國立政治大學資訊科學系

Department of Computer Science

National Chengchi University

碩士論文

Master’s Thesis

在點對點網路上針對串流資料傳播的品質保證

Quality Assurance of Streaming Data Dissemination over P2P Network

研究生：邱威中

指導教授：連耀南

中華民國九十八年十二月

December 2009

摘要

網路技術發展的日新月異帶領了眾多新網路服務的崛起，例如即時影音串流這類的多媒體服務。但即時影音串流服務所產生的龐大資料流和傳輸延遲時間的嚴格限制也隨之而來的為網路環境帶來許多挑戰，在這些條件下，傳統Server-client 拓樸架構將 client 要求的影音資料以單一鏈結傳輸時，常會因為頻寬不足而面臨嚴重的封包遺失，或是資料流擁擠造成的額外傳輸延遲使得封包無法達到即時性的需求。P2P 網路擁有 server-client 架構所難以達到的規模伸縮性，且對於節點、鏈結失效所引起的傳輸錯誤也較能容忍，更重要的是，它有效的分散了原本負載在少數 link 上的龐大資料流。因此 P2P 架構近年來風行於即時影音串流服務。

目前 P2P 網路的拓樸多是隨意形成，當網路成員規模龐大時，由傳送端出發到遠方的接收端，途中可能經過無數的鏈結，每一個鏈結都會由於頻寬的不足使得資料流遭受某種程度的品質損害，另一方面，對即時影音服務而言，若資料流的累積延遲時間超出可容忍範圍時，無法為使用者接受。

本研究嘗試找出一個較好的拓樸用以傳輸多媒體資料流，使得位於最遠端節點的累積延遲亦能為使用者接受，且資料品質的損害程度最小。我們將之建置成一 NP-Complete 複雜度的問題模型，名為 MLDST。而解法則是修改 Dijkstra single-source shortest-path 演算法，並加上每個節點承擔下游節點數量及延遲時間限制而來。我們以 PlanetLab 環境在實際的網路上進行實驗，證實我們的演算法比傳統的 Minimum-Spanning Tree 及 shortest path spanning tree 有更好的影像品質。

Abstract

Numerous new network services arise with the advanced development of network technologies, such as real-time multimedia streaming services. But challenges to network environment come along with the enormous traffic of data flows and rigorous restriction to transmission delay of real-time multimedia streaming services. Under this circumstance, conventional server-client topology suffers from serious packet loss and packet delay due to the overload of servers and their accessing links. Also, extra transmission delay may make packets fail to meet the requirement of real-timed services. Peer-to-peer network is more scalable than server-client model, and is much more tolerable to the transmission errors caused by node or link failures. More importantly, it effectively distributes load from the server to peers. As a consequence, peer-to-peer service architecture becomes very popular for real-time multimedia streaming services recently.

Peer-to-peer networks are mostly formed in random fashion. As the size of network grows, packets may have to travel through numerous links to reach far-end receivers. The quality of data may be damaged by insufficient bandwidth of links. For real-time multimedia services, it is not acceptable to users if the cumulated packet delay exceeds a tolerable limit.

Our research is trying to find a better topology to transmit multimedia data flows which makes the cumulated delay of the most-far-end user be tolerable and the damage of data quality is minimized. The problem is modeled as a MLDST problem, which is a NP-Complete problem. To solve the problem, we modified Dijkstra's single-source shortest-path algorithm by bounding the node degree and adding delay constraint. The experiments were carried out on real network environment through PlanetLab. Experiments show that our algorithm outperforms traditional MST and

shortest path spanning tree.

I、Introduction

1.1 Peer-to-Peer Multicasting Network

As advance of network technology, many new network services emerged rapidly, such as real-time multimedia streaming service. Conventional server-client model, as depicted in figure 1, is no longer adequate to support these kinds of services because of the extremely heavy traffic load they generate and the stringent time requirement they ask. Server-client topology puts entire traffic on a single link connecting the server and each requesting clients. Under this circumstance, the link bears a great burden such that packets may suffer from huge packet loss and excessive long delay. Therefore, server-client model may not be a good option for real-time multimedia streaming service.

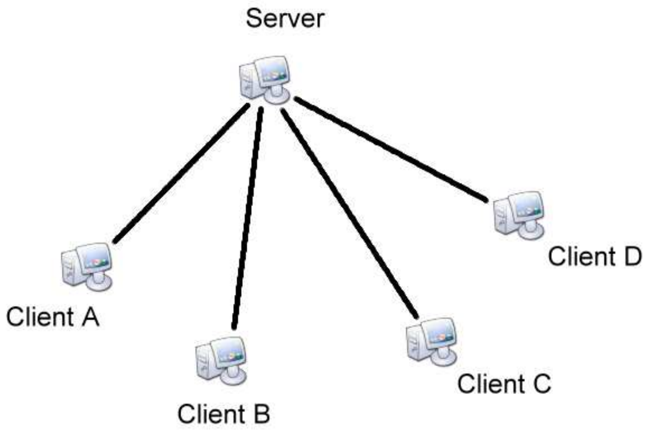

Figure 1 Architecture of server-client model

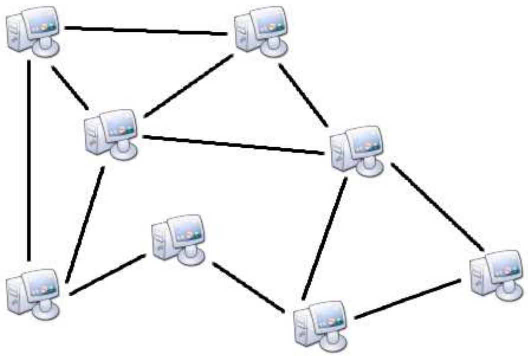

Figure 2 Architecture of peer-to-peer network

On the other hand, the new-emerging peer-to-peer network, originated from BitTorrent distributed file sharing system, is more adequate for real-time streaming services, as depicted in figure 2, due to its high scalability which server-client model can hardly reaches as well as its better tolerance of node failure. Moreover, it is worth noticing that peer-to-peer fashion effectively distributes traffic load from heavy loaded links to all other links, since peers do not only download video but also upload it to other users who want to watch. As a consequence, peer-to-peer network becomes more and more popular for real-time streaming services.

Unfortunately, although peer-to-peer network surpasses server-client model in several aspects, there are still inevitable problems yet to be overcome. Peer-to-peer networks are most formed freely without consideration of either the balance of peer load or the depth of the spanning tree. Furthermore, the popularity of error prone wireless links is increasing rapidly recently such that not only delay time, but also packet loss rate, must be taken into consideration. Whenever the size of the network grows enormously, number of long paths and overloaded peers, accompanying with long transmission delay and high packet loss rate, increases as well. This problem has been well studied and a large amount of researches has been proposed [4, 7, 9, 10, 15, 17, 18]. Spanning trees are widely applied to peer-to-peer network since the very nature of itself, such as easy to build and maintain, well constructed for data transport, quick reaction to nodes failures. However, most of these solutions can only take care of one quality parameter such as delay time.

Focusing on the video streaming services, we propose to build a multicasting tree that can preserve the data quality at a user-acceptable level.

