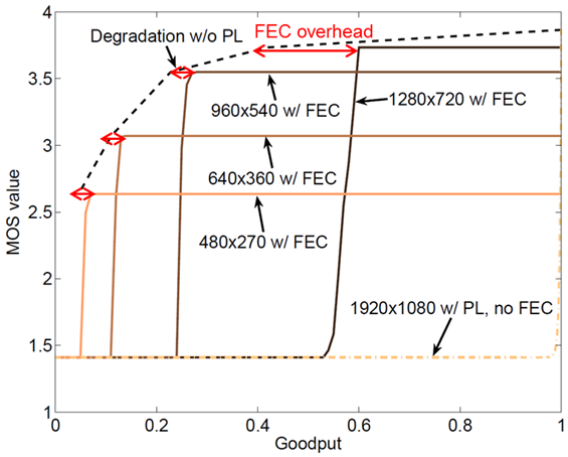

Figure 3: Extension of the PDH

y-axis denotes the corresponding user-perceived quality which is computed with full reference metrics as presented in [6].

The goodput can be reduced either in case of packet loss, as indicated by the case of streaming a video in 1920x1080 resolution with no FEC. For this case, the user perceived quality decreases rapidly and gets minimal for values of ~2% packet loss, i.e., a goodput of 0.98. In case of degradation w/o packet loss, we study the streaming of the video clip in lower resolutions in our example. As can be seen in the figure, the user perceived quality decreases much slower and keeps on a tolerable value for the lowest resolution, 480x270 pixels. Accordingly, the goodput is around ~15% compared to the maximum resolution. The other cases denote controlled service degradation with FEC mechanisms. As an example, streaming a video with a resolution of 1280x720 with FEC is considered. Switching to this resolution reduces the goodput of ~ 60%. We add a FEC mechanism which adds an overhead $\varepsilon \approx 20\%$, resulting in a goodput of ~60%. This is illustrated by the red arrow. However, by transmitting more symbols it is now possible to cope with packet loss rates up to ~40% 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, as illustrated by the results in Figure 3.

III. Prospects for Realization in the Future Internet

This section details the functional blocks needed for a video streaming service using scalable video coding and forward error correction mechanisms and discusses where to place them in a Future Internet Architecture.

A. Functional Elements: Stream Splitter, FEC Encoder, FEC Decoder, Monitoring Unit and Decision Unit

The functionalities to enhance a video streaming service using SVC with FEC capabilities are depicted in Figure 4:

- **SVC Splitter**, an entity which can adjust the SVC stream, e.g. with respect to the user perceived quality, cf. [6].
- The **FEC Encoder** encodes the stream according to given parameters.
- The **FEC Decoder** decodes the stream.
- **Network Observer** denotes an entity or an interface which provides information on the current network conditions. This includes information like available bandwidth, grade of congestion and packet loss
- The **Decision Unit** implements the extended PDH and maximizes the QoE of the given SVC video stream and also

for the current network conditions. The resulting stream and FEC parameters are then passed to the corresponding functional blocks.

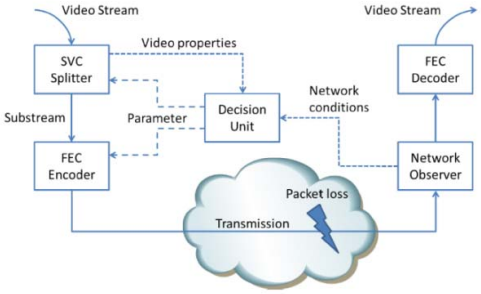

Figure 4: Functional blocks for the investigated video streaming architecture

B. Implementation Possiblities with GAPI and Forwarding on Gates/Sonate

A common implementation of such a system for today's Internet would base on UDP/IP and just consider the end hosts. The SVC Splitter, FEC encoder and the decision unit would be placed on the sender host and the remaining two entities on the receiver side. While this solution is easy to deploy, the implementation overhead is large. In special, the network observer component has to be implemented, although it is not directly related to the problem itself. Moreover, this component is even interesting for other applications. But such applications would have to implement it as well, despite the fact that these common components could be reused theoretically. In order to decrease the implementation effort for all applications, a common library for this component would be beneficial.

