

SmartPeerCast: a Smart QoS driven P2P live streaming framework

Wenyi Wang · Yaowu Chen

Published online: 5 June 2010

© Springer Science+Business Media, LLC 2010

Abstract The P2P swarm technologies have been shown to be very efficient for medium scale content distribution systems in the last few years, such as the file sharing and video-on-demand (VOD) applications. However it is still an open topic about how to deploy the P2P paradigm for the real time video broadcasting (RTVB) applications. The P2P RTVB application is different from the cache based P2P system because it has more stringent restrictions for startup time and packet loss ratio. In this paper, an adaptive media broadcasting P2P framework named SmartPeerCast which employs the media transrating service to control the quality of service (QoS), is proposed. SmartPeerCast achieves a network awareness, codec awareness, and high performance RTVB service with four key designs: (1) It groups the newly joined peers into different quality clusters by their uploading capability. This clustering mechanism avoids the bandwidth bottleneck between the heterogeneous peers of the overall P2P overlay by only forwarding the same quality stream over the peers in the same cluster. (2) The streaming quality is adjusted adaptively between the sending and the receiving peers by a Smart QoS algorithm to compensate for the network jitters to reduce the receiving peer's playback jitter. (3) The receiving peer monitors the data forwarding QoS of the sending peer to select the best suitable parent node dynamically. The SmartPeerCast uses this Smart QoS framework to implement an incentive mechanism to award the peers with high uploading contributions by migrating them to a higher quality cluster. (4) A transrating engine is used at the leaf nodes of the high quality cluster to forward the stream with suitable bits rate to the nodes of the low quality cluster; this transrating service not only can fully utilize the uploading bandwidth of the peers in the higher quality cluster but also avoids the bandwidth bottleneck of stream forwarding between the heterogeneous peers. Our experiment results and the real deployment show that SmartPeerCast can eliminate the bandwidth bottleneck and content bottleneck between the heterogeneous peers with a smaller startup time and packet loss and it is a high performance and medium scale P2P RTVB framework.

Keywords P2P · Broadcasting · Live streaming · QoS · Transrating

W. Wang (✉) · Y. Chen

Advanced Digital Technology and Instruments Institute, Zhejiang University, Hangzhou 310027, China
e-mail: walker_wwy@hotmail.com

1 Introduction

Over the last decade, the Internet protocol (IP) based multimedia applications are growing tremendously and becoming the killer applications. Most of them are implemented as the client/server architecture with proprietary protocols; such as the VOD applications, User-Generated-Content (UGC) web sites (ex, the YouTube [21]), Push VOD applications (ex, the Netflix [18]), and the real time video broadcasting (RTVB) applications (ex, the IP video surveillance system). These centralized systems always have the scale issue as the user base continues to grow due to the bandwidth and storage space bottleneck of the servers. Taking the RTVB applications as an example, the performance data of these systems; such as the startup time, packet delay, and playback jitters, heavily depend on the transferring network bandwidth, so the scale of RTVB application is limited by current Internet framework and its transfer mode. For example, if a broadcasting server wants to support a flash crowd accessing of 10,000 users for a video stream encoded with 512 Kbps, A 5 Gbps (512 Kbps * 1,000) uploading bandwidth must be provided at least for it. In the current Internet framework, a huge investment for the bandwidth needs to be in place to have this service. Multicast can resolve the scale issue by only replicating one copy of the packet over the same physical link, but it has not being deployed successfully over the Internet [34] and works only in the Intranet scope. Researchers thus have resorted to application-level solutions using P2P systems, which build an overlay network out of unicast tunnels across cooperative participating users called overlay nodes, and multicast is then achieved through data relaying among these nodes[45].

In P2P streaming systems, there are two main types of overlay architectures to organize the peers for data forwarding [31, 33, 34, 36]: a) tree based and b) mesh based. The tree based systems use push method to disseminate the data over the application level multicast (ALM) tree, as the researches [5, 9, 11–13, 24, 41] have proposed. The ALM tree structure uses the same idea as the multicast technology, so it is easy to implement and maintain. There are two kinds of ALM tree overlay structure; one is single ALM tree, such as DirectStream [13], OverCast [24], ESM [9] and ZIGZIG [41], and the other is the multiple ALM trees as proposed by SplitStream [5], P2VOD [11], and P2CAST [12]. At the ALM overlay network, every peer or node uploads the data to others while receiving the data to reduce the server's loading. It improves the scale of the media broadcasting system by utilizing every peer's uploading bandwidth. But the ALM overlay network is an application layer protocol, so it has the performance penalty comparing with the multicast and it has more transfer delay of the packet. The depth of an ALM tree node is deeper and the transfer delay of the packets between the source and this node is longer. Unlike the multicast topology, the nodes of the ALM trees are deployed over the Internet widely and heterogeneously, so the extra transfer delay of the packets will be added due to the bandwidth bottleneck between the heterogeneous nodes. For example, if a peer in the data forwarding path has a smaller uploading bandwidth than the media stream's bits rate, all the descendent nodes of this peer will meet the bandwidth bottleneck when receiving this stream and have very heavy playback jitters in their player. System's maintaining effort is another impact to the ALM tree's performance. When a peer joins and leaves the ALM tree, the system needs take some time to fix the data forwarding path and this process will interrupt the data forwarding of the media stream. This interruption often will introduce the playback jitters to the peers. As the SplitStream [5] presents, the multiple ALM tree overlay combining the Multi-Description Coding (MDC) layer encoding technology can address some key performance parameters of the P2P RTVB application, such as the startup time delay and the packet loss. Although the MDC concept is a good way to improve the playback performance between the heterogeneous peers, none of the available decoders

nowadays has supported the MDC technology. Neither the single tree nor the multiple trees of the ALM utilize the leaf node's uploading bandwidth. Thus, the ALM tree framework does not have the overall best performance as the mesh framework [34].

As the researches [1, 2, 6–8, 10, 22, 25, 26, 28, 29, 32, 35, 37, 38, 40, 42, 45] have discussed, the peers of a mesh overlay network can get the data from multiple source peers and forward the data to multiple children peers comparing with the tree based structure, so the mesh structure can support larger scale systems and are more tolerant to peers churn. The mesh framework uses the receiver driven framework to pull data from the parent nodes and can utilize every peer's uploading bandwidth. Unlike the receiving node in the ALM tree framework which can only get the data from its parent nodes, the receiving node of the mesh P2P network uses a scheduler to monitor and select the suitable parent node periodically and globally. This global source node selection mechanism can utilize all peers' upload bandwidth and increase the topology resilience to the node failure or departure due to the increased probability of available distinct network paths. However, streaming applications over the mesh overlay face two important challenges [27]. First, due to the inherent sequential media encoding and play-out, packet dissemination and data requests must follow the temporal ordering of the content closely at the source. This constraint may be slightly reduced by the implementation of play-back buffers when delays permit it. Second, the limited look-ahead content availability; especially in the case of live streaming scenarios, limits greatly the flexibility in terms of content download/upload through such architecture.

From the application point of view, the P2P streaming applications can be divided into two categories: (1) P2P VOD application. It is similar to the P2P file sharing application of BitTorrent [16] and eMule [17] by using the peer's cache to reduce the server's loading. But it is different from the P2P file sharing application. The peers in the P2P VOD system often only cache the movie segments which they have played. If the movie data hasn't been played or cached yet, the peer will request it from the central server. Unlike the P2P file sharing application, the chunks must arrive at the peers of the P2P VOD system in the playback order, so the chunk replication algorithm of the P2P VOD application is more complex and challenge. The P2P VOD system can use the large cache of the peer to improve the VOD playback performance as presented by the PPLive [22] and the PPStream [20]; they cache one or multiple files at the peer's storage to improve the performance. (2) P2P RTVB application. It is totally different from the P2P VOD application because it is more stringent for startup time and packet delay; the cache algorithms of the P2P file sharing and VOD system cannot be applied to the RTVB system. For the P2P VOD system, it is still an efficient way to improve the overall performance of playback jitters and streaming quality by increasing the peer's cache buffer size. But the bigger cache buffer also will increase the initial startup time delay [32]; it is not acceptable for the RTVB application which needs as small an initial startup time as possible. Thus, the cache mechanism used at the P2P VOD system cannot benefit the P2P RTVB system and needs some improvement. At the same time, the P2P VOD system also uses the cache algorithm to reduce the playback jitters caused by the network congestion between the heterogeneous peers. The P2P VOD clients compensate the packet delay by increasing the peer's cache and its player buffer size [6]. This cache mechanism fails to work for the P2P RTVB system either because the P2P RTVB application has little tolerance for the packet delay. The P2P RTVB clients must drop any delayed packets at its player to have a continuous real time stream playing and, the delayed packets would be taken as being lost. Consequently, it generates the playback jitters at the player. From the above discussion, the traditional framework and algorithms of the P2P content distribution systems cannot meet the

requirements of the P2P RTVB application because it has more stringent requirements for the startup time and the packet delay. A new P2P framework and algorithm need to be created and designed for the RTVB application.

