PALMS : Reliable P2P Live Media Streaming

Poo Kuan Hoong, Hiroshi Matsuo *†

Abstract—**In recent year, Peer-to-Peer (P2P) approach for media streaming has been studied extensively. In comparison to on-demand media streaming, P2P live media streaming faces a much stringent time constraint. In order to improve the performance metrics, such as startup delay, source-to-end delay, and playback continuity, we present PALMS, a P2P approach for live media streaming where node employs gossip based pull and push protocols to receive and forward media data among connected nodes. We present a simple heuristic mechanism for the pull protocol in the selection of media segments and peers. Besides the pull method, a push method is deployed to increase the streaming quality. We also adopt a randomized push protocol in order to increase the probability of media data delivered to connected nodes. We know that the presence of free-riders could degrade the delivered streaming quality. In PALMS, a simple tit-for-tat incentive mechanism is adopted to discourage the existence of free-riders. We conducted simulations and performance comparisons for PALMS. Experimental results demonstrate that PALMS can deliver better streaming quality and more resilience towards the heterogeneity of network bandwidths as compared to some of the existing protocols.**

Keywords: peer-to-peer, live media streaming, overlay, gossip.

1 Introduction

In recent years, due to the popularity of file sharing phenomenon that leverages on the power of the normal connected computers, peer-to-peer (P2P) network has been the center of significant research. P2P approach for media streaming has been studied extensively too in recent years by some of the following authors [5],[6],[8],[10],[13]. For P2P streaming, the basic building block behind P2P streaming system is the nodes. Nodes can act both as clients and servers at the same time. Nodes form an overlay and simultaneously forward and receive media stream data to and from other connected nodes.

The focus of our paper is mainly on large scale live streaming under the P2P unstructured overlay network. In live P2P streaming, the media stream is a continuous flow of media data encoded from the streaming server. Media content generated must be delivered to participating nodes under a tight time constraint. Nodes should be able to receive media content before the playout deadline or the media content will be discarded. Inspired by the file swarming mechanism such as *BitTorrent*[3], participating nodes will swarm around newly generated media content. Newly generated media content is retrieved by receiver nodes and in turn actively contribute the media contents to other connected nodes. One good example of such application is the distribution of television programs from a single source broadcaster to a large number of dynamic and heterogeneous clients/end-users over the Internet. In this context, we present our approach to large scale P2P streaming for live and non-interactive media streaming, called PALMS or (acronym for *P2P Unstructured Live Media Streaming*). PALMS is based on data-driven and receiver-based overlay media streaming. It is designed to operate in conditions where nodes have heterogenous bandwidths and resources. Moreover, with PALMS, we also address the issue of free-rider nodes. Similar to the 'tit-for-tat' incentive policy used by BitTorrent, we introduce a simple heuristic reward incentive to discourage the existence of free-riders in PALMS.

Although our approach is much similar to BitTorrent, the requirements for live streaming application differ from normal P2P file sharing application. Live streaming deals with the transmission of practically infinite-sized media streams, which are to be played by the nodes upon reception before the playout deadline. Requirements for streaming to end users for P2P live media streaming applications include (a) locate peers with the desired media segments before the playout time deadline, (b) choose peers that are likely to provide good performance for playback, and (c) manage parallel download and upload to connected neighbor nodes (d) managing connected peers in the network due to the dynamicity of peers joining and leaving and (e) to discourage the existence of selfish peers that choose only to download media contents without uploading.

The rest of the paper is organized as follows : In Section II, we discuss the overview system overview of PALMS, that comprises of two main protocols: (i) pull protocol and (ii) push protocol. In section III, the main algorithms for PALMS is explained. We describe the details of the

*Nagoya Institute of Technology, Nagoya-shi, Showa-ku, Gokiso-cho, Aichi, JAPAN, 466-8555. Email:{khpoo@matlab., matsuo@}nitech.ac.jp

†This work was supported by the by the Ministry of Education, Culture, Sports, Science and Technology, Government of Japan, and also by a grant from the Hori Foundation Science Promotion Foundation, Japan.

simulation setting in Section IV. Section V provides the result of performance evaluation based on simulation in various conditions. Finally we present our conclusions and discussion on future works in Section VI.

