Bottom-Up Trie Structure for P2P Live Streaming

Boyuan Zhang, Changcheng Huang, James Yan
Department of Systems and Computer Engineering
Carleton University, Ottawa, Canada
boyuan@sce.carleton.ca, huang@sce.carleton.ca, jim.yan@sympatico.ca

Abstract— By simultaneously providing live video and audio contents to millions of users around the world, peer-to-peer live video streaming (P2P LVS) has become one of the most popular Internet applications in recent years. However, current P2P LVS software has problems such as non-smooth playback and long start-up delay for end users. To address these issues, we design a P2P-based multi-bit Trie structure, called Bottom-Up Trie (BU-Trie), for distributing P2P live contents. Different from other approaches, BU-Trie is a Trie formed and built inversely from leaf nodes (or child nodes) back to the root node (or parent node).This architecture consists of two phases: a diffusion phase and a swarming phase. The main design goal of the diffusion phase is to group the local peers together by discovering physical locations of peers, and design the paths for fast distributing live streams from the source node to end users. The objective of the swarming phase is to find an optimal way for exchanging the video stream chunks within a local group. We propose an algorithm called Most Popular Chunk First (MPCF) and apply it for the swarming phase for efficient chunk exchange. Performance evaluation of the proposed BU-Trie shows that, when compared to other approaches, the sequential throughput of video chunks is increased. The inter-domain traffic, the traffic between different Internet service providers (ISPs), is reduced as well. Such a reduction would benefit carriers economically.

Keywords-component; P2P; live streaming; Bottom-up; Trie; tree-based; Swarming; Diffusion

I. INTRODUCTION

Peer-to-Peer live video streaming (P2P LVS) has become more and more popular in modern Internet and computer communications. Due to the high demand for live contents, many video-on-demand application providers, such as PPlive [1], have added live broadcast features to their software in recent years. However, due to the dynamics of peers and the inefficient structure of the network, many users have experienced problems such as non-smooth playback and long startup delay. Other issues, including playback continuity and packet loss, have also been discussed in previous research [2].

Some approaches have been designed to improve the quality of P2P live video streaming in recent years. In general, those approaches can be divided into two categories: tree-based overlay structure [3-6] and mesh-based overlay structure [7-12]. A tree-based overlay is where a tree-like video streaming is created, with the content source node being the root node and viewers being the child nodes in the tree structure. The video stream is distributed by pushing packets from parent nodes to their children nodes. One major issue for tree-based overlay network is that the system is vulnerable to unpredictable peers' behaviors [7]. For example, all the child nodes will be disconnected from the network if their parent node suddenly becomes unavailable. However, some approaches have been designed to improve the performance, such as multiple parent nodes for each child node.

For a mesh-based overlay network [9-13], peers form a randomly connected overlay. The video streaming is divided into small chunks. Depending on the algorithm applied, the missing video chunks are pulled by one peer from its neighbor peers who have already had the desired data. The mesh-based structure is robust to the unpredictable peers' behaviors since each peer always maintains a list of its neighbor peers and has multiple neighbor peers to download video chunks from. However, mesh-based overlay has poor performance for live contents due to the limited availability of future contents and the poor sequentiality of the received video chunks, which may result in non-smooth playback and long waiting time [9].

By carefully analyzing the advantages/disadvantages of previous research and the behavior of users during a P2P live streaming session, we design the Bottom-Up Trie (BU-Trie), a multi-bit Trie structure for P2P live streaming. The rest of this paper is organized as follows. The detail design of BU-Trie and its features are given in Section II. The swarming phase and our MPCF algorithm are described in Section III. Section IV shows the evaluation of this design by simulation and comparisons. Finally, Section V concludes this paper and identifies directions for further work.

II. DESIGN OF BU-TRIE AND DIFFUSION PHASE

In this section, we show the detailed design of Bottom-Up Trie and how to build it inversely from leaf node to root node.