In this paper, a framework by combining the ALM tree and mesh framework of the P2P overlay together is proposed. Our proposal not only keeps the simplicity and easy maintaining features of the ALM tree framework, but also utilizes all peers' uploading bandwidth like the mesh framework to improve the startup time delay and playback jitters performance of the RTVB application. As Fig. 1 shows, the framework we propose in this paper is named SmartPeerCast. It is a revised multiple ALM tree architecture; the nodes at the different ALM tree also can serve each other. The SmartPeerCast is composed of three type of nodes; source, tracker, and peer. The original input video stream is replicated by the transrating engine [3, 30] at the source node to three different quality output channels of high, medium and low bits rate stream, which have the same resolution and scenes. The peers are grouped into three ALM trees of high, medium and low quality stream by the tracker node. The tracker assigns a newly joined peer to one of the above three ALM trees by its uploading capability; the peer with higher uploading bandwidth joins the ALM tree of the higher quality stream. Besides the tracker node, the peer node may also uses the transrating engine to replicate the stream by reducing the quality, especially the leaf nodes of the higher quality stream ALM tree. SmartPeerCast uses the transrating engine at these leaf nodes to forward the lower quality streaming to the nodes of the lower quality stream ALM trees. A Smart QoS framework is designed between the receiving and the sending peers to monitor the forwarding service quality of the sending peer. The receiving peer will trigger the sending peer to adjust the streaming quality dynamically by the transrating engine to avoid the playback jitters when the Smart QoS data show that there is some network link congestion. When the sending peer's QoS cannot meet the minimum playback

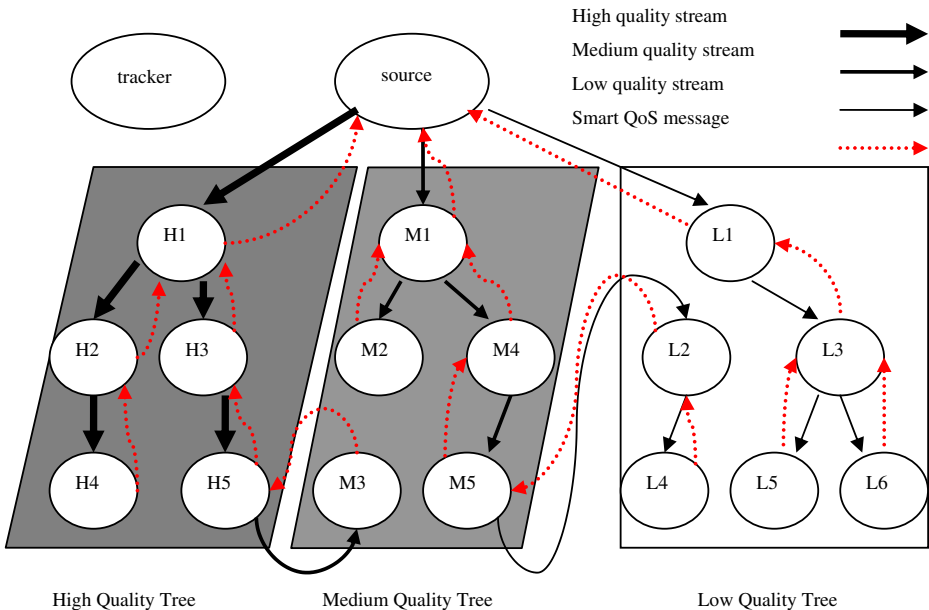


Fig. 1 SmartPeerCast overlay overview

jitters requirement, the receiving peer will drop it and select a better one. The receiving peer also reports the sending peer's QoS to the tracker. The tracker implement an incentive mechanism based on the QoS data to award or punish the sending peer; the peer with less contributing uploading bandwidth will be punished by allowing only joining the low quality stream ALM tree. SmartPeerCast contributes four improvements for the P2P RTVB systems: 1) Eliminating the bandwidth bottleneck between the heterogeneous peers by clustering them according to their uploading bandwidth contribution and only forwarding the same quality stream at the given quality cluster. 2) Reducing the playback jitters of the receiving peers by adjusting the streaming quality dynamically between the sending and the receiving peers based on the transrating engine. 3) Guaranteeing the fairness of the peer's contribution and receiving by Smart QoS. 4) Utilizing the leaf node's uploading bandwidth of the higher quality stream ALM tree by reducing the video stream quality and forwarding it to the nodes of the lower quality stream ALM tree.

The remaining of the paper is organized as follows: In Section 2, the related research works on the P2P RTVB framework are reviewed, and then the proposed algorithm's advantages and disadvantages are analyzed. In the third section, the SmartPeerCast framework, its major components, and the algorithms are presented; the Smart QoS algorithm is also covered in this section. The performance evaluation and experiments' result are discussed in the fourth section. Finally, the paper is concluded in Section 5 along with the proposed future works.

2 Related works

Although lots of researches have been done for the P2P multimedia applications in the past few years, most of them are focusing on the VOD applications as we have mentioned in the first section, but it is still unclear about how to use the P2P paradigm to improve the scale and the performance in many areas of the RTVB system. The P2P RTVB applications normally face three challenges: (1) Sustained bandwidth from sending peers; the challenge is that all sending peers must have the enough uploading bandwidth to meet the smooth playback requirement. (2) Startup delay; the RTVB system has a more stringent startup time than the VOD application, and most of RTVB applications can't tolerate the startup time by more than 2 s. (3) Playback media quality which depends on the completeness of the data received before its playback deadline.

As the paper [44] has surveyed, a few live streaming application protocols have been designed during the last few years to meet the challenges of both media quality and timeliness of stream reception; different approaches have been proposed to strike a trade-off between these two strongly-correlated factors in all cases. Table 1 presents a global overview of our surveyed P2P RTVB applications by grouping them with the overlay topology, data chunk exchange design, data encoding schema, heterogeneity, locality, incentives mechanism, and performance measurement. The design comparison at the peer choice, peer and content meta-data repository, and uploading source schemas [44] have been omitted, because there are little design trade-offs for these metrics at most of the P2P live streaming applications. In the next paragraph, the proposals listed in Table 1 will be described, and the differentiation between them and the SmartPeerCast system will be highlighted.

PRIME is a P2P RTVB system proposed in [35]. The peers are organized into both the trees and a random mesh overlay. It uses two phases action of diffusion and swarm to replicate the data. Peers receive media content from multiple sources by using the MDC data encoding schema. Likes our proposal, PRIME can fully utilize the resource of the leaf

Table 1 Summary of the surveyed P2P RTVB systems

	Overlay topology	Data chunk exchange design	Data encoding schema	heterogeneity	locality	Incentives mechanism	Performance measurement
PRIME	Tree+Mesh	Pull	MDC	bandwidth	Not Used	Not Used	Simulation
SIM	Tree	Pull	MDC	Bandwidth + latency	Proximity information + Multicast Island	Not Used	Simulation
R ²	Mesh	Push	Random network coding	Not Used	Not Used	Not Used	Prototype implementation
PULSE	Mesh	Pull	Multiple sources	Bandwidth	take into account	Tit-for-Tat + additional excess	Simulation + Prototype
POEMS	Tree	Pull	MPEG4 layer coding	Latency + AVO	Proximity information	Not Used	Simulation
OSN	Tree	N/A	Single stream	Homogeneous peers	Geographical distance	Employing the idle node	Simulation
Dag-stream	Tree	N/A	MPEG-21 gBSD coding	Adaptation capability + bandwidth + CPU	Proximity information	Not Used	Simulation
SmartPeerCast	Tree+Mesh	Push	Adaptive single stream, but codec awareness	Bandwidth + CPU power	Proximity information	Tit-for-Tat	Simulation + Prototype

nodes at the trees. However, PRIME system doesn't use any incentive mechanism and quality control to adapt the network transfer jitters such as employed in our system.

SIM [26] is a hybrid overlay topology of P2P and IP Multicast. Peers in SIM first form an overlay tree using a scalable protocol. They then detect IP multicast islands and employ IP multicast whenever possible. At SIM proposal, the receiving peer's media quality is limited by the minimum upload bandwidth of the intermediate peers in the branch, since each client is connected to the source through a single tree branch. A peer clustering mechanism is deployed at our proposal to address above issue of SIM.

R² is a P2P RTVB system proposed in [43]. Peers are organized in a mesh based overlay. The system is based on Random Push with Random Network coding. Unlike the other live streaming systems, the playback at R² is synchronized for all peers. However, the R² system doesn't consider the peer's heterogeneity and take advantage of the incentive mechanism.

PULSE is a P2P live streaming system proposed in [39]. Peers are organized in a mesh-based overlay and data distribution follows a pull-based scheme. The system uses a primary optimistic tit-for-tat peer selection policy, and an additional excess-based altruistic incentive. Its major contribution is to design and use an incentive mechanism for the sending peer's selection.

POEMS [1] group the peers into different ALM trees based on the MPEG-4 multiple layers' encoding feature. Peers with high uploading bandwidth are selected to receive and forward the base layer of the MPEG-4 stream and peers with low uploading bandwidth can only receive and forward the enhancement layers of the MPEG-4 stream. In POEMS, peers containing audio video objects (AVO) are assigned higher priority than peers offering less important AVOs. Once a peer is selected for a high quality object, then it is marked as unavailable in all other overlay networks to avoid its being selected for low quality object. It is helpful to avoid congestion over certain network links, but it can't fully utilize the resources of the peers with high uploading bandwidth even though when they still have the capability to serve more sub-streams.

OSN [4] is a framework designed for P2P RTVB applications. The appropriate idling peers in the ALM tree overlay can be found and incorporated into the multicasting tree to reduce the network traffic and improve the overall system performance. Our proposal uses the similar incentive concept as OSN where the server will record and store every peer's contribution in bandwidth and uploading time. However, OSN didn't explain how the incentive mechanism is in place to motivate the peers to contribute their resources for video forwarding. It remains to be a trusted overlay network in the sense that all peers and the source server are stable. It also assumes that all peers are homogeneous.

Like the POEMS system, Dag-stream [23] is another P2P RTVB application using content adaption technology. Dag-stream proposes a solution to distributed adaptation and streaming using MPEG-21 generic Bitstream Syntax Description (gBSD). The peers are organized by tree based overlay and served based on a hierarchy of adaptation and streaming requirements. The peer's heterogeneity is handled in terms of its processing power and bandwidth. The tree overlay is constructing based on the video adaption requirements of the clients. Comparing with our proposal, Dag-stream proposal didn't discuss the incentive mechanism about how to encourage the peers to contribute their CPU cycles and uploading bandwidth. Different from the Smart QoS algorithm of our SmartPeerCast proposal, Dag-stream system only considers the video adaption among heterogenous peers when incorporating them into the overlay tree, but can't adjust the video adaption dynamically during the streaming phase according to the variations of its parent CPU loading and the network jitters.

