**MULTIMEDIA COMMUNICATIONS TECHNICAL COMMITTEE**
***http://www.comsoc.org/~mmc***

# *MMTC Communications - Frontiers*

**Vol. 11, No. 2, March 2016**

## CONTENTS

# Free Viewpoint Video Streaming: Concepts, Techniques and Challenges

*Árpád Huszák*
*Budapest University of Technology and Economics, Budapest, Hungary*
*Multimedia Networks and Services Laboratory*
*huszak@hit.bme.hu*

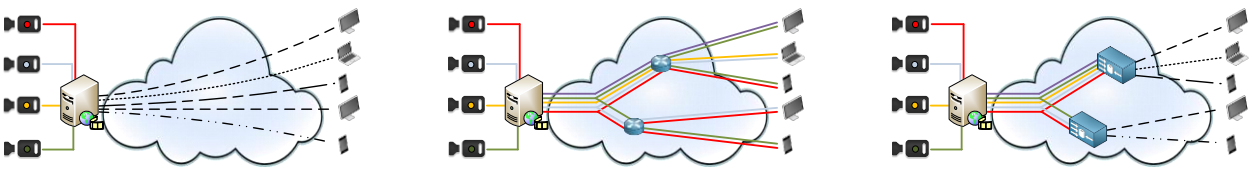
## 1. Introduction

In contrast to traditional 3D videos, which offer the users only a single viewpoint, Free-Viewpoint Video (FVV) is a promising approach to allow free perspective selection while watching multi-view video streams. The user-specific views dynamically change based on the user's position [1], and they must be synthetized accordingly.
The unique views are synthesized from two or more high bitrate camera streams and the corresponding depth maps [2] that must be delivered over the network and displayed with low latency. By increasing the number of deployed cameras and the density of the camera setup, the free-viewpoint video experience becomes more realistic. But on the other hand, more camera streams require higher network capacity. Therefore, viewpoint synthesis is a very resource hungry process and there is the need to find the best tradeoff between the quality of the synthetized view, which is related to the number of the delivered camera streams, and the processing time of the algorithm.
The required camera streams may change continuously due to the free navigation of viewpoint, hence effective delivery schemes are required to avoid starvation of the viewpoint synthesizer algorithm and keep the network traffic as low as possible. Moreover, packet losses due to congestion and the increased latency can disturb the user experience. In order to support more multi-view videos in IP networks, a simple approach is to minimize the bandwidth consumption by transmitting only the minimal number of camera views, as it was investigated in [3][4][5].

## 2. FVV architecture models

From architectural point of view, the FVV streaming models can be categorized based on the location of the virtual viewpoint synthesis in the network. The first category depicted in Fig. 1(a) is the server-based model, where all the camera views and corresponding depth map sequences are handled by a media server that receives the desired viewpoint coordinates from the customers and synthesizes a unique virtual viewpoint stream for each user. In this case, only unique free viewpoint video streams must be delivered through the network. The drawback of the server-based solution is that the computational capacity of the media server may limit the scalability of this approach and the service latency is also higher. In case of interactive real-time services, latency is one of the most critical parameters. If remote rendering is used, the control messages must be delivered to the rendering server and the generated stream must be forwarded back to the user, causing significant time gap between triggering the viewpoint change and the synthetized view playout. Moreover, in case of large number of customers, the centralized approach can suffer from scalability issues.

The approach in the second architectural solution, shown in Fig. 1(b), is to deliver reference camera streams and depth sequences directly to the clients to let them generate their own virtual views independently. In this approach the limited resource capacity problem of the centralized media server can be avoided, but huge network traffic must be delivered through the network caused by multiple camera streams. Multicast delivery can reduce the overall network traffic, however the requested camera streams by a user is changing continuously that must be also handled using advanced multicast group management methods. The benefit of client-based architectural model is that it has the lowest latency values, because the viewpoint synthesis is performed locally and the user control can be processed immediately by the rendering algorithm. Unfortunately, rendering FVV video streams at an interactive frame rate is still beyond the computation capacity of most devices, especially in mobile terminals.



