Advanced Free Viewpoint Video Streaming Techniques

Árpád Huszák

huszak@hit.bme.hu

Budapest University of Technology and Economics, Department of Networked Systems and Services, Hungary

Abstract

Free-viewpoint video is a new type of interactive multimedia service allowing users to control their viewpoint and generate new views of a dynamic scene from any perspective. The uniquely generated and displayed views are composed from two or more high bitrate camera streams that must be delivered to the users depending on their continuously changing perspective. Due to significant network and computational resource requirements, we proposed scalable viewpoint generation and delivery schemes based on multicast forwarding and distributed approach. 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. Moreover, a predictive multicast group management scheme was introduced in order to provide all camera views that may be requested in the near future and prevent the viewpoint synthesizer algorithm from remaining without camera streams. The obtained results showed that even 42% traffic decrease can be realized using distributed viewpoint synthesis and the probability of viewpoint synthesis starvation can be also significantly reduced in future free viewpoint video services.

Keywords: Free Viewpoint Video, multicast streaming, viewpoint prediction, distributed networks

1. Introduction

Free viewpoint video (FVV) is an interactive multimedia service offering similar functionalities that are known from 3D computer graphics. FVV allows users to choose own viewpoint, viewing direction and interactive free navigation within a visual scene likewise 3D computer graphics applications. The main difference is that FVV targets real world scenes, captured by real cameras, without using 3D graphical models. Slightly different viewing angles can be requested by the customers as they control their own viewpoint position and perspective, e.g. by moving or turning their head or changing position in a room. The customers of these interactive multimedia services may control the viewpoint and generate new virtual views of a dynamic scene continuously. The uniquely generated and displayed views are composed from several high bitrate camera streams that must be delivered from the cameras to the viewpoint synthesis algorithm that can be deployed at the media server, at the client or in the network. The free viewpoint video experience becomes more realistic as the number of camera views used to sample the scene increases. On the other hand, more camera streams requires higher network capacity, because the network traffic load increase, as well. Without advanced stream delivery schemes, the network bandwidth required to transmit multiple camera views for the viewpoint synthesizer deployed in the user equipment can overload the links of the network.

FVV has been drawing more and more attention due to promising features and various application scenarios. Free viewpoint streaming with its advanced features is foreseen as the next big step in 3D video technology beyond stereoscopy. However, a commercial free viewpoint television (FTV) service will be similar to the IPTV solutions, the difference is that not only one stream belongs to a TV channel, but several video streams [1]. The other difference is that the displayed media content is also dissimilar due the individual viewpoints of the same scene demanded be the users.

The research activity on this topic is very intensive because the key techniques of FVV are still not efficient enough to provide services with acceptable quality. Viewpoint synthesis is a very computational hungry process, therefore the existing algorithm are still trying to find the tradeoff between the video quality of the synthetized view and the processing time of the algorithm. Two methods can be used to synthase an individual viewpoint from the camera sequences: Light Field Rendering (LFR) [2] and Depth Image Based Rendering (DIBR) [3]. The LFR algorithm interpolates a virtual view from multi-camera images, while DIBR uses fewer images and a depth map to establish new views [4].

For transmission over limited channels FVV data must be compressed efficiently. Multi-view video compression techniques have been widely studied and powerful algorithms were proposed. Although many efforts have been done to compress LFR and DIBR, transmitting issues have not been deeply investigated.

Although different approaches have been used to generate 3D video, all these approaches make use of multiple views of the same scene. Delivery of multi- view FVV is different from traditional video streaming in the following points. An FVV service requires several video streams captured by different cameras recording the scene from different locations. Hence, the streaming costs more bandwidth than single video stream, therefore scalable quality of service is an important issue. 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. Current IP multicast routing protocols (e.g. PIM-SM) exploit shortest path tree logical layout for point-to-multipoint group communications that significantly reduces the network bandwidth. To synthesize a view using DIBR, the user must receive two continuously changing camera views instead of one due to viewpoint variance of users. Therefore, it is desired to have a smart view selection strategy to minimize the total bandwidth consumption in the networks in order to provide scalable multi-view FVV service.

The camera streams required by customers may change continuously due to the free navigation of viewpoint, hence the variation of visual quality due to view switching must be also handled to avoid starvation of the viewpoint synthesizer algorithm. Rendering FVV video streams at an interactive frame rate is still beyond the computation capacity of most devices. 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 servers and sending the 2D image of rendered scene back to user devices to display. However, the distributed rendering solution can overcome bandwidth and computational limitations, new questions arises e.g. regarding to architectural issues.

In this paper scalable FVV viewpoint generation and delivery schemes are proposed based on multicast forwarding and distributed approach. To prevent the user's viewpoint synthesizer algorithm from remaining without camera streams (starving), a predictive multicast group management method is introduced in order to provide all camera views that may be requested in the near future. We examined the multicast groups join latency and viewpoint movement features to find optimal threshold values that minimizes the starvation probability but avoids unnecessary camera stream forwarding. Multicast delivery can be used together with the distributed viewpoint generation approach. However, the viewpoint prediction based multicast group management can be also efficient solution, the localization of the viewpoint generation node is a significant issue, as well. Hence, in this paper the distributed viewpoint synthesis functionality was investigated from the network layout point of view. Our aim was to find the optimal arrangement of the distributed viewpoint synthesis processes by allowing network nodes to act as proxy servers with caching and viewpoint synthesis functionalities. We also proposed a method on order to minimize the traffic load of the FTV service without overloading the computational and storage resources of the network components. The performance of the proposed streaming techniques was analyzed in Ns-2 simulations.

The rest of this paper is organized as follows. The background of free viewpoint video viewpoint synthesis and streaming methods are presented in Section II. In Section III, the proposed distributed viewpoint synthesis approach based on multicast delivery for FVV services is introduced. The obtained performance results are presented in Section IV. The summary of the paper and the conclusions can be found in the last section.

2. Overview of Free-viewpoint video

Media content delivery requires high link capacity and low latency in order to provide acceptable quality of media streams. The transmission of traditional high resolution single-view videos is still challenging, but in case of multi-view videos this challenge becomes more interesting.

2.1. Multi-view video coding

To synthetize a virtual viewpoint from existing camera views, the camera streams must be forwarded to the renderer that can be deployed i) in the user equipment, ii) in a media server, or iii) distributed in the network. Without compression, the delivery of camera stream set would be impossible.

The two-view stereo (stereoscopic) video is the simplest scenario that consists of two videos representing the left and right views from two slightly different viewpoints corresponding to the distance of human eyes. Since two nearby views have similar content, compression is based on both adopting the traditional intra-view prediction along each view and performing inter-view prediction between two adjacent views.

With the advances of Depth Image Based Rendering (DIBR) approach [3][7], the views can be reconstructed from a video signal and a depth map. In case of video plus depth (V+D) approach the 3D information are separated into color and depth channels, where the depth information can be transformed to a monochromatic, luminance-only image taking values between 0 and 255 as shown in Fig. 1. In general, the depth channel requires an extra 10–20% of bit-rates to encode the depth information [8]. The free viewpoint video technique is usually based on the V+D representation. The most common solution requires two video streams and corresponding depth sequences to synthetize an individual virtual view.

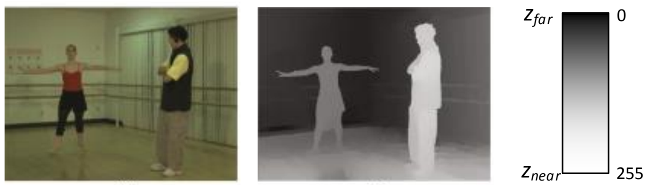

Fig. 1. Video-plus-depth representation

2.2. Viewpoint synthesis

Image based view synthesis in real time is still an open research problem that gains a lot of attention. It does not use any 3D geometry at all. The intermediate virtual views are generated from available natural camera views by interpolation. The main advantage is a potentially high quality of virtual views without 3D scene reconstruction. However, dense sampling of the real world with a sufficiently large number of natural cameras is necessary. Due to large numbers of cameras tremendous amount of image data needs to be processed. If the number of used cameras is too low, interpolation and occlusion artifacts will appear in the synthesized views causing reduced quality. Several image based solutions have been proposed [13][14][15] that often have problems in terms of both computation time and perceptual quality of synthesized views.

