LayeredCast - A Hybrid Peer-to-Peer Live Layered Video Streaming Protocol

Masoud Moshref, Reza Motamedi
AICTC Research Center
Department of Computer Engineering
Sharif University of Technology
Email: moshref,motamedi@ce.sharif.edu

Hamid R. Rabiee
AICTC Research Center
Department of Computer Engineering
Sharif University of Technology
Email: rabiee@sharif.edu

Mohammad Khansari
Information Technology Faculty
Education and Research Institute for ICT
Email: khansari@itrc.ac.ir

***Abstract*—Peer-to-Peer overlay networks are an attractive foundation for video streaming. However, live Peer-to-Peer media streaming systems face many challenges such as bandwidth heterogeneity, node churn, and selfish nodes. Although many tree based and mesh based streaming protocols have been proposed, each has its own drawbacks such as unreliability and unfairness in tree based and long startup delay and complex scheduling in mesh based protocols.**

In this paper, we propose a new video streaming protocol called LayeredCast main features of which are: 1) Hybrid: Drawbacks of the simple approaches are compensated using a hybrid of mesh and tree overlays. 2) Layered Video: Provides an adaptive scheme to enhance the video quality using a layered video codec for heterogeneous clients. 3) QoS: LayeredCast scheduling aims at moving complexity of Multi-Service network core to the network clients application layer, thus providing better QoS over simple regular networks. LayeredCast's tree network pushes the base layer to all peers while the enhancement layers and missing base layer segments are pulled over a mesh network by peers with extra bandwidth using a new data-driven scheduling scheme. We have evaluated the performance of LayeredCast on an innovative simulation framework. Simulation results verify better performance of LayeredCast in term of decodable video frames over CoolStreaming, especially when network resources are limited.

I. Introduction

The accessibility of broadband Internet to home users has pushed video broadcasting applications up in the list of the most favorite IP-based applications. Such applications involve delivering the data generated in a single sender to a set of users which are scattered around the world. Since their advent, Peer-to-Peer applications have mitigated many problems previously unsolved in the field of communication. File sharing, distributed file systems, distributed databases, and many other applications have used the idea of resource sharing behind Peer-to-Peer systems. In this architecture, end-systems receive the stream content form a self-organized efficient overlay over the IP network instead of joining an IP Multicast [1] session. On this overlay, media content is distributed, and each end-system plays the role of a relay that retransmits the contents to some other participating peers in the network.

Different Peer-to-Peer streaming systems differ in their topology control and scheduling mechanism, thus topology and scheduling can be used to classify these systems into tree based, mesh based and hybrid systems. Many protocols form a tree structured overlay and push data along the branches to the leaves. Tree based structure inspired by IP Multicast is the most simple structure used to provide application layer multicast service. On the other hand, some other protocols use a data driven approach and form a mesh with loosely bounded links. In this scheme, data forwarding is dynamically determined according to data availability in the peers. Recently, a new class of Peer-to-Peer streaming protocols which uses hybrid overlay structures has emerged.

Our proposed protocol, LayeredCast, uses layered video in order to distribute video with adaptive quality. Adaptive video enables the system to serve clients with heterogeneous bandwidths. In the layered coding, a video bitstream is partitioned into layers so that the base layer is decodable by its own, while the upper ones are only decodable if the lower layers are decodable. If the base layer were well protected, a minimum picture quality would be guaranteed. Therefore, although LayeredCast regularly transmits base layer over a tree overlay and enhancement layers on a mesh, it compensates base layer loses in mesh. LayeredCast uses the hybrid topology in order to have low delay advantage of tree topologies and reliability and fairness of mesh ones. Besides the robust adoptable mechanism for streaming layered video, we proposed a new heuristic for NP-Hard problem [2] of scheduling layered video over mesh with the aim of maximizing video play back quality measured via number of decodable frames.

The rest of the paper is organized as follows: The next section will present some related hybrid protocols. We will introduce our protocol in four subsections under the Section III. In Section IV, simulation results will be discussed and at last, Section V discusses on future work we planned to do on LayeredCast.