2 PALMS Approach

2.1 System Overview

In this section, we present our approach for providing streaming services in PALMS. We discuss the main components of PALMS, the protocols used as well as simple description of the basic heuristic algorithms. For the ease of exposition, we refer to the media source as the *streaming server* and receivers as *clients*. The term *peers* and *nodes* are interchangeable, and refer to the all the clients. Basically, PALMS is comprises of the following components :

1. a centralized bootstrapping point cum streaming server
2. a randomly connected directed graph mesh network
3. clients/peers

Our design goal is to use a simple yet effective design for each component so that PALMS can be easily deployed.

PALMS consists of two main protocols namely the Pull protocol and Push Protocol. Briefly, the functionalities of the two protocols are:

- **Pull Protocol:** Media segments are pulled by receiver nodes upon request made to neighbor nodes. Based on the *buffer content* information obtained from neighbor nodes, a simple heuristic approach is used to select which media segments to be retrieved from neighbor nodes. The detailed push protocol is explained in the next section.
- **Push Protocol:** A simple heuristic algorithm is employed to choose which media segments to be pushed. Media segments that are chosen to be pushed are stored in the *push segment matrix*, PSm. The PSm is derived from information gathered from all connected neighbor nodes. Besides that, we also introduce of a small probability of randomized forwarding for the push protocol. Our objective for the randomized forwarding is to achieve higher delivery ratios even under high churn rates or heterogeneity of network bandwidths.

2.2 Initiation Process

For PALMS, streaming server, s is known to every node. It serves as a centralized bootstrapping point for the initiation process for node joining to PALMS. Even though having a centralized bootstrapping point can be a potential bottleneck for the whole network. Nevertheless, since the streaming server is the only source for media streaming, thus if the streaming server failed then it will cease the transmission of media streaming for all clients.

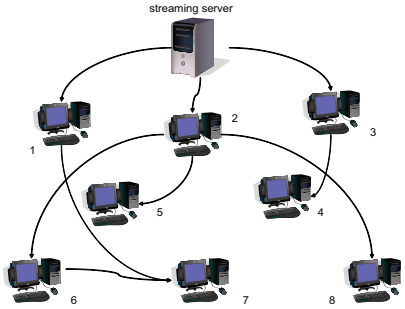


Figure 1: The organized view of PALMS that consists of 8 nodes

As mentioned earlier, PALMS consists of random set of nodes, N that forms a connected mesh overlay directed graph as shown in Figure 1. Each node $n \in N$ has an unique identification ID such as the IP address. A new node p that wishes to join the streaming process will make a contact via HTTP to the streaming server s. The streaming server s possesses the list of all connected nodes in the streaming network. Streaming server will pick a random node from the list and redirect the new node p to a random node q. During the initiation process, all newly joined nodes will synchronize its local clock with the streaming server. Each node also makes contact periodically to the streaming server via HTTP and sends update of its current list of connected neighbor nodes. Moreover, each node also update the streaming server periodically the ratio uploaded media segments over downloaded media segments. Lastly, node that intends to leave the streaming network will inform the streaming server by sending *Leave* message when it departs.

2.3 Media Stream

The streaming data is a continuous media data that is encoded at the streaming server. Streaming server will stream series of segments with monotonically increasing sequence number at a constant streaming bit rate to connected nodes. Each nodes receives new media content from neighbor nodes will provide the following content to other connected neighbor nodes. For PALMS, real-time transport protocol (RTP) is adopted as the transport protocol between connected nodes.

2.4 Membership Management

In PALMS, participating nodes form an unstructured overlay connected network. Each node maintains a certain number of connected neighbor nodes which it retrieves and forwards media segments to each other. However, it's not practical for a node to maintain a membership table for all connected nodes, that is also known as *view* of the whole network system. The size of the *view* grows as proportionate with the size of the network system. The amount of overheads for every node to exchange information with all other nodes will be huge too.

As a result, PALMS employs a scalable membership management protocol similar to SCAMP [4]. The membership management protocol uses the gossip-based information for dissemination of membership information. Every node connects to a set of nodes that is known as *neighbors*. Instead of maintaining a full *view* of the system, every node maintains a *partial view* of the connected nodes. Through continuous gossiping of membership information with neighbor nodes, it helps the node to build good conductivities with other nodes, thus increasing the resilience to network failures or node disconnections. Periodical membership information also acts a "heartbeat"-like mechanism to detect failure nodes, nodes that experiencing long time congestion or node that leaves the streaming network. Such nodes will be removed from the *partial view*.