In case of DIBR at least two camera streams and the corresponding depth map sequences must be available at the renderer to generate an individual viewpoint [16][17] as illustrated in Fig. 2. The color image and an associated depth map along with camera calibration information, any pixel of the image can be projected into the 3D space and then projected back onto an arbitrary virtual camera plane, creating a virtual image. Conceptually, this method can be understood as a two-step process [18]: i) 3D image warping: it uses depth data and associated camera parameters to back-project pixel samples from reference images to the proper 3D locations and re-project them onto the new synthesized image space; and ii) reconstruction and re-sampling: determination of pixel sample values in the synthesized image.

In the first FVV solutions offline viewpoint generation was mainly used in film production, e.g. for stop-motion special effects in movies or for sports effects systems, like “LiberoVision” [19]. Fortunately, the increased computational and network resources makes interactive real-time FVV services available, too.

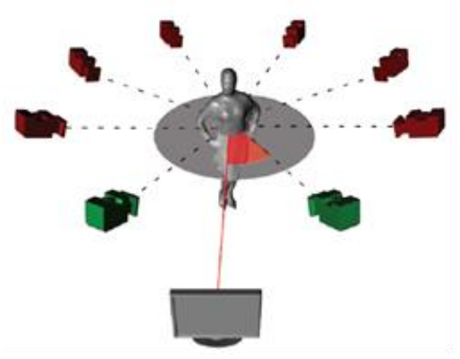

Fig. 2. Viewpoint synthesis

The accuracy of the depth data significantly impacts the quality of the generated virtual view. The amount of distortion increases with the distance of the virtual view from the original view, thus drastically limiting the potential navigation range using single video plus depth. The synthesis ability of image based representation has limitations on the range of view change and the quality depends on the scene depth variation, the resolution of each view, and the number of views, as well. However, using more than one view with corresponding depth channels, the potential navigation range can be widened e.g. by pairwise switching. The next step is to deliver the required V+D streams to the renderer device that can be a server or a client terminal with strong computational resource constraints.

2.3. Streaming

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. 3.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 syntheses 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. The second solution (Fig. 3.b) is to deliver required camera streams and depth sequences to the clients to 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

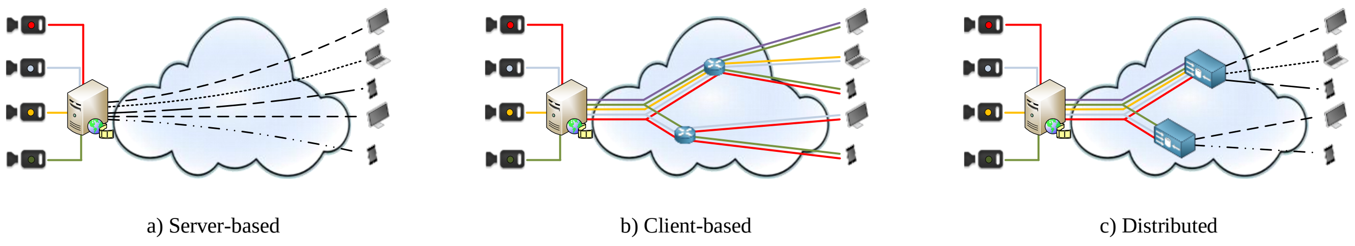

Fig. 3. FVV streaming model categories based on the location of the virtual viewpoint synthesis

through the network. 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 third model is a distributed approach (Fig. 3.c), where the viewpoint rendering is done in distributed locations in the network. The distributed model can avoid bandwidth and computational resource overloads and handle the user requests in a scalable way.

Most of the published works assume client-based viewpoint synthesis and focus on multi-view video delivery. Authors of [20] are proposing a LFR based and QoS aware FVV streaming solution. The paper focuses on I-frame retransmission and jump frame techniques in the application layer based on RTP/RTCP to support different level of QoS. A streaming system for DIBR based FVV over IP networks was introduced in [21]. The proposed solution divides video streams into depth video, texture video and common video, and transmits them in RTP/RTSP individually making the service more robust against transfer errors, however it did not solve view switching and synchronization problems. Kurutepe et al. [24] presented a multi-view streaming framework using separate RTSP sessions to deliver camera views allowing the client to choose only the required number of sessions. The proposed scheme utilizes currently available streaming protocols with minor modifications. Camera switching was not investigated in their work.

Selective streaming is a method to reduce the bandwidth requirements of multi-view video, where only a subset of views is streamed depending on the user's viewing angle. To select which views should be delivered, the viewer's current head position is tracked and a prediction of future head positions is calculated as reviewed in [22]. In order to conceal prediction errors, low quality versions of other views can be also be streamed as presented in [23]. However, the selective streaming method suffers from fast head movements. They showed that delay in stream switching, which is determined by the frequency of switching frames, may degrade the perceived quality.

Very efficient way of reducing traffic load is multicast delivery, suited for both video on demand (VoD) as well as live multimedia applications. However, multicasting can be applied only if a group of users are interested in the same content in the same time. In case of FVV, the displayed view is different for each user, but the required camera streams can be the same. In case of FVV multicast delivery, streams of camera views are transported over separate IP multicast groups. The users can selectively join to multicast groups that are used for delivery of camera stream, which is required to synthetize the desired viewpoint. Multicast transmission is effective to reduce the network load, but continuous and frequent viewpoint changes may lead to interrupted FVV service due multicast group join latencies. Therefore, the required camera streams may arrive too late and starve the FVV synthesizer process.

Multicast FVV transport solutions is a promising delivery scheme, however it was not investigated deeply. Authors of [25] and [26] proposed a multi-view video streaming system based on IP multicast, where the camera streams are transmitted using multiple-channels to support various users who have different available bandwidth. Other advanced ideas for transmission, like multipath delivery, P2P or cloud-assisted techniques for multiview video streaming were reviewed in [27].

2.4. Distributed FVV Streaming

The only paper we found in the literature regarding to distributed 3D services was authored by Petrovic et al. [28]. They proposed an end-to-end delivery model for 3D video applications, which leverages distributed system architecture to reduce the bandwidth and processing cost at the server and the end-hosts. Their prototype implementation demonstrated that highly heterogeneous clients can coexist in the system, ranging from auto-stereoscopic 3D displays to resource-constrained devices.

Authors of [29] proposed a cloud based view synthetization architecture for interactive multi-view video systems facing with limited bandwidth constraints. They introduced a synthesized reference view selection optimization problem aimed at finding the best subset of reference views to be transmitted to the decoder, where the subset is not limited to captured camera views as in previous approaches but it can also include virtual viewpoints, too. In [30] a cloud-based free viewpoint video rendering framework for mobile phones over cellular networks was presented. More specifically, a novel resource allocation scheme was proposed that jointly considers rendering and rate allocations between cloud and client to optimize the quality of experience.

Our work differs from the above work in that: i) in previous works camera switches were not studied from the starvation of the viewpoint synthesizer algorithm point of view; ii) how to trigger multicast group join and leave control messages and iii) none of the papers deal with the layout issues of a distributed FVV system.

3. Distributed viewpoint synthesis approach based on multicast delivery

Free viewpoint video and television services allow users to individually change the desired viewpoint of a video scene that is captured by several. In order to produce the requested viewpoint, the camera streams must be delivered to the viewpoint synthesizer algorithm. Due to very high storage capacity and computational resource requirements, we propose to distribute the viewpoint synthesis process in the network and deliver the camera streams using multicast streaming. The distributed architecture combined with multicast routing can solve the scalability problems and keep the traffic load as low as possible.

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. These proxy servers share their resources for viewpoint synthesis, recoding and caching purposes. Therefore, the user is not connected directly to the media server, but asks the most appropriate proxy server for a synthetized stream with the desired viewpoint. The aim of the proxy servers is to gather the camera streams that are needed to serve the connected clients and originate the unique streams as illustrated in Fig. 4.