The main idea of BU-Trie is to let each node in the network download video from its closest neighbor nodes. We use the term "closest neighbor nodes" to refer to the nodes that are geographically close to each other. In order to determine whether the locations of two nodes are geographically close to each other, we compare the IP addresses of the two nodes. According to [14], most of the IP address allocations are related to nodes' physical locations, and the physical locations of IP prefix rarely changes. In this case, if the IP prefixes of two nodes are the same, for example, the higher 24 bits in IP addresses are the same, then we consider these two nodes are geographically close to each other. Or otherwise, even though the physical locations of two nodes with the same IP prefix are not close to each other, they are usually in the same network domain or under the same ISP, where closeness means the cost is cheaper because they are in the same ISP.

978-1-4577-2053-6/12/$31.00 ©2012 IEEE

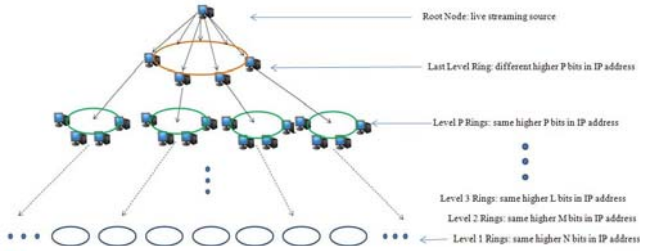

Figure 1. Bottom-Up Trie Architecture

By using the philosophy described above, we can break down all peers and organize them into a hierarchy of small groups, where all peers in the same group are close to each other. We assume there is a registration system for a P2P application. The registration system can be either centralized or distributed.

To start, each peer contacts the registration server with its IP address. The server will compare all peers' IP addresses and divide the peers into different groups according to the higher N bits in their IP addresses. Note that the peers in the same group will have the same higher N bits in their IP addresses. In other words, only the last 32-N bits in their IP addresses are different for the peers in the same group. As shown in Figure 1, each circle represents a group. At level 1, all peers in the same circle (or group) have the same higher N bits of their IP addresses where N is set to be 28 as an example. Therefore, there are maximum 16 different peers within the same Level 1 group after the first division. It is easy to see that, based on our definition of closeness, the peers within the same group are close to each other.

After Step one, each peer will be informed of its neighbor information by the registration system. The peers in the same group will form a mesh network which applies the DHT building process with fingers if necessary. Each peer will also be assigned an identifier based on its last 32-N bits of IP address.

In Step 3, one peer in each group is selected as the parent peer for a P2P application. We assign a key value to each live streaming source. The key value is then hashed to a node identifier. In each Level 1 group, we select the node with closest node identifier in the last 32-N bits as the parent for the specific application. For example, if the last 4 bits of the hash value of a video session is 0101, then the nodes with the same last 4 bits of IP addresses in each group will be chosen as the parent node in this group for this specific video session. If no node has the same last 4 bits of IP address equal to 0101, then the next closest node in terms of IP address will be chosen. The host node will serve as a parent in the Level 1 group. It should be noted that different applications will select different parent nodes. This will allow loads to be distributed relatively evenly among different peer nodes.

In Step 4, the registration system selects a value M as the number of bits to compare for Level 2 in the hierarchy, where $M<N$. The Level 1 groups with the same first M bits in their IP addresses will be in the same Level 2 group. There will be maximum 2^{N-M} Level 1 groups in a Level 2 group. The parent node in each Level 1 group for each application will serve as the representative for the Level 1 group in Level 2.

In Step 5, the registration system will inform each parent of Level 1 its neighbors in Level 2. These parent nodes then form a Level 2 DHT ring with fingers if necessary. A node identifier based on the bits M to N will be assigned to each Level 1 parent as their Level 2 identifier. For each application, the node with closest Level 2 identifier for the bits N to M of the hash of the specific application key value will be selected as the parent of Level 2 nodes.