II、Related Work

There already exists many peer-to-peer solutions [4, 7, 9, 10, 15, 17, 18] and can be roughly classified into two categories [16]. In mesh-pull based systems, videos are divided into small clips for distribution. If a user demands to watch a video, he/she must send out request messages to ask his/her neighbors whether they have the clips of the video or not. After being notified by response messages, the user retrieves those clips from possessing neighbors. The control messages generated create a large overhead. Moreover, extra delay arises from the round trips of requests and responses. Typical mesh-pull based systems are CoolStreaming [12], PPlive, and Chainsaw [13].

In tree-based systems [15], video data distributes in clips as well. Nodes simply receive data from their parents after they demand for it [15]. The overhead caused by large amount of messages is avoided. Typical tree-push based systems are Chunkyspread, Splitstream, ESM, and ZIGZAG [15].

Major characteristics and constraints of both types of systems are summarized as follows.

1. *Both types of systems have a much better scalability than client-server systems.*
2. *Existing solutions do not take into account packet loss rate of links.*

   Existing researches did not focus on packet loss but only strive for meeting the delay requirement of real-time streaming services. In our opinion, restricting delay under a user-acceptable level is quite sufficient, there's no need to pursue the minimization of packet delay. On the other hand, minimizing data loss is more important than minimizing delay. Knowing that there are many poor-resourced links residing in real network, packets traveling through those links may have a great probability being dropped. Furthermore, if those links reside in a long transmission path, the data transmitted along this path will suffer from great damage in quality because of numerous packet losses.
3. *With respect to a video clip, all participating peers form a spanning tree topology.*
4. *The topology of the spanning tree is form randomly without any control such that long paths often presented.*

As the size of network grows, packets may have to travel through numerous links to reach far-end receivers. The longer the path, the higher the packet loss rate and longer transmission delay. No one would like to watch a soccer game from the Internet and to see the winning goal few seconds after hearing his/her neighbors' screaming. Therefore, both packet loss rate and delay time must be controlled under respective thresholds.

Most current solutions model a P2P multicasting network into a spanning tree problem. However, traditional spanning tree algorithms have some structural characteristics that may become obstacles on the way to reach our objective.

Different spanning tree algorithm constructs trees with different criterions and thus forms unique structure characteristics. The typical two spanning tree algorithm are minimum spanning tree and shortest-path spanning tree.

A minimum spanning tree has a minimum total cost. It often generates a long tail in the tree [17]. A long tail may cause a large hop count and long delay as well as higher packet loss rate.

As for single-source shortest path algorithms, which also generate spanning trees, their objective is to find a shortest path from the source to all other nodes. The node degree in the resulting spanning tree is unbounded. Large node degree will increase the processing time within a node and thus increase total delay.

This paper proposes to model the problem into a Minimum Loss Diameter Spanning Tree (MLDST) problem which can meet stringent delay requirement and minimize the data loss. The rest of this section will briefly review some existing solutions of peer-to-peer streaming services, especially for ZIGZAG system, which will serve the benchmark in the evaluation of our solution, and a few constrained spanning tree problems.

2.1 CoolStreaming

Different from tree-shaped overlay, Tree-push systems, in which video is pushed from original source to peers, peers of Mesh-pull system form a mesh-shaped overlay and pull contents from each other.

CoolStreaming [12] and PPLive are two well-known Mesh-pull systems. In Mesh-pull system, a video is divided into media chunks for users to request from channel server. Apart from channel server, tracker server maintains a list of hosts who are interested in watching the same video. Hosts on the list establish TCP/UDP connections to deliver video chunks cooperatively.

The message which hosts use to communicate with each other is called Buffer Map. Buffer Maps indicate available video chunks that a host has and willing to share. Once a host receives a Buffer Maps from another host, it can request for what it needs, the requested chunks will then be scheduled to be delivered to it. Figure 3 illustrates a CoolStreaming system.

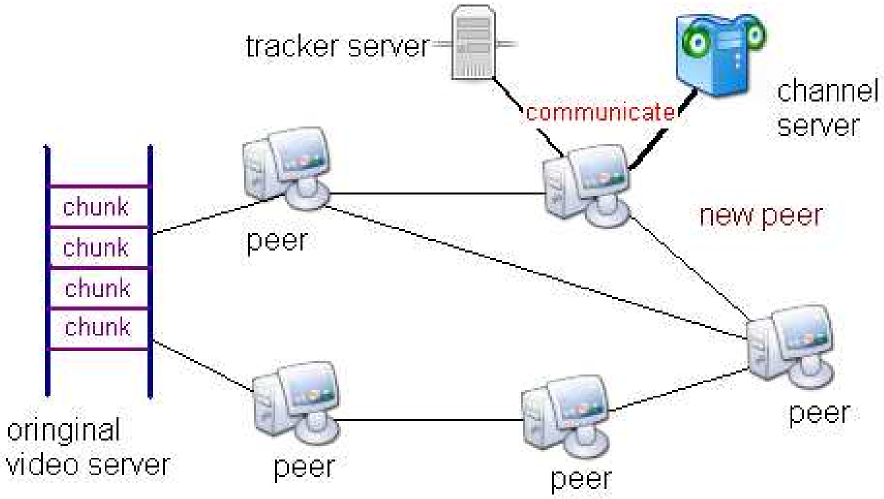

Figure 3 Architecture of a CoolStreaming system

In CoolStreaming, playback progresses of the peers are semi-synchronized and any segment downloaded after its playback time will be useless. For failure recovery, CoolStreaming maintains a stable number of peers in the member list since peers accidentally either depart or crash. It also allows each node periodically establish new partnership with randomly selected nodes; as a result, nodes have better chance to find partners of great quality.

2. 2 Chunkyspread

Chunkyspread [17] is a Tree-push approach and constructs a single-source multicast group among a set of end users. To disseminate a video to end users, it splits video into M pieces and each piece is transmitted through one multicast tree. That is, there will be M multicasting trees, which need not to be node disjoint. Using multiple trees for data dissemination provides fine-grained control over member load, reacts quickly to membership changes, scales well and has low overhead.

The rest part of Chunkyspread focuses on load balance and the quick reaction to peer churn, which is not our concern, so that we will not discuss them further.

2. 3 ZIGZAG

ZIGZAG [15] is a single source Tree-push streaming application which had been proved to be height logarithmic and able to bound node degree in a constant. This helps reduce the number of processing hops on the delivery path to each client while avoiding network bottleneck and long end-to-end delay. ZIGZAG organizes members into a hierarchy of bounded-size clusters and builds a multicast tree rooted at a media server. The administrative organization of peers is depicted in Figure 4.