Removal of failed or departed nodes from the *partial view* will result in the activation of the node replacement policy to establish a new random connection with other nodes from the full *view*. For PALMS, at all time, a fixed number of connected nodes are maintained for all nodes. The node replacement policy has two main purposes : (i) it helps the node to maintain the fixed number of connected neighbor nodes (ii) it helps the node to discover better connectivity and disconnect from free-rider nodes that do not contribute its resources for media streaming. Basically, nodes with higher upload bandwidth will be given a higher priority to establish new connection.

2.5 Proposed Method

The basic idea for PALMS is simple and intuitive. PALMS is based on data driven and receiver-based media streaming protocol. PALMS employs the pull-push mechanism which is similar to the swarm-like content delivery mechanism of BitTorrent. During the beginning of the streaming session, each node will use the pull method to obtain media segments from neighbor nodes. And after a certain period of time, each node has received certain amount of streamed media segments and is able to contribute the media segments by relaying segments to its neighbors through the push mechanism. The pairing of the pull and push protocol improves the utilization of resources and reduces the delivery delay. More over, a simple data incentive policy is employed to discourage the existence of free-riders.

2.5.1 Pull Protocol

In order for a node p to pull required segments from neighbor node q, information of current *buffer content*, b_c of q is required. Each node in PALMS can use the gossip protocol to update its current *buffer content*, b_c periodically with neighbor nodes. The *buffer content*, b_c contains three tuples : sequence number of buffer content ID_b_c, size of *buffer content*, $Size\_b_c$ and segments currently available which are represented by bit vector sequence number. Each segment is identified by sequence number generated by the streaming server and time-stamped. Clock time for all node is synchronized with the streaming server during the initiation process. Bit vector of 1 and 0 are used where 1 indicating that a segment is available and 0 otherwise. At normal circumstances, requests from neighbor nodes will be delivered as soon as possible after the requests are received.

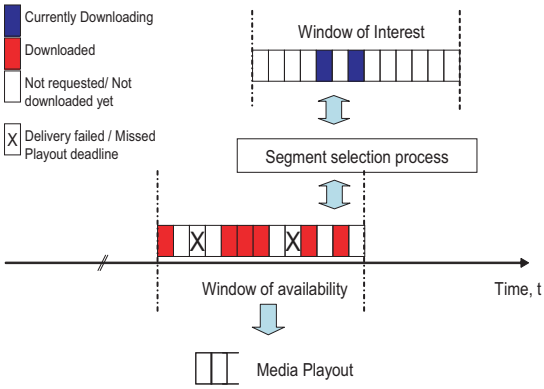


Figure 2: Data buffer for PALMS node

2.5.2 Push Protocol

One disadvantage of the pure pull protocol is that there will be frequent requests from receiver nodes flowing through the network and this is taking up network bandwidth and processing time. Moreover, there are chances that all peers decide to request the same segment from the same supplier node, thus increasing the delivery delay and queueing time.

One of the solutions for the problems stated above is to pair the push protocol with the pull protocol. The push protocol complements the pull protocol and helps to increase the delivery ratios despite high churn, heterogeneity of network bandwidth or network congestion.

In order to keep a minimum overhead, a simple heuristic algorithm for push protocol is employed. The algorithm assists the supplier nodes to execute the push protocol for selecting which segments to be pushed with the aim to increase the robustness of the streaming protocol. Basically, every node determines which segments to be pushed based on the information stored and shared in *Push Segment Matrix*, PSm.

2.5.3 Randomized Push Protocol

In order to increase the probability of transmission of media segments for PALMS, a randomized push protocol can be used as a proactive solution for errors occurred

in transmission. Inspired by [2], as a small percentage of redundant segments to randomized neighbor nodes, errors in transmission can be corrected. For PALMS, we employ a randomized push mechanism with a low probability (0.01 - 0.03) to neighbor nodes selected from the *view*. Naturally, randomized push mechanism will lead to minor redundancy of segments deliveries as well as there will be a small addition of overheads incurred. Nevertheless, we believe such small redundant segments will be detected easily and discarded due to the design and nature of PALMS for live media streaming that is time sensitive. Since each segment has a certain lifetime and is required to be received and played before the playout deadline. As a result, a small randomized push protocol increases the resilience towards network failures or high churn rate.