For example, Level 1 parent nodes with the same higher 24 bits of IP addresses can be organized to form the Level 2 group. Then for the hashed key value of the same live streaming source, the most matching node in each Level 2 group, for example the node has the same bits 25 to 28 or the node with closest bits 25 to 28, will be chosen as Level 2 parent in the level two group. Up to this point, the second bottom level of the BU-Trie is finished.

We use the same strategy to recursively build the third level, the fourth level, etc., until the source of the application is selected as the last level. Here we assume that the key of the live content is hashed to the IP address of the source. Figure 1 shows the overall architecture, where $P<...<L<M<N<32$. It is easy to see that the overall structure forms a multi-bit Trie. As described above, our unique construction process starts from the bottom level and moves level by level to the root node. Therefore it is called a BU-Trie (Bottom Up Trie) structure.

The diffusion phase starts right after building the BU-Trie. During the diffusion phase, the root node continuously push video stream to each of the next level parent nodes. Then these parent nodes will further push the stream to their child nodes. This process will continue until level-1 parent nodes are reached. Since the uploading bandwidth is limited, a level-1 parent node will randomly push video chunks to different peers in the same group. As a result, each peer as a leaf node may receive random number of video chunks. The advantage of this diffusion phase is that video stream can be fast distributed from the root node to a wide area covered by the Trie. Even though there may be many missing chunks at this moment, the speed of distribution is high, since the video stream is diffused along the Trie that is built based on closeness.

III. SWARMING PHASE AND MPCF ALGORITHM

The swarming phase of BU-Trie design is the second phase that starts in parallel with the diffusion phase. If we consider the entire diffusion phase as a push-driven process that allows some end users to receive some number of video chunks. Then the swarming phase is a pull process that allows end users to exchange and download the video chunks that are missing. By considering the characteristic of P2P live streaming, we propose an algorithm called Most Popular Chunk First (MPCF) for the swarming phase.

In MPCF algorithm, each peer maintains bitmaps of all other peers in the same local group indicating which chunks are available. However, contrary to Rarest-First algorithm, MPCF lets peer download the most popular chunk first, i.e., always download the chunk that is shared by the most number of peers. If the most popular chunk is unavailable, for example due to

peer churn or limited node capacity, then the next popular chunk will be selected and downloaded. MPCF is ideal for real-time multimedia applications such as P2P live streaming due to the following reasons:

1. In real-time multimedia applications, the older chunks, the chunks that are generated earlier, are always shared by more peers and therefore are more popular. Based on the definition of MPCF, peers tend to download older chunks ahead of newer chunks. So MPCF can achieve very high sequentiality compared to Rarest-First algorithm which tends to download newer chunks ahead of older chunks. Here, the sequentiality means the percentage of chunks that are downloaded sequentially. Since the buffer size is limited, older chunks may be obsolete. By downloading oldest chunks first, we avoid the situation that some old chunks are starved in the sense that they have been deleted before fully downloaded. Therefore, with high sequentiality, MPCF is better than Rarest-First algorithm for live video streaming applications. Also, in Rarest-First algorithm, the rarest-shared chunk always has limited availability due to limited uploading bandwidth. If many peers want to download this rarest-shared chunk simultaneously, there might be longer delay or waiting time due to the limited availability. However, for MPCF, the most popular chunk is shared by the most number of peers, so it has high availability. Even though many peers want to download this chunk simultaneously, the waiting time is short.

2. By comparing with Naive Sequential algorithm, both of them tend to download older chunks ahead of newer chunks. However, MPCF is better in the sense that it does not enforce sequential requirement. This allows more flexibility in selecting peers. In some cases, peers running sequential algorithm may need to wait because some chunks may not be available due to limited bandwidth, while peers running MPCF can move on to next most popular chunks that have bandwidth to download. This will maximize bandwidth utilization.