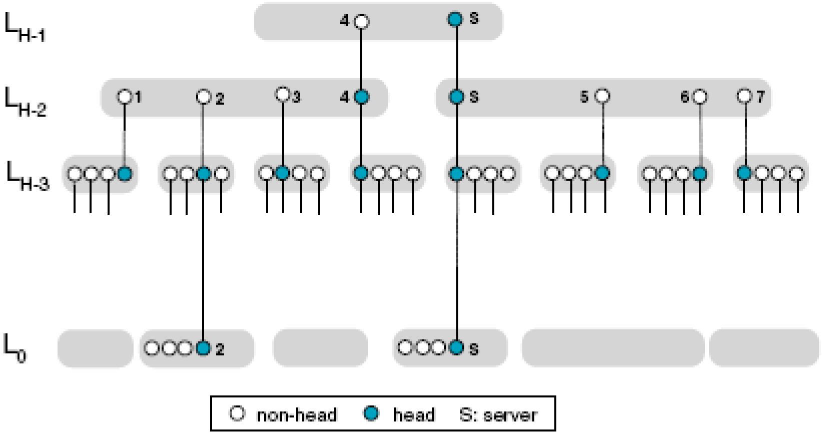

Figure 4 Administrative organization of peers in ZIGZAG

Peers in Figure 4 are organized in a multi-layer hierarchy of clusters and defined recursively according to following rules (where *H* is the number of layers and *k* > 3 is a constant):

- Layer 0 contains all peers.
- Peers in layer *j* < *H*—1 are partitioned into clusters of size of [*k*, 3*k*]. Layer *H*—1 has only one cluster which has a size of [2, 3*k*].
- A peer in a cluster at layer *j* < *H* is selected to be the head of that cluster. This head becomes a member of layer *j* + 1 if *j* < *H* – 1. The server *S* is the head of any cluster it belongs to.

The cluster size is upper-bounded by 3*k*. The above structure implies $\mathrm{H} = \Theta(\log_{k} \mathrm{N})$ where N is the number of peers. Any peer at a layer *j* > 0 must be the head of the cluster it belongs to at every lower layer.

This administrative organization does not infer a data delivery topology. Instead, a multicast tree with some given rules for transmission is depicted in Figure 5.

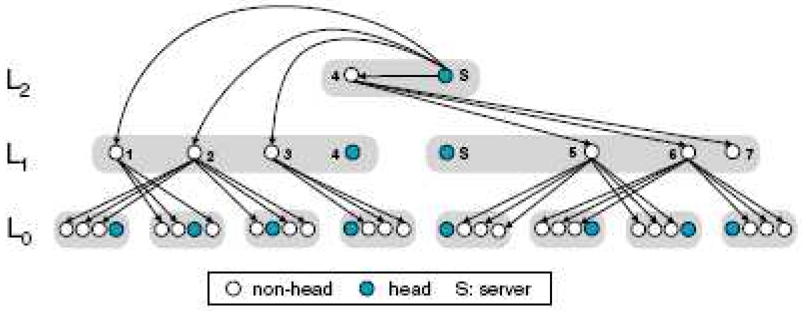

Figure 5 The multicast tree of ZIGZAG

Notice that cluster members do not receive contents from their heads but from heads of other clusters. A peer, when not at its highest layer, cannot have any link to or from any other peer. It can only link to peers which belong to other cluster at the lower layer, as one may see in the figure. This mapping structure is one of the major contributions of ZIGZAG.

This paper also proved several theorems, including the worst case node degree, the height of the multicast tree and other worst case control overhead. Similar to our consideration that node degree must be bounded, ZIGZAG limits its worst case node degree to be O (k^2) where k is a constant. On the other hand, letting the height of multicast tree to be logarithmic is also a constraint to transmission path length, which proved to be O ($\log_k N$) where N is the number of peers.

2.4 Chainsaw

Chainsaw [13] is a request-response based high bandwidth data dissemination protocol drawing upon gossip-based protocols and BitTorrent. The source node, called a seed, generates a series of new packets with monotonically increasing sequence numbers.

Peers in Chainsaw request data from others and thus make Chainsaw a Mesh-pull approach. Peers maintain the states of their neighbors. The majority of the information they posses is a list of packets that each neighbor has. Peers are notified of new packets by their neighbors and must explicitly request a packet from a neighbor in order to receive it. Every peer maintains a window of interest, which is the range of sequence numbers of packets that the peer is interested in acquiring at current time. It also maintains and informs its neighbors about a window of availability, which is the range of packets that it is willing to upload to its neighbors. The window of availability will typically be larger than the window of interest.

A peer keeps track of what packets it has requested from every neighbor and ensures that it does not request the same packet multiple times. It also limits the number of requests to some given neighbors to ensure balanced member load.

2.5 Splitstream

In Splitstream [2], which is a Tree-push approach, data is divided into several disjoint sections called stripes. To do the dissemination, one tree is built for each stripe. So, in order to receive the complete stream, a node must join every multicast tree.

Every node plays the role of an interior node in exactly one tree. Therefore it ensures that each node will never upload more than it receives. Figure 6 depicts how Splitstream transmits video stripes through multiple trees. Data is split into two stripes. An independent multicasting tree is built for each stripe.

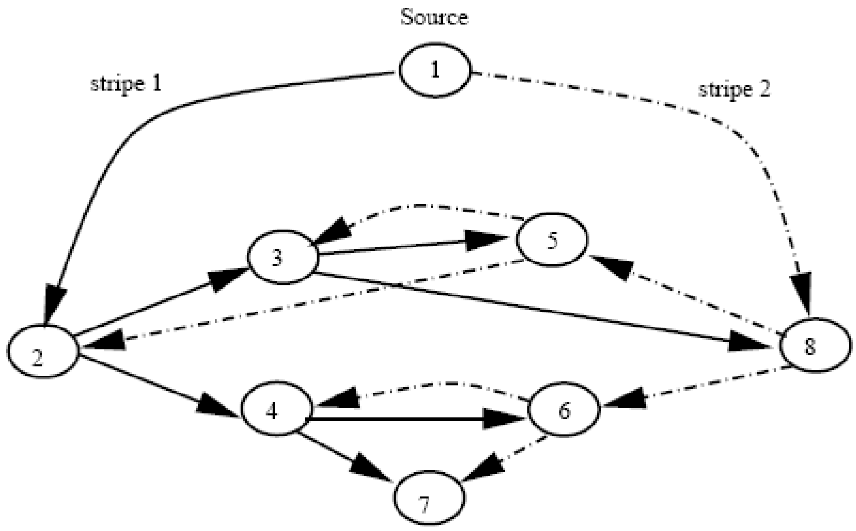

Figure 6 Example of Splitstream

Furthermore, each node is only responsible for data forwarding on one of the stripes. If a node suddenly leaves the system, at most one stripe is affected. Therefore, robustness is also improved.

2.6 ESM

The ESM system [12] constructs a Tree-push based overlay which is distributed, self organizing, performance-aware. The tree is optimized primarily for bandwidth, and secondarily for delay.

Each ESM node maintains information of members which are randomly selected as a subset. If a node wants to join the broadcast, it retrieves random members participating in the broadcast from the source. Nodes update its information about other members in periodically.

Each ESM node maintains the application-level throughput it is receiving in a recent time window. If its performance is significantly below the source rate, then it selects a new parent to download data from. ESM employs a default detection time of 5 seconds, which means it takes 5 seconds at most for a node to switch to a new parent after it detects a low performance. But the protocol running on transmission path influences the choice of this value since switching to a new parent requires going through a slow-start phase, which may take 1 to 2 seconds to reach the full data rate.

2.7 Spanning Tree for Data Collection

In [16], authors analyzed several spanning tree topologies. Traditional spanning tree topologies were examined and compared. In a shortest path spanning tree, the distance, say, the total weight of the path, from the root node to all other nodes is minimized. Such a tree is easily constructed by Dijkstra's algorithm, denoted as SP. A fewest hop spanning tree that minimizes the number of hops along the path from each node to the root node is denoted as FH. The other typical problem is the minimum spanning tree problem, which minimizes the total sum of edge weights and can be constructed by Prim's or Kruskal algorithm and is denoted as MST.