3 Algorithm

In this section, we present the algorithm for pull and push protocol that make up the core of PALMS system. Pull and push protocol determine the selection of segments to be pulled and pushed, the selection of peers and the recovery process for segments loss.

3.1 Algorithm for Pull Protocol

As mentioned earlier, PALMS uses a simple heuristic pull mechanism to identify a subset of supplier nodes to receive segments from neighbor nodes and at the same time to maximize overall streaming quality based on the playout deadline. Based on the information of *buffer content*, b_c gossiped by neighbor nodes, a heuristic segments scheduling algorithm is used to properly retrieve the required segments. The pull mechanism requires coordination between supplier nodes and receiver nodes. Each receiver node is responsible for the selection of which segment to be pulled from the subset of supplier nodes.

Every node maintains a sliding *window of interest*, which contains the list of segments that are to be pulled from neighbors. Figure 2 illustrates the fundamental concept of the sliding window. While a sliding *window of availability* contains the list of segments available for each node and this is the information for *buffer content* shared with other neighbor nodes. The sliding window moves forward in accordance to the largest time-stamp of segment in the *window of interest*. Without a doubt that using a large window improves the availability of media segments and thus increase the effectiveness of media delivery. On the other hand, more buffer space and a higher startup latency are required.

We assume that each peer knows the incoming and outgoing bandwidth. Let *ibw* denote the incoming bandwidth and *obw* of the outgoing bandwidth. Upon receiving the *buffer content*, b_c from neighbor nodes, the receiver node first check the availability of the segments. Similar to the protocol used by BitTorrent, PALMS employs the *Rarest First* algorithm for segment selection. Absent segments with the larger time stamps will be given higher priority to be pulled. With the objective to reduce transmission delay, segments will be pulled from supplier nodes with higher upload bandwidth, *obw*. In order to reduce redundancy, the pull algorithm determines the next segments to be pushed by neighbor nodes. Information from the *push segment matrix*, PSm, is taken into consideration for the selection of segments to be pulled. The simplified pull algorithm is shown in Algorithm 1. [1]

3.2 Algorithm for Push Protocol

A good selection strategy is required to distribute the segments through the push protocol. It should ensure that every node pushes different segments in order to reduce redundancy. Finally, it should also take into account the requests from neighbor nodes.

Each neighbor node will try to allocate different segments into the *Push Segment Matrix*, PSm to be pushed. A simple roulette wheel selection scheme is applied for the next time interval for each node to push the available segments. Segments with the highest time-stamp or *least sent* will be given higher priority to be allocated into the *Push Segment Matrix*, PSm. Each node keeps a counter of how many times each segment is sent. Segment with the least number of times sent will be chosen. The simplified push algorithm is shown in Algorithm 2.

Let A denote the number of neighbor nodes, and Seg is the list of available segments to be pushed that each node possesses. Each node possesses a subset of Seg. The *Push Segment Matrix*, PSm, is represented by : A$\times Seg$. PSm^t denote the PSm at time t.

$$PSm^t = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

As for loss recovery, receiver nodes that failed to receive expected segments from the push protocol should request it through the next pull protocol cycle. Besides that, any segments that failed to arrive from the pull or push protocol, a NACK feedback is triggered and sent to neighbor nodes. Such node will be given higher probability to receive that segments through the next cycle for the push mechanism.

3.3 Incentive Algorithm

Inspired by the *Tit-for-Tat* policy by BitTorrent, PALMS employs similar policy to discourage the existence of free-riders. Free-riders that only willing to download segments

[1] For convenience, description of the algorithms are simplified.

and do not contribute its resources for uploading will receive a small portion of the media segments and the connection will be chocked. Since the nature of live streaming is bounded by tight time constraint, free-riders will experience a longer delay in receiving media segments and thus suffer a poor quality of streaming.

In the context of PALMS, a simple numeric value is used for the incentive policy. Each node maintains a simple record of the previous activities with every other nodes. Numeric value for the ratio of uploaded over downloaded media segments is recorded and stored. Such value is periodically updated and also sent to the streaming server for storage purpose. In the node selection process for the push protocol, the supplier node chooses nodes with higher numeric value to receive the push segments.