Based on the discussion above, it is reasonable to conclude that MPCF algorithm can achieve higher sequentiality than Rarest-First algorithm and higher throughput than Naive Sequential algorithm, which is ideal for real-time multimedia applications such as P2P live streaming. At the beginning of the swarming phase, peers in the same local group will download the bitmap from the local DHT table, which indicates the chunks each peer has downloaded. By checking the bitmap, peers will select the chunks that are owned by the most number of peers in the local group. When the total uploading bandwidth for the most popular chunk is exhausted, the next popular chunk will be selected for downloading. The bitmap in the local DHT table will be updated if any peer has a new chunk or deletes an old chunk. Every time a peer has the capacity to download a new chunk, it will check the local DHT table to see whether the bitmap has changed. If the bitmap is changed, it will update its own bitmap and proceed to the selection of the next most popular chunk.

By applying the diffusion phase of the BU-Trie, each new peer can easily determine its position in the Trie for certain live streaming. At the same time, by applying the swarming phase, peers can easily determine which video chunks need to be downloaded and where to download the desired video chunks. If a leaf node happens to leave, only the DHT table in the lowest level group the leaf node belongs to needs to be updated. On the other hand, if a parent node at Level *x* leaves a session, another node within the same group in Level *x* will be selected immediately as the parent. Related DHT labels in the level x+1 to the lowest level will be updated. These DHT tables are the closest ones to the leaving node and therefore can be updated very fast. In general, the BU-Trie and the MPCF algorithm achieve our goal to localize traffic as much as possible and henceforth minimize delay. Each application session has its own BU-Trie. This will allow peers to share load across different live sessions and avoid bottleneck problem.

IV. Performance Evaluation

To evaluate the performance of the BU-Trie, some simulations are designed and implemented. We use Opnet [15] as our simulation tool.

A. Simulation Design and Assumptions

Due to the complexity and the limitation of the software, we cannot simulate a real life P2P LVS network. Instead, we created 128 nodes in total for our simulation. As shown in Figure 2, the 128 nodes are divided into 4 domains. The nodes in the same domain have same or similar IP address prefix, for example, we assume that each domain represents an ISP. We also assume that the 32 nodes in each domain are physically close to each other and have less delay. We run the simulation with heterogeneous nodes. For example, nodes have different uploading bandwidths from 5Mbps to 20Mbps. Our simulation focuses on one live streaming session. Therefore one node is selected as the source node as shown in Figure 2. The source node keeps generating live video chunks, where each chunk has unique size of 512KB. In this simulation, we build a 3-level BU-Trie, where we set N=28, M=16. Figure 3 shows the BU-Trie structure. Each level 1 ring has 8 peers, where the 8 peers have the same higher 28 bits in their IP addresses. One of the 8 peers in each ring is then selected as the parent peer node. These 16 parent nodes form the level 2 rings based on the higher 16 bits in their IP addresses. After that, 4 level 2 parent nodes are selected from the 4 level 2 rings, and form the level 3 ring. Therefore the level 3 ring has 4 parent nodes, and one of them is the source node.