II. Related Work

Previous protocols have used hybrid structures in different schemes but these schemes can be categorized in the following classes: 1) Some protocols use hybrid structures to separate data and control traffic. 2) Another scheme is to use one overlay as a compensatory structure. Thus content delivery in the reserve overlay is invoked in case the primary overlay fails to deliver media content. 3) In final class, media is distributed over both overlays. The first overlay is used to deliver part of video contents, while the second overlay is used to consume extra network resources to improve video quality. AnySee2 [3], for instance, is using a hybrid topology and divides the transmission of control messages and media data into two different overlays, one with tree topology which is used to transfer control messages, and another one with mesh structure which is used to deliver data messages to end systems. AnySee2's scheduler on the other hand is very similar to the one in CoolStreaming [4], but some new ideas are introduced like zonal buffer request. Although Anysee2 uses a hybrid topology, the previously defined roles of each overlay structure, prevents the topology from benefiting of both their advantages. In mTreeBone [5], the main idea is to identify a set of stable nodes to construct a tree based backbone with most of the data being pushed over the backbone tree. The stable nodes along with other participating peers are organized through an auxiliary mesh overlay (compensatory role) which facilitates the tree bone to accommodate node dynamics and fully exploit the available bandwidth between overlay nodes. The system's main drawback arises when an unlimited video is being streamed, e.g. a TV channel. In this case, the threshold cannot be calculated optimally. In this scheme, the mesh topology is only used when the tree fails to deliver video data, thus the protocol is suitable for high dynamic circumstances. Bullet [6] uses both the tree and the mesh structures to send video data packets. It distributes video contents as much as possible through the tree using the push mechanism and exploits the unused uplink bandwidth in a mesh structure by pull requests. Bullet developers also used tree structure to distribute video segment advertisements. Advertisement packets collected in the root node will be distributed downward in the tree structure in each Epoch. LayeredCast actually benefits from advantages of the last two categories. Although both overlays are used to deliver media content, the mesh structure plays a compensatory role when the tree fails to deliver the base layer. Since enhancement layers can be only decoded when the base layer is available in a peer, layered video priorities suggest that mesh overlay should be used to deliver base layer; otherwise, the consumed bandwidth would be in vain.

Another set of related work are those which have taken a more theoretical approach. In [7], [8], it is assumed that all peers in the network are partners. This full mesh structure enables each peer to provide media for all other peers in the networks. In [7] a centralized optimization problem is solved to optimally divide each peer's bandwidth among all other peers in the network and a central greedy scheduling algorithm is used to determine each peer's schedules. Although the successive water-filling algorithm used in [8] is not centralized, but its sequential nature imposes a large delay over the algorithm performance. Both schemes could not be used in real P2P networks since their premises are totally different to those of large P2P networks. In contrary, the PWF heuristic algorithm proposed in LayeredCast uses the same rationale in [7], [8] to discriminate between its neighbors based on the content reserve level, but in a totally decentralized and applicable fashion.