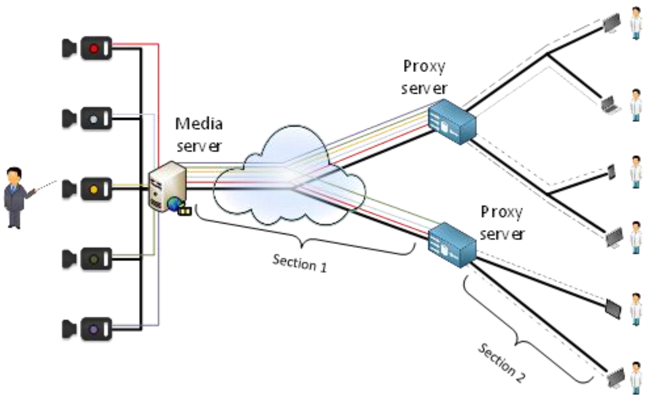

Fig. 4. Proxy server based view synthesis

Basically, a proxy server may cache the segments of a conventional video stream, but in case of FTV services it is preferred to support codec functionalities and viewpoint synthesis, too. We modeled the proxy element as presented in Fig. 5.

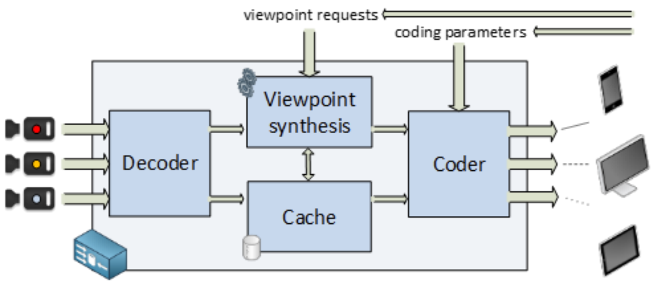

Fig. 5. Caching and rendering proxy model

The received multicast camera streams (all or only a set of streams) are processed by the decoder module and stored in the cache to handle delay variation. Based on the incoming viewpoint requests, the viewpoint synthesis module will generate the new stream that will be coded according to users' coding setups (resolution, coding quality). The outputs of the proxy server are the unique, user specific streams.

In this paper our aim was to analyze how to localize the proxy servers in order to find optimal proxy server layout for FTV service. Actually, different objectives can be targeted to optimally locate the proxy servers, nevertheless in this work our objective was to generate the lowest traffic load, but avoid the overload of computational and storage resources. In order to provide seamless viewpoint changes, the set of required camera stream must be available at the proxy server. Multicast delivery of camera streams from media server to proxy server is an appropriate solution, however due to network latency and frequent changes of the viewpoint, the required camera streams may arrive too late, interrupting the FVV synthesis and playout. Therefore, we propose a seamless FVV streaming scheme based on user viewpoint prediction. In order to avoid the starving of the FVV synthesizer, we prefetch the camera views that will be probably required for the viewpoint generation.

The first part of our work focuses on the optimal proxy layout issues, while the second part introduces predictive multicast group management solution to prevent the viewpoint synthesizer algorithm from remaining without camera streams.

3.1. Optimization of distributed viewpoint synthesis

In order to find the optimal arrangement of the distributed viewpoint synthesis processes, the network architecture must be introduced first.

As Fig. 4. shows, the path between the media server and each client can be divided into two parts. In *Section_1* (from media server to proxy server) the real camera streams are delivered, while *Section_2* (from proxy server to the client) the user specific views are transferred. The camera streams must be always forwarded with the highest quality and full resolution, requiring higher bandwidth, but fortunately multicast delivery mode can decrease the overall occupied bitrate on the links. In *Section_2* the streams are unique, so multicast is not an option. Opportunely, the synthetized streams may be coded with lower bitrate, e.g. if it is played out on a mobile terminal with low resolution display. 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. The goal is to determine the proxy locations to minimize the overall number of link usage:

$$\min\left\{\Phi_{UC}+\Phi_{MC}\right\}, \tag{1}$$

where Φ_{MC} stands for the overall number of multicast links in *Section_1* and Φ_{UC} is the number of network links used to deliver user specific unicast streams in *Section_2*, respectively.

Multicast significantly alleviates the overhead on senders by allowing them to reach the entire group with the transmission of a single packet. While, multicast routing ensures that only one copy of each packet will traverse each link, significantly reducing the network load. The gain of multicast in terms of network resource consumption was firstly analyzed by Chuang and Sirbu [31]. Their scaling law shows that in case of multicast, the average number of hops from the source to randomly chosen m distinct destinations in the shortest-path tree is $E[H_N]\cdot m^{0.8}$, where $E[H_N]$ is the average number of hops of a message to a uniform location in the graph containing N nodes. Hence, in case of unicast delivery to m different users, the hop number is $m\cdot E[H_N]$.

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 is performed in the proxy servers located δ hops from the root (Fig. 6.).

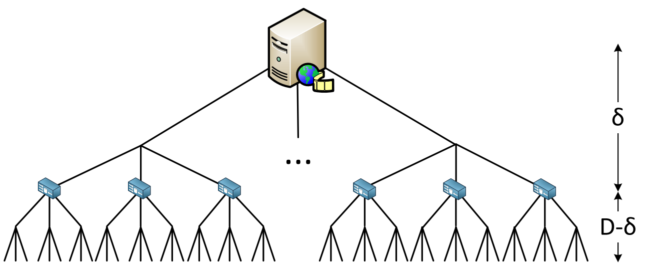

Fig. 6. Considered network topology: *k*-ary tree

In a simplified k-ary tree distributed FVV architecture the proxy servers are placed at level δ in the hierarchical tree. Hence, there are k^{δ} proxy servers served by multicast camera streams. Assume that only one multicast camera stream is forwarded. Now, if we select n not necessarily distinct proxy servers, each such selection will require a path through one of the k^l links at level *1*. Hence, picking one of the proxy servers at random is equivalent to picking one of the k^l links at level *1* at random. Thus, the probability that this particular link is in the delivery tree after n proxy servers have been selected is given by [32]:

$$1-\left(1-k^{-l}\right)^{n} \tag{2}$$

All such probabilities are independent since the sum of the averages is independent of correlations, and so the average number of links in the delivery tree at level l is

$$\varphi_{MC}=k^{l}\left(1-\left(1-k^{-l}\right)^{n}\right). \tag{3}$$

Assuming that the bitrate of multicast camera streams are equal, the traffic load correlates linearly to the number of links in delivery tree. If there are c cameras (multicast groups) deployed, the average number of links in the multicast delivery tree (*Section_1*) can be calculated as follows:

$$\Phi_{MC}=c\cdot\sum_{l=1}^{\delta}k^{l}\left(1-\left(1-k^{-l}\right)^{n}\right) \tag{4}$$

Calculating the number of used links in *Section_2* is simpler, because the user-defined streams are forwarded from the proxy servers to the users in unicast mode. According to Fig. 6., there are D-δ unicast hops from proxy to client, hence the total number of hops in *Section_2* can be calculated as follows:

$$\Phi_{UC}=M\cdot\left(D-\delta\right) \tag{5}$$

In equation (5), M stands for the number of users. In order to find the appropriate hierarchical level for proxy server locations, we must estimate the consumed network resources in the proposed distributed FVV architecture:

$$\Phi_{MC}+\Phi_{UC}=c\cdot\sum_{l=1}^{\delta}k^{l}\left(1-\left(1-k^{-l}\right)^{n}\right)+M\left(D-\delta\right) \tag{6}$$

We performed numerical analysis to investigate how the hierarchical level of proxy servers impacts the overall link usage, thereby the traffic load in a basic network with k-ary tree topology. The optimal level of proxy servers (δ) depends on the parameter k, the number of users (M) and the depth of the network topology (D) based on equation (5). Unfortunately, the classical way of marginal value calculation cannot be derived:

$$\frac{\mathrm{d}\left(\Phi_{MC}+\Phi_{UC}\right)}{\mathrm{d}\,\delta}=0 \tag{7}$$

We analyzed different parameters, how they modify the optimal hierarchical level of proxy servers. In a basic k-ary topology, where k=5, tree depth D=8 and number of cameras c=3, the lowest link usage of the networks that can be calculated by (5), depends on the number of clients, as Fig. 7. shows.