From all the above researches, POEMS and Dag-stream are the most similar to ours because they both use the video adaption technology. But differences are: (1) The SmartPeerCast is a codec awareness video adaption framework by using the transcode and transrating engine of ffmpeg [14], while the POEMS uses MPEG-4 layer encoding and the Dag-stream uses the MPEG-21 standard. (2) In our proposal, the video stream quality is adapted dynamically along with the transferring link condition and the peer's CPU loading. The video adaption is not only used to eliminate the heterogeneity between the peers, but also used to smooth the streaming playback caused by the network jitters. (3) Instead of using the trees overlay, the tree and mesh overlay together are combined to organize the peers in our proposal. This tree and mesh hybrid overlay improves the overall system stability and resource utilization. (4) None of the POEMS and Dag-stream proposal implements the incentive mechanism, but a tit-for-tat like incentive mechanism to award and punish the peer's contribution based on the Smart QoS framework is designed into our proposal. By eliminating the free riders with this incentive framework, the overall system performance is enhanced also.

3 The SmartPeerCast framework and algorithms

Figure 2 shows the SmartPeerCast implementation overview. The SmartPeerCast overlay network is composed by three type nodes; tracker, source, and peers. SmartPeerCast defines the three nodes' function and role as the description below.

The tracker node; it is demonstrated in the Fig. 2. SmartPeerCast uses the tracker node to manage the registration of the peers and the source node; such as the leaving and the joining events. The tracker node is the super node of the SmartPeerCast and its main function are defined and implemented as below: a) To manage and assign the newly joined peer to different quality ALM tree. Every ALM tree takes one of the tree output channels in

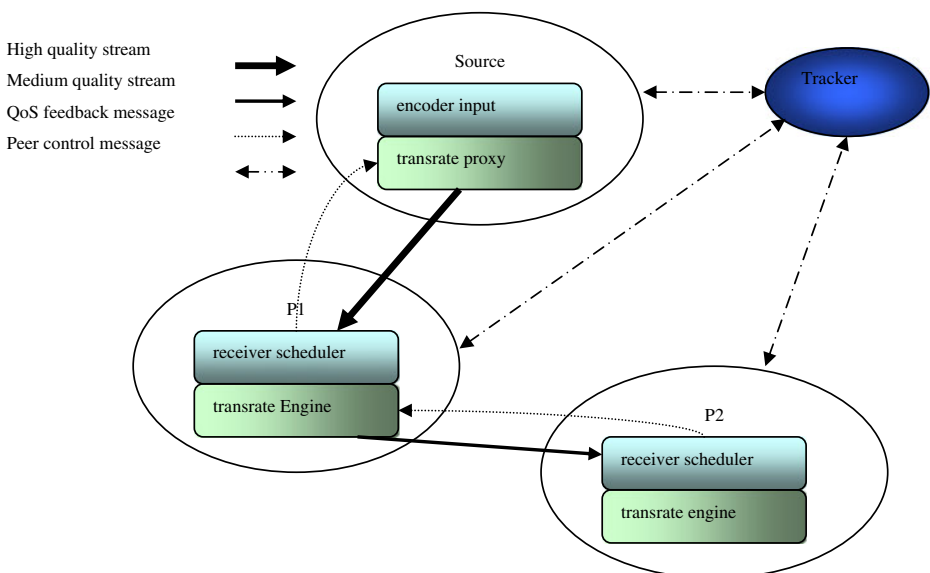


Fig. 2 SmartPeerCast implementation overview

the source node as the root node. The peer with the higher uploading bandwidth is assigned to the ALM tree rooted at the higher quality output channel. The peer registers to the tracker with the following node information; $\langle \text{peer id, upload bandwidth, the maximum acceptable input connections, stream id} \rangle$. Please refer to the Table 2 for the definition of the notation used here. b) To handle the peer's leaving event and the peers churn of the ALM trees. At the ALM tree overlay network, the peers churn must be fixed as soon as possible so that the corrupted streaming forwarding path can be recovered quickly to avoid the playback jitters. There are two possible ways for a peer to quit an ALM tree in the SmartPeerCast; one is the graceful way and the other is the forced way. If the peer quits the ALM tree gracefully, it is unregistered by the tracker with the following message; $\langle \text{peer id, stream id} \rangle$. After the tracker receives this message, it removes the peer from the stream ALM tree indicated by the *stream id*. For the other cases that the peer quits the ALM tree in a forced way, such as an accident of the system crashing, the peer's children will detect this event by the "keep-alive" message between them and reports the leaving event to the tracker node. Once a peer leaves the SmartPeerCast ALM trees either in the graceful or the forced way, all the descendant nodes of the peer must run the Algorithms 1 to find a new parent to fix the peer churn in the ALM tree. c) To award and punish the peers based on its uploading bandwidth contribution. The Algorithm 2 presents the details. The tracker node collects the sending peer's QoS data during the stream forwarding and acts as the central server of the incentive mechanism. If the sending peer's QoS could not meet the stream quality of the ALM tree, it would be migrated to another ALM tree which conveys a different quality stream.

The source node; its overview is shown in the Fig. 3. The source node's function is to provide the real time video encoding data to the whole broadcasting service and acts as the root node of the ALM trees. In current applications and researches, the developers and the researchers often use a TV capture card or an IP camera at the source node to provide one MPEG-4 or H.264 encoding stream as the output channel. But SmartPeerCast uses a different source node implementation. The source node of SmartPeerCast is designed as a logic unit by combining one channel MPEG-4 or H.264 encoder and the transrating proxy

Table 2 Notations

Term	Definition
T	The tracker
S	The streaming source server
P_i	Peer i
sid	The real time live streaming id. It indicates the encoder's output live streaming. All the three output streaming channels in the source node have the same <i>sid</i> .
Q_i	The quality level of P_i . SmartPeerCast defines three quality levels as HIGH_QUALITY, MEDIUM_QUALITY and LOW_QUALITY.
$peers_clusters[i]$	It is the array of the peer clusters. SmartPeerCast groups all the peers into three clusters with the different quality level of HIGH_QUALITY, MEDIUM_QUALITY and LOW_QUALITY. All peers at the same cluster receive the same quality streaming data indicated by the cluster's quality level.
ub_i	The total uploading bandwidth of P_i .
fb_i	The total consumed uploading bandwidth of P_i when forwarding the streams to its children.
sb_{sid}	The bandwidth requirement of the HIGH_QUALITY output streaming channel in the source node for the stream <i>sid</i> .

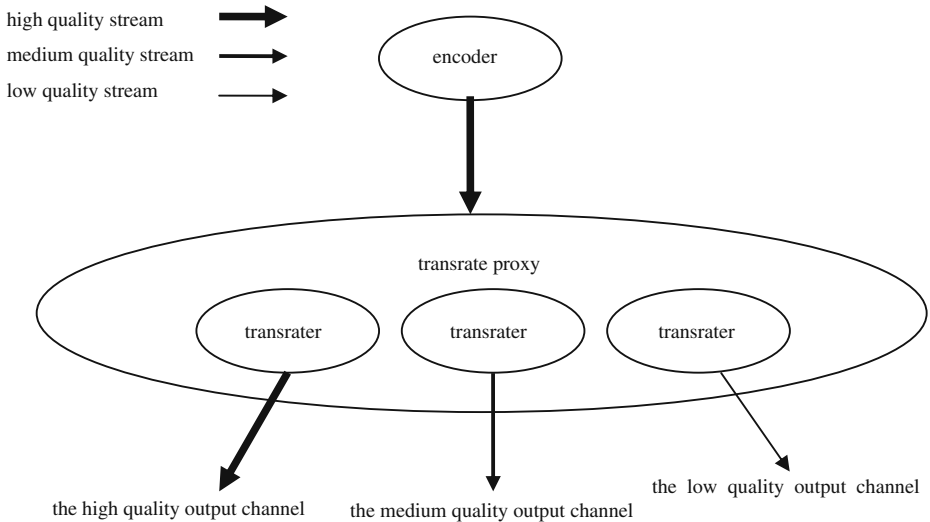


Fig. 3 The source node overview

together to provide three output channels of real time encoding stream simultaneously. The source node uses the transrating proxy to replicate the original one input stream channel to three output channels with different quality (or profile) streams. These three output channels output three different levels of quality with real time streams in the SmartPeerCast source node; the high quality stream with 2 Mbps bits rate, the medium quality stream with 1 Mbps bits rate, and the low quality stream with 500 Kbps bits rate respectively. An ALM tree is created for every output channel to broadcast different quality real time stream. Every ALM tree only allows the peers with the enough uploading capability to forward the stream which it receives to join. Thus, the peers are grouped into three clusters based on their uploading bandwidth. This three different quality clusters' definitions in the SmartPeerCast are similar to the three layers' definitions for the MPEG-4 audio and video objects in the research [1]. But the SmartPeerCast is different from the research [1] because the SmartPeerCast source node is codec awareness and can use any encoding format at the input encoder. If the output connections of a lower quality output channel were saturated, the higher quality output channels still could serve the peers with the lower uploading bandwidth by using the transrating engine service in the source node. The transrating engine usage is described in the section of "The New Peer Joining Algorithm".

The Peers; they are the majority nodes in the SmartPeerCast. The peers are assumed to be heterogeneous with different uploading bandwidth because the SmartPeerCast is targeted to be the framework of an Internet widely RTVB application. The transrating engine and the receiver scheduler are the two important modules of the peer. The sending peer uses the transrating engine to achieve two functions; (1) adjusting its uploading stream quality dynamically based on its receiving peers' QoS events to smooth the playback jitters of the receiving peers when there occurs the network link congestion. The Algorithm 2 presents the detail description for this QoS framework and, (2) reducing the stream's quality between the heterogeneous peers to avoid the bandwidth bottleneck when the sending peer with the high quality stream forwards the stream to one of its intermediate descendant node which has the low uploading bandwidth. With the transrating engine, the peers in the high quality stream ALM tree can serve as the parent node of the peers in the medium and the

low quality stream ALM tree. Thus the SmartPeerCast improves the overall system's performance by fully utilizing the ALM trees' leaf nodes. The receiver scheduler is the second important module running in the receiving peer in a periodical interval. SmartPeerCast configures this interval value to ten seconds. Similar to the unchoking algorithm used in the BitTorrent, this receiver scheduler is used in the SmartPeerCast receiving peers to monitor the forwarding QoS of their parent nodes. Whenever the sending peer cannot meet the receiving peer's QoS requirement, the receiving peer runs the Algorithm 1 again to select a new parent.

```

Join Broadcasting Tree ( $T$ ,  $sid$ ) {
    /* (1) register itself ( $P_i$ ) to the tracker  $T$  */
    register_to_tracker( $P_i$ ,  $sid$ ,  $ub_i$ );
    /* (2) to get the quality level of  $P_i$  */
     $Q_i$  = receive_quality_level( $T$ );
    /* (3) to get the list of potential candidate peers which can serve as the  $P_i$ 's parent node
        and assign the peers at the candidate_peers to the different clusters according to his
        quality level value
    */
    candidate_peers = receive_candidate_peers( $T$ )
    peers_clusters  $\leftarrow$  candidate_peers

    /* (4) To find the best suitable node from the peers_clusters array with the same or above
        quality levels
    */
    level =  $Q_i$ ;
    while ( level <= HIGH_QUALITY ){
         $P_j$  = find_best_hit( peers_clusters[ level ] );
        if (  $P_j$  is found )
            break;
        else
            level = level + 1;
    }
    /* (5) To stream the data from the selected parent peer */
    transrate = false;
    if (  $Q_i$  <  $Q_j$  )
        transrated = true;
    handshake_and_streaming( $P_j$ , transrated )
}