III. LayeredCast

A. Peer Architecture

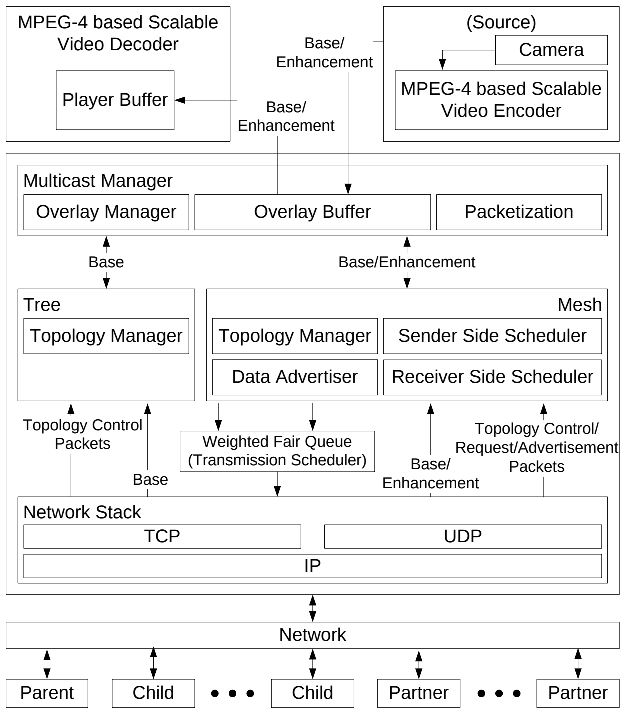


Fig. 1. The node architecture of LayeredCast

A LayeredCast node architecture is depicted in Figure 1. As Peer-to-Peer nature of the system suggests, all nodes have the same structure, except for the *source node* which is capable of capturing video. In the top level of this layered design, video camera and video player are responsible for encoding and decoding video data. In the lower layer, *Multicast Manager* contains *Overlay Buffer*, *Overlay Manager*, and *Packetization* modules. Overlay Buffer stores video segments which can be feed to other peers in the network. The Overlay Manager synchronizes the two Overlay Topology Managers below and the Packetization module packeteizes video segments to be transmitted over the network. Mesh and Tree Overlay Managers reside in the lower level. While the *Tree Overlay Manager* has a simple structure, the *Mesh Overlay Manager* has more complex structure and comprises *Data Advertiser*, *Receiver Side* and *Sender Side Scheduler* in addition to *Topology Manager* module. Data Advertiser advertises segments which are stored in the Overlay Buffer and the latter two modules are described in detail in the following sections. *Transmission Scheduler* resides in the lowest level of this design and controls packet transmission sequence.

B. Topology Control

As LayeredCast uses both a tree and a mesh topology, we need to design and construct both topologies. We tried to decrease the coupling of these protocols and manage them only on separated layers interacting with predefined interface to the overlay protocols to be able to reuse the architecture with different protocols.

1) Tree: Every overlay protocol with tree topology should specify its solutions for four problems: locating potential parents, selecting the best parent, refining the structure, and avoiding a loop construction. Each node in the tree uses a centralized node to get a list of limited number of potential parents in LayeredCast. This list is sorted by node's reliability and depth. Upon receiving the list, the node can select the best parent by calculating and comparing the RTT for each node.

The new node sends a JOIN-REQ message to the selected parent node. If the parent's available bandwidth equals or exceeds the amount needed for data transmission in tree, parent responds with JOIN-RSP and reserves the required bandwidth; otherwise, it sends a JOIN-DENY message. Then the accepted child sends JOIN-ACK if it does not need to improve its position. As we used the tree structure to multicast only base layer blocks, it does not need to be refined seriously. However, its refinement based on the reliability measures would be appropriate in presence of node churn. In order to avoid a loop construction, each node sends a detection packet to its children containing its signature on changing its parent. A node receiving the packet checks if it contains the node's signature and disconnects from the parent if detects a loop; otherwise, adds its signature and passes the packet to its children. The detection packet method has less overhead than putting signature on data packets. However, it increases the complexity of protocol.

2) Mesh: Mesh overlay network is used along with the tree network to pull enhancement video layers and to retransmit missing base layer segments. In order to build a mesh structure, we need to determine a solution for each of the following problems: locating potential neighbors, selecting neighbors, and refining mesh structure. To join the mesh, a new node asks a centralized bootstrap server for a depth-based sorted list of nodes already joined to the network. However, after joining the network, nodes can find out about more peers via neighbor list which is embedded in JOIN-RSP, JOIN-ACK, and advertisement messages they receive from their immediate neighbors.

Each node tries to find a distinct provider for each required video layer and accepts a join request if it has enough free bandwidth and video block in the required window of interest. Although the join mechanism in mesh is similar to the mechanism in tree, we have noticed that we can use the information that nodes got about their neighbors in advertisement packets to decrease the join delay time. The central server knows the number of required layers for each node; however, it could not tell which part of video they are playing or where is the window of interest in each node. Therefore, a suggestion mechanism is designed in which a list of potential neighbors is provided in response to a denied join request. This knowledge could be very helpful when the churn rate is high as the regular waterfall order in nodes' buffer breaks in the mesh.

Mesh refinement in LayeredCast runs to allay content bottleneck [9]. Content bottleneck occurs when a node could not find a needed block in its neighbors. Data clustering on overlay network clusters, poor network construction method, or node churn are some reasons behind content bottleneck. If a content bottleneck occurs, scheduler finds out that it cannot request wanted blocks from the layer provider(s) and notifies the overlay controller to drop the neighbor and to find a new one.