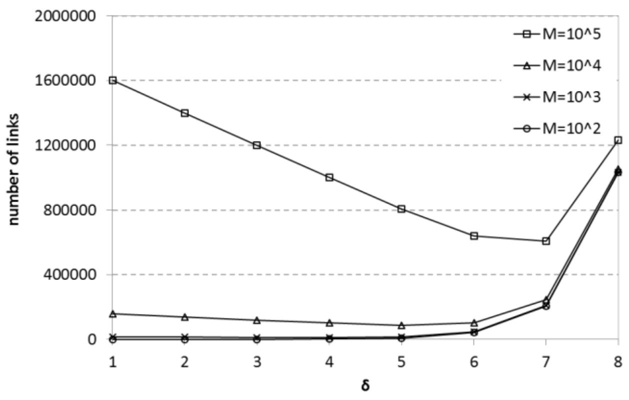

Fig. 7. Overall link usage in *k*-ary tree network with *D*=8 levels, depending on number of FVV customers

According to the presented results, if the number of FVV users (*M*) decreases, the view synthesis process must be located closer the root (media server) of the network. E.g. in case of 10^5 users, the ideal proxy server location is at level δ=7, while having only 100 customers, the lowest number of overall hop number can be achieved if the viewpoint synthesis is deployed 2 hops from the media server (root).

The optimal proxy server location is also influenced by the number of FVV cameras. In the proposed distributed FVV model, each camera stream is forwarded in separate multicast groups. In order to introduce how camera number modifies the traffic load in the network, we set *k*=3, *M*=1000 and *D*=8. The calculated average numbers of links in the delivery paths are shown in Fig.9.

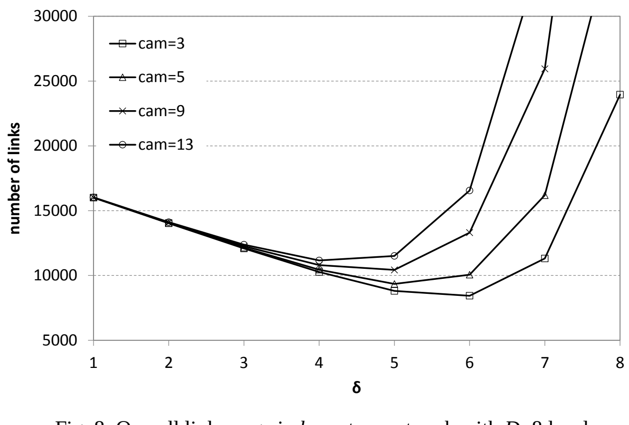

Fig. 8. Overall link usage in *k*-ary tree network with *D*=8 levels, depending on number cameras

By increasing the number of cameras, the total hop number, as well the traffic load is also increasing in the multicast part of the network (*Section_1*). As the calculations show, the optimal hierarchical level of proxy servers, where the link usage is the lowest, is closer to the media server if more cameras are deployed. In opposite, e.g. having only three cameras, the lowest number of links usage can be achieved if the view synthesis is performed at level δ=6 that is further from the media server.

Finally, we analyzed the ratio of average number of links in the multicast (*Section_1*) and unicast (*Section_2*) delivery tree. In a simple *k*-ary tree network, the total hop number of multicast links is higher, if the path between the media server and the view synthesis functionality (proxy server) is longer, because the proxy servers are served in multicast delivery mode. Similarly, if the proxy server is close to the media server, the multicast traffic hop number will be negligible, however on the other hand the longer unicast path will cause higher load in unicast *Section_2*. Fig. 10. shows the numerical result of a *k*-ary tree FVV architecture with *D*=8 levels, where *k*=5, camera number is 7 and the number of users is 10^5.

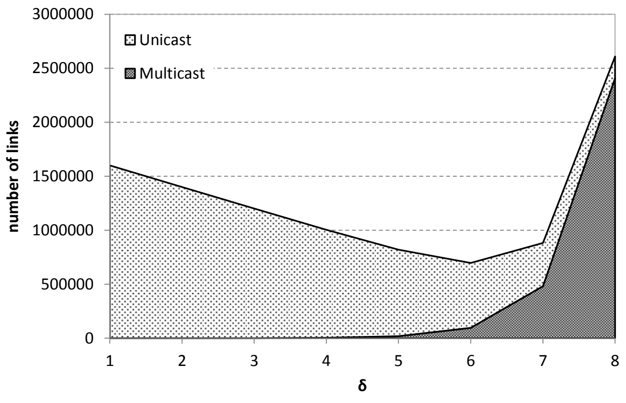

Fig. 9. Overall link usage in *k*-ary tree network

According to the calculations, the lowest number of delivery paths was used, when the proxy servers were located at level δ=6 in the hierarchical FVV architecture.

The *k*-ary tree topology is a simplified layout for analytical investigation. Actually, the real network graph representations are more complex, hence finding the proxy server locations is more difficult.

The described proxy localization problem can be mapped to a knapsack combinatorial optimization task [33], if the items are considered to be the proxy servers, the value is the occupied total bandwidth (actually, it is the inverse of the bandwidth, because we want to minimize it and not maximize) and the limited weight is the computational and link capacity limit. Unfortunately the knapsack problem is known as NP-complete problem, so there are no guaranties that the optimal proxy topology setup can be found with acceptable runtime. In case of brute force method, all possible proxy location must be examined for each client. The complexity of the brute force optimization approach is $O(k2^n)$, where n stands for the number of possible proxy server locations and k is the number of users. Finding the optimal distribution of the viewpoint synthesis process is hard even in static network environment, where the clients do not change their point of access and the required camera streams necessary for the viewpoint production do not vary. In reality the problem is more difficult due to the continuously changing environment.

3.2. Predictive multicast group management

In our proposed scheme we use multicast delivery of camera streams from media server to proxy servers, however due to network latency and frequent changes of the viewpoint, the required camera streams may arrive too late, interrupting the FVV synthesis and playout. The interruption can become more serious in case of fast viewpoint changes.

To generate a desired virtual perspective, the proxy must be joined to multicast groups that contain the required camera views. When the user changes his viewpoint and new camera views are needed, the proxy server must join to a new multicast group ensuring the actually needed camera streams. If the multicast group change (leaving the old multicast group and joining the new one) is performed only when the new virtual view already must have appeared on the users screen, there will be an interrupt in the FVV playout, because the lately requested camera view will not be received in time to synthetize the new viewpoint. Therefore, our aim was to propose a viewpoint prediction based solution for camera view handoffs to minimize the probability of the synthesis process starvation. To prevent the user's viewpoint synthesizer algorithm from remaining without camera streams, multicast group join threshold is introduced in order to start prefetching camera views that may be requested in the near future.

The following scenario introduces how the proposed threshold is used to prefetch camera streams based on viewpoint prediction (Fig. 10.). Using the proposed prediction model in this sample scenario and supposing that the viewpoint of the client is moving from the *blue* camera view position towards the *yellow* one, the desired view will reach *Threshold_*1 initiating a joint message to the *yellow* camera multicast group. While the viewpoint of the client is within the threshold zone, it will become a member of three multicast groups (*blue*, *green* and *yellow*). If the viewpoint is moving towards the *yellow* camera position and reaches *Threshold_*2, the proxy should leave the *blue* multicast group.