Algorithm 1: Pull Protocol

```
1 while seg ∉ PSm;
2 For seg ∈ window_interest do;
3  determine the rarest segment;
4 if (playout deadline is satisfied)
5  {enqueue segments};
6 else {drop the segment};
7 pull segments;
8  terminate;
```

Algorithm 2: Push Protocol

```
1 For seg ∈ window_availability do;
2  determine the least sent segment || highest time−
       stamped segment;
3 if (playout deadline threshold is satisfied && turn to
      push)
4  {enqueue segments into PSm};
5 else {drop the segment};
6  {enqueue segments into PSm};
7  For all nodes that satisfied the ratio of upload/
       download do;
8  push segments;
9  terminate;
```

4 Simulation Scenario

In order to evaluate our approach, we have conducted a simulation-based evaluation of the performance for PALMS. Our simulation implements the proposed pull and push protocol for PALMS that follows the algorithms described in the previous section. To evaluate the quality of media streaming, we implemented a simple MPEG4 video streaming system and run it on a Network Simulator, *ns2* [1] simulation environment. Our streaming system is based on the Microsoft MPEG4 video encoding/decoding source codes. The streaming rate is set as 400 Kbps. Figure 4 shows the *ns2* simulation environment setup [7].

We used GT-ITM [11] to create a flat random graph as our underlying network topology. We vary the number of nodes participate in the streaming networks from 100, 200, 300, 400 to 500 nodes. The average degree of the graphs is 5. We randomly select a few nodes to depart or

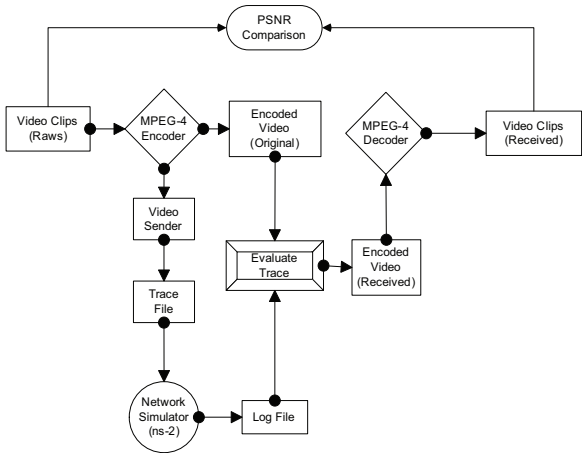


Figure 4: Structure for NS-2 Simulation Environment

join at different periods of time. Each link in the graph has a delay value, which is generated at random. To evaluate the performance of PALMS, we compare the proposed protocols of PALMS against two existing streaming protocol, Chainsaw [9] and PeerCast [12].

4.1 Evaluation Metrics

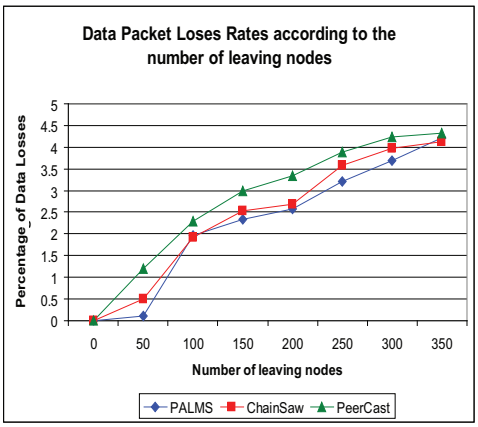
We evaluate the performances of the PALMS against two other existing P2P media streaming protocol using the following metrics:

1. The Peak Signal to Noise Ratio (PSNR)
2. Percentage of Data Lost according to number of departure nodes
3. Comparison of time for the first segment to arrive
4. Comparison of Overhead

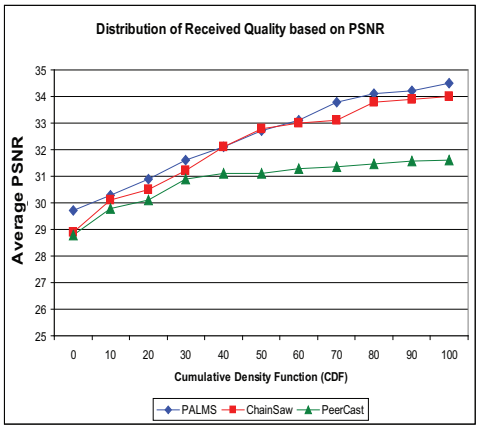
5 Simulation Results

We compare the performances of PALMS against two existing protocols, tree-based overlay multicast streaming protocol; PeerCast and unstructured overlay multicast protocol; Chainsaw. PeerCast consists of end-system nodes on the Internet that execute multicast information dissemination applications. Each PeerCast peers act both as clients and servers in terms of their roles in serving multicast requests. PeerCast system uses the P2P membership protocol to organize the loosely coupled and widely distributed end-system nodes into a P2P network that carries the multicast service. PeerCast peer-to-peer network is a Distributed Hash Table (DHT) based structured P2P network. While Chainsaw is a push-based randomly constructed graph streaming protocol. Each node in Chainsaw creates a list of desired segments and receives segments randomly from neighbor nodes.