C. Mesh Scheduling

Using a data driven scheme for the mesh overlay, each node's scheduler uses information about video segments available in the node itself and in its neighbors to determine segments to be requested from each neighbor and to distribute outgoing bandwidth among them. It worth mentioning that the size of video segments is fixed and each frame contains video data from one GoP (Group of Picture) of a single enhancement layer. Scheduling process is invoked periodically every Δ seconds, namely every *Epoch* and includes following activities.

1) Sender Side Scheduling: In each peer, the sender scheduler module distributes outgoing bandwidth among the peer's neighbors for the next Epoch every time the scheduler is invoked. A Pseudo Water-Filling algorithm is devised for this purpose. The rationale behind this algorithm is that the optimal case in a fully dynamic P2P Streaming happens when all peers have the same amount of data in their buffer [7].

As stated earlier, every peer in the system finds a neighbor to play the role of layer provider for each enhancement layer it is willing to fetch from the mesh overlay, thus a contract is made among two peers. Sender scheduler uses a token allocation scheme to distribute outgoing bandwidth among neighbors to which it has agreed to provide enhancement layers. Total number of tokens is equivalent to the number of segments the peer can provide during an Epoch.

In order to avoid a bandwidth bottleneck, the scheduler initially allocates the minimum bandwidth declared in the contract, i.e. if Δ equals play back time of $four$ segment and a neighbor contract includes first and third enhancement layers, $eight$ tokens will be allocated to that neighbor in the first step. In the second step, content reserve level (i.e. segment received but not played) and possible segment demand (i.e. segment available in peers buffer but not received in the neighbor) of all neighbors are evaluated, and the remaining tokens are distributed in a water-filling fashion, with the goal of equalizing the content reserve level of all neighbors.

Number of tokens allocated to each neighbor is piggybacked on segment advertisement messages, thus each neighbor is informed of the number of segments it can request from the peer.

2) Receiver Side Scheduling: A windowing Mechanism in addition to a utility based scheme is used to decide which segment to request from which neighbor. As Figure 2 depicts, *Window of Interest* (WoI) comprises three sub-windows, namely urgent, relaxed and aggressive sub-windows. The height of the WoI is set proportional to the peer's incoming bandwidth, thus different nodes could enjoy different types of service in accordance to their bandwidth. Aggressive sub-window covers base layer segments and its width is set in a way each base layer segment would be covered by this sub-window just once. Urgent window covers enhancement layer segments which have a single chance to be requested, i.e. WoI would not cover these segment the next time scheduler is invoked and other interesting segments are covered by relaxed sub-window. Every time the scheduler invoked, WoI is pushed forward Δ seconds to maintain fixed distance with play back pointer and cover fresh segments.

In the first phase, receiver side scheduler initially checks the base layer buffer. If a missing segment is found in aggressive window (this may happen due to delay loss or parent departure in tree), the scheduler tries to fetch base layer segments using its mesh allocated bandwidth. A simple *Round Robin* among neighbors is used for this purpose and selected neighbors' tokens are consumed for each requested base layer segment. In

