Journal of Network and Computer Applications 34 (2011) 137–150

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Peer-to-peer multipoint video conferencing with layered video

İstemi Ekin Akkuş [a], Öznur Özkasap [b,*], M. Reha Civanlar [c]

[a] Max Planck Institute for Software Systems, Kaiserslautern, Germany
[b] Department of Computer Engineering, Koc University, Istanbul, Turkey
[c] Department of Electrical and Electronics Engineering, Ozyegin University, Istanbul, Turkey

ARTICLE INFO

Article history:
Received 12 January 2010
Received in revised form
10 July 2010
Accepted 13 August 2010

Keywords:
Layered video
Peer-to-peer
Multipoint video conferencing
Scalable video
Multiple description coding
Multi-objective optimization

ABSTRACT

A peer-to-peer (P2P) architecture for multipoint video conferencing using layered video coding at the end hosts is proposed. The system primarily targets end points with low bandwidth network connections and enables them to create a multipoint video conference without any additional networking and computing resources beyond what is needed for a point-to-point conference. For P2P multipoint video conferencing applications, wide-area collaboration is significant for connecting participants from different parts around the globe to support collaborative work. In our system, peers collaborate for streaming video, and the motivation behind the use of layered video is to overcome the problem of denying video requests by peers and assure that each participant peer can view any other participant at any configuration. Layered video encoding techniques usable within this architecture are discussed. A protocol for operating the system has been developed, simulated and its performance has been analyzed. Furthermore, a multi-objective optimization approach has been developed to simultaneously minimize the number of base layer receivers and the delay experienced by the peers while maximizing the granted additional requests to support peers having multiple video input bandwidths. The use of the multi-objective optimization scheme is demonstrated through an example scenario and simulations. A prototype has also been implemented, and the system has been formally specified and verified.

1. Introduction

The Internet has revolutionized people's communication methods by first replacing traditional pen&paper letters with e-mail and then traditional telephony with voice over IP (VoIP). Also, image and video coding have become more common with the increasing computing power and its decreasing cost. Unfortunately, the increase in the universal access bandwidth to the Internet has not been as steep as that of the computing power of end hosts. Besides, its cost does not become cheaper as speedily. Several Instant messaging (e.g., ICQ, Microsoft Messenger) and voice over IP applications (e.g., Skype, VoipBuster) allow pair-wise video communications; however, multipoint (MP) video conferencing is still not very popular mostly because of the bandwidth demands of video transportation. Low bandwidth connections like wireless GPRS and even 3G systems are barely enough for point-to-point video communications let alone supporting MP video. Moreover, users tend to consume as much of the available bandwidth as possible to increase their video quality and, hence, a MP video system that increases the demand for bandwidth with the number of participants cannot be popular.

The bandwidth demand of a MP video system can be reduced using a special network-based equipment called Multipoint Control Unit (MCU) (ITU-T Study Group XV—Recommendation, 1993). The MCU acts as a single-point recipient for each participant, thus needing a large bandwidth connection only for itself. It prepares a MP video representation that can fit into a smaller bandwidth and sends it to each participant. However, because of the complexity and cost of the operations of the MCUs, they are mostly used by large business applications that can afford such equipment. They also suffer from single-point-of-failures and hence are not failure transparent.

Multicasting is another approach to reduce bandwidth demands of MP video conferencing whenever the underlying network supports it. The additional advantage of a native multicast-based solution is the reduced operational complexity (Civanlar et al., 1997), but it requires router support. Deploying multicast-supporting routers on the global Internet has not been popular since they may increase operational and security risks. Therefore, MP video conferencing using this approach is not practically viable, either.

An alternative approach is to use P2P principles to distribute video signals efficiently among participants in MP conferencing.

* Corresponding author. Tel.: +90 212 3381584.
E-mail address: oozkasap@ku.edu.tr (Ö. Özkasap).

1084-8045/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2010.08.006

There exist several P2P video streaming and conferencing systems including (Chu et al., 2001; Cui and Nahrstedt, 2003; Tran et al., 2003; Hosseini and Georganas, 2003; Civanlar et al., 2005; Luo et al., 2007; Ponec et al., 2009). We review and discuss related studies in Section 2. In this context, our earlier work proposes a P2P MP video conferencing approach (Civanlar et al., 2005). That system is based on a distributed P2P approach, where peers collaborate for streaming video of source peers, and each participant could see the video of any other participant in most cases. Wide-area collaboration is significant for MP video conferencing applications that connect participants from different parts around the globe to support collaborative work.

In this article, we propose a P2P MP video conferencing system based on layered video. The assumption is that each peer in the system is able to send and receive one full quality video stream at a time. That is, our P2P MP video conferencing approach would be able to work with peer and network resources needed for a point-to-point video conference. A major feature of our system is the use of layered video (i.e., base + enhancement layers). By using layered video encoding, a peer can initiate a video stream and forward a video stream both in half quality. The motivation behind the use of layered video is to overcome the problem of denying video requests by peers and assure that each participant peer can view any other participant at any configuration. Although some users may have to view base layer quality video, we show that this is only a small percentage of the participant count and tends to decrease as the participant count increases; thus, making the system scalable. In this study, we formally specify and verify the correctness our protocol. We also propose a multi-objective optimization approach and corresponding formulations for the optimal operation of the system.

Major contributions of our study are summarized as follows.

- We propose a P2P MP video conferencing system that makes use of layered video. The participants (i.e., the peers) are assumed to have at least the networking resources that are enough to be used in a point-to-point video conference, that is, they are able to send one video signal and receive one video signal at the same time.
- A fully distributed algorithm and architecture of the system is developed and presented. Layered video techniques that can be employed within our system are described. Although all participants can receive one other participant's video signal of their choice, some peers only receive base layer quality video. Optimizations to minimize the number of base layer receivers are proposed and presented along with simulation results.
- A multi-objective optimization framework has been developed to consider the end-to-end delays between peers and their heterogeneous bandwidth connections to the Internet. Objective functions to minimize the number of base layer receivers, to minimize the maximum delay experienced and to maximize the number of additional requests granted are defined and derived. Formulations to achieve these objectives are developed.
- We also describe our prototype implementation, peer protocol details, and summarize formal specification and verification aspects of the system.

The article is organized as follows. Next section presents related work in P2P MP video streaming and conferencing. The system description and the use of layered video are given in Section 3. Section 4 presents optimizations to minimize the number of base layer quality video receivers in a given configuration, the effect of multiple outputs and simulation results. Our multi-objective optimization framework is discussed in Section 5. Section 6 presents the details of our system implementation and peer protocol. The formal specification and verification aspects of the application are summarized in Section 7. Finally, Section 8 includes the concluding remarks and suggests future directions.

2. Related work

In this section, we first review related work on P2P video streaming applications and video conferencing tools discussing their benefits and shortcomings. Then, we describe well known, free or commercial conferencing systems, and how they address the challenge of MP conferencing. We also discuss how our study differs from the related prior work.

2.1. P2P video streaming and conferencing systems

There are a large number of P2P video streaming systems reported in the literature. A recent study (Liu et al., 2008) provides a survey on P2P solutions proposed for live as well as on-demand video streaming. Several popular techniques for P2P media streaming on the Internet utilize multicast model at the application layer where the main benefit is overcoming the lack of large-scale multicast deployment at the network layer. In the application layer multicast – also known as End System Multicast (ESM), P2P Multicast, Overlay Multicast – end hosts are used to relay data instead of routers as in native multicast.

Narada (Chu et al., 2000) is a self-organizing and self-improving protocol for conferencing applications and adapts to network dynamics. After maintaining a connected graph called mesh among the peers, Narada constructs a shortest path spanning tree on the mesh whenever a source wants to transmit content to a set of receivers. The mesh is improved by adding links or removing them in an incremental fashion, considering stress, i.e., the number of identical copies of a packet carried over a physical link, relative delay penalty, ratio of the delay between two members in the mesh to the unicast delay between them, and resource usage. Results indicate that Narada is a promising approach for conferencing applications in dynamic and heterogeneous Internet settings.

An implementation of video conferencing through ESM has been presented in Chu et al. (2001). Although the system employed P2P techniques for MP video conferencing, it assumes that the peers have large upstream bandwidths and there is only a single source. Although its applicability to multi-source video conferencing is mentioned, its practical usage in this context is not that common and related performance results are not reported. Also, since the links are added by probing, maintaining the mesh becomes costly as the group size increases.

NICE (Banerjee et al., 2002) is an application layer multicast protocol which aims larger receiver sets than targeted in Narada (Chu et al., 2000). NICE clusters peers into a hierarchical structure. The hierarchy of clusters is useful for the scalability of the system. The hierarchical structure implicitly defines the multicast overlay paths, where each cluster head forwards data to other peers in the same cluster. The cluster heads form another cluster in the higher level and receive the data from their heads. Clustering also brings the advantage of faster recovery on leaves or failures of peers, since it is localized. The increase in the control overhead is logarithmic as the group size increases, which makes NICE scale better than Narada for large receiver sets. The performance may be increased by using randomized forwarding when there are high packet losses and host failures (Banerjee et al., 2003).

ZIGZAG (Tran et al., 2003) is another P2P system developed for single-source media streaming for large receiver groups like NICE. ZIGZAG and NICE differ in their multicast tree construction and maintenance mechanisms. Although ZIGZAG also possesses a

hierarchical cluster structure for peers, the cluster members are not used to forward the content to the peers. Instead, the so-called *associate* heads from the upper layer are used. This gives the ability of recovering fast from failures. ZIGZAG also targets to maintain the height of the multicast tree and thus, to minimize the end-to-end delay from the server to the peers.