The spanning trees created by FH tend to be shallow and fat with the average node degree being fairly large. On the other hand, FH minimizes the data loss when a node or link fails. MST grows deeply and skinny because its only criterion is the total edge weight. SP tends to have smaller node degrees and grow deeper than FH depending on the edge weight. Figure 7 shows all four spanning trees.

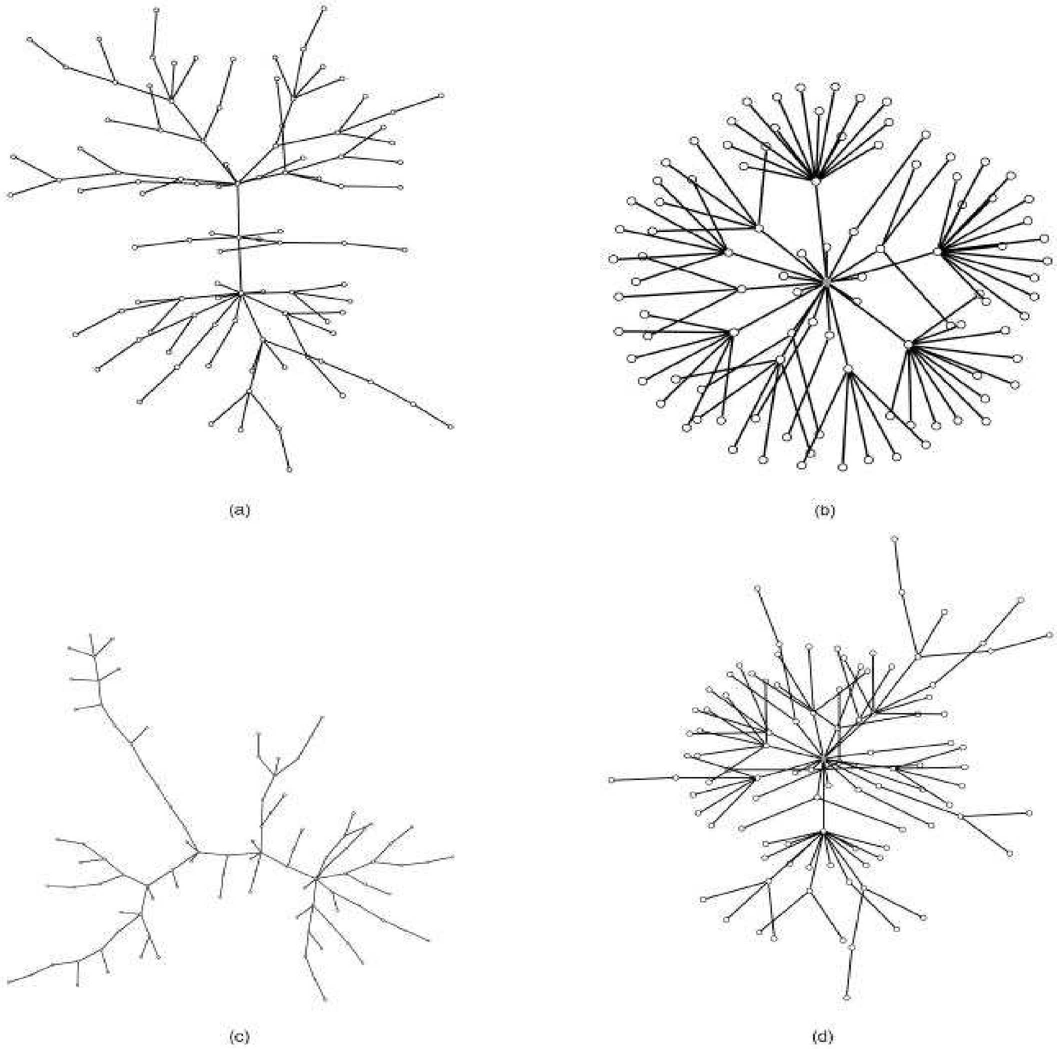

Figure 7(a) SP (b) FH (c) MST (d) RB

They then propose a RoBust spanning tree (RB) that combines the nature of spanning tree algorithms mentioned above. By a linear combination of input parameters such as hop count and path weight as follows.

$\lambda \times$ *hop count* + *(1* $-\lambda \times$ *path weight),* where $\lambda_i = 1 - \frac{h_i}{\epsilon_1}$, $0 \leq \lambda \leq 1$.

h_i represents the hop count and ϵ_1 stands for the eccentricity of a node. The eccentricity of the node is the largest of the shortest paths from that node to all other nodes.

They proposed two implementation schemes to build the proposed tree:

centralized and distributed one. Figure 7 (d) is the product of centralized scheme. It clearly has the characteristics of different algorithms. Experiment results show that RB performs better in many aspects.

2.8 Spanning Tree Problems with Constraints

2.8.1 Bounded Diameter Spanning Tree

Bound-Diameter Spanning Tree and related problems are discussed in [3]. Diameter is defined as the longest path from an arbitrary node to any other destination.

BDST problem is defined as: Given a graph G (V, E), weight $W(e) \in \mathbb{Z}^+$, for each e ∈ E, positive integer D ≤ |V|, positive integer B, find a spanning tree T for G such that the sum of the weights of the edges in T does not exceed B and such that T contains no simple path with more than D edges. It remains to be NP-complete for any fixed D ≥ 4, even if all edge weights are either 1 or 2. It can be solved easily in polynomial time if D ≤ 3, or if all edge weights are equal.

BDST can be the base of our problem model if we replace its link weight by the parameters we interested in. Since the number of peers in a peer-to-peer network will mostly be large, our modeling should be of NP-complete complexity as well.

2.8.2 Minimum Diameter Spanning Tree

Consider a Euclidean graph that every link of the graph has a weight. Instead of minimizing total cost of the spanning tree, Minimum-Diameter Spanning Tree tends to minimize the diameter of the tree.

MDST (Minimum-Diameter Spanning Tree) problem is described in [3] and [6]: Given a graph G = (V, E) and a cost function $W(e) \in \mathbb{Z}^+$, for all e ∈ E, find a spanning tree T for G, such that $MAX_{\text{simple path } p \in T} \sum_{e \in p} W(e)$ is minimized. This problem can be solved in polynomial time O (n^3).

In [1], it proposed a distributed algorithm to find a MDST. The link weights were defined in positive value. The main contribution of this algorithm was that it achieved an efficient time complexity of O (n) and meantime an O (*nm* log *n* + *nm* log *W*) bits communication complexity, where *W* was the largest link weight in the network, *n* was the number of vertices and *m* was the number of edges.

2.8.3 Spanning Tree with Bounded Node Degree

Finding a Minimum Bounded Degree spanning tree is also a NP-Complete problem [3]:

Given a graph G (V, E), positive integer $K \leq |V|$, a cost function $c: E \rightarrow R$, find a spanning tree of minimum cost for G in which no vertex has a degree larger than *K*. This problem remains NP-complete for any fixed $K \geq 2$.

In [14], it proposed a polynomial time algorithm that returns a spanning tree of minimum cost and bounded node degree. Furthermore, it not only set an upper bound on node degree but also a lower bound on node degree. This property harmed the data quality when lower bound is large. Although this paper provided a polynomial time algorithm for this problem, it is not adequate for our model because of different objective.