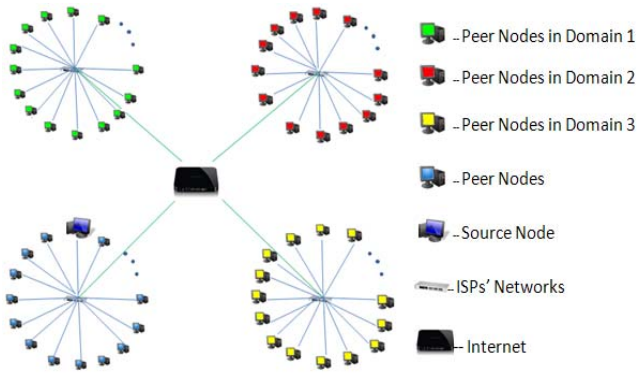

Figure 2. Network Topology for Simulation

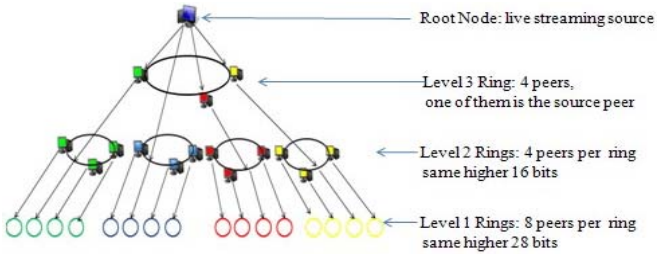

Figure 3. BU-Trie Structure for Simulation

Three tests are made with this simulation. Test 1 is the Sequential Throughput test. Sequential Throughput is defined as the number of sequential chunks downloaded per second. In other words, we measure the average downloading speed of the in-order video chunks. For P2P LVS applications, higher sequential throughput means better playback continuity. Test 2 is the end-to-end (ETE) delay test. The ETE delay is measured by the waiting time from the time of sending one chunk by the source node to the time of receiving this chunk by a leaf node. Shorter ETE delay means less initial buffering time. Test 3 measures the average number of inter-domain packets, i.e., the average number of packets that are transmitted between different domains. We compare the results with three other algorithms: Rarest-First (RF), Naive Sequential (NS), and Hybrid (half RR half NS).

B. Simulation Results

Test 1 is the Sequential Throughput test. We record the total number of chunks received by the node and the total number of chunks in sequence during the simulation. Without loss of generality, we randomly pick a node (node_6) to show our analysis as an example.

Figure 4 shows the simulation results obtained from the random picked node (node_6). The left figure shows the total number of packets received by node_6 versus time. The right figure shows the total number of packets received sequentially versus time. Based on our definition, the slopes of the two lines represent the average throughput and the average sequential throughput. The two slopes shown on Figure 4 are 7.5459 (packers/sec) and 7.0651(packers/sec).

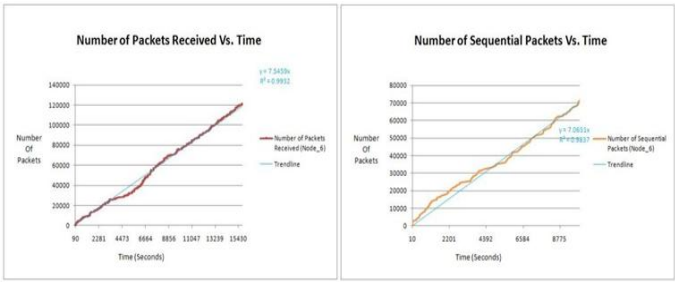

Figure 4. Throughput Test Results of a Random Picked Node

We repeat this analysis for the remaining 31 nodes in domain 1, and calculate the average value for all the four algorithms. Figure 5 shows the average total throughput and average sequential throughput test results. As stated above, the slops of the lines represent the TR and the STR. The numerical results are shown in Table 1.