Cui and Nahrstedt (2003) proposed a layered P2P streaming mechanism for on-demand media distribution. This work points out the asynchrony of user requests and heterogeneity of peer network bandwidth. As the solution, cache-and-relay and layer-encoded streaming techniques are proposed. The solution has been shown to be efficient at utilizing peers' bandwidth, scalable at saving server bandwidth consumption, and optimal at maximizing streaming quality of peers.

Hosseini and Georganas reported a 3D video conferencing application using ESM (Hosseini and Georganas, 2003). The authors point out that the current applications for P2P media streaming are focusing on distributing a video content, be it on-demand or live, from a single source to a large group of receivers. However, a video conferencing application targets a smaller group, mostly from 4 to 10 participants, where each of these participants can be the source of a video signal. They prefer a centralized algorithm for tree construction for the recipients since it can react faster to changes during join or leave events. Rendezvous Points (RP) decide how the multicast trees for different sources are built and which peer will be acting as a parent for another peer. Although using a centralized approach decreases reaction time to changes and makes good use of the idle bandwidths, it suffers from single-point-of-failures. Controlling multiple multicast trees also increases operational complexity.

Ponec et al. considered P2P multi-rate multi-party conferencing systems (Ponec et al., 2009). In a multi-rate setting, different receivers in the same group can receive video at different rates using layered video. In particular, that study focuses on issues related to multi-rate setting in comparison to the case of single-rate video (Chen et al., 2008). It studies optimal usage of peer uplink capacities, and for P2P utility maximization in a multi-rate multicast it provides a novel multi-tree formulation. Lim et al. proposed another approach named N-Tree, a bandwidth fair application layer multicast for multi-party video conferencing (Lim et al., 2009). It builds a distribution tree for each source, and aims to satisfy requirements of latency and multicast bandwidth fairness. The N-Tree algorithm is shown to be convenient for video conferencing with small number of participants.

Application layer multicast protocols can be classified according to their construction of the data network, as receiver-driven, source-driven, and data-driven (Silverston and Fourmaux, 2006). In the source-driven approaches such as Narada (Chu et al., 2000, 2001), NICE (Banerjee et al., 2002, 2003), ZIGZAG (Tran et al., 2003), and PeerCast (Bawa et al., 2003), there is a single source of the content and peers cooperatively stream the content from the source to the other receivers. The system builds a control plan among the peers. The data plan is always a spanning tree using the control plan with its root at the sender. In the receiver-driven approach, the receiver actively decides, which peers should be the senders of the desired video. Multiple sources contribute parts of the content and the receiver puts them together. Spanning trees are built at the receivers so that multiple senders can be organized to send data to them. The data-driven approach does not clearly separate both plans; the peers exchange information about the availability of the pieces of the media. No precise direction for the data flow is defined.

Our prior work based on a P2P approach for multipoint video conferencing (Civanlar et al., 2005) mainly assumes that the participants (i.e., the peers) could be the source of one video signal only. Also, they could only receive and send one video signal at a time. In other words, they have the computational power to produce one video signal and their bandwidth is limited and only enough for streaming one video signal upstream and downstream. This way, the networking and computing resources of an MP video conference do not exceed that of a point-to-point video conference.

The restriction that the upstream bandwidth is only enough to send one video signal may cause a request for a particular participant's video to be denied. This is because a peer cannot send its own video signal if it is already relaying another peer's video signal and a peer cannot relay if it is already sending its own video signal. Civanlar et al. showed that this does not happen most of the time, and in fact each participant could view another participant's video under most practical cases (Civanlar et al., 2005). However, the number of the cases in which a request is denied increases as the number of the participants increases causing the system to be unscalable.

2.2. Popular conferencing systems and multipoint support

We now consider well known, free or commercial conferencing systems such as Skype, Gmail chat and Apple's iChat, and discuss how they address the challenge of MP conferencing.

Skype provides MP audio conferencing, but video conferencing in Skype is only point-to-point. Recently, they have added a group calling feature which is limited to 5 persons; however, it is not known what the bandwidth requirements are (Skype, online). GMail chat has a video conferencing module (Google gmail chat, online), but it works only for two persons. Also, it uses a special hardware provided by Vidyo (Vidyo, online). Apple's iChat provides MP video conferencing; however, one of the peers has to have enough download and upload bandwidth to initiate the conference, where it presumably acts as a software MCU (Apple iChat, online). BNI solutions' IPContact provides MP video conferencing without an MCU; however, all participants need to send audio and video to every other participant, requiring much more bandwidth than a point-to-point video conference (BNI solutions' IPContact, online). Nefsis provides dedicated cloud computing resources for video conferencing. Users automatically connect to geographically close servers distributed on the Internet to have a low-latency experience (Nefsis, online). On the other hand, our solution does not depend on a server's existence to provide multipoint video conferencing. A recent study (Lu et al., 2010) provides a survey of free multi-party video conferencing systems, and a measurement work to compare four representative systems including Nefsis in terms of their performance, mechanisms and quality of experience.

As a general comparison to prior work, our approach in this study is based on the use of a distributed P2P architecture which does not need any special hardware or network infrastructure support. Its P2P architecture prevents single-point-of-failures and provides failure transparency. In contrast to the prior work, our approach does not assume high bandwidth connections and it makes use of layers of video with low bandwidth requirements (Akkus et al., 2006). There is no additional networking and computing resources needed at the end points more than that of a point-to-point video conference. Our platform targets small groups of participants (i.e., participant count <10), where all peers can act as sources of video without interrupting each other. Next section describes the details of our system model and use of layered video.

3. System description and use of layered video

3.1. System description

Each peer in our system is assumed to be able to send and receive one full quality video stream at a time. Thus, our approach to P2P multi-point video conferencing would be able to function with limited peer and network resources just needed for a

point-to-point video conference. We define a *chain* as the ordered group of peers that receive the same video signal. A peer sending its own video signal is named as *head of a chain*. A *relay* is a peer that is forwarding the video signal it is receiving, to another peer. The use of layered video allows a peer to send two separate video signals in lower quality instead of sending one video signal in full quality when necessary (i.e., in the case that a peer is already relaying another peer's video signal and also needs to send her own video signal). The motivation behind the use of layered video is to overcome the problem of denying video requests by peers and assure that each participant peer can view any other participant at any configuration. This is accomplished by allowing a peer to be a relay and a head of a chain at the same.

Our algorithm describing the actions to be performed by a participant who receives a video request is given in Fig. 1. In particular, the algorithm shows the actions of a participant named v when another participant u requests the video of the participant v. It checks if v is the head of a chain as well as if v is relaying for another peer w. All possible actions for placing u in a chain and the video quality u will receive (base or full) are defined accordingly. Not all configurations require a peer to send two separate video signals, so the upload bandwidth can be used only for relaying purposes. In this case, using layered video allows to make use of the available bandwidth to enhance the video quality that is being relayed, but also makes it possible to accommodate requests that are for the relaying peer's own video signal if any.

As shown in the example chain of Fig. 2(a), without using layered video, the request of participant 4 to view participant 2's video must be denied since the upstream bandwidth of participant 2 is already in use. Also, moving participant 2 to the end of the chain does not help because, then, participant 1 cannot see participant 3 anymore. The use of layered video and how it solves the problem so that participant 4's request can be granted, is shown in the example chain of Fig. 2(b).

The use of a distributed P2P architecture brings the advantage of enhanced fault tolerance. Peers can leave the system at any time, either at will or unintentionally (i.e., due to a crash). If a peer is sending its own video signal (i.e., the peer is a head of a chain) when this happens, the members of the chain would immediately notice it and would act accordingly (i.e., stop receiving the video signal of the peer that has just left the system and may request another peer's video signal). If this peer is a relaying peer in a chain, the head would notice the situation from missing keep-alive messages that peers must send to their respective

Participant *u* requests video signal of participant *v*

```
If v is chain head
    If v is relaying for another peer w
        If u can forward base (i.e., u is not a chain head
                or u is sending base quality to its own chain)
            Insert u into v's chain with base only
        Else
            Add u at the end of v's chain with base only
    Else
        If u can forward full (i.e., u is not a chain head)
            Insert u into v's chain with full quality
        Else
            Add u at the end of v's chain with base only
Else
    If v is relaying for another peer w
        Start new chain with base only and add u
    Else
        Start new chain with full quality and add u
```

Fig. 1. Algorithm for granting a video request.

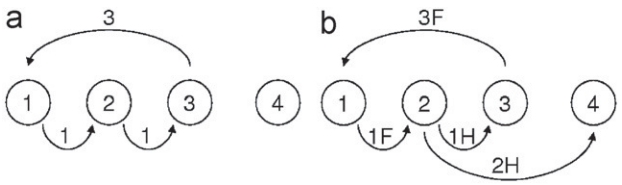

Fig. 2. (a) Participant 4 cannot see participant 2. (b) Participant 4's request can be granted. (F stands for full quality video (base & enhancement), H stands for half quality video (base)).

heads and rearrange its chain accordingly. A peer that is both a head and a relaying peer would not cause a problem either when it leaves or experiences a problem. The members of the leaving peer's chain would react when stopping to receive the video signal and the head would rearrange its chain accordingly.

3.2. Use of layered video

The base layer is used whenever an intermediate peer, a relay, receives a video request from another participant and it cannot be moved to the end of its chain. The relay then stops forwarding the video signal it is receiving in full quality, that is, the base and the enhancement layers, and starts to forward only the base layer. The rest of its upstream bandwidth is used for sending the base layer of its own video signal to the participant that requested it. In this section, two approaches for generating layered video usable within the system, namely scalable video and multiple description coding, are described: We explain the issues related to these approaches and how they can be implemented with standard encoders, such as JSVM (JSVM Software, online) and Nokia (Nokia H.264 codec, online).

3.2.1. Scalable video approach

Scalable video coding techniques are gaining popularity with H264/SVC allowing encoding of the video in different quality layers so that according to the bandwidth restrictions corresponding layers can be transmitted or stored (Draft ITU-T Recommendation and Final Draft International Standard, 2003). We can use all scalability types or their combinations to obtain our two layers. To demonstrate the related issues, we describe two examples of scalability that can be employed in our system.