2.9 Analysis of Spanning Tree-Push Solution

Tree-push based systems [8] have superior advantages to mesh-pull based systems for delay consideration. Once a tree is built, the routing of transmission is fixed such that there is no need for peers to communicate with each other for data sharing. This characteristic is important for the system to meet the stringent delay constraint.

However, traditional spanning tree algorithms have some structural characteristics that may become obstacles on the way to reach our objective. Different spanning tree algorithm constructs trees with different criterions and thus forms unique structure characteristics. The typical two spanning tree algorithms are minimum spanning tree and shortest-path spanning tree.

A minimum spanning tree has a minimum total cost. It often has a long tail in the tree. A long tail may cause large hop count and long delay if link weight is link delay. Long delay is an enemy in a streaming service because it significantly increases the waiting time on user end, or even increases the number of dropped packets which exceeds playback point, and thus reduces the quality of service. Note that the startup delay is also included in the metrics of quality of service.

As for single-source shortest path algorithm, which also generates spanning trees, their objective is to find a shortest path from the source to all other nodes. The node degree in the resulting spanning tree is unbounded. Large node degree will increase the processing time within a node and thus increase total delay.

III、Proposed Approach

3.1 MLDST (Minimum virtual Loss Diameter Spanning Tree)

In order to reduce the damage associated with the long transmission path, the objective of this research is to find a multicast tree for a given peer-to-peer IPTV network that demands quality of service. Specifically speaking, we want to build a spanning tree which has minimum data loss rate under several constraints.

“Diameter” of a spanning tree is defined as “the longest path from the root to all other nodes”. Delay diameter is the diameter of a spanning tree when link weight is delay. In other words, the delay diameter is the longest total delay time among all possible paths from the root to all other nodes. Likewise, loss diameter is the diameter of a spanning tree when link weight is loss rate. In other words, the loss diameter is the loss rate of the node that receives the least amount of data. However, since loss rate is not addable but multipliable in nature that makes the computation a very complicated task for a graph based algorithm, we further define the following two terms to represent real Loss Diameter in order to simplify algorithmic computation. *Virtual path loss* is simply the summation of packet loss rates of all links in a path, while *virtual Loss Diameter* is the largest virtual path loss in a tree.

Please note that the real packet loss rate of a path should be $1-\prod(1-p_{i,j})$, $\forall$ i, j $\in$ T, we use direct summation of loss rate to simplify the model hoping its solution may lead to a good solution.

By taking one parameter as our objective and other two as the constraints, we propose a new spanning tree construction model.

Although a distributed model is more appreciate to construct a multicasting tree in real network, we model the problem in centralized fashion at current stage. The problem will be extended to distributed version only after we gain a better understanding on the centralized version.

The objective function of MLDST is defined below:

Given a graph G (V, E), where V = {v_1, v_2, v_3,…v_n} is the set of user nodes, E = {$e_{i,j}$ | v_i, v_j ∈ V} is the set of possible interconnections between pairs of nodes, we define $d_{i,j}$ be the delay time spent on $e_{i,j}$ and $p_{i,j}$ be the the packet loss rate on $e_{i,j}$. Thus, for each edge $e_{i,j}$, we have a two-attributes weight for link $e_{i,j}$, ($d_{i,j}$, $p_{i,j}$). Tree T is a *k*-nary spanning tree rooted at v_1 with respect to G.

Define: *Delay Diameter*: $Maximum_{\text{simple path } S \in T} \sum_{e_{i,j} \in S} d_{i,j}$

virtual Loss Diameter: $Maximum_{\text{simple path } S \in T} \sum_{e_{i,j} \in S} p_{i,j}$

Delay Diameter = $Maximum_{\text{simple path } S \in T} \sum_{e_{i,j} \in S} d_{i,j}$, is defined as the maximal accumulated delay from v_1 to any other node. *Virtual Loss Diameter* = $Maximum_{\text{simple path } S \in T} \sum_{e_{i,j} \in S} p_{i,j}$, is defined as the maximal accumulated packet loss rate of any simple path rooted at v_1.

The optimization function is defined as follows:

Given G(V, E), delay bound (*D*) and degree bound (*b*), find a spanning tree T rooted at v_1, such that:

virtual Loss Diameter is minimized, while *Delay Diameter* < *D* and node degree < *b*.

Our objective is to minimize the virtual Loss Diameter in a spanning tree while Delay Diameter and node degree are both bounded. In MLDST, virtual Loss Diameter is just taken as an index to simplify the model for good MLDST.

Delay Diameter represents the largest accumulated delay from root to any other nodes. By bounding it, we can assure the worst case transmission delay is under control. On the other hand, the bound of Delay Diameter can also determine the size of MLDST, since the larger the bound is defined, the longer a single path could be.

3.2 NP-Completeness of MLDST

(A) MLDST is in NP:

We first show that MLDST∈ NP. Assuming that we are given a graph $G(V, E)$, two parameters on each edge, say, delay d and packet loss rate p, and two predefined bound $D>3$ and B. There is a k-nary spanning tree T given, where maximum $d_{1,m} \leq D$, $\forall\ m \in T$ and $k \leq b$. Then we verify this instance by checking if maximum $\sum p_{1,m}$, $\forall\ m \in T$ is the minimum amount all possible solutions. The verification algorithm performs in polynomial time.

(B) MLDST is NP-Complete:

As we illustrated in Section 2.8.3, a BDST (Bounded Diameter Spanning Tree) problem is a NP-Complete problem if node degree is greater than 3. We can BDST to MLDST straightforwardly. Let graph G (V, E), edge weight $W = \{ w_{ij} \mid v_i, v_j \in V \}$, a total weight bound C' and a diameter bound B' be a valid instance of BDST, we construct the corresponding instances G of MLDST as follows. We let $V=V'$, $E=E'$, $B=B'$ and a large node degree bound $D' = |V|$, as well as $p_{ij}=d_{ij}=w_{ij}$, for all edges. We can easily prove by contraction that an optimal solution g to G with a minimum virtual Loss Diameter x must be a solution to G'. First, we can see that the diameter bound B' must be satisfied. Next, g must be smaller or equal to total weight bound C. Otherwise, we can find another solution g' to G' with a total weight $y \leq C < x$. In that case, we can use g' to solve G to obtain a solution with a virtual Loss Diameter y, which is a contradiction.

From (A) and (B), we can say that BDST can be reduced to MLDST. As a result, we prove that MLDST is a NP-Complete problem if node degree bound is greater than 3.

3.3 Design Concepts and Objective

According to the problem model describing in previous sections, we designed a heuristic solution for MLDST. Since our objective is to minimize the virtual Loss Diameter, we prefer a single-source shortest-path algorithm rather than a minimum spanning tree algorithm, whose objective is to minimize the total cost of the tree, which may create large diameter paths.

Our heuristic algorithm follows Dijkstra's algorithm's footstep. We modify Dijkstra's algorithm by bounding Delay Diameter and node degree and searching for a spanning tree which has a minimum virtual Loss Diameter.

The issues in distributing environment, such as peer churn and membership change, are left behind in proposed solution. Currently, we only focus on the centralized version for the purpose of proof of concept.

3.4 Heuristic MLDST

Our heuristic algorithm is quite simple and easy to understand. Every edge has two network parameters, delay and packet loss rate. While executing a Dijkstra's algorithm, total delay and total packet loss rate of each intermediate path is calculated. We modified the original Dijkstra's algorithm so that Delay Diameter and the degree of each node will be constantly examined to meet the constraints.