(a) server-based (b) client-based (c) distributed

**Figure 1**. FVV streaming model categories based on the location of the virtual viewpoint synthesis

The third model is a distributed approach, Fig. 1(c), where the viewpoint rendering is done in locations distributed over the network. The user is not connected directly to the media server, but asks for the most appropriate proxy server for a synthetized stream from the desired viewpoint. Remote rendering provides a simple but effective solution, because both bandwidth and computation problems can be solved by synthetizing virtual views remotely on a powerful server at the price of increased latency [6]. Even if the distributed rendering solution can handle some of the bandwidth and computational limitations, new questions arises, *e.g.*, how to optimally design the FVV network architecture.

Possible answers to these questions have been proposed in [7]. Our aim was to find the optimal deployment locations of the distributed viewpoint synthesis processes in the network topology by allowing network nodes to act as proxy servers with caching and viewpoint synthesis functionalities. The other goal was to propose viewpoint prediction based multicast group management method in order to prevent the viewpoint synthesizer algorithm from remaining without any camera streams.

### 3. Optimized FVV network topology

The distributed approach provides a tradeoff between the server-based architecture and the client-based one, because it can avoid bandwidth and computational resource overloads and handles the user requests in a scalable way. Our goal was to optimize the FVV service topology by minimizing the traffic load without overloading the computational and other resources of the network components. In order to find the optimal arrangement of the distributed viewpoint synthesis model, the network architecture must be overviewed first.

The path between the media server and each client can be divided into two parts: *i)* from the media server to the proxy server, where the real camera streams are delivered and *ii)* from proxy server to the client, where the user specific views are transferred [7]. By locating the viewpoint synthesis functionality closer to the camera sources, the high bitrate camera streams will use less network links, therefore occupying less total bandwidth in the network. On the other hand, the proxy servers will have to serve more clients, so the total network traffic of the unique user specific streams will be higher.

In order to analytically investigate the optimal hierarchical level of the proxy servers, *k*-ary tree is considered. The depth of the tree is *D*, with the source at the root of the tree, while all the receivers are placed at the leaves and the viewpoint synthesis are performed in the proxy servers located $\delta$ hops from the root as illustrated in Fig. 2.

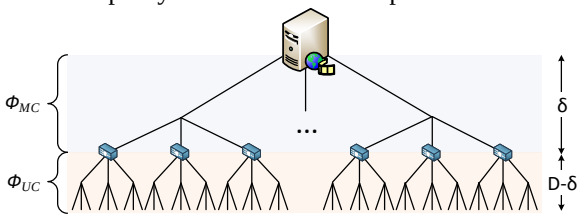



**Figure 2**. Distributed *k*-ary tree network topology

The goal is to determine the proxy locations to minimize the overall number of link usage:

$$\min\left\{\Phi_{UC}+\Phi_{MC}\right\} \qquad (1)$$

where $\Phi_{MC}$ stands for the overall number of multicast links from the media server to the proxy server and $\Phi_{UC}$ is the number of unicast links used to deliver user specific streams from proxy server to the client, respectively.

To calculate the number of multicast links ($\Phi_{MC}$) we adapt the results of Phillips et al. [8] to the multi-view video scenario. The unicast part ($\Phi_{UC}$) is easier to calculate. There are $D$-$\delta$ unicast hops from proxy to client as shown in Fig. 2, hence the total number of hops is $\Phi_{UC}=M\cdot(D-\delta)$, where $M$ is the number of users. Assuming $n$ proxy servers placed at level $\delta$ in the hierarchical tree, the summarized network resources can be calculated as follows, where $c$ stands for the number of deployed cameras:

$$\Phi_{MC}+\Phi_{UC}=c\cdot\sum_{l=1}^{\delta}k^{l}\left(1-\left(1-k^{-l}\right)^{n}\right)+M\left(D-\delta\right) \quad (2)$$

The number of FVV cameras and the number of users influence the optimal proxy server location. In order to show how the number of cameras modifies the traffic load in a *k*-ary tree network, we set *k*=3, number of users *M*=1000 and network depth *D*=8. The numbers of occupied links in the delivery paths are shown in Fig. 3.
By increasing the number of cameras, the number of hops and the traffic load increase in the multicast part of the network ($\Phi_{MC}$). Thus, it is worth locating the proxy servers closer to the camera sources. In an opposite case, where there are only three cameras, the lowest number of link usage can be achieved if the view synthesis is performed at level $\delta$=6 that is further from the media server. The *k*-ary tree topology is a simplified layout for analytical investigation. In fact, finding the optimal proxy server locations in dynamically changing network is extremely difficult.