```

Algorithm 1 the new peer joining

3.1 The new peer joining algorithm

The Algorithm 1 shows the procedure as to how the new peer P_i joins the SID stream ALM trees (refer to the Table 2 for the notation of P_i and SID). There are five steps for the new

SmartPeerCast peer to join the video broadcasting service and play the real time streaming:

- (1) Registration. The new peer P_i connects to the tracker T with the message $\langle peer\ id, uploading\ bandwidth, the\ maximum\ acceptable\ input\ connections, stream\ id \rangle$. The P_i 's uploading bandwidth value included in this initial registration message is very important because the tracker node decides which stream quality ALM tree P_i can join based on its uploading bandwidth value. The quality level value is calculated in the following step; the SmartPeerCast client runs the wizard to help user configure its uploading bandwidth when it is running for the first time. If a false uploading capability were provided by P_i , the Smart QoS algorithm could detect this false uploading bandwidth declaration and would then punish the peer by migrating it to a low quality stream ALM tree. Besides the punishment, the Smart QoS algorithm can also award the peer with the large uploading contribution by migrating it to a high quality stream ALM tree even though the user configures the uploading bandwidth to be smaller than its real contribution.
- (2) Calculating the stream quality level that P_i can receive. In the SmartPeerCast framework, P_i can only receive the stream with the bits rate less than its uploading bandwidth. For example, if P_i 's available uploading bandwidth is 700 Kbps, it can only receive the low quality stream with less than 700 Kbps bits rate broadcasting from the low quality output channel of the source node. The tracker node assigns the P_i 's quality level based on its initially configured uploading bandwidth value.
- (3) Grouping the peers to three different quality clusters. After P_i connects to the tracker node, it receives all the candidate peers which can serve as the parent node. P_i traverses the candidate list to divide them into three different quality clusters according to the candidate's uploading bandwidth. The candidates list includes all the peers with a higher quality level than P_i because these higher quality peers can provide the media stream forwarding service to P_i in SmartPeerCast by the transrating engine.
- (4) Finding the parent node. P_i traverses the three clusters to select the best suitable peer as the parent. First, it searches in the cluster with the same quality level as itself. If all peers in this cluster are saturated and cannot serve as the P_i 's parent, it then continues to search the peers in the clusters with higher quality level until the best suitable node is selected or this process quits as a failure. The peers in the same cluster as P_i will have the higher priority than the peers in the higher level quality clusters when P_i is selecting the parent node. If a peer in the same cluster can be selected as the parent node, the parent node needs not to waste its computing cycles to run the transrating engine. The above policy makes the SmartPeerCast overall performance better.
- (5) Receiving and playing the stream. P_i does the hand shake with its parent node; if the handshake were OK, P_i would start to receive the streaming data and play it, and if the nodes in the higher quality cluster were selected as the parent node, this parent node would have the higher quality than P_i . The transrating engine is used the at the parent node to reduce the stream bits rate when forwarding to avoid the transferring bottleneck occurring at the P_i 's descendant nodes in the ALM tree.

3.2 The Smart QoS algorithm

Figure 4 shows the Smart QoS framework between the receiving and the sending peer. The player buffer in the receiving peer is divided into three segments by the high and low water mark position. The current playback position in the player buffer is used to compare with the low and the high water mark to trigger the QoS events. The receiving peer P_i 's QoS events are used to indicate the receiving quality from the sending peer P_j . When P_j receives these QoS events, it adjusts its forwarding stream's bits rate dynamically by the transrating engine to smooth the P_i 's playback jitters. Two types of QoS events are defined in the

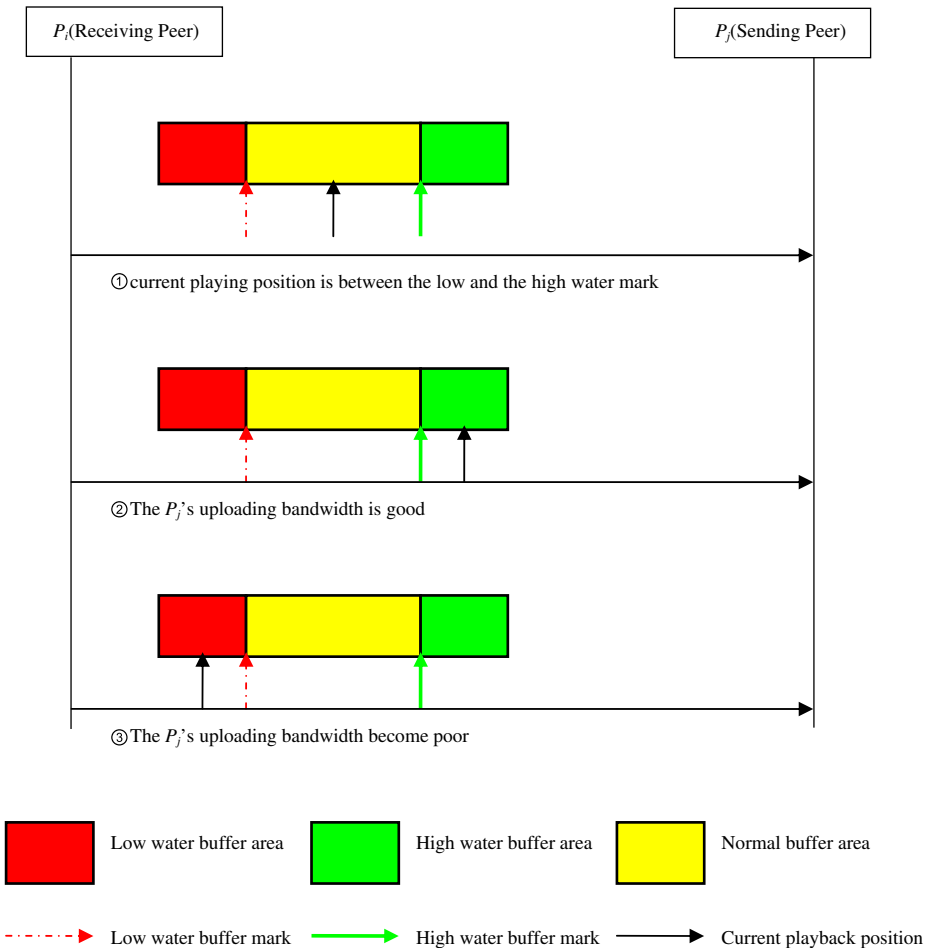


Fig. 4 Smart QoS overview

Smart QoS algorithm and they are triggered by the low and the high water mark boundary checking respectively. The Smart QoS uses the above two QoS events and the transrating engine to eliminate the P_i 's playback jitters when there is network link congestion between P_i and P_j . The jumping of ②→③ in the Fig. 4 illustrates the first type QoS event. It is triggered when the current playback position is going down from the high water buffer area to the normal buffer area. This QoS event indicates that there is a bandwidth bottleneck when P_j is forwarding the stream to P_i because the P_j 's data forwarding speed is far slower than the P_i 's playing speed and, the streaming data cached at the P_i 's player buffer is dropping quickly. When the sending peer P_j receives this QoS event, it must kick off the transrating engine to reduce the forwarding stream's bits rate to avoid the packet loss and the playback jitters occurring in P_i . The transrating engine is used here to reduce the stream's bits rate to compensate for the dropping of the throughput. The second type QoS event is triggered when the current playback position is growing from the normal buffer area to the high water buffer area as the jump ①→② showed in the Fig. 4. This second type QoS event indicates that the receiving peer P_i 's playback speed has caught up with the P_j 's data

forwarding speed. It shows that the network throughput between P_j and P_i becomes larger than the stream's bits rate P_j forwarding. P_j increases the stream quality by the transrating engine to improve the P_i 's playback quality when this QoS event is received. The transrating engine is used here to fully utilize the network bandwidth to have a better stream quality. The Algorithm 2 shows the Smart QoS algorithm detail implementation.

```

L :           the  $P_i$ 's player buffer size in bytes
LWM:         the low water mark of buffer  $L$ , it's defined as  $L * 20\%$ 
HWM:         the high water mark of buffer  $L$ , it's defined as  $L * 80\%$ 
p:           the current playback position in buffer  $L$ 
Tlwm:       The total time that  $P_i$  is staying with condition " $p \leq LWM$ "
Thwm:       The total time that  $P_i$  is staying with condition " $p \geq HWM$ "

do_play() {
    // (1) Do the initial buffering until the high water mark is reached. The startup time is decided by this initial buffering
    // process.
    while(  $p \leq HWM$  )
        receive_packet_from(  $P_j$  );
    // (2) kick off the playback
    while ( bQuit ) {
        player_play_packet();
        receive_packet_from(  $P_j$  );

        // First type QoS Event: Triggering the first type of QoS event as the jump ②→③ shows in the Fig. 4
        // to reduce the stream quality
        if(  $p \leq LWM$  ) {
            if ( $p$  is jumped from the normal buffer area to the low water mark buffer area)
                trigger_qos_event(  $P_j$ , reduce_quality );
            increase Tlwm;
        }

        // Second type QoS Event: Triggering the second type of QoS event as the jump ①→② shows in
        // the Fig. 4 to increase the stream quality
        if(  $p \geq HWM$  ) {
            if ( $p$  is jumped from the normal buffer area to the high water mark buffer area)
                trigger_qos_event(  $P_j$ , increase_quality );
            increase Thwm;
        }
    }
}