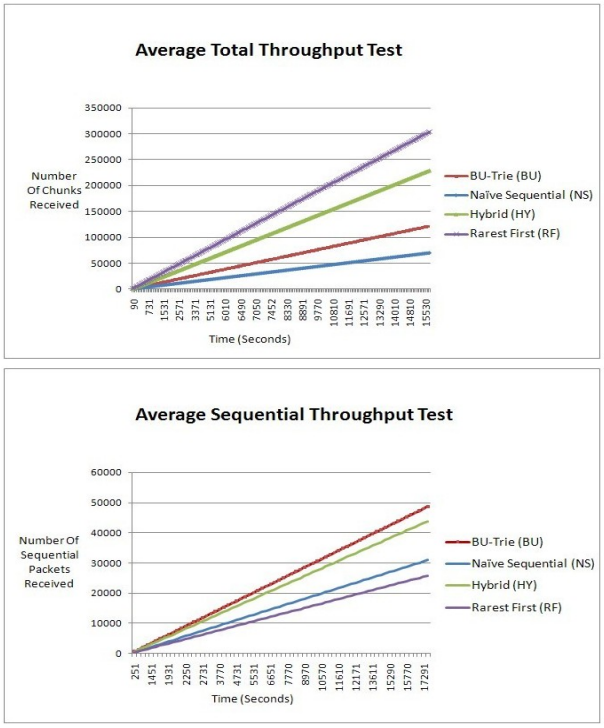

Figure 5. Sequential Throughput Test Results

Table 1 shows the average throughput and average sequential throughput by using the slope of the Figure 5 multiplied with 512k (the size of each chunk). We can see that even though the Rarest-First algorithm has the highest total throughput, the chunks downloaded are not in order. As a result, the playback continuity is poor due to the low sequential throughput. For Naive Sequential algorithm, all downloaded chunks are in order, but the throughput is relatively low since most of the peers download the newest chunk at the same time. BU-Trie has higher total throughput than the Naive Sequential and much higher sequentiality than the Rarest-First. As a result, BU-Trie achieves the highest sequential throughput as predicted.

Table 1. Sequential Throughput Test Results

Test	Average Throughput (Mbps)	Sequential Throughput (Mbps)	Sequentiality
BU	3.857	3.618	0.938
RF	10.098	1.917	0.19
NS	2.303	2.303	1
HY	6.917	3.416	0.494

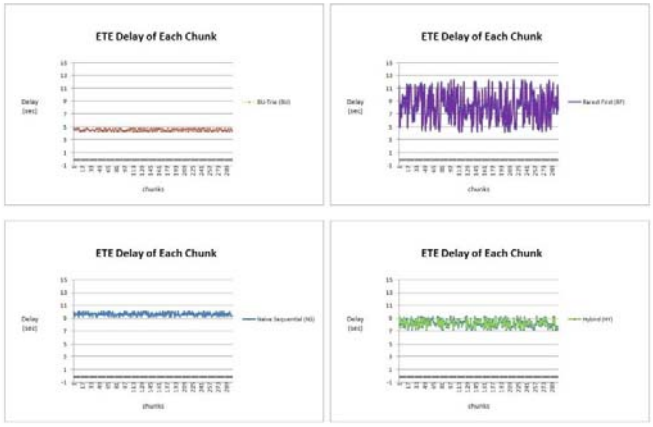

Figure 6 End-to-End Delay Test

Test 2 is the end-to-end (ETE) delay test. Figure 6 shows the ETE delay of each chunk received by the randomly picked node_6. For BU-Trie, the ETE delay is relatively low and stable. The delay is stable, because it has similar downloading order as the NS algorithm. However, the delay of BU-Trie is much lower, this is due to: 1. MPCF algorithm makes sure the next downloaded chunk always has the largest uploading bandwidth, and it allows peers to move on to the next popular chunk if there is a bandwidth bottleneck. 2. The BU-Trie allow peers to exchange data with the peers that are close to each other, for example, within the same ISP or physically close to each other. As a result, the propagation delay is much lower.

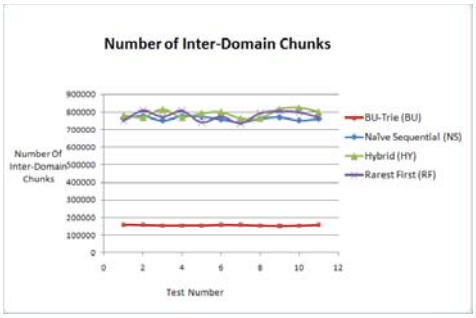

Figure 7. Number of Inter-Domain Packets

Table 2. Average Number of Inter-Domain Chunks

Algorithm	Average Number of Inter-Domain Chunks
BU	15612
RF	76270
NS	80334
HY	76412

Test 3 is made to count the average number of the inter-domain packets during 1 hour live streaming session. The reason we make this test is because P2P applications always generate huge amount inter-domain traffic between different ISPs, which is not preferred by ISPs. We repeat the test 10 times for each algorithm, the results are shown in Figure 7 and Table 2. The results show that the number of inter-domain packets is reduced significantly by applying the BU-Trie design. Since most of the traffic in the lower level of the BU-Trie is within the same domain, the traffic between different ISPs is much less as predicted.