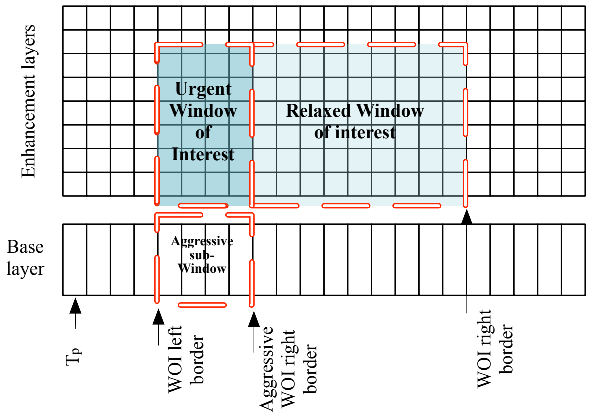


Fig. 2. Window of interest (WoI) and its sub-windows

the second phase, the receiver scheduler aims at maximizing value of requested segments given available segments in neighbors and the peer itself and the value of each segment. A utility function is designed to value video segments. Value of each segment is proportional to the layer number and the number of remaining request chance of the segment. As the utility function depicted in Figure 3 shows, value of the most valuable segment with $n+1$ remaining request chance is lower than value of the least valuable segment with n remaining request chance. Thus, an optimization problem of Eq. 1 should be solved in which w_s is segment s's value, N is set of all neighbors, S is set of all segments in the WoI and S_n is set of all segment in the WoI available in neighbor n's buffer. BW_{mesh_n} is neighbor n's allocated bandwidth and Δ is Epoch duration. y_{sn} is equal to one when the segment s is available at neighbor n's buffer and variable x_{sn} would equal to one if segment s is going to be requested from neighbor n.

In order to solve the optimization problem of Eq. 1, given the neighbors' and the peer's buffer content and the segments' values, in addition to the number of tokens remaining from phase one, the receiver side scheduler forms a bipartite weighted graph. Nodes in the first party represent enhancement

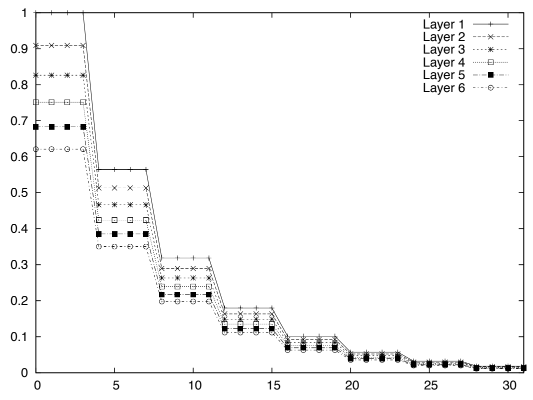


Fig. 3. The utility function

layer segments covered by WoI which are not yet received in the peer. Second party's nodes represent peer's neighbors. If a segment s which is covered by the WoI is available at neighbor n and has not been requested before, a link with weight $value(s)$ would connect node s from the first party to node n from the second party. If the segment s from layer l_i is covered by relaxed sub-window, the weight of its connecting links would be dropped to the minimum value unless the node n_j had accepted to be the provider for that specific layer in their contract. In fact, the peer delays requesting segments beyond contracts to the time the segments happen to reside in the urgent WoI or when a neighbor allocates extra tokens.

$$max \sum_{n \in N} \sum_{s \in S} w_s \times x_{sn} \tag{1}$$

$$subject\ \ to \begin{cases} \forall n \in N \sum_{s \in S_n} \frac{x_{sn} \times sizeof(s)}{\Delta} \leq BW_{mesh_n} \\ \forall s \in S \sum_{n \in N} x_{sn} \leq 1 \end{cases}$$

$$variables\ \ x_{sn} \in \{0, y_{sn}\}$$

The node representing peer n_x is then duplicated T_x times where T_x is the number of tokens neighbor n_x has allocated to the peer, thus the maximum matching of the constructed bipartite graph is the solution to the maximization problem of Eq. 1. The *Hungarian Method* can be used to solve this maximum matching problem with the complexity of $O(n^3)$ where n is the maximum of the number of segments and the total number of tokens assigned to the peer by neighbors.

D. Transmission Scheduling

The transmission scheduler comprises leaky buckets and a combination of priority and weighted queues. A leaky bucket for each child in the tree overlay, shapes tree traffic to avoid link congestion in the receiver side. Besides, a queue is formed for each neighbor or child of the peer in addition to a single high priority queue for control messages to address congestion in the sender side. A priority scheduling scheme is used to prioritize control message but a weighted fair queuing mechanism is used to select next transmitted packet when the control queue is empty. The weight values are set in each

Epoch in accordance to the number of tokens assigned to each neighbor. Moreover, the scheduler attempts to drop video packets of removed neighbors and the GoPs deadlines of which are passed. To address congestion in access link of senders, the available bandwidth is divided between mesh and tree structure according to the base layer and enhanced layers bit rate.