If a node exceeds the degree limit, the link with high loss rate will first be abandoned. Priority is given to loss rate, instead of delay. That is, we disconnect links with higher loss rate prior to the ones with longer delay. Once a link is disconnected, data must be rerouted to downstream peers.

The resulting MLDST is only responsible for transmitting one piece of data. To receive complete data, many MLDSTs must be constructed for each piece of data. These multicasting trees need not to be fully disjoined.

Notice that data partitioning is not our concern.

3.5 Pseudo Code of Heuristic MLDST

```
Heuristic MLDST (G, w, s)
Initialize−Single−Source(G, s)
        do for each edge (u, v) ∈ E[G]
            // use the weight of (u, v) to update current shortest path
            do RELAX(u, v, w)
    for each edge (u, v) ∈ E[G]
        do if d[v] > d[u] + w(u, v)
            then return FALSE
                // check if total delay exceeds the bound
                do if d[v] ≥ Delay Bound
                then return FALSE
                //check if node degree exceeds the degree bound
                do if Degree[v] ≥ Degree Bound
                    then return FALSE
```

```
Return TRUE
```

This algorithm is modified based on Dijkstra’s algorithm by adding two checking processes into the loop. While algorithm is running, it checks if the constraints are both satisfied. The rest part of the algorithm remains the same with original design.

IV、Performance Evaluation

We evaluated our proposal using the real network experimental environment, PlanetLab. By examining the quality of transmitted video data, we evaluated Heuristic MLDST against other tree construction algorithms including minimum spanning tree, shortest-path spanning tree, and ZIGZAG tree. The first two trees were used as baseline performance and ZIGZAG is the competitor of our algorithm.

Experimental data, including PSNR, transmission delay of each packet and total packet loss rate were recorded from the far-end node of Delay Diameter of the tree of all four topologies. It goes without saying that we supposed to extract data from far-end node of virtual Loss Diameter for the worst case of packet loss since we were mainly examining PSNR. In our experimental topologies, these two nodes just happened to be same one.

We calculate PSNR frame-wisely, the average value of frame-wise PSNR, and further examine total packet loss rate and average packet delay. Average packet delay is calculated based on received packets. Total packet loss rate is the proportion of the number of received packets and the number of transmitted packets.

4.1 Experimental Envirement

We first selected high performance peers on PlanetLab and constructed our experimental topology. 100 pinging messages were sent to each of carefully selected peer to measure their performance. Peers with shorter delay were given priority to participate in the experiment. Fifteen peers were selected to participate in the experiment. In MLDST, both delay and packet loss rate were taken into consideration, while MST and SP trees took delay only.

Degree bound was set to 3, while delay bound was set to 1200 ms based on the analysis of measured delay between peers and user tolerance. We used the same set of peers to build all four trees.

Since the characteristics of a real network change constantly, only a snapshot of the network were measured. Therefore, this experiment had its own limits. A dynamic version, which is beyond the scope of this research, will be more appreciate to model the real network. Our experiments were carried out several hours right after the characteristics measurement was performed (i.e. the instantiation of the graph) assuming that the fluctuation of network condition is acceptable.

The measured average delay (ms) and packet loss rate of each pair of selected peers are shown in Table 2. These values were averaged over 100 probes. Spanning trees mentioned in the following sections were based on these parameters.

At the beginning of each experiment, we transmitted via the constructed spanning tree for 30 seconds a MPEG-1 video clip, which has a bit-rate of 1150 Kbps, 29.97 fps and sized 352 x 240.

Table 1 List of Participating Nodes

Node	Domain Name of the Node
1	Planetlab1.csie.nuk.edu.tw
2	Planetlab1.sfc.wide.ad.jp
3	Pub1-s.ane.cmc.osaka-u.ac.jp
4	Planetlab-1.calpoly-netlab.net
5	Planetlab1.utdallas.edu
6	Planetlab1.postel.org
7	Nodea.howard.edu
8	Pl1.csl.utoronto.ca
9	Planetlab1.cs.cornell.edu
10	Planetlab1.georgetown.edu
11	Planetlab1.utep.edu
12	Vn1.cs.wustl.edu
13	Plgmu2.ite.gmu.edu
14	Planetlab2.eecs.northwestern.edu
15	Planetlab1.cs.stevens-tech.edu

Table 2 Measured Delay between pairs of nodes

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1		62.921 0%	52.851 0%	168.371 0%	229.373 0%	155.906 7%	225.173, 0%	249.227 3%	250.927 0%	248.442 0%	243.048 0%	298.532 0%	241.738 0%	226.291 0%	249.703, 1%
2	62.921 0%		13.550 0%	129.115 0%	156.392 0%	238.352 0%	198.364 0%	208.153 8%	198.313 0%	189.622 0%	166.590 0%	210.924 0%	179.413 0%	201.738 0%	199.311 2%
3	52.851, 0%	13.550 0%		133.236 0%	161.906 0%	242.550 0%	200.089 0%	222.582, 4%	207.746 0%	193.160 0%	170.692 0%	229.359 0%	183.490 0%	205.906 0%	200.082, 1%
4	168.371 0%	129.115 0%	133.236 0%		51.709 0%	39.710 0%	87.952 0%	100.219 1%	95.665 0%	84.500 0%	62.024 0%	105.195 1%	74.632 0%	97.153 0%	90.672 1%
5	229.373 0%	156.392 0%	161.906 0%	51.709 0%		38.027 0%	21.031 0%	63.309 3%	50.595 0%	54.414 0%	21.133 0%	76.432 0%	38.966 0%	61.632 0%	55.076 1%
6	155.906 7%	238.352 0%	242.550 0%	39.710 0%	38.027 0%		66.934, 0%	102.996 3%	98.577 0%	96.059 3%	77.291 0%	117.965 0%	75.557 0%	112.538 0%	105.960 1%
7	225.173 0%	198.364 0%	200.089 0%	87.952 0%	21.031 0%	66.934 0%		35.948 6%	16.670 0%	2.863 0%	61.453 0%	73.782 0%	37.021 1%	43.514 0%	35.560 3%
8	249.227 3%	208.153 8%	222.582 4%	100.219 1%	63.309 3%	102.996 3%	35.948 6%		34.650 4%	50.533 2%	66.028 1%	93.811 0%	26.701 3%	63.344 13%	40.121 5%
9	250.927 0%	198.313 0%	207.746 0%	95.665 0%	50.595 0%	98.577 0%	16.670 0%	34.650 4%		15.842 0%	60.841 0%	71.744 0%	30.755 0%	28.830 0%	11.950 4%
10	248.442 0%	189.622 0%	193.160 0%	84.500 0%	54.414 0%	96.059 3%	2.863 0%	50.533, 2%	15.842 0%		57.334 0%	55.044 0%	25.527 0%	18.909, 0%	12.735 3%
11	243.048 0%	166.590 0%	170.692 0%	62.024 0%	21.133 0%	77.291 0%	61.453 0%	66.028 1%	60.841 0%	57.334 0%		94.552 0%	48.132 0%	70.415 0%	64.020 0%
12	298.532 0%	210.924 0%	229.359 0%	105.195 1%	76.432 0%	117.965 0%	73.782 0%	93.811 0%	71.744 0%	55.044 0%	94.552 0%		48.613 1%	78.899 0%	62.460 0%
13	241.738 0%	179.413 0%	183.490 0%	74.632 0%	38.966 0%	75.557 0%	37.021 1%	26.701 3%	30.755 0%	25.527 0%	48.132 0%	48.613 1%		39.227 0%	36.900 0%
14	226.291 0%	201.738 0%	205.956 0%	97.153 0%	61.632 0%	112.538 0%	43.514 0%	65.344 13%	28.830 0%	18.909 0%	70.415 0%	78.899 0%	39.227 0%		25.438 2%
15	249.703 1%	199.311 2%	200.082 1%	90.672 1%	55.076 1%	105.960 1%	35.560 3%	40.121 5%	11.950 4%	12.735 3%	64.020 0%	62.460 0%	36.900 0%	25.438 2%	