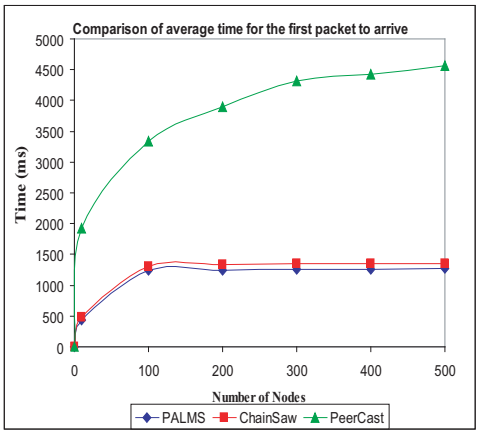
Figure 3(a) shows the result of comparison for percentage of data lost according to the number of departure nodes.



(a) Data Loses Rate according to number of departure nodes



(b) Distribution of Received Quality based on PSNR



(c) Comparison of average time for the first segment to arrive

Figure 3: Comparison Simulation Results

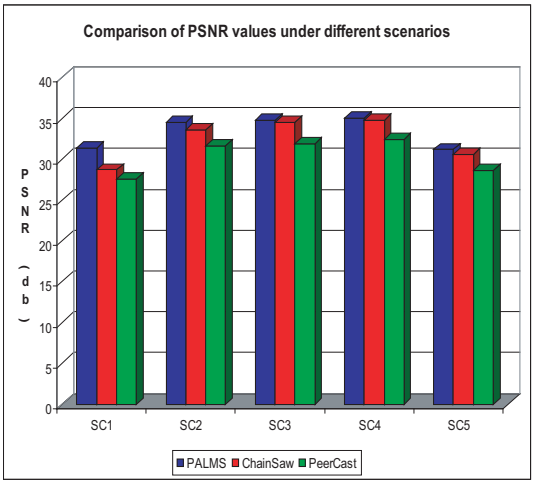


Figure 5: Averaged PSNR for different scenarios

The percentage of data lost for PALMS is smaller as compared to PeerCast and Chainsaw. The simulation result shows that PALMS is much more robust as compared to PeerCast and Chainsaw. PeerCast shows higher percentage of segment lost as nodes are frequently redirected and reorganized in the tree structure during the departure of nodes. On the contrary, PALMS and Chainsaw do not require any restructuring as it consist of an unstructured overlay network. If a node leaves the network, new connections with other nodes can be established easily. With the node replacement policy adopted by PALMS, it's able to maintain a fixed number of connected neighbor nodes while discovering better connections. Moreover, with the combination of the pull and randomized push protocol, PALMS is able to reduce the percentage of segment loss during transmission.

Figure 3(b) depicts the averaged PSNR values for PALMS in comparison to PeerCast and Chainsaw. The peak signal to noise ratio (PSNR) is the measurement of the quality of received streamed video by the nodes in comparison to the original video quality. The streaming quality for live media streaming under PALMS shows better quality as compared to PeerCast and Chainsaw. This is due to the fact that time is the main constraint for playout deadline in live media streaming. As PALMS behaves similarly to BitTorrent, nodes show the swarm-like effect towards newly generated media segments. Moreover, with the combination of pull and push protocol, media segments are selected heuristically and received before the playout deadline.

Naturally, there is always a trade-off between failure resilience and maintenance overheads. In the context of PALMS, maintenance overhead involves the exchanges of membership information among neighbor nodes as well as buffer content. Figure 7 depicts the comparison of overhead between PALMS, PeerCast and Chainsaw. As expected, the overhead for PALMS is slightly higher. Nevertheless, the overhead incurred is still slow as compared to the overall load for media segments.