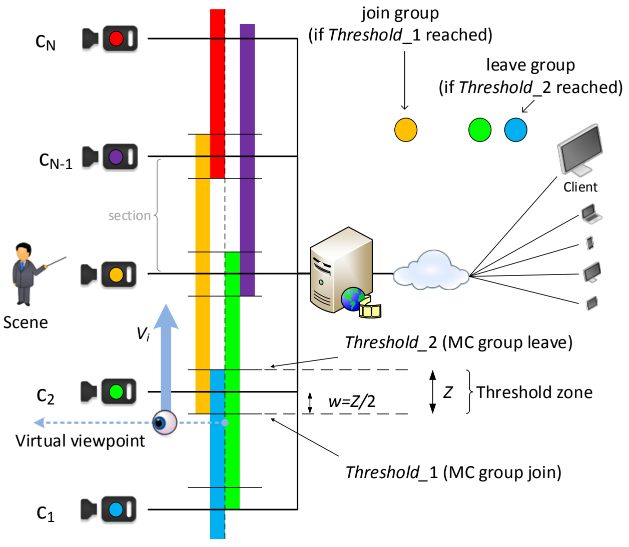

Fig. 10. Multicast FVV: Multicast group join thresholds

However, Fig. 10. shows a linear camera topology setup, the cameras can be deployed in plane (2D) and in space (3D) as well. In the latter cases not only two camera streams are required for the viewpoint synthesis, but three or even four that makes the threshold area determination more difficult. Our goal was to keep the threshold area as low as possible to reduce the number of multicast group memberships, so the overall network bandwidth, but keep it large 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 were examined. Assuming a linear camera row, where x_i denotes the actual viewpoint position and v_i the velocity of the viewpoint in time t_i, the next viewpoint location in time t_{i+1} can be expressed as follows

$$x_{i+1} = x_i + v_{i+1}\left(t_{i+1} - t_i\right). \tag{8}$$

Depending on the velocity of the viewpoint in the next moment (v_{i+1}), the view synthesis algorithm may require new camera views, despite that v_{i+1} is not known in t_i time, so it must be estimated based on previous viewpoint movement behavior. We used linear regression method to estimate the next viewpoint by calculating the average viewpoint velocity values from previous viewpoint coordinates.

To determinate the threshold values and zones of the viewpoint coordinates that triggers the multicast join and leave processes, the required time duration (d_m) from sending a multicast join message by the proxy server to receiving the first packet of the camera stream is necessary. After the reception of the new camera frames, the proxy server can render and forward the required views. The proxy server (Fig. 5.) can only decode the multicast stream after receiving an I-frame, therefore the I-frame period time

(d_I) must be also taken into consideration. Within $T_D=d_m+d_I$ time the viewpoint location should not move to another section of the camera row, where new camera streams are required for the viewpoint synthesis, otherwise the synthesis process will stall. In other words, if the new camera video packets arrive within T_D time, the proxy server can render the required virtual views based on the new camera stream without starvation. Therefore, the T_D time constraint is considered only for *Section_1* (Fig. 4.), while the rendering process time and the delivery delay through *Section_2* is not included in T_D.

In our proposed method the threshold zone dimensions (Z) is determined as follows (see Fig. 10.)

$$Z \geq 2\left(v_{i+1} \cdot T_D\right) \tag{9}$$

where T_D is assumed to be the sum of d_m (RTT (round-trip time) between the proxy server and the FVV media server) and d_I (time distance between the I-frames), while v_{i+1} is estimated as

$$v_{i+1} = \frac{\sum_{j=1}^{i} v_j}{i}. \tag{10}$$

In some cases the d_m parameter can be even lower than the RTT, if the join message goes through a multicast router that already forwards the required camera stream to other destination. If the camera view must be inquired from the media server, the multicast join latency will be equal to RTT. In order to minimize the viewpoint synthesis algorithm starvation, we used $\max\left(d_m\right) = RTT$ in our model. According to Fig. 10., the threshold zone size can be calculated also as

$$Z = Threshold\_2 - Threshold\_1. \tag{11}$$

The threshold values in each section can be determined based on the camera coordinates (c_k) and the threshold zone size (Z) as $c_k \pm Z/2$. In the forthcoming evaluation section, w=Z/2 was used as a parameter, named window size.

From architectural point of view, the proposed solution requires multicast support in the network layer. The generally used PIM-SM [34] or PIM-DM [35] protocols are applicable for the presented free viewpoint video streaming service without any modification. Using PIM-SM rendezvous point (RP) and routers with multicast support are necessary elements of the network, while the control of group management packages must be done in the application layer. Synchronization of camera streams are also required in order to perform seamless camera handovers. Using the RTP/UDP timestamp feature this problem can be handled.

4. Simulation results

In order to examine the achievable gain of the proposed viewpoint synthesis process distribution in a multicast free viewpoint television network, simulation environment was used. Due to very high simulation runtimes, the performance analysis was performed using a simple network, which spanning tree has only several hierarchical levels. First, the impact of proxy server localization on network traffic was analyzed with our simulation tool that was implemented in Java, while the proposed threshold based multicast group management method was examined using Ns-2 [36]. We used Ns-2 because it has built-in PIM-DM multicast routing implementation.

4.1. Evaluation of optimized proxy server deployment

Beside the analytical analysis of distributed viewpoint synthesis, simulations were also performed in order to evaluate the benefits of the proposed approach. The implemented Java simulation tool provides an interface to design the FVV network topology, set link capacities, add users and their requested viewpoint coordinates, determine the camera set and define network elements to act as router or proxy server. Proxy server (with viewpoint synthesis functionality) can be added manually to a network, or using the optimization process. To find the global optima, we used a brute force approach. Unfortunately it works only for small network (cca. 20 network elements and max. 60 clients) due to long simulation runtime.

In the first phase of the evaluation process, the analytical results were compared with the outcome of the simulations. Afterwards the proposed proxy arrangement scheme was analyzed from the number of hierarchical layers, number of clients, and camera bitrate point of view.

Taking the limitation of both the analytical and simulation examination into consideration, a four level (D=4) binary tree (k-ary tree, where k=2) network topology was used for the comparison tests with 30 clients and 1 Mbps video stream bitrates. However, the analytic model can handle higher k values and more hierarchical levels (D), the simulation runtime makes it not possible to increase these parameters. On the other hand, the simulation tool is able to work not just with k-ary tree layout, but with more complex network topologies. Fig. 11. shows the comparison results of the theoretical and simulation based analysis, where impact of the number of reference cameras was examined. The solid lines show the analytic estimation of the total traffic in the network, while the dashed lines sign the simulation results. Two scenarios were analyzed with *cam*=5 and *cam*=9 reference cameras deployed. The results show that in all of the cases the second layer (δ=2) was resulted as the most appropriate layer for proxy deployment.

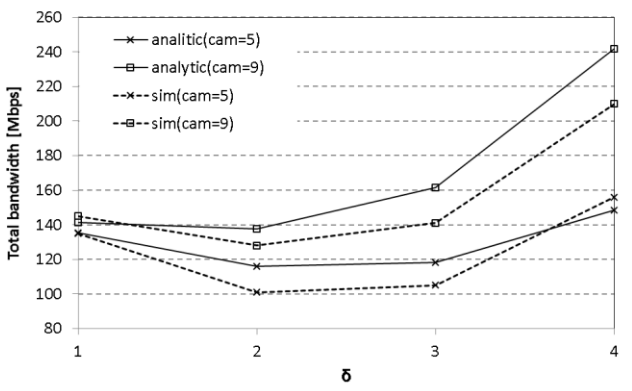

Fig. 11. Comparison of simulation and analytic approach

According to the obtained simulation results, we found 4.6% and 8.5% difference in average compared to the theoretical result for *cam*=5 and *cam*=9 scenarios, respectively. The analytical model estimates the most appropriate hierarchical level for the proxy server deployment, therefore in the comparison analysis the proxies were allowed to be located at the same hierarchical level in the simulations. However, the advantage of the simulation tool is that it can analyze all possible arrangements, not just those ones, where all the proxy servers are at the same level. The other significant advantage of the implemented simulation tool is that it takes network capacity into account and discards all layouts, where any of the links is congested or proxy server is overloaded. The distortion analysis of virtual views due to packet losses is out of the scope of this paper.

In order to measure the performance of the proxy server layout optimization scheme, two network topologies were used. One with three hierarchical levels (*D*=3) and the other one with four levels (*D*=4). The default values of link capacities were set to 100 Mbps, while assuming high definition reference camera system, the stream bitrates were 10 Mbps.

The brute force algorithm compares all proxy setup combination and checks whether it is a correct topology arrangement or not. While the network links capacities and the proxy server computational resources are limited, in some of the network layouts the proxy servers and the links can be overloaded. These incorrect proxy server arrangements must be rejected. Otherwise, service guaranties cannot be offered.