Temporal scalability: The bandwidth requirements of each layer should be equal. This can be achieved simply by dividing the video stream in the temporal dimension. As an example, the base layer can have half the frame rate of the original video.

In Table 1, we show the average PSNRs for base and enhancement layer frames when a standard test sequence (Foreman) is encoded using JSVM (JSVM Software, online) with 7.5 frames in base layer and 7.5 frames in the enhancement layer at the given bandwidths. As can be seen from this example, if we use temporal scalability and if we divide the frames equally between the base and the enhancement layers, because of our equal bandwidth between layers constraint, the enhancement layer frames will have better quality. This introduces "quality jitter" which may be disturbing for the users. If we were to use the same quality for both layers, the bandwidth use of the enhancement layer would be less (Akkus et al., 2006). Alternatively, fewer frames can be included in the base layer, to make the SNRs equal, but the resulting base layer frame rate may be inadequate. For the example above, the base and enhancement layer PSNRs are equal if the base layer is set at 5 fps. One important issue is that once the base layer frame rate is selected, all the users in the chain will receive the same video. So, it will be difficult to let the users choose between the amount of "quality jitter" and "low frame rate".

Table 1
Average PSNR and bit rate values: Temporal and SNR scalable base and full quality layers.

Quality	Average			
	PSNR(Y)	PSNR(U)	PSNR(V)	Bit Rate (kbps)
Single Layer	34.3292	38.7423	40.0163	63.7320
Base Layer (Temporal)	31.8586	37.4623	38.5050	31.5066
Base + Enhancement (Temporal)	33.6630	38.8204	40.1227	63.9414
Base Layer (SNR)	30.3959	36.9938	37.7932	31.7730
Base + Enhancement (SNR)	32.9896	38.4540	39.6059	64.1106
Base (5fps)	32.05	38.40	38.88	29.02
Base + Enhancement	32.0210	38.44	38.90	59.06

Table 2
Bitstream contents of the full quality layer with SNR scalability.

Layer	Resolution	Frame rate	Bit rate	Minimum Bit rate
6	176×144	15	31.00	31.00
7	176×144	15	64.00	

SNR scalability: To achieve SNR scalability, JSVM uses progressive refinement (PR) slices whose coding symbols are ordered by their importance. PR slices can be truncated at any point, providing a rate interval rather than rate points (JSVM Software, online). So, one can extract a base layer at almost any required bandwidth. The SNR scalable encoding results for the above example can be found in Table 1. The bandwidth requirements for the corresponding video that can be extracted from the bitstream are shown in Table 2. The base layer can now have the same frame rate (i.e., 15 fps) as the full quality video and the bandwidth is shared equally between two layers. A similar solution can be obtained using spatial scalability as well. The problems with these last two approaches as opposed to temporal scalability are the reduced coding efficiency and the increased encoder complexity.

3.2.2. Multiple description coding approach

Multiple description coding (MDC) (Goyal, 2001) is another alternative solution for transmitting video in our P2P system. In one particular implementation of this approach, odd numbered frames and even numbered frames may be predicted only from each other, creating two independently decodable threads. This approach may also be used to increase the system's loss resilience by allowing forwarding of the description experiencing smaller number of packet losses at the relay nodes. The quality is increased if more than one description is received. However, there is redundancy in the layers because of the independent decodability feature causing the MDC approach to be inefficient compared to scalable coding. The results of the MDC approach for the above example are reported in Table 3 and in Table 4 using two publicly available codecs. Here, we used JSVM's base layer as an H.264 encoder. It has better results in terms of PSNR than (Nokia H.264 codec, online), because it uses two reference frames for encoding a frame. However, Nokia H.264 codec works in real-time on an ordinary PC. An important advantage of this approach is that it can easily be implemented with almost any video codec.

4. Minimizing the number of base layer receivers

Clearly an important operation target of our system is to reduce the number of users who have to receive base layer video only. To this end, we propose optimization heuristics that try to minimize this number. Finding an optimized solution to this problem would require *global* knowledge of the peers that receive base layer video. Instead, we propose a distributed, greedy algorithm and we show via simulations covering *every possible request configuration* that our algorithm works well.

4.1. Optimization heuristics

Let n be the number of participants and r be the set of video requests, defining a particular configuration. The number of full quality receivers is $k=f(n, r)$. The aim of the optimizations is to maximize k, when the number of participants changes and/or the request of a participant changes, namely $k' = f(n',r')$.

For example, as shown in Fig. 3, assume that a participant (e.g., 2) is relaying another participant's (e.g., 1's) video signal at full quality, i.e., base and enhancement layers. When yet another participant (e.g., 3) requests 2's video signal, 2 has to drop the relayed video signal (of 1) to base layer and forward only the base layer, so that it can send its own video to 3. This makes all participants receiving the relayed video from 2, and participant 3 to receive base quality video. Assuming that 2 is located right after 1 in a long chain, letting it relay base quality video would reduce the received video quality for a large number of participants. However, if 2 could be moved toward the end of the chain, this large number of participants can continue to receive full quality video.

If in the configuration of Fig. 3, participant 5 was also sending its own video signal, then moving 2 to the end of the chain would cause all the participants in 5's chain to receive base quality video. In this case, chain lengths for 2 and 5 can be compared and the participant with a longer chain can be moved to (or left at) the end. An example demonstrating this situation is depicted in Fig. 4.

Similarly, when inserting participants into chains, the lengths of the involved chains should be taken into consideration. If a participant is the head of a chain sending full quality video, we should avoid using it as a relay node. As an example, assume that a participant, P, that is a chain head sending full quality video, requests participant T's video signal and T is already sending at base quality. In this case, P could drop its video to base and relay T's video signal. However, doing so would cause the chain members of P receive only base quality video. A better solution would be adding P to the end of the chain, so that it would continue to send its video at full quality. However, if the last member, L, of the chain is also a chain head sending its video, a comparison between P's and L's chain lengths should be carried out. The participant with longer chain would go to the end of the chain, minimizing the number of base quality video receivers. This greedy approach ensures that every time a participant requests video from another one, the configuration stays with maximum number of full quality receivers.

Our complete decision algorithm including the optimizations to minimize the number of base layers is given in Fig. 5.

Table 3
Average PSNR and bit rate values: MDC scalable base and full quality layers. (H.264 - JSVM).

Quality	Average			
	PSNR(Y)	PSNR(U)	PSNR(V)	Bit Rate (kbps)
Single Layer	34.3292	38.7423	40.0163	63.7320
Base Layer (even frames)	32.5192	37.8689	38.9798	31.6908
Base Layer (odd frames)	32.4926	37.9254	38.9997	31.6272
Combined (even + odd frames)	32.5059	37.8972	38.9898	63.3180

Table 4
Average PSNR and bit rate values: MDC scalable base and full quality layers. (Nokia H.264 codec, online).

Quality	Average			
	PSNR(Y)	PSNR(U)	PSNR(V)	Bit Rate (kbps)
Single Layer	34.3292	38.7423	40.0163	63.7320
Base Layer (even frames)	30.2404	37.0603	37.8799	32.045
Base Layer (odd frames)	30.4107	37.0932	37.9618	32.167
Combined (even + odd frames)	30.32555	37.07675	37.92085	64.212

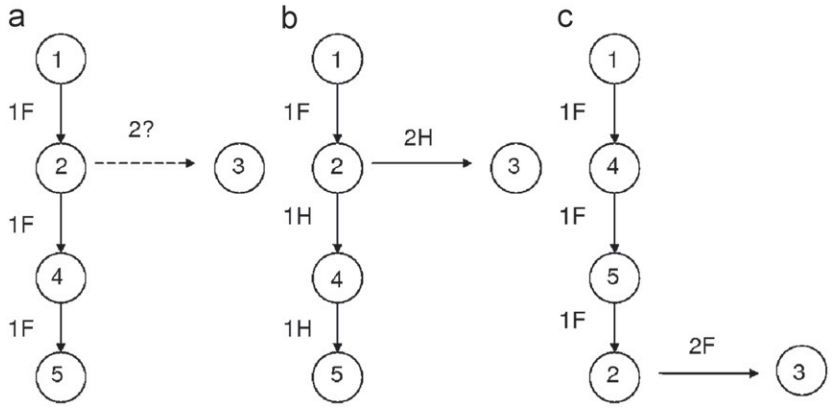

Fig. 3. An example of chain optimization (a) Participant 3 requests a relaying participant's video. (b) Chain without optimization. (c) Chain after Participant 2 is moved to end.

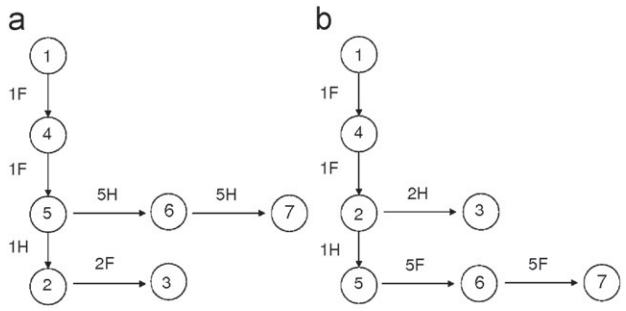

Fig. 4. (a) Chain after Participant 2 is moved to the end. (b) Chain after Participant 2 is moved just before the end.

The algorithm shows the actions (based on the optimizations described above) of a participant named v when another participant u requests video of the participant v. It checks the cases of whether v is the head of a chain, whether v is relaying for another peer w and if so whether v can be moved to the end of w's chain. All possible actions for placing u in a chain, changing the location of v (if needed based on the optimizations), and video quality (base or full) decisions are defined accordingly in the steps of the algorithm.