From the architectural point of view, moving this functionality into the stack would be even more beneficial. It enables the stack itself to access the measurement data collected by this component. In order to enable the application to access the measurements, a suitable application programming interface is needed. Furthermore, the architectural view broadens the problem to multiple such communications running in parallel. In case of limited resources the network has to decide how to distribute the resources to the video transmissions. More abstract, that is a question of how to distribute the QoE among the transmissions. Should the bandwidth be used for one video transmission in order to maximize its QoE or should the bandwidth be equally distributed in order to have fair QoE for all? But an equal distribution might not be optimal due to the discrete steps and the overhead caused by a special bandwidth, as shown in Figure 4. As a solution, the application should tell the network more details about its streams by defining requirements.

The G-Lab API (GAPI) [7] provides access to measurement data and enables the application to specify functional and non-functional requirements, like "bandwidth >= 150kbit/s and loss rate < 2%". If these requirements cannot be fulfilled (e.g. indicated by the network observer, which is situated in the stack) the stack will inform the applications via events. Based on the events, the decision unit can adjust the sender's parameters. Different to the situation without the application requirements, the network now has a clue about the minimum requirements of each stream.

With dynamic stacks supporting functional composition and GAPI (like e.g. "Forwarding on Gates" (FoG) [8] or SONATE [10]) it might even be possible to move all other components from the application to the stack. The main benefit would be that the decision unit is than a part of the stack and would give the network a better handle to adjust the amount of used bandwidth per stream.

Furthermore, Future Internets supporting functions within the network might use the FEC en-/decoder as dynamic functional block for securing lossy links. FoG provides this opportunity. As shown in a demo [9], FoG can integrate such functional blocks in a scalable way. Furthermore, the re-use of such blocks for multiple transmissions would increase the bandwidth handled by one entity. This results in a higher number of source symbols which can be protected with the presented FEC mechanism more efficiently.

IV. Future Steps

Future steps concerning the proposed streaming service are twofold. First, the mechanisms itself is currently implemented and evaluated for static network conditions. This has to be extended to deal with dynamic network conditions through an online adaptation of the SVC splitter and the FEC encoder. Further, guidelines on the accuracy and precision of the network measurements have to be investigated in detail. Second, the current version of the SVC splitter and FEC en-/decoder are implemented in the common way for today's Internet. We would like to integrate them to the FoG and SONATE approach in order to investigate the additional benefits from re-using them and to analyze the impact of the requirements on the behavior of parallel streams.

Acknowledgements

This work was funded by the Federal Ministry of Education and Research of the Federal Republic of Germany (support code 01 BK 0806, G-Lab) and by the European FP7 Network of Excellence "Euro-NF" through the Specific Joint Research Project "PRUNO".

V. References

[1] ITU-T Recommendation H.264: Advanced video coding for generic audiovisual services, Telecommunication Standardization Sector of ITU, March 2010.

[2] M. Luby: LT Codes. 43rd Annual IEEE Symposium on Foundations of Computer Science, November 2002.

[3] DVB TM Joint Task Force on Upper Layer Forward Error Correction: Digital Video Broadcasting (DVB); Upper Layer Forward Error Correction in DVB. DVB Document A148, March 2010.

[4] T. Zinner, O. Abboud, O. Hohlfeld, T. Hoßfeld, P. Tran-Gia: Towards QoE Management for Scalable Video Streaming. 21th ITC Specialist Seminar on Multimedia Applications – Traffic, Performance and QoE, March 2010.

[5] T. Hoßfeld, M. Fiedler, T. Zinner: The QoE Provisioning-Delivery-Hysteresis and Its Importance for Service Provisioning in the Future Internet. Proceedings of the 7th Conference on Next Generation Internet Networks (NGI), June 2011.

[6] T. Zinner, O. Hohlfeld, O. Abboud, T. Hoßfeld: Impact of Frame Rate and Resolution on Objective QoE Metrics. 2nd International Workshop on Quality of Multimedia Experience, June 2010.

[7] F. Liers, T. Volkert, D. Martin, H. Backhaus, H. Wippel, E. Veith, A. A. Siddiqui, R. Khondoker: GAPI: A G-Lab Application-to-Network Interface, 11th Würzburg Workshop on IP (EuroView), Germany, Würzburg, August 2011.

[8] F. Liers, T. Volkert, A. Mitschele-Thiel: A Flexible Abstraction for the Future Internet, 8th Würzburg Workshop on IP (EuroView), Würzburg, Germany, August 2008.

[9] F. Liers, T. Volkert, A. Mitschele-Thiel: Scalable Network Support for Application Requirements with Forwarding on Gates, EuroView2011, Würzburg, Germany, August 2011.

[10] P. Müller, B. Reuther, M. Hillenbrand: Future Internet: A Service-Oriented Approach – SONATE, EuroView 2007, Würzburg, Germany, July 2007