4.2 Experiment Results

4.2.1 Topologies

Figure 8 shows the result of running Heuristic MLDST algorithm on the nodes we selected from PlanetLab. Considering the scale of this experiment, we notice that a degree larger than three will overload a single peer. Figure 8 depicts the topology of MLDST, video is transmitted from the source peer “planetlab1.csie.nuk.edu.tw” to every other peer in this graph. In all four quality evaluations, we measure the quality of the video received by the peer with the longest delay. In MLDST, “planetlab2.eecs.northwestern.edu” is the selected node.

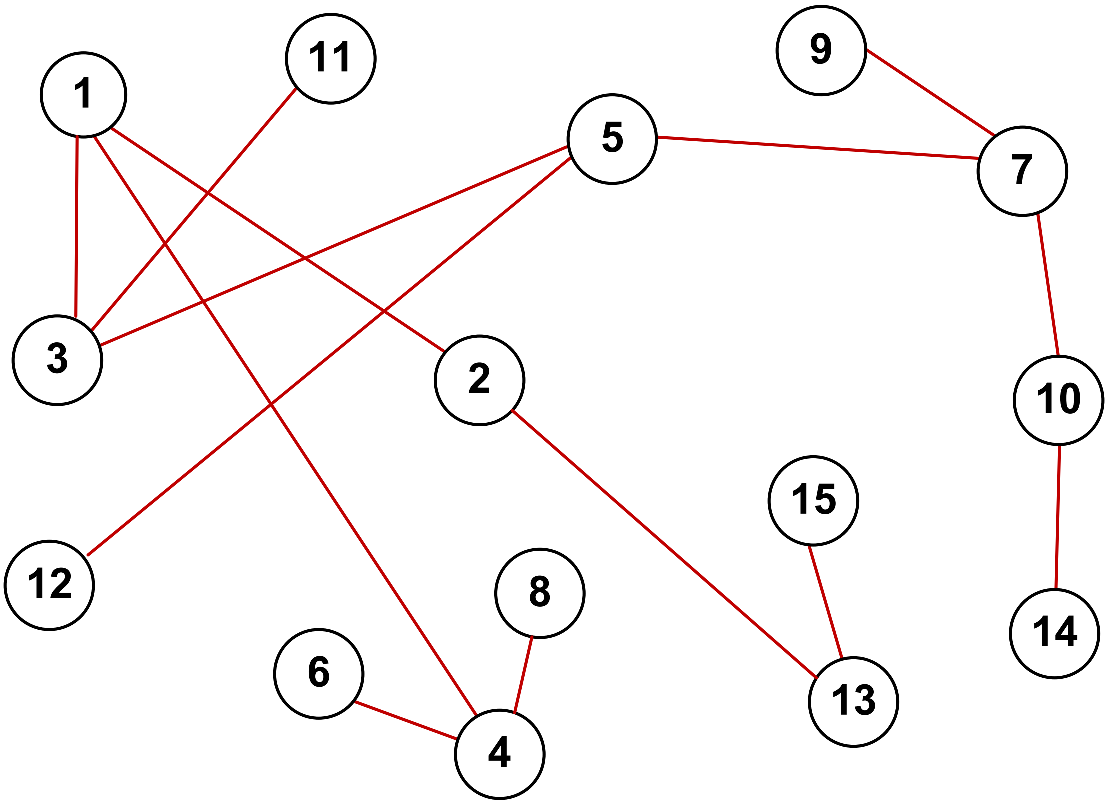

Figure 8 MLDST on PlanetLab

A minimum spanning tree is constructed using Kruskal algorithm in figure 9. As we mentioned above, the only parameter that it adapt is delay. The video is also transmitted from source peer “planetlab1.csie.nuk.edu.tw” to every other peer. At the other end of Delay Diameter, “vn1.cs.wustl.edu” is selected to measure the result.

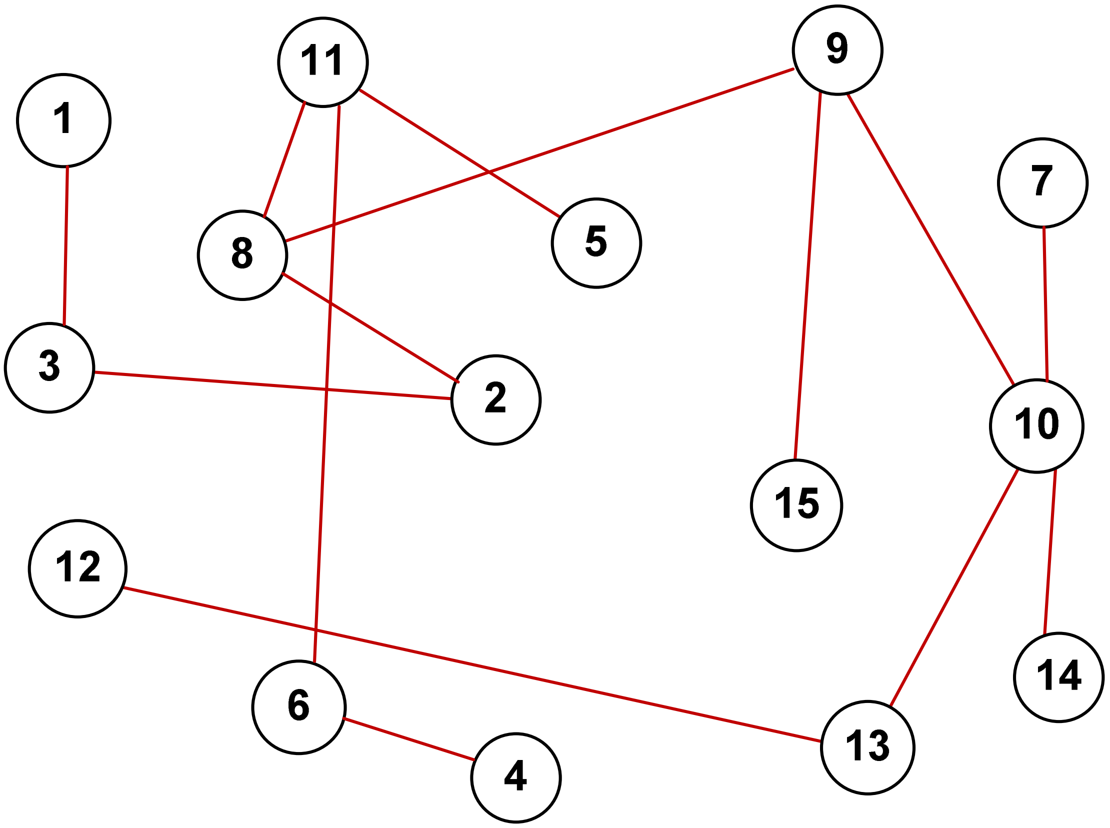

Figure 9 MST on PlanetLab

We construct shortest path spanning tree using original Dijkstra’s algorithm in figure 10. Same as minimum spanning tree topology, shortest path tree only consider delay as link parameter. Video is transmitted from source peer “planetlab1.csie.nuk.edu.tw” to every other peer. At the other end of Delay Diameter, “planetlab2.eecs.northwestern.edu” is selected to measure the result.