4.2. Validation via simulations

To evaluate our optimization heuristics, we performed simulations covering *all* possible request configurations with a conference up to 10 participants. A participant p in a conference with n participants can request the video of any other participant in the conference. This creates $(n-1)^n$ combinations. For example, in a conference with 4 participants, there will a total of $3^4=81$ separate cases. In each case, every participant requests the video signal of any other participant selected randomly. We ran our simulations for the algorithm without the optimizations (shown in Fig. 1) and for the algorithm using our optimization heuristics (shown in Fig. 5). For these simulations, we have created each possible request configuration (e.g., 81 for a conference with 4 participants) and obtained and analyzed the chains that were created. Fig. 6 shows that the percentage of cases where base layer is required (i.e., at least one peer receives base layer quality video) to all possible cases increases with the increasing number of participants; therefore, causing the probability of any

Participant *u* requests video signal of participant *v*

```
If v is chain head
    If v is relaying for another participant w
        If v can be moved to the end of w's chain
            Move v to the end of w's chain
            Reconsider u requests v
        Else
            If u can forward base (i.e., u is not chain head
                                   or u is sending base quality to its own chain)
                Insert u into v's chain with base only
            Else
                Check chainlengths of u and last member in v's chain
                If u has longer chain
                    Add u to v's chain with base only
                Else
                    Insert u into v's chain
                                   before the last member with base only
    Else
        If u can forward full (i.e., u is not chain head)
            Insert u into v's chain with full quality
        Else
            Check chainlengths of u and last member in v's chain
            If u has longer chain
                Add u to v's chain with base only
            Else
                Insert u into v's chain
                               before the last member with base only
Else
    If v is relaying for another participant w
        If v is relaying full for w
            If v can be moved to the end of w's chain
                Move v to the end of w's chain
                Add u to v's chain with full quality
            Else
                Swap v with first non-chainhead in w's chain
                Add u to v's chain with base only
        Else
            Add u to v's chain with base only
    Else
        Add u to v's chain with full quality
```

Fig. 5. Complete decision algorithm with optimizations to minimize the number of base layer receivers.

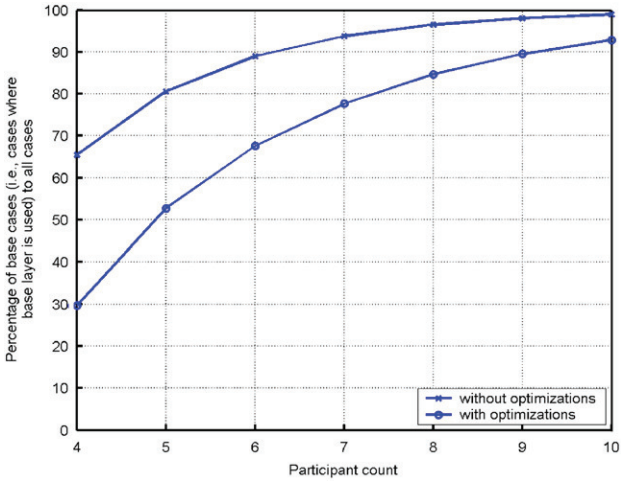

Fig. 6. Percentage of configurations containing base quality receivers versus number of participants.

participant to receive base layer quality video to increase (i.e., the bigger the conference, the higher the probability of a peer receiving base layer quality video). This increase is steeper without the optimizations.

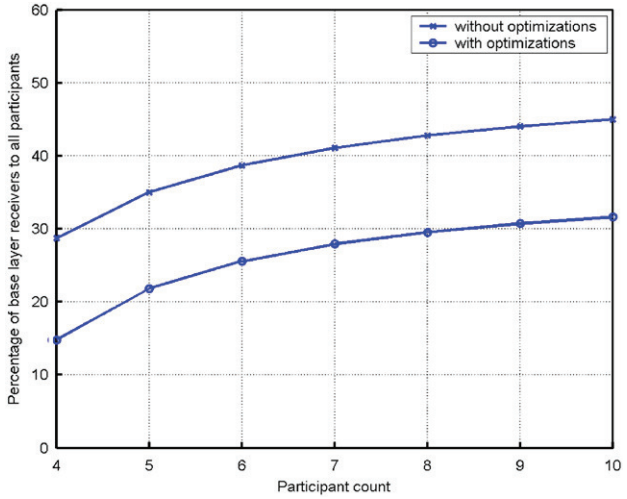

Fig. 7. Percentage of base quality receivers to all receivers under all configurations versus number of participants.

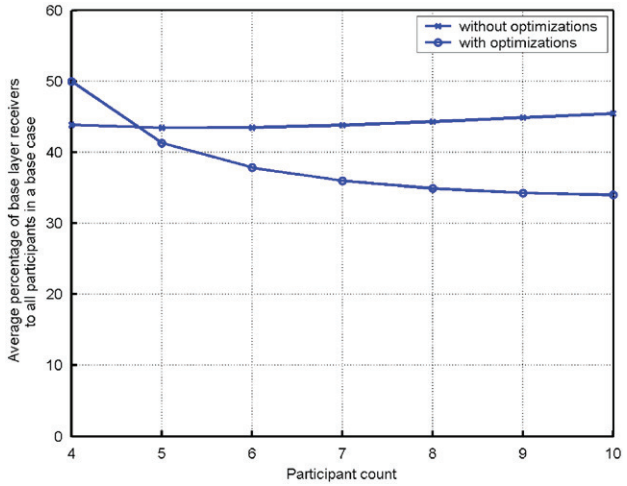

Fig. 8. Average percentage of base quality receivers versus number of participants.

Fig. 7 shows the percentage of total base layer receivers for a given number of participants. In a conference with 4 participants, there will be a total of 81 cases, with a total number of 324 receivers. The number of total base layer receivers in all 81 cases without employing the optimizations is 93, which gives 28.70% for participant count 4 (i.e., the first data point in Fig. 7 for the 'without optimizations' line). Our heuristics cannot fully prevent base layer quality video to be used; however, it helps to significantly reduce the total base layer receivers for a given participant count as shown in Fig. 7.

The number of base layer receivers increases with the participant count, even with our optimizations. However, Fig. 8 shows the average percentage of base layer receivers to the total participants, *only for cases where there was a base layer quality video receiver*. For example, in a conference with 4 participants, there were 24 such cases out of a total 81. The total number of participant receiving base layer quality video was 48, giving an average of 2 participants. So, Fig. 8 shows that only 50% of peers (i.e. 2/4) in such cases receive base layer quality video for 4 participants in a conference with our optimization heuristics. For the remaining 57 cases out of 81, there was no need for base

layer quality video and *all peers receive full quality video*. As shown in the figure, the percentage of average base layer quality video receivers decreases as the number of participants increases with our optimization heuristics.

From Fig. 8, it may look like without our optimizations in place, the system would still have a low percentage of base layer receivers. However, we should note that there were *many more base layer requiring cases* although the average percentage of base layer receivers seems manageable. For example, for a conference with 4 participants, there were 53 cases out of 81 requiring base layer quality video instead of 24 with optimizations. This number increases with the increasing participant count much more steeply without the optimizations than with the optimizations, as shown in Fig. 6. As a result, although the average may look close, the total number is much bigger as shown in Fig. 7, showing the benefits of our optimization heuristics.

Besides this issue, the geographical location of the users and their heterogeneous connection bandwidths would also play a role on the ordering of the participants in a chain. In that case, a trade-off between maximizing the number of full quality receivers, minimizing the maximum delay any participant experiences, and maximizing the number of granted additional requests must be made. Our proposed system targets small groups of participants. In order to support low-latency chains, and therefore to increase the perceived QoS, we are providing a multi-objective optimization framework, where the latencies between the peers are also considered while employing different chain configurations. These are addressed in the context of our multi-objective optimization framework (Section 5).

4.3. The effects of multiple outputs

The obstacle emerging from the assumption that the end-hosts do not have enough bandwidth was overcome by using two layers of video, so that virtually two output bandwidths were created. In real life, peers can have bandwidth that can support the case of two or more video signals are received and/or sent. In this section, we investigate the case of multiple video outputs. Effects of multiple inputs will be explained in Section 5.

When peers have multiple outputs, a relay receiving a video request, does not have to drop the forwarded video quality to half; it just uses the other output bandwidth to send its own video signal. In order to support these peers, the availability of a spare output is checked, before the requester is added to the chain with base quality. If there is spare bandwidth for a full quality video, the requester receives full quality.

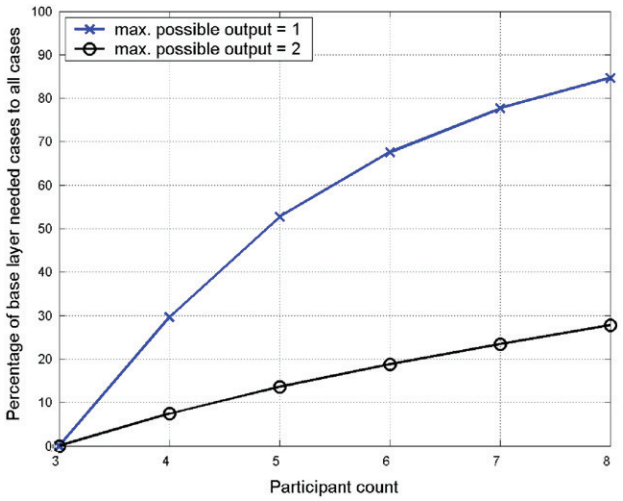

Fig. 9. Percentage of base layer needed cases in all cases vs. participant count.

Figure 9 shows that the increase of the percentage of cases, in which there are base layer receivers, in all cases is much smaller when the maximum possible output is 2 (i.e., 85% vs. 29% in a conference with 8 participants). In addition, the probability of a relay's dropping the forwarded video's quality to base layer decreases, since the spare output can be used for sending the relaying peer's own video signal whenever a request is received. This causes the percentage of average number of base layer receivers in a configuration to decrease.