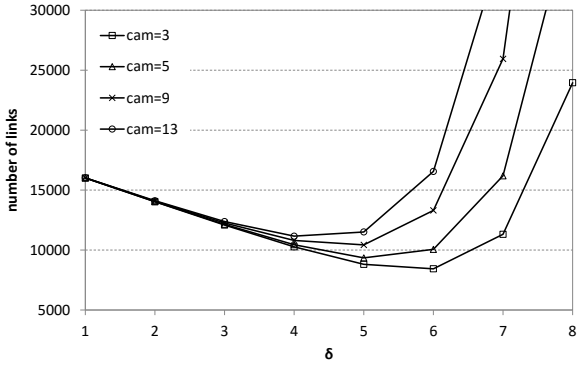



**Figure 3**. Overall link usage in *k*-ary tree network

The distributed architecture combined with multicast routing can solve the network overload problems and keep the traffic load as low as possible. However the increased latency of control messages can decrease the experienced user quality.

**4. FVV multicast**

In order to support more multi-view videos in IP networks, a simple approach is to minimize the bandwidth consumption by transmitting only the minimal number of views required. Multicast transmission is effective to reduce the network load, but continuous and frequent viewpoint changes may lead to interrupted FVV service due to the multicast group join latencies. To prevent the user's viewpoint synthesizer algorithm from starving, effective multicast group management methods must be introduced that can rely on viewpoint prediction. Therefore, our aim was to propose a viewpoint prediction based group management solution to minimize the probability of the synthesis process starvation.
Current IP multicast routing protocols (*e.g.*, PIM-SM) exploit shortest path tree logical layout for point-to-multipoint group communication that significantly reduces the network bandwidth.
In case of multicast free viewpoint video streaming each camera view is encoded and forwarded on a separate channel to the users. The separate channels (camera views) can be accessed by joining the multicast group that contains the needed camera source. Users can switch views by subscribing to another multicast channel, while leaving their present one.
If the multicast group change (leaving the old multicast group and joining the new one) happens only when the screen playout of the new virtual view is due, there will be an interruption in the FVV experience, since the lately requested camera view stream will not be received on time to synthetize the new view. Therefore, our aim was to propose a viewpoint prediction based solution for camera stream handoffs to minimize the probability of the synthesis process starvation.
To prevent the user's viewpoint rendering algorithm from starvation, the multicast group join message must be sent in time in order to provide all camera streams that may be requested in the near future. The join message must be sent when the viewpoint coordinates reach a predefined threshold coordinate value. While the viewpoint of the client is within the threshold zone, it will become a member of three multicast groups (*e.g.*, blue, green and yellow), as illustrated in Fig. 4. When the viewpoint coordinates leave the threshold zone, the client should receive only the two

required camera streams.

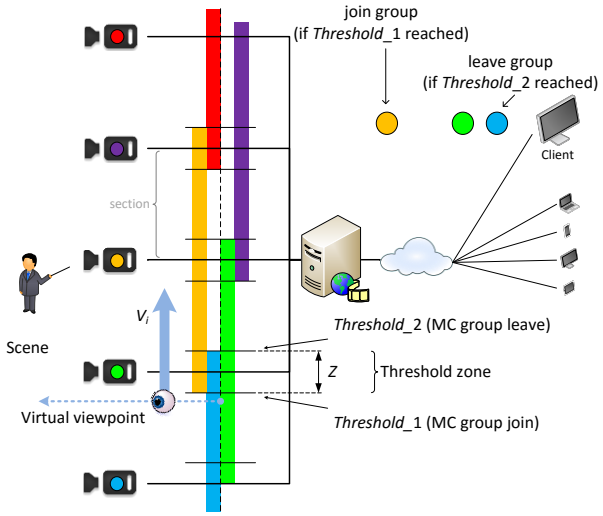



**Figure 4**. Multicast group join thresholds

An optimization goal can be to keep the threshold area as narrow as possible to reduce the number of multicast group memberships, so that the overall network traffic is reduced, but keep it wide enough to avoid playout interruption during viewpoint changes. In order to find the optimal threshold values, the multicast groups join latency and viewpoint movement features must be considered. Different algorithms can be used for viewpoint estimation such as linear regression or Kalman-filter [7]. To determinate the threshold values and the zone width ($Z$) of the viewpoint coordinates that trigger the multicast join and leave processes, the required time duration ($T_D$) from sending a multicast join message to receiving the first I-frame of the camera stream and the estimated viewpoint velocity ($v$) are used.

$$Z \geq v \cdot T_D \qquad (3)$$

Controlled threshold zone setup can minimize the starvation effect. The comparison of viewpoint velocity values and the caused starvation ratios are presented in Fig. 5.