V. Conclusions and Future Work

In this paper, we designed the BU-Trie for P2P live streaming application. The BU-Trie can be easily built by following our simple algorithm. By applying MPCF algorithm, BU-Trie design can achieve relatively high sequential throughput and lower end-to-end delay, which reduce the initial buffer time and improve the playback continuity. Since the BU-Trie prefers downloading locally, the number of inter-domain packets can be reduced as well. We will continue to work on this brand new Trie structure and do more performance study to demonstrate its full benefits.

Acknowledgment

This work was supported by the NSERC grant #CRDPJ 354729-07.

References

[1] *PPLive,* http://www.pptv.com/en/.

[2] X. Liu, H. Yin, C. Lin, Y. Liu, Z. Chen, X. Xiao, "Performance analysis and industrial practice of peer-assisted content distribution network for large-scale live video streaming." *IEEE AINA 2008,* pp.568-574, Okinawa, March 2008.

[3] Z. Lu, Y. Li, J. Wu, S. Zhang, Y. Zhong, "MultiPeerCast: A tree-mesh-hybrid P2P live streaming scheme design and implementation based on PeerCast." *IEEE HPCC 2008,* pp.714-719, Dalian, September 2008.

[4] C. Xu, G. M. Muntean, E. Fallon, A. Hanley, "A balanced tree-based strategy for unstrucatured media distribution in P2P networks." *IEEE ICC 2008*, Beijing, May 2008.

[5] H. Liu, I.Wu, F.Jen, "MeTree: A Contribution and Locality-Aware P2P Livee Streaming Architecture." *IEEE AINA 2010,* pp.1136-1143, Perth, March 2010.

[6] X.Tu, H.Jin, X.Liao, J.Cao, "Nearcast: A locality-aware P2P live streaming approach for distance education." *ACM TOIT2008,* Vol 8 Issue 2, New York, February 2008.

[7] N. Magharei, R. Rejaie, and Y. Guo, "Mesh or multiple-tree: A comparative study of live P2P streaming approaches." *IEEE INFOCOM 2007*, Anchorage, May 2007.

[8] N. Magharei, R. Rejaie, "PRIME: Peer-toPeer Receiver-Driven Mesh-Based Streaming*", IEEE/acm transactions on networking, vol. 17, no. 4*, August 2009

[9] K. Sripanidkulchai, "The Feasibility of Supporting Large-scale Live Streaming Applications with Dynamic Application EndPoints", *ACM SIGCOMM 2004*, Portland, August 2004.

[10] X. Zhang; J. Liu; B. Li; Y.-S.P.; , "CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming," *IEEE INFOCOM 2005,* Miami, March 2005.

[11] B. Fan, D. Andersen, M. Kaminsky, K. Papagiannaki, "Balanceing Throughput, Robustness, and In-Order Delivery in P2P VoD," *ACM CoNEXT 2010*, Philadelphia, November 2010.

[12] Y. Zhou, D. Chiu, J. Lui, "A Simple Model for Analyzing P2P Streaming Protocols*," IEEE ICNP 2007,* Beijing, 2007

[13] B. Zhao, J. Lui, D. Chiu, "Exploring the Optimal Chunk Selection Policy for Data-Driven P2P Streaming Systems," *The 9th Conference on Peer-to-Peer Computing,* 2009

[14] M. J. Freedman, M. Vutukuru, N. Feamster, H. Balakrishnan, "Geographic locality of IP Prefixes." *ACM IMC 2005*, Berkeley, 2005.

[15] *Opnet*,http://www.opnet.com