```

Algorithm 2 Smart QoS

There are total two steps in the Smart QoS implementation as demonstrating in the Algorithm 2. The first step is to prepare the receiving peer P_i 's initial playback buffer. The initial buffering can reduce the playback jitters and improve the playback experience of the RTVB clients and, an increase of the initial buffer can have better performance improvement in the playback jitters. But, it also increases the startup time delay when buffering the data. The startup time is always the most critical performance requirement for

the RTVB application. Most of the RTVB applications, such as the IPTV and IP Video surveillance system, require the startup time to be less than two seconds. SmartPeerCast uses the high water mark of the receiving peer's player buffer as the initial buffer size which is defined as the 80% of the whole player buffer length in the Fig. 4. After the initial buffering step, the receiving peer P_i kicks off the stream play by keeping receiving the data from the sending peer P_j as the Algorithm 2 shows in the second step. There are two possible types of QoS events during the second step running process: (1) Bad QoS Event; it is shown by the first type QoS event case in the Algorithm 2. The receiving peer P_i finds that its current playback position is dropping below the low water mark and reports the QoS event to the sending peer P_j . P_j then reduces the forwarding stream quality immediately after receiving this QoS event. The receiving peer P_i 's continuous playback is not interrupted and kept smooth by the transrating action in P_j . (2) Good QoS Event; it is shown by the second QoS event case in the Algorithm 2. This time P_i finds that its current playback position is increasing from the low water mark to the high water mark and reports the event to P_j . P_j tunes the stream quality up to meet the bandwidth changing. P_j forwards the better quality stream to P_i so that it can have a better playback experience when the network condition becomes better. The two types of QoS events used in the Smart QoS algorithm make the sending peer change the stream's bits rates adaptively according to the condition of the network link congestion, so the receiving peers can always have the smooth playback experience by receiving the best suitable quality stream over the current network condition. Besides monitoring the stream forwarding QoS between the sending and the receiving peers, Smart QoS algorithm also counts the sending peers' uploading contribution. Algorithm 2 defines the parameter T_{lwm} to count the total time that the current playback position is below the low water mark during the playing in P_i . The T_{lwm} value varies during the playback QoS switching between P_j and P_i . After the second type QoS event occurs, the T_{lwm} value continues to increase until P_i 's current playback position returns back to the normal buffer area. Thus, if the overall QoS between P_i and P_j is worse, a larger T_{lwm} value will be gotten at this step. The receiver scheduler algorithm discussed in the next section uses the T_{lwm} value to kick off the new parent selection when it reaches the threshold. The T_{lwm} value is also used in the tracker node to award and punish the peers' bandwidth contribution in the next section's algorithm.

3.3 The receiver scheduler algorithm

Algorithm 3 shows the receiver scheduler implementation. It is running periodically as the interval Δt at the receiving peer. The receiving peer uses the scheduler to choke or unchoke the sending peers. In the Algorithm 3, the receiving peer P_i monitors the sending peer P_j 's uploading QoS to select the new parent node when the old one cannot provide a good QoS. The Bandwidth Sharing Index (BSI) in SmartPeerCast is defined to demonstrate the forwarding quality difference between the sending and the receiving peer. Equation 1 defines the BSI below:

$$BSI = \frac{T_{lwm}}{T_{total}} \quad (1)$$

In Eq. 1, T_{lwm} presents the total time that the current playback position is below the low water mark during the playing in P_i as it is defined in the Algorithm 2. T_{total} presents the total streaming time between P_j and P_i . BSI is defined as the time when P_i plays the low quality stream versus its total playback time. According to its definition, BSI value indicates

P_i 's playback jitters probability when streaming from P_j . When the link's *BSI* value between P_i and P_j becomes bigger, it shows that this link transferring quality becomes worse. The worse forwarding quality in this link may be caused by the network congestion or the resources' competing in P_j . Algorithm 3 uses a constant *BSI* threshold value to judge if the sending peer could or could not provide the required streaming quality. The fourth step shows the detail: If the link's *BSI* value between P_i and P_j goes beyond the *BSI* threshold, it means that P_j is no longer a good parent node. P_i needs contact the tracker node again to select a new parent and, the new parent finding process shall repeat the Algorithm 1 run. In the third step, the tracker checks the sending peer's *BSI* value to decide if the peer has configured the fake uploading bandwidth and acts as a leech node. SmartPeerCast configures the *BSI* threshold value to be 5%, 10%, and 20% respectively in our experiments. If P_j 's *BSI* value goes beyond the threshold value when it is forwarding the stream data to P_i , the tracker thinks P_j is facing a forwarding bottleneck and it cannot stay at the same ALM tree as P_i . P_j is then migrated to the lower quality stream ALM tree by the tracker. The above guideline is the incentive mechanism used at the SmartPeerCast; it is used to reward or punish the peers' bandwidth contribution.

Δt : the scheduling interval ($\Delta t = 10s$)

T_{total} : the total streaming time from the parent node (ex, P_j)

BSI: the uplink bandwidth sharing index between P_i and P_j

jitter_theshold: a const threshold value to indicate if the sending peer can provide enough forwarding quality or not

```
Scheduler( $T, P_j$ ) {
    //(1) calculate the uplink bandwidth sharing index between  $P_i$  and  $P_j$ .
     $BSI = T_{lwm} / T_{total}$ ;
    //(2)  $P_j$  meets a forwarding bottleneck
    if(  $BSI \geq$  jitter_theshold ) {
        //(3) reports the BSI value of  $P_j$  to the tracker so that the tracker can award or punish  $P_j$ 
        // according to its bandwidth contribution or cheats
        update_peer_uplink_capability( $T, P_j$ );
        drop the connection between  $P_i$  and  $P_j$ ;
        //(4) run the algorithm 1 again to rejoin the video broadcast tree and select a new parent
         $P_j =$  Join Broadcasting Tree(  $T, SID$  );
    }
}
```

Algorithm 3 the scheduler

4 Performance evaluation

The startup time and packet loss ratio are the two key performance parameters of the P2P RTVB application. To prove and verify the SmartPeerCast performance, the NTCUns [15] simulator is used to run and check the prototype implementation. The experiment's target is to show that the SmartPeerCast is a stable solution with good performance for a medium scale IP surveillance system. For example, it can at least support 500 live streaming playback PC clients simultaneously from one IP camera output channel.

4.1 Experimental setup

The NCTUns is a Linux kernel based network simulator and it can run the real application inside the simulated network node directly. It can provide the same P2P network running environment as the real Internet based P2P applications. The SmartPeerCast prototype implementation is based on the PeerCast [19] open source project by adding our new algorithms. Because the NCTUns can run the prototype binary code as a normal Linux application from the simulated network node inside, it is very easy for us to design and implement our experiments. Since the SmartPeerCast applications can be run directly in both the NCTUns simulator and the real Linux environment without any modifications, it is believed that the experiments results gotten from the NCTUns should be same as the case running over the real Internet.

In our experiments, the NCTUns simulator is running at the high performance PC with Intel Duo Core CPU, and 2 GB DDR memory. Because the NCTUns can only support maximum 64 network nodes at one Linux PC, total 10 cascaded connecting PCs in an intranet network are used to simulate the total 390 peers in our experiments. Every emulation station is configured as a sub network with 39 nodes as shown in Fig. 5. Two peers at the 10th emulation station are picked up and configured as the tracker and the source node respectively. The peers are divided into three groups and then organized by the SmartPeerCast application to the ALM trees which can play and forward the high, the medium, and the low quality stream respectively. The uploading bandwidth of the nodes at high, medium, and low quality clusters are configured as 400 Kbps, 200 Kbps, and 100 Kbps respectively. The simulated network is an Autonomy Network (AS) with 1Gbps throughput in the backbone. As shown in Fig. 5, the overlay network is divided into ten sub network areas by the virtual routers at the simulation stations. The nodes with different quality uploading bandwidth are placed over the ten sub networks. For simulation station

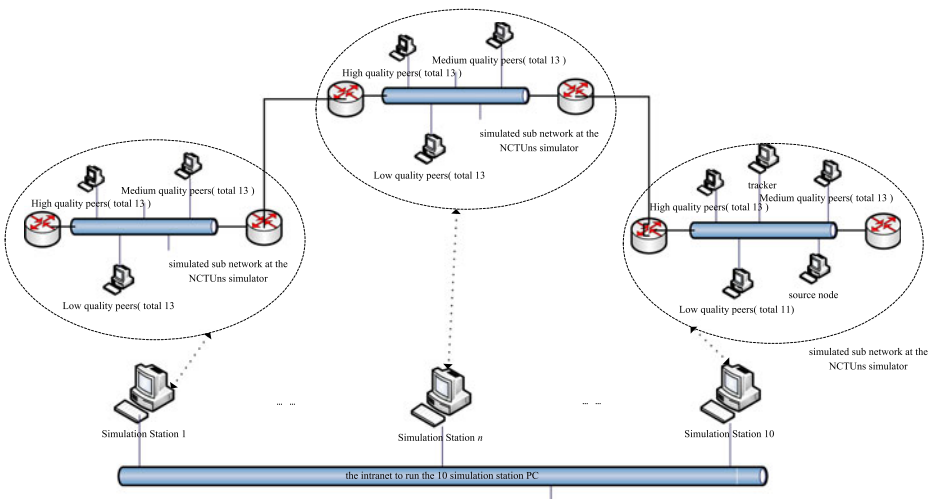


Fig. 5 The network topology of NCTUns simulation experiment. Ten connected PCs are running at an Intranet to simulate 10 connected sub networks. Every PC runs a NCTUns simulator as a simulation station to simulate 39 SmartPeerCast nodes, so there are total 390 peers running at our experiments. At simulation station 1–9, same account peers (total 13) of high, medium, and low quality are placed at one station (or one sub network). At station 10, two of the low quality nodes are replaced with the source node and the tracker node respectively

1–9, there are 13 high, medium, and low quality peers placed in the simulated sub networks respectively, and two of the low quality peers at the 10th simulation station are replaced onto the tracker and the source node. To make the experiments simple to implement and easy to analyze, a packet generator in the source node is used to simulate the three output channels of the real time stream with 200 Kbps, 100 Kbps, and 50 Kbps Const Bits Rate (CBR) encoding. For the real RTVB application implementation, this packet generator can be replaced with the HTTP based real time encoder directly; such as an H.264 IP Camera. In our experiments, every peer's initial buffering time is configured as ten seconds and the peers join the video broadcasting tree in the flash crowd mode. All peers connect to the source node simultaneously to request the stream for every experiment run. Every experiment lasted for ten minutes and every 20 experiments were run for three different *BSI* threshold values (%5, %10, and %20) respectively to collect the data.