5. Multi-objective optimization

Efforts to minimize the number of base layer receivers, as described in Section 4, do not consider the delay experienced by the peers, nor the heterogeneity of their connection bandwidths. Furthermore, it was assumed that end-points had low bandwidth connections that can support only one video signal to be sent and received, although some peers may have higher bandwidth and asymmetric (i.e., download rate $>$ upload rate) Internet connections. In this section, we describe our multi-objective optimization approach (Akkus et al., 2007) that aims to minimize the number of base layer receivers, minimize the maximum delay experienced in a chain, and maximize the number of additional requests granted.

5.1. Assumptions and problem definition

Each peer is assumed to have a connection bandwidth that can support at least one full quality video (i.e., base *and* enhancement layers) to be received and sent at the same time. Some peers may have download bandwidths to enable them to receive more than one full quality video simultaneously. Likewise, some peers may have upload bandwidths that can support more than one full quality video at the same time. However, having a spare upstream bandwidth is always beneficial, since it can be used without dropping the quality of the video sent. Therefore, we do not investigate them here.

We assume that each head of a chain obtains the packet delay time to its chain members, by gathering the round-trip-time (RTT) values during the session. RTT values are assumed to be symmetric, so that the delay in the direction from peer i to peer j (i.e., $d_{i,j}$) is the same as the delay in the direction from peer j to peer i (i.e., $d_{j,i}$). These delay values are stored in a table by the head and updated periodically. A time synchronized algorithm (Civanlar et al., 2005) may be used for measuring one-way delays accurately; however, the RTT information seems to be sufficient without complicating the system. The computational burden of the system is shown to be small (Civanlar et al., 2005). Delays other than the end-to-end delays (e.g., processing, switching, forwarding) are assumed to be negligible compared to the network delays.

We also assume that a participant can make at most two video requests, even if it has a bandwidth that is sufficient to view more than two video signals. The reason behind this assumption is that the focus of a participant cannot cover interactions with more than two peers at the same time. Although this may look like as an artificial assumption, the situation is similar to whenever a group of people interact with each other in a face-to-face conference. For example, as argued by Hosseini and Georganas (2003), in a video conference *mostly* two persons are in a participant's view. Also, if a peer were allowed to make three video requests, it might cause problems while granting requests. Suppose p already watches the videos of $r1$ and $r2$ and forwards them to other peers. p's third

request can be granted by another peer $r3$ by adding it to the end of $r3$'s chain where p would not have to relay it to another peer. However, if another peer s requests p's own video signal, the request would be denied, because p's uploading bandwidth can only support two base layers at a time and is already occupied fully (i.e., sending $r1$'s and $r2$'s video signals). Limiting the number of video requests to two prevents this situation and guarantees that each *first* request to be granted, even in the case of additional requests.

Each head that receives a video request needs to configure its chain accordingly to grant the request and to achieve the defined objectives. This is done using only local information, and brings the advantage of not setting up a global information exchange mechanism to be used whenever a request is made. Omitting this overhead also allows the head to decide fast, since the chain of a peer needs to be updated dynamically whenever a video request is received by the head.

One can optimize each of the operating objective separately, but as we will show in Section 5.3, these objectives can conflict, so that one solution for an objective would cause another objective to fail. Therefore, we need an approach that will optimize all objectives simultaneously. One can choose one objective to be achieved and set constraints for the others; however, this approach has the disadvantage of having to choose the main objective to be achieved. Moreover, particular values have to be selected for the constraints.

We claim that any comparison between the objectives to decide their order of importance cannot be justified. Instead, we combine these objectives by making use of participants' preferences. We believe that this is a reasonable solution, since the peers are the ones who would get affected by the chain configuration that is going to be employed by the head.

The solution space consists of the possible chain configurations headed by a peer v. The use of combinatorial optimization by exploring all the possible solution space gives a computational complexity of $O(n!)$. This would be problematic when the chain of a peer becomes very long (i.e., $l_v > 10$). Since our target group size for this application is small (i.e., participant count < 10), the enumeration of possible chains is not costly. This way, each head can update its chain configuration dynamically and in real-time.

5.2. Optimization objectives

In this section, we define the optimization objectives to consider the delays experienced by the peers and to support peers that have asymmetric bandwidth connections. In order to formulate the objectives, we define the following variables.

- i: id of the peer
- c: a *possible* chain configuration
- l_i: the length of the chain headed by peer i (i.e., the cardinality of the set consisting of the peers that receive peer i's video signal)

Let $f_{v,c}(o)$ be an integer valued function of the positions of peers that return the id of a peer given its position, o, in a *possible* chain configuration c headed by peer v. Then $f^{-1}_{v,\,c}(j)$, the inverse function of $f_{v,\,c}(o)$, gives the position of the peer with the id j in a *possible* chain configuration c.

Objective 1: *Minimize the number of base layer receivers*. The $k_{v,c}$ variable is defined to formulate the objective function. It gives the number of lower quality video receivers in a *possible* chain configuration c headed by peer v.

Consider the example chain configuration of peer 1 in Fig. 11(a). Peers 3 and 4 in peer 1's chain are also chain heads. Let $f^{-1}_{v,\,c}(j)$ be the inverse function of $f_{v,\,c}(o)$, so that it gives the position of the peer with the id j in a *possible* chain configuration c. In this example, $f^{-1}_{1,\langle 4,3,5\rangle}(3)=2$ and $f^{-1}_{1,\langle 4,3,5\rangle}(4)=1$. Let o be the smallest of these values, namely 1. Also, suppose that peer 3 has a chain length of 1 and peer 4 has a chain length of 2 as depicted in Fig. 11(a). So $k_{v,c}$, the number of lower quality video receivers in a possible chain configuration c headed by peer v, is calculated as

$$k_{v,c}=\begin{cases} 0 \quad \text{if } o=l_v \textbf{ or } \textit{no such peer} \\ l_{f_{v,c}(o)}+1+\sum_{j=o+1}^{l_v-1}(l_{f_{v,c}(j)}+1) \text{ otherwise} \end{cases}$$

The number of base layer receivers in a possible chain configuration c is 0, if $o=l_v$ (i.e., the corresponding peer is at the end of the chain) or there is no such peer that is acting as the head of a chain and a relay simultaneously, so that every peer receives full quality video. Otherwise, it is calculated as given above.

Remember that peers 3 and 4 are chain heads and relays at the same time. This means that the peers in peer 3's chain (i.e., peer 2) and the peers in peer 4's chain (i.e., peer 6 and peer 7) receive only the base layer, like peers 3 and 5. In this particular chain configuration of peer 1, the total number of base layer receivers is 2+1+1+1 = 5. (i.e., the chain length of peer 4 + peer 3 + the chain length of peer 3 + peer 5). Note that, if peer 5 was also a chain head; since it would not relay, its chain would receive full quality video and thus, would not be included in the sum. Consequently, the first objective function $g_v(c)$ is defined as

$$g_v(c)=k_{v,c} \tag{1}$$

Objective 2: *Minimize the maximum delay experienced by a peer*. Since the maximum delay in a chain configuration is experienced by the peer at the end of the chain, we aim to minimize the delay of that peer. We define

$cd_{v,c}$**:** delay experienced by the peer at the end of a *possible* chain configuration c headed by peer v.

The head receiving a video request calculates the delay value of each possible chain configuration by adding the one-way delay values between the peers.

$$cd_{v,c}=\sum_{j=1}^{l_v} d_{f_{v,c}(j-1),f_{v,c}(j)}$$

The second objective function $h_v(c)$ is defined as

$$h_v(c)=cd_{v,c} \tag{2}$$

Objective 3: *Maximize the number of additional requests granted*. The objective function takes the value of 1 if the additional request of a peer can be granted, whereas it is -1 if the request is to be declined. This is because a peer is allowed to make only one additional request if it has additional bandwidth (i.e., two requests in total) as described in Section 5.1. Every head investigates each possible chain configuration c whenever a peer makes an additional video request to the corresponding chain head.

$$s_{v,c}=\begin{cases} -1 \text{ if the request is not granted} \\ 1 \text{ if the request is granted} \end{cases}$$

The third objective function $m_v(c)$ is defined as

$$m_v(c)=s_{v,c} \tag{3}$$

5.3. Formulation

We use the weighted sum method (Zadeh, 1963) to determine the best solution. The issue of determining importance weights to be assigned to each objective is overcome by employing a preference mechanism. Peers being aware of the optimization

objectives choose one of the objectives as their preference. These will be exchanged during the initialization of the conference. Peers may change their preferences during the conference, but they need to inform the others. This should not be difficult, as they would know who is in the conference. This is a plausible assumption, because peers need to know other peers in the conference to make an arbitrary video request. Besides, our system targets a low number of participants. The assigned importance weights of each objective function f_i is defined as

$$w_{f_i} = \frac{p_{f_i}}{n} \tag{4}$$

where p_{f_i} represents the number of peers that prefer f_i to the other optimization objectives. n is the number of participants in the conference.

The importance weights are determined using the number of participants in the entire conference (i.e., n), rather than the number of participants in a corresponding chain. The reason is that a peer's preference (e.g., a head of a chain) may affect other peers' (e.g., the peers in its chain) received video quality (Akkus et al., 2007).

Consider the conference in Fig. 10 with 5 participants. Suppose that peers 2 and 3 prefer minimum delay and peers 1, 4 and 5 prefer maximum video quality. Peers 1 and 2 are geographically closer, so the chain configuration in Fig. 10(a) will be employed by the chain head (i.e., peer 1) to minimize the delay. This will force the other chain head (i.e., peer 2) to send its own video signal in base layer quality, although *all* peers in its chain prefer maximum video quality. Since they can receive only the base layer, their preferences will have no effect, even if they constitute the majority in the conference. Therefore, rather than using only the preferences in the corresponding chain, all peers' preferences are taken into account while determining the importance weights. As a consequence, the configuration in Fig. 10(b) should be used to satisfy the majority of the peers (i.e., 3 out of 5 peers).