IV. Simulation Framework & Simulation Results

Our simulation framework has three phases (Figure 4): In the first phase, we have generated our underlay topology using the two levels top-down model in the Brite topology generator [10] with 15 AS and 10 routers in each AS using the BA model in each level. The bandwidth of the core network is assumed high enough to let the congestion occur only in the access links. Then each scenario has been run ten times to locate 50 nodes in different parts of the network with a Weibull [11] random lifetime. We used OMNet++ simulator [12] as our simulation foundation. Besides, INET and OverSim [13] packages are used to simulate network protocols and configure the P2P overlay network. Next, we implemented overlay protocols and multimedia applications on top of them. In the third phase, we used Graphviz library and Pajek software to visualize the overlay graph. We have used the FGS codec traces [14] containing a 800kbps base layer and four 200kbps enhancement layers. There are three bandwidth schemes in our simulation results: Tight in which half of the nodes could receive one and others two enhancement layers; in Generous, the second half can receive all layers. Finally, all nodes in High scheme download all layers. The resource index [15] of the schemes is 1.07, 2.26, and 2.4 respectively. Quality of a played frame has been measured by the number of its layers and the *Quality Percentage* measure is the ratio of quality of played frames to the required quality per each node. The join delay is computed by the average time a layer has been without provider and the computation complexity is the CPU ticks used by our implementation obtained by a profiler tool. Figure 5(a) shows the effect of mesh improvement on the average quality percentage over the simulation time in which more quality improvement is gained for high churn rates and tighter bandwidth schemes. Base layer compensation is one of the mesh uses effect of which is depicted in Figure 5(b). The figure shows that base layer preservation in the mesh network helped us to improve video quality up to 10 percent without losing the real-time nature of tree structure. Besides, we tested our protocol with 16, 8, 4 seconds for initial buffer length and it could keep the video quality upper than 95 percent for 16 and 8 seconds. Hybrid structure, neighbor suggestion, bootstrap list creation algorithm, and mesh improvement algorithm are reasons behind small initial buffer requirement. We have assumed that when a node leaves the network gracefully, it could inform its neighbors, parent, and children one second before. However, the protocol could resist ungraceful leaves using precise timeout mechanisms in nodes' interactions as depicted in Figure 5(c).

Using neighbor suggestion in mesh topology earned us 50 percent improvement in mesh join delay, but the interesting ob-

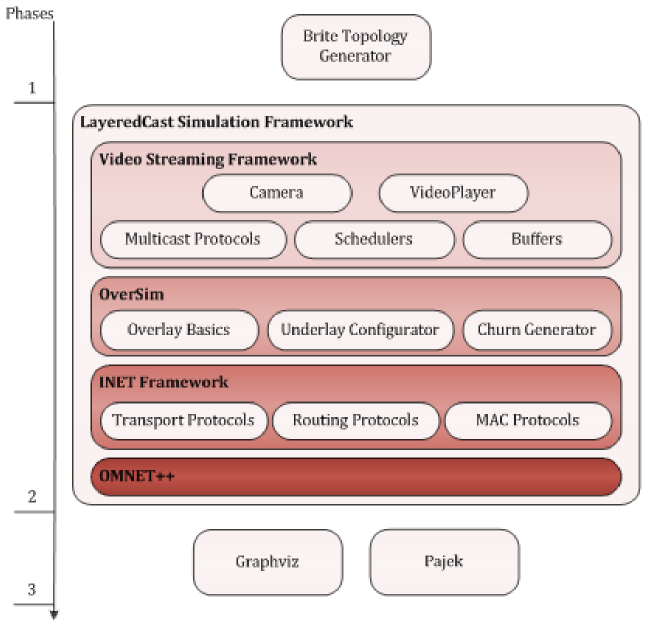
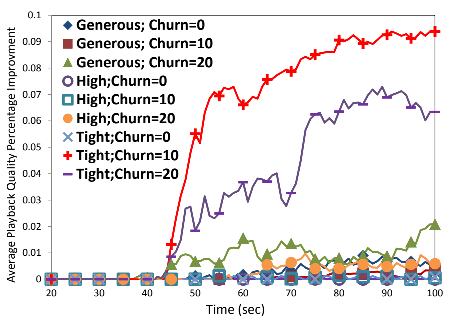