By increasing the number of hierarchical level, the number of total layouts grows exponentially, similarly to the optimization process runtime. Table 1 shows the number of total and acceptable proxy server arrangements in a two, three and four level hierarchical network. Note that most of the total layouts are not correct due to unserved clients criteria, link or proxy server overloads. The distributed proxy server brute force optimization algorithm ran fast for two and three levels, but it took 66 hours for a network with four hierarchical levels containing 21 network elements and 60 clients.

Table 1. Number of proxy server layouts

Hierarchical levels (*D*)	Total layouts	Correct layouts	Duration [ms]
2	8	1	1
3	512	124	190
4	2 097 152	704 969	239 927 390

In order to synthetize a desired viewpoint according to the user's request, at least two camera streams are necessary. In the first simulation scenario we analyzed how the camera stream bitrate effects the total occupied bandwidth in a network with three hierarchical levels (*D*=3). As Table 1. shows, in this case 512 different proxy arrangement exist that are signed with #0 to #511 in the following figures, while the different layouts are referenced with #0 to #2097152 in case of *D*=4 . The camera bitrate has impact on the first section of the delivery path, from media server to proxy server, as the obtained simulation results (Fig. 12.).

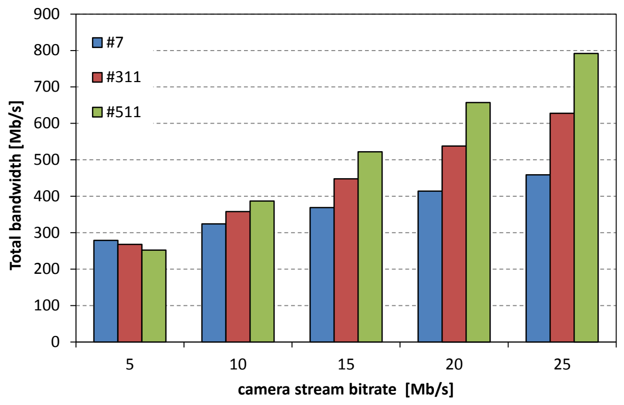

Fig. 12. Comparison of layouts from reference camera bitrate point of view

According to the obtained results, the best performance was achieved in case of layout #7 when the camera bitrate was 10 Mbps or higher, while the worst performing layout was #511. Layout #311 had an average performance. Fig. 13. shows the proxy server locations in these arrangements.

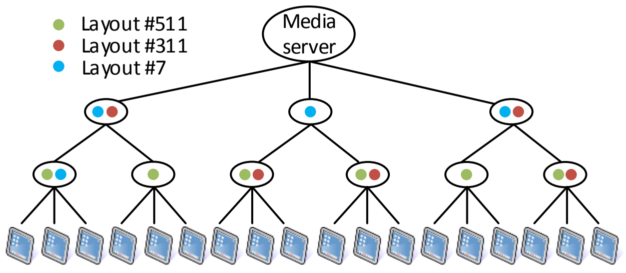

Fig. 13. Proxy server layouts

The optimally distributed viewpoint synthesis functionality decreases the total traffic by 42% compared to the worst case scenario even in this simple network topology.

To evaluate the performance of the proposed distributed viewpoint synthesis scheme, we compared the total network traffic of all proxy server topology setups. Of course, those layouts were discarded, where link congestion or computational overload of proxy server occurred. In the following measurement, we have compared all possible proxy layouts of network topologies with three (Fig. 13.) and four hierarchical levels from the network traffic point of view. All the network elements, except media server, were able to serve as a proxy server with viewpoint synthesis functionality. Due to presentation reasons, the analyzed layouts were ordered by the measured total network traffic and illustrated in Fig. 14. The results are shown in a single figure for both network topologies, hence the overall bandwidth values of the four level network topology are on the left vertical axis of the figure, while the right vertical axis is scaled for the three level network topology. There is also difference in the horizontal axis scale.

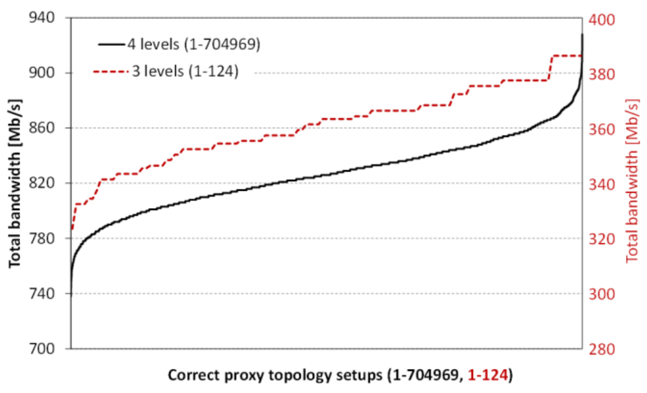

Fig. 14. Network traffic of correct layouts in a three and four level topology

The simulation results make it possible to compare the occupied bandwidth of all proxy server layouts in a fixed network structure. As Fig. 14. shows, there are significant differences in performance. In case of a three level network (D=3) the overall traffic can be reduced by 12% by selecting the most appropriate layout, while in a four level network (D=4) the gain is 16% compared to the worst proxy server arrangement. The obtained results prove that it is worth to consider viewpoint synthesis distribution for free viewpoint television service and optimize the layout of proxy servers in order to minimize the network traffic.

The other important constituents of a free viewpoint video streaming service are the clients. Therefore, we analyzed the performance variation from the number of users point of view. In this measurement, a three level network hierarchy was assumed, where 512 different layout combinations exist, but only 124 are correct. The total network traffic in case of 18 and 60 clients are presented in Fig. 15. The proxy server layout combinations are ordered by the occupied bandwidth values, similarly to the previous figure.

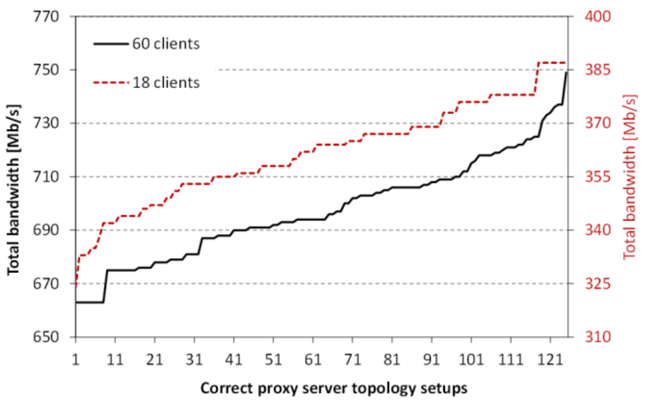

Fig. 15. Comparison of layout performance from client density point of view

The simulation results show that the difference between the worst layout and the optimal one is more than 10% in both cases. Comparing the occupied network bandwidth of the optimal viewpoint synthesis distribution arrangements, we can find that having 3.3 times more clients results only 2.1 times higher traffic load. The reason of this behavior is that due to high number of client, most of the camera streams are already forwarded through *Section_1* (media server to proxy server) of the delivery path (Fig. 4.). Hence, by increasing the number of clients, the traffic load on *Section_1* will not increase, only in *Section_2*.

4.2. Evaluation of threshold-based predictive multicast group management

Even if the optimal layout of proxy servers are found, the networks latency and the frequent viewpoint changes can lead to late camera stream delivery causing starvation in the viewpoint synthesis process. The proposed predictive multicast group management scheme uses viewpoint prediction and a threshold for camera stream prefetching as it was presented in section 3.2.

The performance evaluation of the proposed predictive multicast FVV streaming model was based on Ns-2 network simulator. In the analyzed network topology the viewpoint synthesizer proxy servers were deployed at the lowest level of a hierarchical tree network topology with four levels (D=4). We used PIM-DM multicast dense mode protocol in the FVV streaming simulations. The default parameter values of the simulated network used for the examination of the predictive FVV streaming scheme are presented in Table 2.

In the evaluation phase, the viewpoint velocity was considered to be measured in camera distance unit. This means that the distance between two neighbour camera positions is considered to be *cam_dist*=1. In the analysed FVV scenario the user viewpoints shift with viewpoint velocity value in random times. The time difference between two viewpoint shifts can be set with the max. timeslot length parameter.