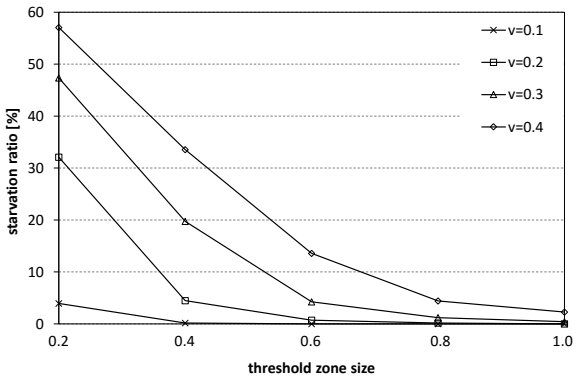



**Figure 5**. Starvation ratio in case of different velocity values and threshold zone sizes

According to the obtained results, setting the threshold zone too narrow can make the starvation ratio reach to as much as 57%, which renders the FVV service unacceptable. However, using adaptive threshold size can make the synthesizer algorithm get the camera views in time in more than 95% of the cases.

## 5. Conclusions

Both stream delivery and viewpoint generation are resource hungry processes leading to scalability issues in a

complex network with a large number of users. The delivery of high bitrate camera views and depth images required for viewpoint synthesis can overload the network without multicast streaming, while at the same time, late multicast group join messages may lead to the starvation of the FVV synthesis process. Distributed viewpoint synthesis approach and prediction based multicast group management schemes can offer scalable solutions for new FVV services and hopefully it can become a popular interactive multimedia service of the near future.

**Acknowledgement**
The author is grateful for the support of the Hungarian Academy of Sciences through the Bolyai János Research Fellowship.

**References**

[1] Gurler, C.G.; Gorkemli, B.; Saygili, G.; Tekalp, A.M., "Flexible Transport of 3-D Video Over Networks," Proceedings of the IEEE, vol.99, no.4, pp.694,707, April 2011.
[2] Christoph Fehn, "Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV", Proc. of SPIE, Vol. 5291, Stereoscopic Displays and Virtual Reality Systems, pp. 93-104, May 2004.
[3] E. Kurutepe, A. Aksay, C. Bilen, C. G. Gurler, T. Sikora, G. B. Akar, and A. M. Tekalp, "A standards-based, flexible, end-to-end multi-view video streaming architecture", in Proc. Int. Packet Video Workshop, Lausanne, Switzerland, Nov. 2007, pp. 302–307.
[4] Li Zuo; Jian Guang Lou; Hua Cai; Jiang Li, "Multicast of Real-Time Multi-View Video," Multimedia and Expo, 2006 IEEE International Conference on , vol., no., pp.1225,1228, 9-12 July 2006.
[5] T.-Y. Ho, Y.-N. Yeh, and D.-N. Yang, "Multi-View 3D Video Delivery for Broadband IP Networks," IEEE International Conference on Communications (IEEE ICC), June 2015.
[6] L. Toni, G. Cheung and P. Frossard, "In-Network View Re-Sampling for Interactive Free Viewpoint Video Streaming, Proceedings of IEEE ICIP, Quebec City, Canada, September 2015.
[7] Árpád Huszák, "Advanced Free Viewpoint Video Streaming Techniques", International Journal on Multimedia Tools and Applications, Springer, ISSN 1573-7721, pp 1-24, November 2015
[8] Graham Phillips, Scott Shenker, Hongsuda Tangmunarunkit, "Scaling of multicast trees: comments on the Chuang-Sirbu scaling law", SIGCOMM '99, New York, USA, 1999.

**Árpád Huszák** received his M.Sc. degree in 2003 as Electrical Engineer from the Budapest University of Technology and Economics (BUTE) at the Department of Telecommunications (Dept. of Networked Systems and Services since 2013) and completed his Ph.D. in 2010. Currently he is with the Department of Networked Systems and Services as assistant professor, but previously he also worked for the Mobile Innovation Center Hungary (MIK) and Ericsson Hungary. He has been involved in many European projects (FP6-IST, FP7-ICT, and Celtic). His research interests focus on network protocols, mobile computing and adaptive multimedia communications.