To show the performance improvement of the SmartPeerCast, comparisons are made with the other two systems in our experiments: (1) PeerCast; it is the original PeerCast implementation with a minor modification. PeerCast is a single ALM tree structure. When a new peer joins the overlay network, a parent node will be selected randomly by the tracker and sent back to the new peer. After receiving the candidate parent, the newly joined peer does the hand shake with it. If the handshake were successful, the new peer would join the ALM tree to receive and play the stream data from the parent. Otherwise, the newly joined peer will repeat the above steps continuously until finding the parent node. Because the original PeerCast can only support one output channel of the real time stream at the source node, its source node is modified to support three real time output stream channels and, every channel is a CBR stream with 200 Kbps bits rate. With this modification, the PeerCast experiments can be run in the same condition as the SmartPeerCast. (2) MultiPeerCast; it is the simple version of the SmartPeerCast. It does not implement the Smart QoS algorithms, transrating engine, and the incentive mechanism. MultiPeerCast uses the same implementation of the tracker and the source node as the SmartPeerCast, so MultiPeerCast still provides three different quality real time stream output channels in the source code. Likes the SmartPeerCast, MultiPeerCast also groups the peers into three different quality forwarding stream clusters.

In our experiments for SmartPeerCast, we don't discuss experiment cases of peer departure because we think it is easy for a peer to find the new parent node and fix the streaming overlay with the help of the tracker node. Whenever the receiving peer detects that its parent node is unreachable, it will invoke Algorithm 1 to rejoin the overlay just like what it does when the forwarding QoS of its parent node degrades as presented at the Algorithm 3. At the same time, because almost all the P2P RTVB systems are based on the ALM tree or its variant, they have covered the peer churns and tree fixing issues already, the more details about this topic can be referred at the researches [1, 4, 23, 26, 35]. Our simulation experiments don't take the transrating CPU loading of the peers into account of peer heterogeneity either because a ffmpeg based on-the-fly transrating of H2.64 video only takes on average 20% of the CPU load of an Intel Duo Core CPU with 2 GB RAM PC. The PCs with this kind of configuration are very common today and can have three simultaneous instances of this on-the-fly transrating. When the SmartPeerCast is targeted as a PC based surveillance system, all the receiving peers are assumed as PC based which are capable enough to transrating and forward the stream on-the-fly, so all peers in our experiments are assumed to be powerful enough to run two transrating instances simultaneously. But the simulation stations (PCs) are not powerful enough for every simulated node to run the real ffmpeg transrating engine due to the large number of simulated nodes at one PC, so the nodes in the experiments only modify the packet

attributes to mark that the packet has been “transrated” when a transrating process is kicked off.

4.2 The startup time performance results

Experiment results for the startup time of the PeerCast, MultiPeerCast and SmartPeerCast systems are compared and shown in the Fig. 6. The results show that the SmartPeerCast can improve the peer’s startup time efficiently because most of the peers in the experiments begin the playback within 15 s. Considering that the peer’s initial buffering time was configured to be fixed at 10 s and, all peers are requesting the real time video stream in the flash crowd mode, SmartPeerCast has achieved a quite good startup time performance. This is because that the SmartPeerCast system groups the peers into clusters to forward different quality stream according to its uploading throughput. The peers in every cluster can only play and forward the stream in the quality it can adapt. The peer with the high uploading bandwidth can only join the ALM tree rooted at the source node’s high quality stream output channel. The peer with the low uploading bandwidth can get the stream from the nodes of the higher quality clusters, but the stream quality will be decreased by the transrating engine when it is forwarded to the receiving node. This design avoids the bandwidth bottleneck between the heterogeneous peers because a peer never has the chance to forward a real time stream whose bits rate is larger than its uploading bandwidth. The SmartPeerCast peers can get the stream faster by eliminating the transfer congestion and utilizing more leaf nodes at the ALM trees. Thus, the SmartPeerCast have the better startup time performance. Comparing to SmartPeerCast, PeerCast peers have the longest startup time in our experimental results. In the PeerCast experiments, the receiving peer with the low uploading bandwidth often selects the peer with the high uploading bandwidth as the parent node. When stream is forwarded to the receiving peer in high quality, the receiving peer cannot forward the stream to its child nodes with the original bit rate anymore due to its limited uploading bandwidth. Thus, it causes the content bottleneck, and all the descendant nodes of the above receiving peer suffer a long startup delay to buffer the data. The experiment gets the worse startup time result when the above receiving peer is the first degree node in the ALM tree to receive the real time stream

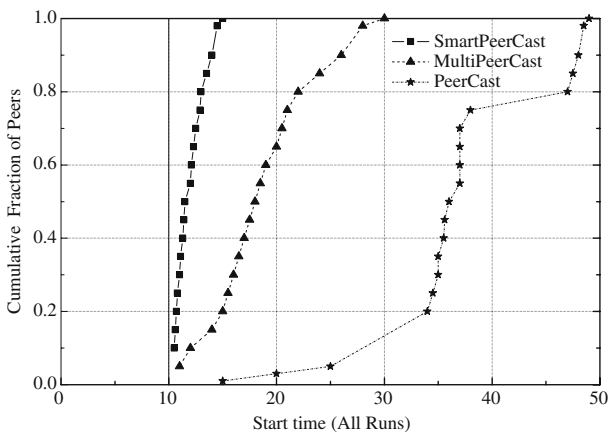


Fig. 6 Cumulative distribution function of the startup time, the data is collected over the total 60 runs (20 runs per *BSI* threshold value). The vertical line represents the earliest possible startup time that a peer could start the stream playback

from the source node's output channel directly. MultiPeerCast has the better startup time performance than the PeerCast because it uses the same clustering mechanism as the SmartPeerCast. It avoids the bandwidth bottleneck occurring between the heterogeneous peers. MultiPeerCast has the worse startup time performance in our experiments results. Comparing to the SmartPeerCast, the leaf nodes of the higher quality ALM trees in MultiPeerCast cannot provide the forwarding service to the nodes in the lower quality ALM trees because MultiPeerCast do not have the transrating engine. Thus, the uploading resources of the leaf nodes in medium and high quality ALM trees are not utilized. In the MultiPeerCast experiments, the peers with the low and the medium uploading bandwidth have longer startup time than the same type of peers in the SmartPeerCast experiment because these peers have more candidate parent nodes in the SmartPeerCast.

4.3 The packet loss ratio performance results

The packet loss ratio is another key performance metric besides the startup time in the RTVB application. The packet loss will cause the receiving peer to have playback jitters. The playback jitter becomes worse with more packets lost in the receiving peer because the decoder will experience the playback jitters when the necessary audio and video frames are dropped. In our experiment, Eq. 2 as defined below is used to measure the packet loss at the receiving peer. In Eq. 2, T_{send} is the sending timestamp of the stream packet and it is assigned by the source node when it is streaming out. $T_{initial}$ is P_i 's initial buffering time and $T_{deadline}$ is defined as the packet's playback deadline in P_i . The Eq. 2 shows the condition when the packet will be considered as lost in P_i . When T_{send} , $T_{initial}$, and $T_{deadline}$ all meet the condition set by Eq. 2, this packet will be dropped in P_i .

$$T_{send} - T_{initial} > T_{deadline} \quad (2)$$

To evaluate the packet loss ratio performance, the total lost packets and the total packets P_i expects to receive need to be counted. In our experiments, P_i begins to count the dropped packets after the stream starts playing, and stop the counting until P_i quits the SmartPeerCast system. The total dropped packets are defined as number N . M is defined as the total packets that P_i expects to receive in the duration when P_i is counting the dropped packets number N . Equation 3 calculates the P_i 's packet loss ratio by dividing N with M :

$$packet\ loss\ ratio = \frac{N}{M} \quad (3)$$

Equation 4 gives the value of M :

$$M = \frac{T_{duration}}{T_{packet}} \quad (4)$$

In Eq. 4, $T_{duration}$ presents the total time that P_i is watching and staying with the stream, and T_{packet} is the playback duration for a packet. In our experiments, $T_{duration}$ is a fixed number in P_i and it always equals to ten minutes in every experiment. T_{packet} is also a constant value in our experiments because it is assumed that the stream is encoded in CBR and, the stream is divided to packets with the fixed playback duration in the source node. The packets have variable data size in different quality stream, but they always have the same playback duration. According to the definition of $T_{duration}$ and T_{packet} , it is concluded that M is also a constant value. Thus, the packet loss ratio defined in Eq. 3 depends on the value of N .

Figure 7 shows the packet loss experiment results for PeerCast, MultiPeerCast, and SmartPeerCast systems. SmartPeerCast has the smallest packet loss ratio of the three systems. The receiving peers in a real deployed SmartPeerCast still have the chance to get better packet loss ratio performance than the results shown in the Fig. 7 by increasing the initial buffer size. PeerCast's packet loss results in our experiments are too poor to be compensated by increasing the peer's initial buffer size. Our experiments results for PeerCast show that PeerCast cannot work at all in a heterogeneous P2P system, especially when a peer with the low uploading bandwidth is selected as the parent node of the peer with the high uploading bandwidth. All the descendant nodes of this low uploading bandwidth peer will suffer the heavy packets loss due to the uploading bandwidth bottleneck. The results also explain why there is so many complaints for PeerCast's bad video broadcasting services over the Internet at the PeerCast forum [19]. As we have discussed at the above section for the startup time performance results, the peers clustering mechanism used in SmartPeerCast prevents the bandwidth bottleneck occurring in the stream forwarding path efficiently. The results in the Fig. 7 show that this mechanism improves the packet loss performance too. MultiPeerCast has the smaller lost packets than PeerCast because it uses the same clustering mechanism as SmartPeerCast, but it has the larger lost packets than SmartPeerCast. Although MultiPeerCast eliminates the bandwidth bottleneck between the heterogeneous peers by grouping them based on the uploading bandwidth, the leaf nodes in the higher quality MultiPeerCast ALM trees cannot provide the stream forwarding service to the nodes in the lower quality ALM trees due to the lack of transrating engine. This will produce more playback jitters in the MultiPeerCast receiving peers because of the content bottleneck [35], especially when all the receiving peers send the requests in the flash crowd mode. Different from MultiPeerCast, SmartPeerCast uses the Smart QoS framework between the receiving and the sending peers to adjust the stream bits rate adaptively based on the network condition. The Smart QoS can have a better control for the packet loss ratio by eliminating the impacts of the network congestion.