Table. 2. Default parameters

Parameter	Default value
simulation time	20 s
link delay	10 ms
total number of clients	350
number of cameras	25
camera stream GOP size	1
packet size	1000 byte
video bitrate per camera	1 Mbps
link bandwidth	10 Mbps
viewpoint velocity (*v*)	avg. 0.3 × camera distance per timeslot
max. timeslot length	avg. 0.05 sec, rand(0; 0.1)
window size (*w*=*Z*/2)	0.3 × camera distance

In the following simulation scenario we analyzed the correlation between the average viewpoint velocity values and the window size. The different expressions such as camera distance (*cam_dist*), threshold zone (*Z*) and window (*w*) are illustrated in Fig. 16. In our measurements the velocity parameter was changed from 0.2 to 1 camera distance units. When the velocity parameter is 1, the viewpoint will skip always to a new section in the camera row and require new camera streams.

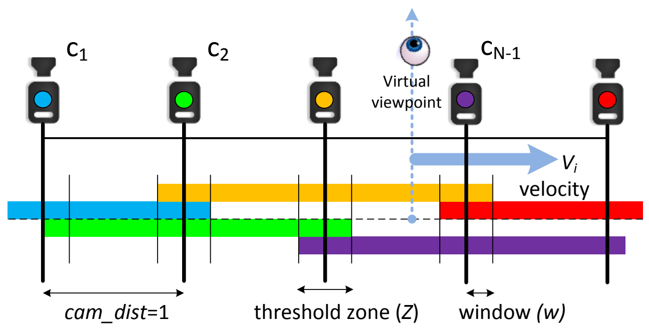

Fig. 16. Main parameters used in predictive multicast group management scheme

The threshold zone is calculated as the double of the window (*w*) parameter, so *w*=0.5 setup means that the viewpoint is always in a threshold zone, thus the client is requesting three camera views continuously. Even if *w*=0.5 there is no guarantee that required camera stream is received in time and no starvation occurs due to network congestion. The simulation results are shown in Fig. 17. If the viewpoint velocity is too high or the network latency increases, the required camera streams will not be available at the proxy server.

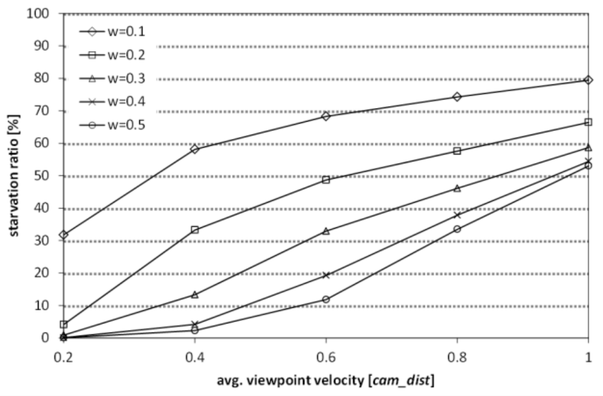

Fig. 17. Starvation ratio in function of viewpoint velocity and window size

According to the simulation results, if the window size (*w*) is set independently from the viewpoint velocity (*v*), the starvation ratio can be very high. However, it is important to note that the quality degradation effect of the starvation significantly depends on its duration. Using the proposed scheme for the window size setup, the starvation ratio can be kept low. E.g., when *v*=0.2 and *v*=0.4, the RTT is 60 ms and the timeslot between the viewpoint shifts is 0.05 s, the proposed window size according to (9) is *w*=0.24 and *w*=0.48, respectively. Using the proposed predictive multicast group management scheme, the starvation ratio in both cases is below 5% as Table 3. shows.

Table. 3. Starvation ratio in case of the proposed scheme

viewpoint velocity (*v*)	proposed window size (*w*)	starvation ratio
0.1	0.12	3.93%
0.2	0.24	4.45%
0.3	0.36	4.22%
0.4	0.48	4.42%

Controlled window size setup can minimize the starvation effect as analyzed in the next simulation scenario. The comparison of viewpoint velocity values and the caused starvation ratios are presented in Fig. 18. Based on the obtained measurement results, if the window size is set based on the velocity of the viewpoint, the synthesizer algorithm will get the camera views in time in more than 95% of the cases as the results show in Table 3. While setting the threshold zone too narrow, the starvation ratio can reach even 57% that makes the FVV service unacceptable.

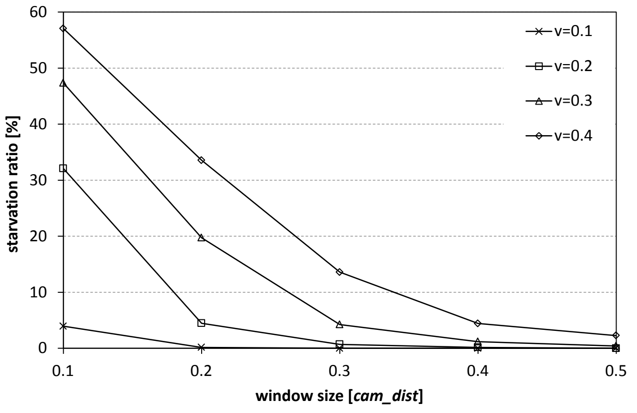

Fig. 18. Starvation ratio in case of different velocity values

In order to provide high quality synthetized viewpoint videos, the deployed number of cameras in the FVV system can be very high. By increasing the number of high bitrate color and depth cameras, the required streams will become more unique and there will be more camera streams that are not requested at all, or only few proxy servers are joined to a specific camera multicast group. Therefore, the multicast join latency will increase, because the probability that a router in *Section_1* already receives the requested stream is lower, so join message and the video packets will travel on longer path. Hence, we have measured how the number of FVV cameras affects the starvation. The obtained results are presented in Fig. 19.

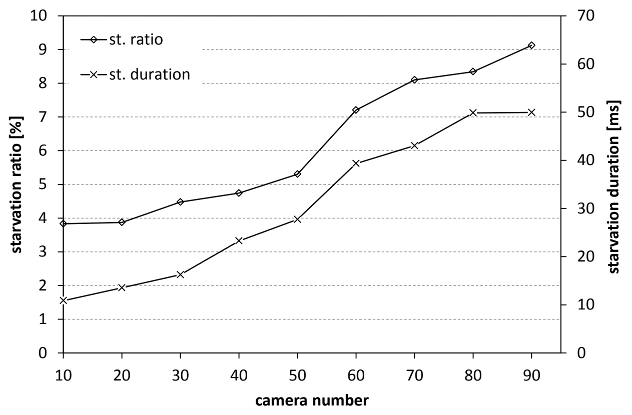

Fig. 19. Starvation ratio and duration

According to the obtained results, as the starvation ratio as the duration is significantly higher if more cameras are used. If only 10 cameras are deployed, ca. 35 users are joined to each camera multicast group, while using 90 cameras this number is only 3.9 in average.

The number of users is the other parameter that influences the number of multicast group members of a camera view. If the FVV system serves more customers, the proxy servers will join more multicast groups, so the latency of camera stream reception can be decreased. The reason is the same as it was described in the previous scenario. Namely, the routers in the FVV network already forwards the desired multicast views to other proxy servers with higher probability. The measurement results are presented in Fig. 20.

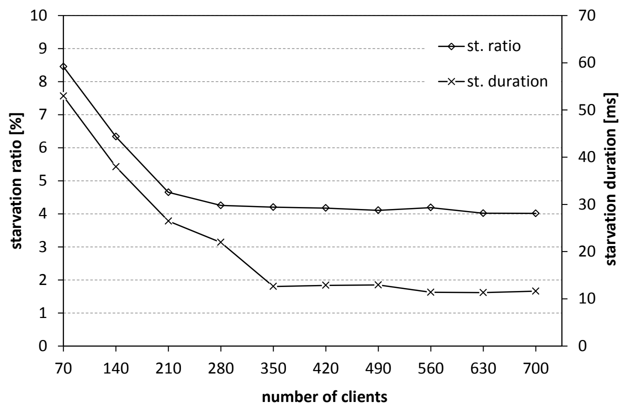

Fig. 20. Starvation ratio and duration from the number of users point of view

Low number of users significantly decreases the performance of the FVV service. According to the measurements, the variation of starvation ratio and its duration became negligible when the number of clients was higher than 300. The reason is that all the routers already forwards all camera streams.