Each head v would calculate the scaled versions of the optimization functions while determining which chain configuration they are going to employ whenever they receive a video request message. The formula for that is

$$f_{i,v,scaled}(c) = \frac{f_{i,v}(c) - f_{i,v,min}(c)}{f_{i,v,max}(c) + f_{i,v,min}(c)} \tag{5}$$

where $f_{i,v,min}$ and $f_{i,v,max}$ represent the minimum and the maximum value of that optimization function, respectively. Such scaling is necessary as the units of these objective functions are different and not comparable with each other (e.g., delay in *ms* versus number of peers). By scaling, the objective functions become comparable. The combined objective function would be $u_v(c)$ as given below.

$$u_v(c) = min\left(\sum_m f_{i,v,scaled} w_{f_i}\right) \tag{6}$$

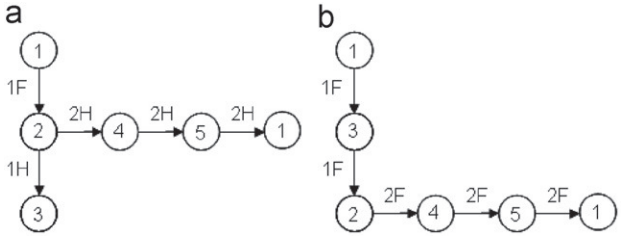

Fig. 10. (a) If the importance weights are assigned only with respect to the preferences of the peers in a chain. (b) If the importance weights are assigned according to the preferences of all peers. The majority of the peers get what they want: maximum video quality.

where m is the number of optimization objectives that are used in the multi-objective solution. The head v would employ c^*, the configuration optimizing the objective function.

5.4. Example scenario: minimize the number of base layer receivers and the maximum delay in a chain

We now illustrate the multi-objective optimization technique with an example scenario. For simplicity and ease of understanding, we assume that all peers have sufficient connection bandwidth only for one full-quality video. Each head v should try *to minimize the number of base layer receivers in its chain* and *minimize the maximum delay experienced in that chain*, at the same time. First, we show that these optimization objectives may be conflicting with each other.

Suppose there is a conference session with 7 participants. Peers 1 and 4 are located in the USA, peers 3 and 5 are located in Turkey, peers 6 and 7 are located in Germany and peer 2 is located in Canada. Typical one-way delay values between the peers are given in Table 5. Let the video request configuration of this conference be the following: Peers 3, 4 and 5 request to view peer 1's video, peer 2 requests peer 3's video and peers 6 and 7 request peer 4's video. So peers 1, 3 and 4 have chains with lengths 3, 1 and 2, respectively.

Suppose that peer 1 needs to configure its chain, so that the number of base layer receivers and the maximum delay experienced by a peer are minimized. The minimum number of base layer receivers is achieved by the chain configuration $\langle 5,3,4 \rangle$ (i.e., peer 4 has the longest chain length, and thus, it should be at the end of the chain, giving a total of 2 base layer receivers). The minimized maximum delay is achieved by the chain configuration $\langle 4,3,5 \rangle$ and yields a maximum delay of 129 ms as calculated from Table 5. These possible configurations can be seen in Fig. 11.

$$g_1(\langle 4,3,5 \rangle) = 5 \quad \text{and} \quad h_1(\langle 4,3,5 \rangle) = \mathbf{129}\,\text{ms}$$

$$g_1(\langle 5,3,4 \rangle) = \mathbf{2} \quad \text{and} \quad h_1(\langle 5,3,4 \rangle) = 200\,\text{ms}$$

Table 5
Sample latency table.

peer id	1	2	3	4	5	6	7
1	0	26	**89**	**24**	**95**	66	70
2	26	0	104	78	108	73	75
3	**89**	104	0	**85**	**20**	55	42
4	**24**	78	**85**	0	**98**	65	71
5	**95**	108	**20**	**98**	0	53	47
6	66	73	55	65	53	0	12
7	70	75	42	71	47	12	0

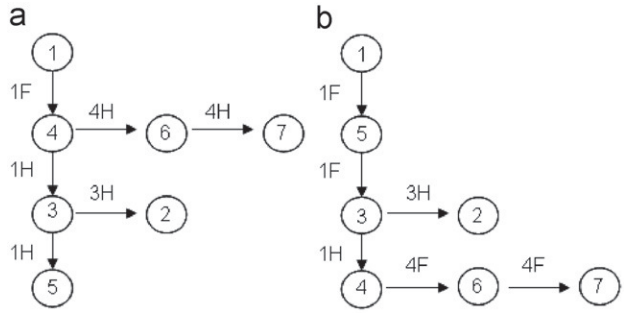

Fig. 11. (a) Chain configuration $\langle 4,3,5 \rangle$ yielding 5 as the number of base layer receivers (Peers 6, 7, 3, 2 and 5) and 129 ms as the chain delay. (b) Chain configuration $\langle 5,3,4 \rangle$ yielding 2 as the number of base layer receivers (Peers 2 and 4) and 200 ms as the chain delay.

Table 6
$g_1(c)$ and $h_1(c)$ values with $w_g = 0.29$ and $w_h = 0.71$.

c	$g_1(c)$	$h_1(c)$	$g_{1,scaled}(c)\ w_g$	$h_{1,scaled}(c)\ w_h$	$u_1(c)$
⟨3,4,5⟩	5	272	0.29	0.68	0.97
⟨3,5,4⟩	3	207	0.10	0.37	0.47
⟨4,3,5⟩	5	129	0.29	0.00	0.29
⟨4,5,3⟩	**4**	**142**	**0.19**	**0.06**	**0.25**
⟨5,3,4⟩	2	200	0.00	0.34	0.34
⟨5,4,3⟩	3	278	0.10	0.71	0.81

If the head (i.e., peer 1) were to minimize only the number of base layer receivers, then the chain configuration it should be employing would be ⟨5,3,4⟩. On the other hand, if it were to minimize only the maximum delay in a chain, then the chain configuration ⟨4,3,5⟩ should be employed. Clearly, these two objectives conflict with each other.

In our scenario, peers 5 and 6 prefer maximum video quality and peers 1, 2, 3, 4 and 7 prefer minimum delay. So, $p_g = 2$ and $p_h = 5$. Then the weights would be $w_g = 0.29$ and $w_h = 0.71$. $g_v(c)$ and $h_v(c)$ values are scaled according to the Eq. (5). The best solution is determined according to the optimization function $u_v(c)$. The entire set of the possible chain configurations and their $g_v(c)$ and $h_v(c)$ values are given in Table 6. According to Table 6, the chain configuration ⟨4,5,3⟩ gives the minimum value of the objective function. This chain is employed by peer 1. Remember that, 4 of 6 peers had told that they would prefer minimum delay over high quality of video. We can see the trade-off between minimizing the delay and minimizing the number of base layer receivers. Peer 1 employs a chain that has the *second* best minimized delay and *third best* minimized number of base layer receivers.

5.5. Simulation results

We performed simulations covering conferencing scenarios with up to 10 participants. Each participant requests the video of another participant randomly. Participants with larger bandwidths make additional video requests. For each conference case with different participant counts, we generated 100,000 cases randomly, in which the number of participants with additional bandwidths is increased from 1 up to the participant count for that case. In the simulations, we assumed that the participants with no additional bandwidth prefer maximum video quality and the rest prefers that additional requests are granted. This may be plausible, because a participant with additional bandwidth would be more likely willing that additional requests are granted. The importance weights for maximizing the video quality and maximizing the number of granted additional requests are calculated accordingly.

The results of the scenarios for each participant count are averaged. Fig. 12 shows the percentages of the rejected requests, granted base layer video receiving requests and granted full quality receiving requests. The percentage of the rejected requests does not exceed 15% and decreases as the number of participants increases. Although the percentage of the base layer video receiving requests increases with the participant count, this increase is asymptotic. Our system was able to grant at least 50% of the requests to receive full quality video. We should also note that these simulations did not cover all possible request configurations as was in Section 4, but a random set of 100,000 cases. Therefore, the results in Figs. 8 and 12 do not contradict.

Instead of rejecting the requests, the system makes use of layered video to grant *every first* request and grants the additional requests according to the best-compromise chain found.

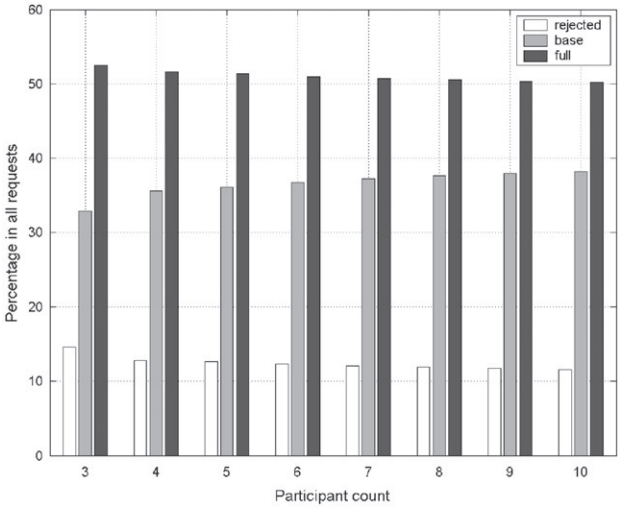

Fig. 12. Percentages of all requests.

Increased participant count increases the probability for a request to receive base quality layer video; however, in cases where base layer video is used, the average percentage of these requests to all requests stays below 40%.