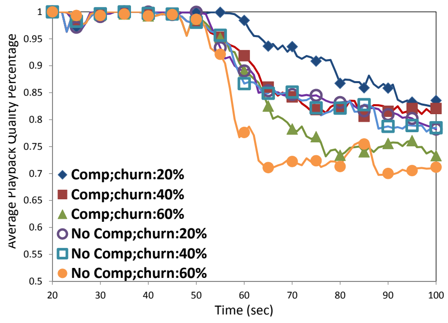
Fig. 4. A logical view of the simulation framework components

servation is that the number of neighbor suggestions increases with node degrees which has correlation with the number of layers and nodes' bandwidth. Besides, nodes suggestion mechanism would be more helpful in unstable networks. Figure 5(d) presents a comparison between applying the proposed pseudo water-filling algorithm and a normal scheme in which all neighbors receive equal tokens. This result verifies that LayeredCast token assignment algorithm improves play back quality by considering status of lagged neighbors. On the other hand, as we have expected, we noticed that by increasing the segment size and decreasing the length of WoI in the scheduler modules, the play back quality and the computation overhead decreased.

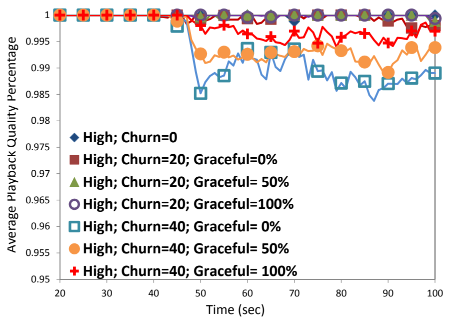
Lastly, we have compared the performance of our mesh protocol with CoolStreaming protocol, which is the base of most of the proposed mesh protocols today. Figure 5(e) and 5(f) show that quality percentage in CoolStreaming protocol is lower than our protocol, especially when the churn rate is high. The effect of churn is obvious in second 26 when nodes start to leave the network. We believe that being aware of layer dependencies and using a bandwidth allocation, a data-aware neighbor selection, and a mesh refinement scheme along with the hybrid structure in LayeredCast protocol are its winning factors over CoolStreaming. However, it should be noticed that the computation overhead of LayeredCast protocol is about 30 times of CoolStreaming protocol, which means that LayeredCast could not be used in devices with limited computation power. Moreover, the play back quality in Tight bandwidth scheme turned down (Figure 5(f)) because content bottleneck occurs when provider nodes leave the network. Provider reservation and multi-provider for each layer could alleviate this situation and are a part of the future work.



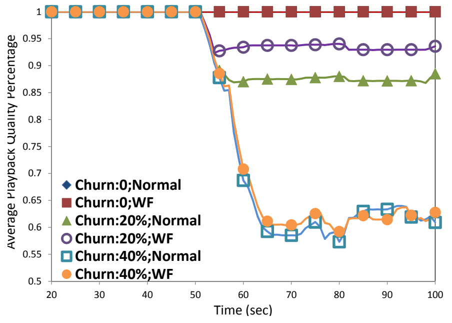
(a) Mesh improvement



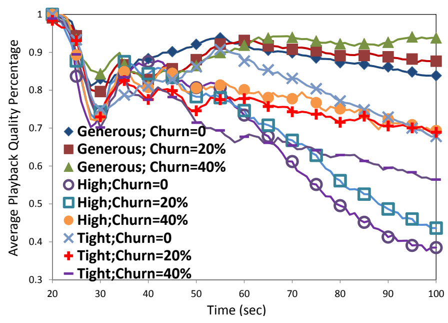
(b) Base-layer compensation



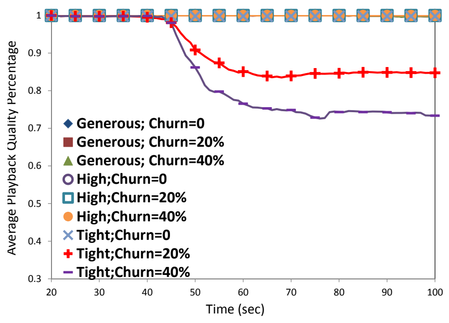
(c) Robustness against ungraceful churns