Figure 3(c) depicts the comparison of average time for the first media segment to arrive. As shown in the results, the averaged time for the first media segment to arrive for PeerCast increases logarithmitically with the number of connected nodes. Primary because PeerCast is based on the overlay tree-structure, the more a node accepts new children, the more resources it shares among connected nodes. Evidently, it takes longer time for the media segment to reach a child of PeerCast. Likewise, for the case of PALMS and Chainsaw, the unstructured overlay network approach is applied. PALMS and Chainsaw have a higher probability of forming connections with nodes with higher bandwidths and resources. Both PALMS and Chainsaw exhibit increment for its average time first media segment to arrive, but soon remain constant with the

Upload B/W	SN 1	SN 2	SN 3	SN 4	SN 5
128 kbps	5%	15%	10%	50%	25%
256 kbps	10%	80%	10%	25%	25%
512 kbps	40%	5%	80%	25%	25%
0 kbps	45%	0%	0%	0%	25%

Table 1: Scenarios for comparing different upload bandwidth under PALMS

number of connected nodes. Based on the results, the average time for the first media segment to arrive is much shorter for PALMS and Chainsaw as compared to PeerCast.

We also conducted simulations and comparisons on PALMS under several network scenarios to evaluate its performance. A comparison is made to evaluate the streaming quality for PALMS as compared to PeerCast and Chainsaw. Streaming quality for PALMS depends on the cooperation of all connected nodes in the streaming network. The streaming quality is poor if the level of cooperation is low even when the network is not heavily congested. Unlike traditional file sharing, live media streaming is bounded by strict time constraints. Thus, cooperation from only a few altruistic nodes cannot provide high quality streaming to all other nodes in a large system. As a result, it is important to investigate the impact of free-riders on the overall streaming quality of PALMS as well as PeerCast and Chainsaw.

Figure 5 and Figure 6 depict the result of PALMS under different network bandwidth scenarios. Parameters for each scenario are shown in Table I. From the results, we can see the effect of free-riders toward the streaming quality of PALMS networks. Inevitably, free-riders decrease the streaming quality due to less cooperation among nodes for uploading received media segments to neighbor nodes. Cooperation among connected nodes is important for PALMS as it is based on pull and push protocols. The increase of free-riders decreases the chances of media segments being pulled by receiver nodes. This in turn reduced the quality of the media streamed. However, in comparison to PeerCast and Chainsaw, the streaming quality of PALMS is still show a significant better streaming quality. It is also interesting to point out that based on different scenarios setting, PALMS exhibits the same streaming quality and is not affected by the effect of heterogeneity of network bandwidths.

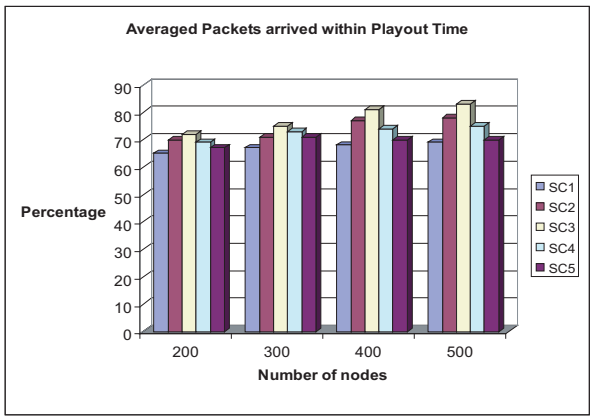


Figure 6: Simulation results for different scenarios

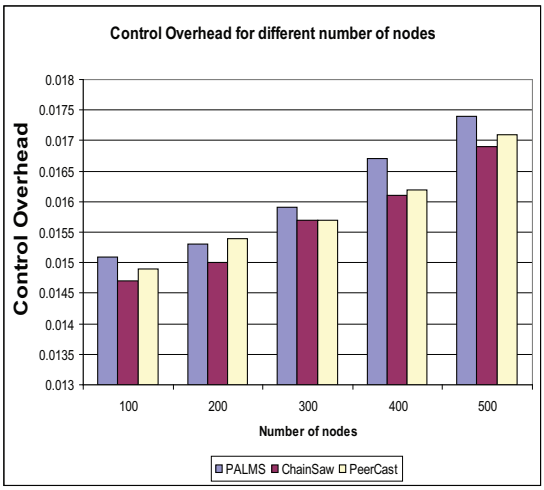


Figure 7: Comparison overhead for different number of nodes

Figure 6 shows the percentage of media segments arrived within the playout deadline. It's expected that scenarios with higher percentage of free-riders increases the media segments delay. Free-riders that are not willing to contribute their upload bandwidth will significantly affect the transmission of segments to all nodes. As a result, for all the segments that are received outside of the playout deadline will be discarded. Nevertheless, with the tit-for-tat incentive policy adopted by PALMS, the effect of free-riders is not so significant.