4.4 The ALM tree depth and delay discussion

The cumulative delay between the relay peers and the source node can be presented by the tree depth of the ALM tree. For the RTVB application, the tree depth presents the total

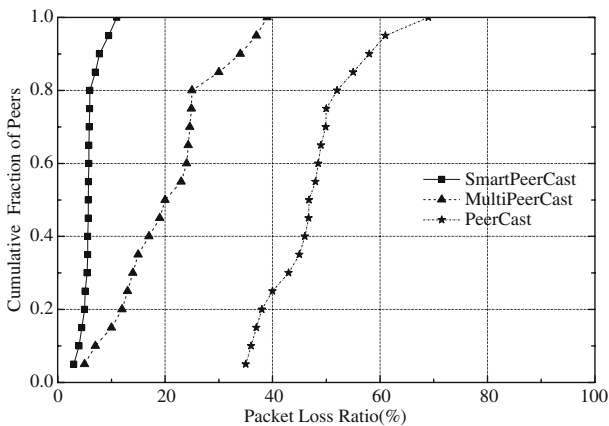


Fig. 7 Cumulative distribution function of the packet loss ratio and the data is collected over the total 60 runs (20 runs per *BSI* threshold value)

forwarding hops between the source node and the leaf nodes of the tree. Because every relay node on the forwarding path will cache the live streaming data at its playback buffer, the higher depth value means the larger cumulative scene delay between the source node and the playback client. Some of the RTVB applications aren't very sensitive to this parameter; such as the TV broadcasting program; but some RTVB applications are very sensitive to the cumulative delay; such as the surveillance system where a playback client always wants to get the real time scene at the source node as soon as possible. Thus, a well designed tree based RTVB system should limit the tree's depth by putting the capable peers as near as possible to the source node of the ALM tree as the Dag-stream [23] has studied at its performance evaluation experiments. Figure 8 shows the tree depth of the total 20 experiments run for PeerCast, MultiPeerCast and SmartPeerCast respectively. From the results, it can be concluded that the PeerCast has the deepest tree depth because it is a single ALM tree overlay structure, but the MultiPeerCast and the SmartPeerCast have shorter cumulative delay (or depth) than the PeerCast by adapting a multiple ALM tree overlay structure. The SmartPeerCast performs the best at the tree depth metric with the help of two designs: (1) The peers are grouped into three clusters due to their uploading bandwidth, thus the single ALM tree overlay at the PeerCast is reorganized into three ALM trees rooted at the source node of the SmartPeerCast system. (2) There are more chances that the high quality peers of the SmartPeerCast to serve more children by the transrating service than the same quality peers at the MultiPeerCast system.

4.5 The incentive scheme effectiveness discussion

In SmartPeerCast, the tracker node maintains a central database storing all information about peers including ID and IP addresses, incentives, and so on as described in the Section 3. When a peer is forwarding the stream to a receiving peer, the receiving peer keeps reporting this parent node's uploading contribution information to the tracker node by using the BSI value. Equation 1 in Section 3.3 defines the BSI which plays the key role in the SmartPeerCast incentive framework. Whenever a sending peer's BSI value is lower than the configured threshold value, the child-nodes of this peer will drop it and find a new parent node; the tracker also will punish it by migrating it to the lower quality ALM trees, and even kick the sending peer out from the service if its BSI value still cannot meet the

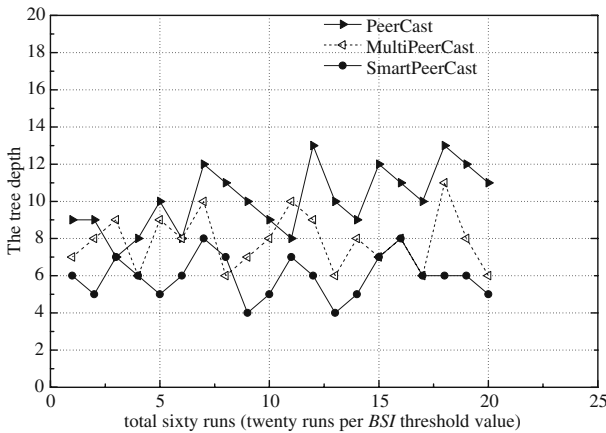


Fig. 8 The tree depth of PeerCast, MultiPeerCast and SmartPeerCast at the total 60 runs (20 runs per BSI threshold value)

threshold value requirement of the lowest quality ALM tree. The tracker node eliminates the leech peers of SmartPeerCast system by keeping collecting every peer's BSI information. In SmartPeerCast, the peer's contribution information (indicated by the BSI value) is only effective within the streaming session and not stored in the database persistently, so the leech peers still have the chance to rejoin the streaming session again by reconfiguring its uploading bandwidth and reconnecting the stream again.

Results of Fig. 9 demonstrate how the SmartPeerCast's startup and packet loss ratio performance are improved by the BSI value based incentive scheme. As state in Section 3.3, the BSI threshold values are configured to be 5%, 10%, and 20% respectively in our SmartPeerCast experiments. Twenty experiments are run per BSI threshold configuration case to calculate the average performance values of the startup time delay and the packet loss ratio. The left side columns of Fig. 9 show the average packet loss ration for the SmartPeerCast receiving peers, and the right side columns of Fig. 9 provide the average startup time of the SmartPeerCast receiving peers for three different BSI threshold value configurations. Results show that a smaller BSI threshold configuration has a better system performance on both the startup time and the packet loss. According to the BSI definition and the design of SmartPeerCast incentive framework, a reasonable smaller system BSI threshold value can help the tracker to detect and deny the leech peers earlier, and to group the peers in different quality clusters more precisely and quickly. But it is not always true that a smaller BSI threshold value is better. If the BSI threshold value is too small, the receiving peers will be too sensitive to tolerate the network jitters between the sending peer and the receiving peer, or the computing resource jitters of the sending peer. It will result in making the peers to reconnect and rejoin the stream broadcasting tree too frequently. The sending peers also must leave and rejoin the stream session frequently to clear its bad incentive records caused by the too sensitive BSI threshold. Thus, the system's overall usability will be reduced substantially due to the frequent nodes' leaving and joining events. It's still an open issue about how to select a BSI threshold value by a theoretical method in the SmartPeerCast system. This issue will remain to be studied in future works.

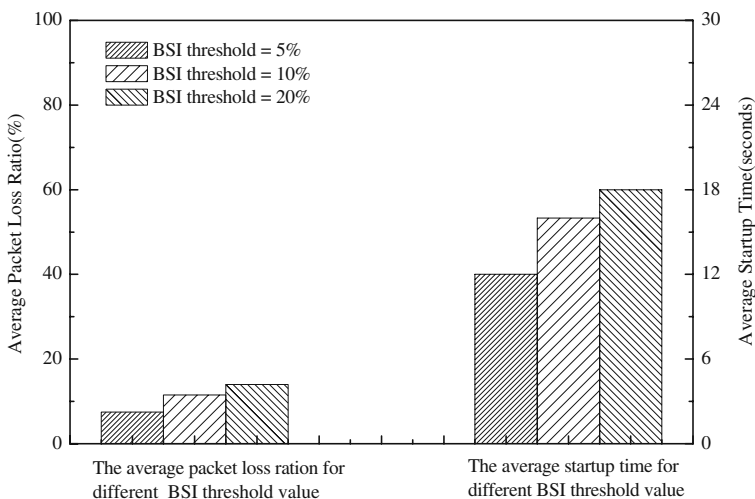


Fig. 9 The BSI threshold value based incentive scheme effectiveness results for the packet loss ratio and startup time performance of the SmartPeerCast system. The data is collected by 20 runs per BSI threshold value

5 Conclusions and future works

In this paper, SmartPeerCast as the smart P2P real time video broadcasting framework is presented with the large scale supporting and QoS enhancement. SmartPeerCast achieves a shorter startup time delay and a smaller packet loss ratio by the new P2P overlay design and the Smart QoS algorithm. SmartPeerCast groups the peers to the different clusters according to their uploading bandwidth. Three different ALM trees are generated based on the clusters to broadcast different quality streams to eliminate the bandwidth bottleneck between the heterogeneous peers. The peers at the same ALM tree use the transrating engine to adjust the stream quality automatically by the QoS framework between the sending and the receiving node. This Smart QoS algorithm also reduces the packet loss efficiently by compensating for the network link congestion. The transrating engine is used between the peers in different ALM trees to fully utilize the leaf nodes' resources which are in the high quality ALM trees. The clustering policy and the transrating engine using across the heterogeneous peers improve the performance by avoiding the bandwidth bottleneck. SmartPeerCast uses the receiving peer to collect the sending peer forwarding QoS and reports it to the tracker which would then awards and punishes the peers based on their uploading QoS records. This incentive mechanism is used to award the peers who contribute more uploading bandwidth with the higher playback quality. The experiment results show that SmartPeerCast is a very scalable P2P and high performance RTVB framework. SmartPeerCast is also deployed into the video surveillance system which is under development.¹

In the planned future works, the peers will be grouped into the clusters based on the round trip time (RTT) values between the peers. The tracker node creates the ALM trees by linking the peers with the shortest physical network path together to check and study if the Smart QoS algorithm can have a better performance than the proposed ALM tree generating method in this paper. According to the research [1], the RTT based method can reduce the data packet replications in the physical link, so we want to perform experiments to verify if the RTT based method can have a better startup time delay and a packet loss ratio performance. We also want to investigate if we can have a better P2P RTVB framework by combining the above metrics together.