(d) Waterfilling against normal method



(e) CoolStreaming performance



(f) LayeredCast performance

Fig. 5. The Average Playback Quality Percentage in each scenario

V. Conclusions & Future Work

We have presented LayeredCast, a hybrid Peer-to-Peer live layered video streaming protocol, which has the advantages of both tree and mesh protocols. LayeredCast uses layered video to serve clients having various downlink bandwidths with different video quality. Moreover, an innovative scheduler is introduced to manage mesh and tree collaboration in order to minimize frame loss and avoid congestion in the network. LayeredCast also tries to create more reliable overlay networks and avoid content and bandwidth bottlenecks using refinement algorithms and bandwidth reservation. Our simulation results approve our claims on performance of our hybrid protocol, LayeredCast, against an available mesh protocol.

As future work, we aim at using incentive mechanisms in LayeredCast design, handling error prone networks with the help of error resilient solutions, and considering the background traffic in the congestion avoidance mechanism. Besides, we planned to implement lateral applications of the universal tree network such as broadcasting PSNR of video frames to be used in scheduling, browsing channels, and distributing incentive information.

References

[1] S. E. Deering, "Multicast routing in internetworks and extended LANs," *SIGCOMM Computer Communication Review*, vol. 25, no. 1, pp. 88–101, 1995.

[2] H. Guo, K. Lo, and C. Cheng, "Overlay networks construction for multilayered live media streaming," in *Proceedings of the Eighth IEEE International Symposium on Multimedia*, 2006, pp. 427–436.

[3] Q. Huang, H. Jin, and X. Liao, "P2P live streaming with Tree-Mesh based hybrid overlay," in *Proceedings of the International Conference on Parallel Processing Workshops*, 2007, p. 55.

[4] X. Zhang, J. Liu, B. Li, and T. Yum, "DONet/CoolStreaming: a data-driven overlay network for live media streaming," in *Proceedings of IEEE INFOCOM*, vol. 3, 2005, pp. 2102–2111.

[5] F. Wang, Y. Xiong, and J. Liu, "mTreebone: a hybrid Tree/Mesh overlay for Application-Layer live video multicast," in *Proceedings of ICDCS*, 2007, p. 49.

[6] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: high bandwidth data dissemination using an overlay mesh," in *Proceedings of ACM symposium on operating systems principles*. ACM, 2003, pp. 282–297.

[7] Z. Li, J. Huang, and A. K. Katsaggelos, "Content reserve utility based video segment transmission scheduling for Peer-to-Peer live video streaming system," in *Proceedings of Allerton Conference on communication, control and computing*, 2007.

[8] Y. Li, Z. Li, M. Chiang, and A. Calderbank, "Video transmission scheduling for peer-to-peer live streaming systems," in *IEEE International Conference on Multimedia and Expo*, 2008, pp. 656, 653.

[9] N. Magharei, R. Rejaie, and Y. Guo, "Mesh or multiple-tree: A comparative study of live p2p streaming approaches," in *Proceedings of IEEE INFOCOM*, 2007, pp. 1424–1432.

[10] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: an approach to universal topology generation," in *Proceedings of MASCOTS*, vol. 1, 2001.

[11] A. Papoulis and S. U. Pillai, *Probability, random variables and stochastic processes*. McGraw-Hill Education (India) Pvt Ltd, 2002.

[12] A. Varga, "The OMNeT++ discrete event simulation system," in *Proceedings of the European Simulation Multiconference*, 2001, p. 319324.

[13] I. Baumgart, B. Heep, and S. Krause, "OverSim: a flexible overlay network simulation framework," in *IEEE Global Internet Symposium*, 2007, pp. 79–84.

[14] P. Seeling, P. D. Cuetos, and M. Reisslein, "Fine granularity scalable video: implications for streaming and a trace-based evaluation methodology," *IEEE Communications Magazine*, vol. 43, no. 4, pp. 138–142, 2005.

[15] Y. hua Chu, A. Ganjam, T. S. E. Ng, S. G. Rao, K. Sripanidkulchai, J. Zhan, and H. Zhang, "Early experience with an internet broadcast system based on overlay multicast," in *Proceedings of USENIX*, 2004, p. 12.