6 Conclusions and Future Work

In this work, we presented a model for PALMS that incorporates the pull and push protocols. We provided a simple and efficient peer selection scheme for the pull algorithm and push algorithm. With the combination of pull and push methods, media segments can be retrieved efficiently and arrived before the playout deadline imposed by the live streaming constraint. We also introduced randomized push protocol that increase the probability of media segments being distributed across the streaming network. Leveraging on the cooperation among all connected nodes, it is shown that PALMS experiences a significant better streaming quality performance. Based on simulation conducted, the results show that the PALMS perform better as compared to PeerCast and Chainsaw

We also examined and evaluated PALMS under different network scenarios. It is shown that the capacity of PALMS grows exponentially with the number of connected nodes. However one of the main problem for P2P media streaming is the existence of free-riders. Free-riders consume resources and bandwidths from the system but do not provide any contribution. We adopted a simple heuristic tit-for-tat policy to discourage the existence of free-riders. The simulation results show that free-riders significantly reduce the streaming quality of other streaming protocols in comparison to PALMS.

Future works can be implemented on the tit-for-tat policy to discourage the existence of free-riders and to achieve fairness among nodes. Study on the mechanism to encourage nodes to cooperate with each others will be taken into consideration. Finally, we intend to implement PALMS on the PlanetLab Internet testbed.

References

[1] Network Simulator, Ns-2. [Online] Available : http://www.isi.edu/nsnam/ns/

[2] S Banerjee, S Lee, B Bhattacharjee, and A Srinivasan, Resilient multicast using overlays, *Performance Evaluation Review*, 2003, Vol. 31; Part 1, pages 102-113.

[3] B. Cohen, Incentives build robustness in bittorrent, *P2P Economics Workshop*, Berkeley, CA, 2003.

[4] A.J. Ganesh, A.M. Kermarrec, & L. Massoulie. SCAMP: peer-to-peer lightweight membership service for large-scale group communication, *In Proc. 3rd Intnl. Wshop Networked Group Communication (NGC'01)*, pages 44-55. LNCS 2233, Springer, 2001.

[5] Y. Guo, K. Suh, and D. Towsley, A Peer-to-Peer on-Demand Streaming Service and Its Performance Evaluation, *Proceedings of 2003 IEEE International Conference on Multimedia & Expo (ICME 2003)*.

[6] M. Hefeeda, A. Habib, R. Rotev, D. Xu, and B. Rhargava, PROMISE: peer-to-peer media streaming using collectcast, *Proc. ACM Multimedia 2003, (Berkeley, CA)*, Nov. 2003.

[7] J. Klaue, B. Rathke, and A. Wolisz, Evalvida framework for video transmission and quality evaluation, *Proc. 13th Int. Conf. Modeling Techniques and Tools for Computer Performance Evaluation*, Urbana, IL, Sep. 2003.

[8] V. N. Padmanabhan, H. J. Wang, and P. A. Chou, Resilient Peer-to-Peer Streaming, *IEEE ICNP*, Atlanta, GA, 2003.

[9] Pai, V. Kumar, K. Tamilmani, K. Sambamurthy, V. Mohr, A. E., Chainsaw: Eliminating Trees from Overlay Multicast, *LN in Computer Science*, 2005, No. 3640, pages 127-140.

[10] S. Ye and F. Makedon, Collaboration Aware Peer-to-Peer Media Streaming, *ACM Multimedia*, NY, 2004.

[11] E. W. Zegura, GT-ITM: Georgia Tech Internetwork topology models (software). http://www.cc.gatech.edu/project, 1996.

[12] J. Zhang, L. Liu, C. Pu, and M. Ammar, Reliable end system multicasting with a heterogeneous overlay network. *Technical Report GIT-CERCS-04-19*, CERCS, Georgia Institute of Technology, April 2004.

[13] X. Zhang, J. Liu, B. Li, and T.P. Yum, CoolStreaming/DONet : A data-driven overlay network for efficient live media streaming. *Proceedings of IEEE INFOCOM*, March 2005.