In some cases, there may be rejected requests; however, all of these requests are *additional* requests, so that the requesters already receive a participant's video. Our system tries to maximize the number of granted additional requests, as long as this does not cause other participants' *first* requests to be rejected.

6. Implementation

In this section, the implementation architecture, interfaces between various modules, states of the participants, and messages exchanged between them will be explained. The peer and tracker software is written in Java to ensure OS independence. The video software is a modified version of the Nokia H.264 codec and is in C. We have tested our system in Linux and Windows platforms. We have tested our software in a local network by setting up conference sessions with small group of participants. The request configuration was designed to trigger the layered video transmission and we have verified that the chains were properly configured and the video quality degraded to base quality where necessary.

6.1. Tracker software

The tracker software's responsibility is to keep track of which peers are online. Each peer has an *id*, a *username* and a contact list and piggybacks its contact list at the end of a sign-in message. The tracker updates the status of the peer to "online" and returns the addresses of the available peers in the contact list. The tracker keeps track of each peer by expecting a periodical check-in message in a certain time. If a peer does not check-in or sends a sign-out message, its status is updated to "offline".

6.2. Peer software

The peer software is responsible for conference management, join, leave, invite, text messaging, video request and video release operations. Furthermore, it manages video module processes and their configurations. It provides a graphical user interface (GUI)

which allows the user to interact with the system. After signing in to the tracker, the peer can see the status of other peers in its contact list. A conference is established after a peer accepts another peer's invitation. Peers that join a conference are called *participants*.

Participants can make video requests to any other participant or can release their video sources if they had been already receiving a participant's video signal. There are two modules in the peer software.

Conference Manager Module is the main application part that is responsible for tasks initiated by the user through the GUI, such as sign-in, sign-out, conference invitation and video request events and sending out their corresponding messages. Furthermore, it listens for messages from the tracker (i.e., the contact list status information) and from other peers and handles these. A video request is handled according to the decision algorithm given in Fig. 5 and appropriate messages are sent. The video related processes are updated if any state variable changes.

Video Module consists of four independent processes. The `Capturer` process gets the frames from the capture device and feeds it to the encoder process. The prototype employs an MDC-like approach where even and odd frames are fed to separate ports of the encoder. The `Encoder` process runs three threads: One is responsible for listening to the configuration changes coming from the Conference Manager module. The other two threads independently encode even and odd frames that are supplied from the capturer process. The encoded frames are sent to the first peer in the chain. The information whether to send both layers or only the base layer and the address of the receiving peer is supplied by the Conference Manager. The `Decoder` process is responsible for receiving encoded packets, decoding them and sending them to the displayer process. It also takes care of relaying operations: according to the information supplied by the Conference Manager module, the received packets are either forwarded to the next peer in the chain in full quality, base quality or none at all. The `Displayer` process is responsible for displaying the received frames in a synchronized fashion.

Conference Manager - Video Interface is between the Conference Manager and Video related processes. Besides creating and destroying video related processes, the Conference Manager uses this interface to update the Encoder and the Decoder processes according to the current state of the peer. The `Encoder` process is updated with the heading information, whether the process should start encoding frames received from the `Capturer` process and the video quality. It is also provided with the IP address of the first member in the peer's chain. The `Decoder` process is updated with the relaying information. It includes whether the peer should be relaying, which layers should be forwarded and the IP address of the peer that is next in the chain.

6.3. Peer states

The actions that can be taken by the peers and responses to the actions of other peers depend on the current status of the peer. There are four possible states of a peer as described below.

Participant: A peer in this state is in idle position. It does not watch the video of any participant, nor does it send its own video signal to other participants. The peer may send a video request to any participant to watch its video. When the peer joins a conference, (i.e., when an online peer accepts an invitation from another peer, or when the invitation sent by this peer is accepted by the invited peer) the peer makes transition into this state.

Member: When a peer is watching another peer's video, it is in `Member` state. A peer in this state cannot make a video request since it already made one. It may release the video source.

Chainhead: The peer in this state has a *chain*. It is sending its own video signal to the next peer in its chain. A peer in this state does not watch another peer's video signal, thus it can make a video request to any participant.

Chainhead_Member: When a peer is watching another peer's video and sending its own video signal at the same time, it is in `Chainhead_Member` state. It is sending its own video signal to the next peer in its chain. A peer in this state cannot make a video request since it already made one. It may release the video source.

6.4. Messages

There are two types of messages used in our implementation, namely messages exchanged between a peer and the tracker, and messages exchanged between peers. These are described next.

Messages exchanged between a peer and the tracker: These messages are depicted in Fig. 13. A `sign-in message` is sent from a peer to the tracker to sign-in. The tracker updates the status of the sender from "offline" to "online". A `sign-in acknowledgement message` is sent from the tracker to the peer as a response to the sign-in message. `Check-in message`s are periodically sent from each peer to the tracker, as long as the peer is online. The id's of the peers in the contact list are piggybacked at the end of the message. If the tracker does not receive a check-in message in a certain amount of time, it concludes that the peer experienced a problem and updates the status of this peer from "online" to "offline". After the receipt of check-in messages, the tracker responds with `check-in reply message`s. This message contains information about the id's of the peers in the corresponding check-in message, their availability status and their addresses. If at any time a peer wants to sign out of the system, it sends a `sign-out message` to the tracker. The tracker updates the status of the sender from "online" to "offline". `Sign-out acknowledgement message` is sent from the tracker to the peer as a response to the sign-out message.

Messages exchanged between peers: These messages are sent from an online peer to one or more online peers. An `invitation message` is used to start a video conference or to invite a peer to an already existing one. The invited peer responds with an `invitation reply message`. If the peer accepts the invitation, the peer is joined to the conference by the inviting peer via an `invitation acknowledgement message`. This message also includes the addresses of the peers already in the conference. Besides this message, the inviting peer also sends an `invitation update message` to the peers already in the conference to inform them about the newly joining peer (i.e., its id, username and address). These messages and their sequences are depicted in Fig. 14. If at any

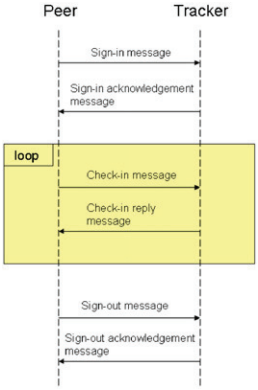

Fig. 13. System Sequence Diagram of signing in, checking in and signing out events of a peer.

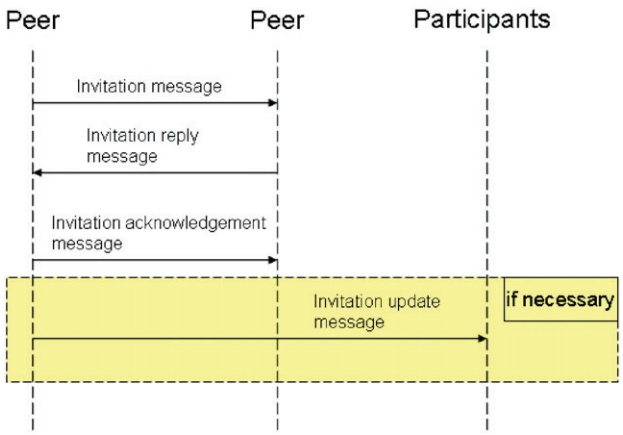

Fig. 14. System Sequence Diagram of inviting an online peer to a conference.

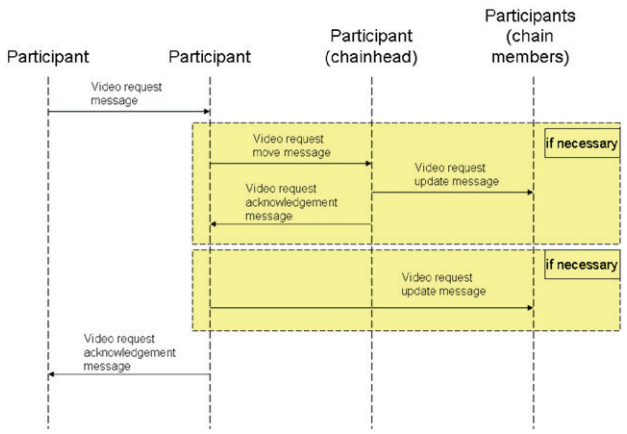

Fig. 15. System Sequence Diagram of sending a video request, video request move, video request acknowledgement and video request update messages. Conference peers that are chain members are the members of the respective chains of the senders.

time, a peer wants to leave the conference, it sends a `leave message` to each peer in the conference. If the peer is a head of a chain, its chain members update their status. If the peer is a member in a chain, it sends a `video release message` (explained below) to the head first, so that it can update its chain accordingly.

After a conference is established, participants can send a `video request message` to any participant to watch their video. The requested participant uses the algorithm given in Fig. 5 to decide where to put the new peer into its chain. If the requested participant is a member in a chain, it may send a `video request move` message to its head, so that the number of base layer receivers is minimized as explained in Section 4. If the requested participant already has a chain, it may also send `video request update` messages to peers that will be *before* and *after* the new member if necessary. The video request update message contains information about the *new previous member*, the *new next member* and the video quality. Lastly, the requested participant responds with a `video request acknowledgement message` to the requesting participant and informs it with its position in the chain (i.e., the previous member and next member in the chain) and the video quality it should be forwarding the video signal (i.e., base or full quality). After the receipt of the `video request acknowledgement message`, the peer starts sending periodical `video keep-alive message`s to its head. This helps the head of the chain to detect problems and rearrange the chain accordingly in a quick fashion. This message also includes information about the current status of the sending peer, such as the length of its chain if any and its state. If at any time a member wishes to stop receiving the video, switch to another peer's video signal or to leave the conference, it sends a `video release message` to its head. The head of the chain rearranges its chain accordingly. The sequences of these messages are illustrated in Figs. 15 and 16, respectively. Besides these messages, the peers can send `chat message`s to all other participants in a conference or another online peer.