Acknowledgement The Authors would like to thank Dr. Albert. S. Wang for his contribution to this manuscript. Dr. Wang was previously with Agilent Technologies and is now a Visiting Fellow at the Advanced Digital Technology and Instrument Institute of Zhejiang University, Hangzhou, 310027, China (e-mail: al-bert_s_wang@yahoo.com).

References

1. Ahmed T, Mushtaq M (2007) P2P Object-based adaptive Multimedia Streaming (POEMS). *J Netw Syst Manage* 15(3):289–310
2. Annapureddy S, Guha S, Gkantsidis C, Gunawardena D, Rodriguez PR (2007) Is high-quality vod feasible using P2P swarming?. *Proceedings of the 16th international conference on World Wide Web*. ACM New York, NY, USA, pp 903–912
3. Assuncao PAA, Ghanbari M (1977) Transcoding of single layer MPEG Video into lower rates. *IEE Proc Vis Image Signal Process* 144:377–383

¹ Project (No. 2005C11001-02) supported by the Key Science and Technology Projects Foundation of Zhejiang Province

4. Cai Y, Zhou J (2006) An overlay subscription network for live Internet TV broadcast. *TKDE* 18 (12):1711–1720
5. Castro M, Druschel P, Kermarrec A-M, Nandi A, Rowstron A, Singh A (2003) SplitStream: high-bandwidth multicast in cooperative environments. Proceedings of the nineteenth ACM symposium on Operating systems principles, ACM New York, NY, USA, pp 298–313
6. Cheng B, Stein L, Jin H, Liao X, Zhang Z (2008) GridCast: improving peer sharing for P2P VoD. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, vol. 4, Issue 4
7. Cheng B, Stein L, Jin H, Zhang Z (2008) Towards cinematic internet video-on-demand. Proceedings of the 3rd ACM SI-GOPS/EuroSys European Conference on Computer Systems 2008. ACM New York, NY, USA, pp 109–122
8. Cheng B, Stein L, Jin H, Zhang Z (2008) A framework for lazy replication in P2P VoD. Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video. ACM New York, NY, USA, pp 93–98
9. Chu YH, Rao SG, Seshan S, Zhang H (2002) A case for end system multicast. *IEEE J Sel Areas Commun* 20(8):1456–1471
10. Dana C, Li D, Harrison D, Chuah C (2005) Bass: Bittorrent assisted streaming system for video-on-demand. Processing of 2005 IEEE 7th Workshop on Multimedia Signal. IEEE, Piscataway, USA, pp 1–4
11. Do TT, Hua KA, Tantaoui MA (2008) Robust video-on-demand streaming in peer-to-peer environments. *Comput Commun* 31(3):506–519
12. Guo Y, Suh K, Kurose J, Towsley D (2003) P2Cast: peer to peer patching scheme for VOD services. Proceedings of the 12th international conference on World Wide Web, ACM New York, NY, USA, pp 301–309
13. Guo Y, Suh K, Kurose J, Towsley D (2008) DirectStream: a directory-based peer-to-peer video streaming service. *Comput Commun* 31(3):520–536
14. <http://ffmpeg.org/>, May 2009
15. <http://nsl.csie.nctu.edu.tw/nctuns.html>, May 2009
16. <http://www.bittorrent.com/>, May 2009
17. <http://www.emule.org/>, May 2009
18. <http://www.netflix.com/Default/>, May 2009
19. <http://www.peercast.org/>, May 2009
20. <http://www.ppstream.com/>, May 2009
21. <http://www.youtube.com/>, May 2009
22. Huang Y, Fu TZJ, Chi DM, Lui JCS, Huang C (2008) Challenges, design and analysis of a large-scale p2p-vod system. *ACM SIGCOMM Computer Communication Review* 38(4):375–388
23. Iqbal R, Shirmohammadi S (2009) Dag-stream: distributed video adaptation for overlay streaming to heterogeneous devices. *Peer-to-Peer Networking and Applications*. URL <http://dx.doi.org/10.1007/s12083-009-0031-0>
24. Jannotti J, Gifford DK, Johnson KL, Kaashoek MF, O’Toole Jr JW (2000) Overcast: reliable multicasting with an overlay network. Proceedings of the 4th conference on Symposium on Operating System Design & Implementation - Vol. 4. USENIX Association Berkeley, CA, USA, pp 14–14
25. Jiang X, Dong Y, Xu D, Bhargava B (2003) GnuStream: a P2P media streaming system prototype. ICME apos:03. Proceedings 2003 International Conference on Multimedia and Expo. IEEE, Piscataway, USA, vol. 2, Issue 6–9, pp II - 325–8
26. Jin X, Cheng K-L, Gary Chan S-H (2007) Scalable island multicast for peer-to-peer streaming. *Advances in Multimedia* 2007(1):10
27. Jurca D, Chakareski J, Wagner J-P, Frossard P (2007) Enabling adaptive video streaming in P2P systems. *IEEE Commun Mag* 45(6):108–114
28. Kim H, Yeom HY (2008) P-chaining: a practical VoD service scheme autonomically handling interactive operations. *Multimedia Tools and Applications* 39(1):117–142
29. Kostic D, Rodriguez A, Albrecht J, Vahdat A (2003) Bullet: high bandwidth data dissemination using an overlay mesh. *ACM SIGOPS Operating Systems Review*, ACM New York, NY, USA, vol. 37, Issue 5, pp 282–297
30. Lavrentiev M (2009) Transrating of coded video signals via optimized requantization. Master Thesis, http://sipl.technion.ac.il/new/Research/Publications/Graduates/Michael_Lavrentiev/Michael_Lavrentiev_Thesis_final.pdf
31. Liao X, Jin H, Liu Y, Ni LM (2007) Scalable live streaming service based on interoverlay optimization. *IEEE Trans Parallel Distrib Syst* 18(12):1663–1674
32. Liao X, Jin H, Liu Y, Ni LM (2007) Scalable live streaming service based on interoverlay optimization. *IEEE Trans Parallel Distrib Syst* 18(12):1663–1674

33. Liu Y, Guo Y, Liang C (2008) A survey on peer-to-peer video streaming systems. *Peer-to-Peer Networking and Applications*. Springer, New York, vol. 1, no. 1, pp 18–28
34. Liu JC, Rao SG, Li B, Zhang H (2008) Opportunities and challenges of peer-to-peer Internet video broadcast. *Proceedings of the IEEE*. IEEE, Piscataway, USA, vol. 96, no. 1, pp 11–24
35. Magharei N, Rejaie R (2007) PRIME: Peer-to-Peer Receiver-driven Mesh-based streaming. *INFOCOM 2007, 26th IEEE International Conference on Computer Communications* 1415–1423. IEEE, Piscataway, USA, pp 1415–1423
36. Magharei N, Rejaie R, Guo Y (2007) Mesh or multiple-tree: a comparative study of live P2P streaming approaches. *IN-FOCOM 2007, 26th IEEE International Conference on Computer Communications*. IEEE, Piscataway, USA, pp 1424–1432
37. Mol JJD, Pouwelse JA, Meulpolder M, Epema DHJ, Sips HJ (2008) Give-to-Get: free-riding resilient video-on-demand in P2P systems. *Multimedia Computing and Networking 2008, Proceedings of the SPIE*. Bellingham WA USA, vol. 6818, pp 681804 (2008)
38. Pai V, Kumar K, Tamilmani K, Sambamurthy V, Mohr A (2005) Chainsaw: eliminating trees from overlay multicast. In: *Peer-to-Peer Systems IV*. Springer Berlin / Heidelberg, vol. 3640/2005, pp 124–140
39. Pianese F, Perino D, Keller J, Biersack EW (2007) PULSE: an adaptive, incentive-based, unstructured P2P live streaming system. *IEEE Trans Multimedia* 9(8):1645–1660
40. Tian Y, Wu D, Ng K-W (2008) A novel caching mechanism for peer-to-peer based media-on-demand streaming. *Journal of Systems Architecture: the EUROMICRO Journal* 54(1–2):55–69
41. Tran DA, Hua KA, Do T (2003) ZIGZAG: an efficient peer-to-peer scheme for media streaming. *INFOCOM 2003, Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*. IEEE, Piscataway, USA, Vol. 2, pp 1283–1292
42. Vlavianos A, Iliofotou M, Faloutsos M (2006) Bitos: enhancing bittorrent for supporting streaming applications. *INFOCOM 2006, 25th IEEE International Conference on Computer Communications, Proceedings IEEE*. IEEE, Piscataway, USA, pp 1–6
43. Wang M, Li B (2007) R2: random push with random network coding in live peer-to-peer streaming. *IEEE J Sel Areas Commun* 25(9):1655–1666
44. Yves DONY (2008) Video-on-Demand over Internet: a survey of existing systems and solutions. Student Thesis, http://www.fundp.ac.be/recherche/publications/page_view/66101/
45. Zhang X, Liu J, Li B, Yum T-SP (2005) DONet/CoolStreaming: a data-driven overlay network for live media streaming. *IN-FOCOM 2005, 24th Annual Joint Conference of the IEEE Computer and Communications Societies, Proceedings IEEE*. IEEE, Piscataway, USA, vol. 3, pp 2102–2111



Wang Wenyi received the B.S degree in electrical engineering in 1999 and M.S degree in electrical engineering in 2002, both from Zhejiang University, Hangzhou, China. He is currently a Ph. D. candidate in the Advanced Digital Technology and Instruments Institute of the same university. Since 2002, he is working as a R&D manager for Alpha Networks Inc. digital home product center. His current interests include multimedia broadcasting and distribution over large scale network and embedded system design for multimedia devices.



Chen Yaowu was born in Liaoning Province, in China, in 1963. He received the Ph.D. degree from Zhejiang University, Hangzhou, China, in 1998. He is currently a professor and the director of the Institute of Advanced Digital Technologies and Instrumentation, Zhejiang University. His major research fields are embedded system, networking multimedia system, and electronic instrumentation system.