5. Conclusion

Free viewpoint video is a promising approach to offer freedom of users' perspective while watching multi-view video streams. Each viewpoint is synthetized from at least two high bitrate color camera and depth camera streams that are used to capture the scene. The delivery of camera views required for viewpoint generation can overload the network without multicast streaming, however late multicast group joins messages may cause the starvation of the FVV renderer process. Both stream delivery and viewpoint generation are resource hungry processes leading to scalability issues in a complex network with high number of users. In this paper a distributed viewpoint synthesis approach and prediction based multicast group management scheme were presented in order to offer scalable solutions for new FVV services. Our aim was to propose optimal layout of the so called proxy servers that are responsible for viewpoint synthesis and transcoding in order to minimize the overall traffic in the network but avoiding link congestions. Besides analytical analysis of *k*-ary tree networks, we investigated the achievable gain by simulations. For simulation analysis a simple hierarchical network with several nodes was used, where the runtime of the brute force optimization algorithm is still acceptable. Optimal placement of proxy servers does not guarantee seamless viewpoint changes, therefore a threshold based multicast group management approach was also proposed. According to the introduced scheme, three camera views are forwards at the camera section borders, instead of the two necessary camera streams. We have formulated how to calculate the threshold area at the camera borders in order to minimize the starvation ratio and its duration. The proposed threshold calculation depends on the viewpoint velocity and the network delay. The obtained results show that the viewpoint synthesizer algorithm gets the camera views in time in more than 95% of the cases if the proposed scheme that is based on the velocity of the viewpoint and the RTT is used. Free viewpoint video streaming is a new form of perspective sensitive media delivery that was not intensively investigated before. Hopefully, FVV streaming will become a popular interactive multimedia service of the near future.

Acknowledgments

The research was supported by the European Union's Seventh Framework Programme under grant agreement n° 288502 (CONCERTO project). The author is grateful for the support of the Hungarian Academy of Sciences through the Bolyai János Research Fellowship.

References

[1] L Chiariglione, Cs A Szabó, “Multimedia Communications: Technologies, Services, Perspectives, Part II: Applications, Services and Future Directions”, Infocommunications Journal VI:(3) pp. 51-59., 2014

[2] M. Levoy and P. Hanrahan., "Light field rendering", Computer Graphics, Proceedings. SIGGRAPH96, August 1996

[3] 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, May 2004, pp. 93-104

[4] Masayuki Tanimoto, Mehrdad Panahpour Tehrani, Toshiaki Fujii, Tomohiro Yendo, “Free-Viewpoint TV”. IEEE Signal Process. Mag. 28(1): pp. 67-76, 2011

[5] K. Mueller, P. Merkle, A. Smolic, and T.Wiegand, “Multiview coding using AVC,” MPEG2006/m12945, 75th MPEG meeting, Bangkok, Thailand, Jan. 2006

[6] P. Merkle, A. Smolic, K. Mueller, T. Wiegand, “Efficient prediction structures for multiview video coding”, IEEE Transactions on Circuits and Systems for Video Technology, Special Issue on Multiview Video Coding and 3DTV, 2007

[7] Fehn C. “3D-TV using depth-image-based rendering (DIBR)”, Proceedings of Picture Coding Symposium, San Francisco, CA, U.S.A., December 2004.

[8] Guan-Ming Su, Yu-Chi Lai, Andres Kwasinski, and Haohong Wang, “3D video communications: Challenges and opportunities”, nternational Journal of Communication Systems, Vol. 24/10, pp. 1261-1281., October 2011

[9] Strohmeier D, Tech G., “Sharp, bright, three-dimensional -- open profiling of quality for mobile 3DTV coding methods”, Proceedings of the SPIE, 2010.

[10] Tech G, Smolic A, Brust H, Merkle P, Dix K, Wang Y, Muller K, Wiegand T, “Optimization and comparison of coding algorithms for mobile 3DTV”, IEEE 3DTV Conference, Potsdam, Germany, 2009.

[11] Merkle P, Morvan Y, Smolic A, Farin D, Muller K, Wiegand T, “The effects of multiview depth video compression on multiview rendering”, Signal Processing: Image Communication 2009, 24(1):73–88.

[12] ISO/IEC JTC 1/SC 29/WG 11. Committee Draft of ISO/IEC 23002-3 Auxiliary Video Data Representations. WG 11 Doc. N8038. Montreux, Switzerland, April 2006.

[13] M. Domański, M. Gotfryd, and K. Wegner, "View synthesis for multiview video transmission," in The 2009 International Conference on Image Processing, Computer Vision, and Pattern Recognition, Las Vegas, USA, 2009, pp. 1-4.

[14] S. Jo, D. Lee, Y. Kim, Ch. Yoo, “Development of a simple viewpoint video system”, IEEE Int. Conf. Multimedia and Expo, Hannover, June 2008, pp. 1577-1580.

[15] H. Kimata, S. Shimizu, Y. Kunita, M. Isogai, K. Kamikura, Y. Yashima, “Real-time MVC viewer for free viewpoint navigation”, IEEE Int. Conf. Multimedia

[16] J. Starck, J. Kilner, and A. Hilton. A Free-Viewpoint Video Renderer. Journal of Graphics, GPU, and Game Tools, 14(3):57-72, Jan. 2009.

[17] Aljoscha Smolic, “3D video and free viewpoint video-From capture to display”, Pattern Recognition Vol. 44 (9), pp. 1958-1968., September 2011

[18] Zefeng Ni; Dong Tian; Bhagavathy, S.; Llach, J.; Manjunath, B.S., "Improving the quality of depth image based rendering for 3D Video systems," Conf. on Image Processing (ICIP), 2009, 7-10 Nov. 2009

[19] http://www.liberovision.com/

[20] Zhun Han; Qionghai Dai, "A New Scalable Free Viewpoint Video Streaming System Over IP Network," Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on , vol.2, no., pp.II-773,II-776, 15-20 April 2007

[21] Goran Petrovicand Peter H. N. de With, “Near-future Streaming Framework for 3D-TV Applications”, ICME2006

[22] Gurler, C.G.; Görkemli, 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

[23] E. Kurutepe, M. R. Civanlar, and A. M. Tekalp, “Client-driven selective streaming of multiview video for interactive 3DTV”, IEEE Trans. Circuits Syst. Video Technol., vol. 17, no. 11, pp. 1558–1565, Nov. 2007.

[24] 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.

[25] 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

[26] HO, Ting-Yu; YEH, Yi-Nung; YANG, De-Nian. “Multi-View 3D Video Multicast for Broadband IP Networks”, arXiv preprint arXiv:1410.3977, 2014.

[27] Chakareski, J., “Adaptive multiview video streaming: challenges and opportunities”, Communications Magazine, IEEE , vol.51, no.5, pp.94,100, May 2013

[28] G. Petrovic and D. Farin, "A distributed delivery model for 3D-video streams." Proceedings of the First International Conference on Immersive Telecommunications, ICST, 2007.

[29] Toni, Laura, Gene Cheung, and Pascal Frossard. "In-Network View Synthesis for Interactive Multiview Video Systems." arXiv preprint arXiv:1509.00464, 2015

[30] Miao, Dan, et al. "Resource allocation for cloud-based free viewpoint video rendering for mobile phones." Proceedings of the 19th ACM international conference on Multimedia. ACM, 2011.

[31] J. Chuang and M. Sirbu, “Pricing multicast communication: A costbased approach,” presented at the INET, 1998.

[32] Graham Phillips, Scott Shenker, Hongsuda Tangmunarunkit, “Scaling of multicast trees: comments on the Chuang-Sirbu scaling law”, SIGCOMM '99, New York, USA, 1999.
[33] H. Kellerer, U. Pferschy, and D. Pisinger. Knapsack Problems. Springer, 2004
[34] Fenner B. et al., Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification, RFC 4601, August 2006
[35] A. Adams, J. Nicholas, W. Siadak, Protocol Independent Multicast - Dense Mode (PIM-DM), RFC 3973, January 2005
[36] Ns-2 – Network Simulator, http:///www.isi.edu/nsnam/ns/index.html