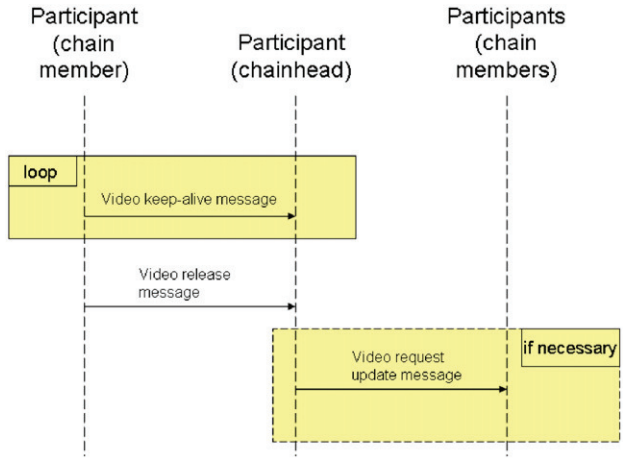

Fig. 16. System Sequence Diagram of sending video keep-alive messages to the head and releasing a video source of a peer.

7. Formal verification of the system

The nature of the P2P architecture requires a distributed algorithm to handle video requests of the peers, without the need of a central node. This necessitates communicating messages between peers. The protocol allows that each peer can make a video request to watch another peer's video at any time, which causes interleaving of these messages. A verification is needed to ensure that the system will function properly with previously described interleavings, so that deadlocks (i.e., peers waiting in a circular fashion for the responses to their requests) or livelocks (i.e., peers not making requests to be able to handle an incoming video request) do not occur.

We have modeled our system and its properties using TLA+ formal specification language (Lamport, 1994). The specifications written in TLA+ are then verified using TLC model checker (TLC model checker, online). During the verification we checked the following properties and ensured that they hold during each conference configuration:

1. A peer making a request will always get an acknowledgement and start watching video.
2. A peer cannot send its own video and relay video both in full quality (i.e., if a peer is the head of a chain and a member forwarding the video it is receiving at the same time, it must be sending its own video and forwarding the video it is receiving in base quality).
3. For a configuration to be considered verified, each peer must have made a request and received an acknowledgement.

Item 1 makes sure that each participant's request can be granted at any time under any configuration. Second item ensures that the upstream restriction is not violated. Item 3 shows that each peer eventually makes a request, thus not waiting forever

without watching the video of a participant (i.e., verifying that there are no livelocks).

All properties have been successfully verified, along with the correctness of peer states and state variables. TLC verifies that the type invariant is not violated during the "next state" action of the system. It is shown that the system works correctly without any deadlocks and livelocks (Akkus, 2007).

8. Conclusion and future directions

We proposed a P2P architecture for MP video conferencing using layered video. This approach makes it possible to grant any participant's video request at any time and under any configuration. Simulations show that with the increasing number of participants, the use of layers and thus, base quality video receivers is inevitable; however, the ratio of the base quality receivers to the total number of participants remains under 50%. We implemented the proposed architecture and developed a fully distributed protocol. Scalable Coding and Multiple Description Coding techniques that can be employed within our architecture are presented.

We developed a multi-objective optimization approach for the P2P video conferencing system. The system makes compromises between the quality of the video peers receive, delays experienced by peers and additional requests of peers with sufficient bandwidth. Our multi-objective formulation successfully finds the best-compromise solution, considering the given objective functions. Since the intended group size is relatively small, generation of possible configurations does not bring much computational burden. This makes the system easy to implement. Furthermore, a formal specification of the system is made using TLA+ and verified with TLC model checker. No errors, deadlocks or livelocks have been found in the protocol.

It is shown that only two layers of video are required to grant each request in a conference where participants have one input, one output and request to watch one participant at a time. We call this type of a conference a basic conference. One can extend this scheme for conferences where participants have n inputs and n outputs and request to watch n other participants. Each request set of participants can be considered a basic conference and each request can be granted by using two layers. This brings an upper bound of $2n$ to the number of required layers of video that can be used in this type of conference to grant each request. Future work may try to determine the lower bound.

Acknowledgements

This work is supported in part by TUBITAK (The Scientific and Technical Research Council of Turkey) under CAREER Award Grant 104E064. A preliminary version of this work appeared in Akkus et al. (2006) and Akkus et al. (2007). We thank the anonymous reviewers for their valuable suggestions and comments.

References

Akkus IE, Civanlar MR, Ozkasap O. Peer-to-peer multipoint video conferencing using layered video. In: Proceedings of IEEE ICIP, Atlanta, 2006.

Akkus IE. Peer-to-peer multipoint video conferencing using layered video. MSc thesis, Koc University; 2007.

Akkus IE, Ozkasap O, Civanlar MR. Multi-objective optimization for peer-to-peer multipoint video conferencing using layered video. In: Proceedings of 16th international packet video workshop, Lausanne, 2007.

Apple iChat. Online: 〈http://support.apple.com/kb/HT2020?viewlocale=en_US〉.

Banerjee S, Bhattacharjee B, Kommareddy C. Scalable application layer multicast. In: Proceedings of ACM SIGCOMM, Pittsburgh, Pennsylvania, 2002.

Banerjee S, Lee S, Bhattacharjee B, Srinivasan A. Resilient multicast using overlays. In: Proceedings of ACM SIGMETRICS, San Diego, CA, 2003.

Bawa M, Deshpande H, Garcia-Molina H. Transience of peers & streaming media. ACM SIGCOMM Computer Communication Review 2003;33(1):107–12.

BNI solutions' IPContact. Online: 〈http://www.bnisolutions.com/products/powerplay/decentralized_mp.html〉.

Chu Y, Rao SG, Zhang H. A case for end system multicast. In: Proceedings of ACM Sigmetrics, 2000.

Chu Y, Rao SG, Seshan S, Zhang H. Enabling conferencing applications on the internet using an overlay multicast architecture. In: Proceedings of ACM SIGCOMM, San Diego, CA, 2001.

Chen M, Ponec M, Sengupta S, Li J, Chou PA. Utility maximization in peer-to-peer systems. In: Proceedings of ACM SIGMETRICS, Annapolis, Maryland, June 2–6, 2008.

Civanlar MR, Gaglianello RD, Cash GL. Efficient multi-resolution, multi-stream video systems with standard codecs. The Journal of VLSI Signal Processing 1997;17(2):269–79.

Civanlar MR, Ozkasap O, Celebi T. Peer-to-peer multipoint video conferencing on the internet. Signal Processing: Image Communication 2005;20:743–54.

Cui Y, Nahrstedt K. Layered peer-to-peer streaming. In: Proceedings of NOSSDAV, Monterey, CA, 2003.

Draft ITU-T recommendation and final draft international standard of joint video specification (ITU-T Rec. H.264 |ISO/IEC 14496-10 AVC); 2003.

Google gmail chat. Online: 〈http://www.google.com/chat/video〉.

Google gmail chat. Online: 〈http://www.google.com/support/chat/bin/answer.py?answer=159499〉.

Goyal VK. Multiple description coding: compression meets the network. IEEE Signal Processing Magazine 2001;18(5):74–93.

Hosseini M, Georganas ND. Design of a multi-sender 3D video conferencing application over an end system multicast protocol. Berkeley, CA: ACM, Multimedia; 2003.

ITU-T Study Group XV - Recommendation H.231: Multipoint control units for audiovisual systems using digital channels up to 1920 kbits/s; 1993.

JSVM Software. Online: CVS Repository of JSVM pserver:jvtuser@garcon.ient.rwth-aachen.de:/cvs/jvt.

Lamport L. The temporal logic of actions. ACM Toplas 1994;16(3):872–923.

Lim BP, Ettikan KK, Lin ES, Phan TK, Thoai N, Muramoto E, et al. Bandwidth fair application layer multicast for multi-party video conference application. In: Proceedings of the sixth IEEE conference on consumer communications and networking conference, Las Vegas, January 11–13, 2009.

Liu Y, Guo Y, Liang C. A survey on peer-to-peer video streaming systems. Peer-to-Peer Networking and Applications 2008;1(1):18–28.

Luo C, Wang W, Tang J, Sun J, Li J. A multiparty video conferencing system over an application-level multicast protocol. IEEE Transactions on Multimedia 2007;9(8).

Lu Y, Zhao Y, Kuipers FA, Van Mieghem P. Measurement study of multi-party video conferencing. In: Proceedings of IFIP networking, Chennai, India, May 10–14, 2010.

Nefsis. Online: 〈http://www.nefsis.com/How-Multipoint-Conferencing-Works/index-multipoint-video-conferencing.html〉.

Nokia H.264 codec. Online: 〈ftp://standards.polycom.com/IMTC_Media_Coding_AG/〉.

Ponec M, Sengupta S, Chen M, Li J, Chou PA. Multi-rate peer-to-peer video conferencing: a distributed approach using scalable coding. In: Proceedings of IEEE international conference on multimedia & expo (ICME), New York, June 30–July 2, 2009.

Silverston T, Fourmaux O. Source vs. data-driven approach for live P2P streaming. Washington DC: ICNICONSMCL'06; 2006.

Skype. Online: 〈http://blogs.skype.com/garage/2010/05/skype_50_beta_1_for_windows.html〉.

Tran DA, Hua KA, T. Do: ZIGZAG: An efficient peer-to-peer scheme for media streaming. In: Proceedings of IEEE Infocom, 2003.

TLC model checker. Online: 〈http://research.microsoft.com/users/lamport/tla/tlc.html〉.

Vidyo. Online: 〈http://newteevee.com/2009/10/29/vidyo-powers-google-video-chat-gets-patent/〉.

Zadeh L. Optimality and non-scalar-valued performance criteria. IEEE Transactions Automatica Control 1963;8:59–60.