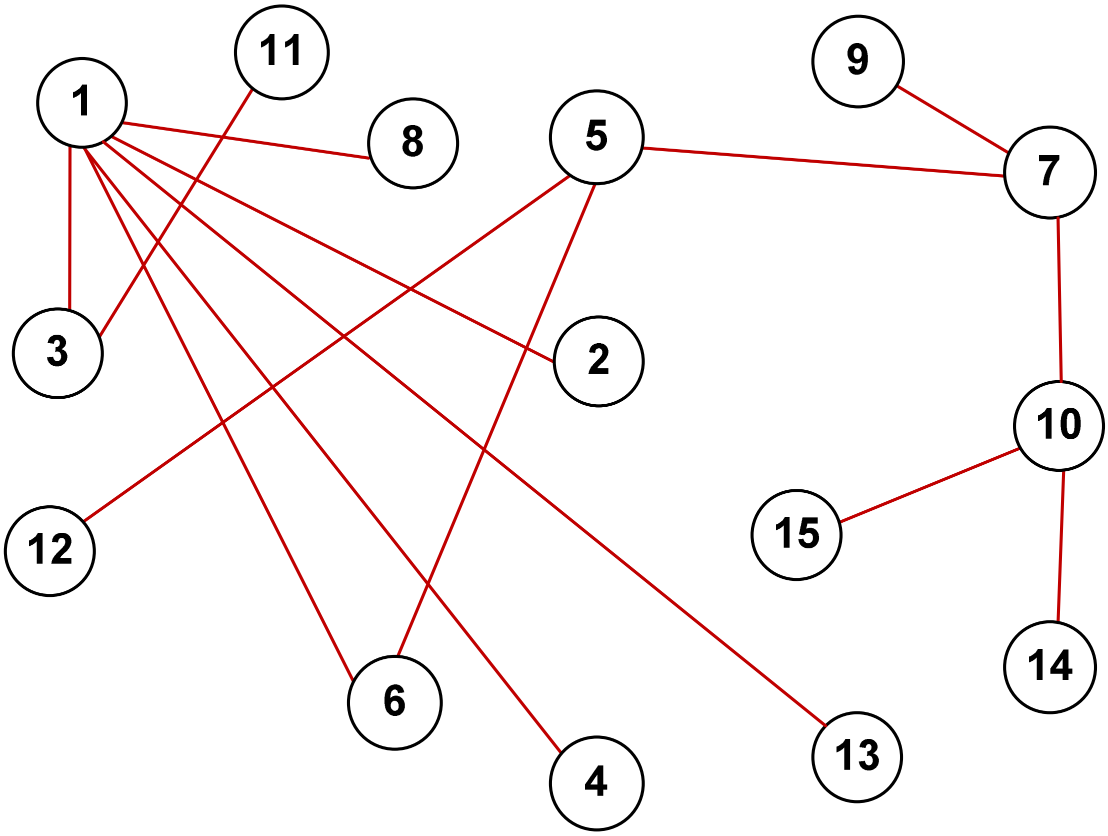

Figure 10 SP tree on PlanetLab

4.2.2 Analysis of Experimental Results

As we can see, MLDST tends to have shorter tail than minimum spanning tree and less number of nodes with outstanding node degree than shortest path tree. Notice that longer path leads to larger delay and higher probability of loss. Furthermore, large node degree leads to large processing delay within those nodes. Moreover, increasing delay hinders the data packets to meet the playback deadline at receivers. Figure 11 shows the PSNR of each transmitted frame.

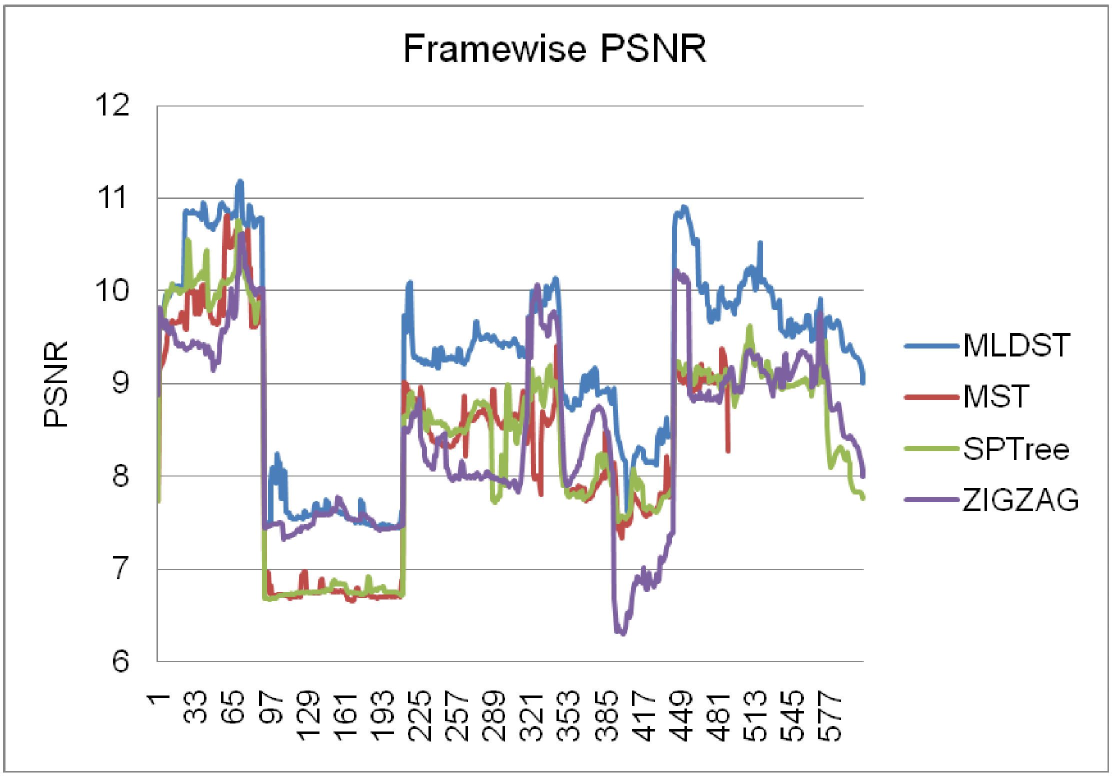

Figure 11 Frame-wise PSNR

Clearly, measured PSNR values of MLDST are about 10% in average higher than those of other tree construction algorithms. From further examination of the result of MST at selected node, we found that the frames of the tail part of the video were all lost during transmission. Notice that the packet loss may be caused by both of network congestion and excessive delay.

Higher PSNR represents better video quality at user end. In summary, MLDST can provide good video quality while total transmission delay is bounded to users' satisfaction.

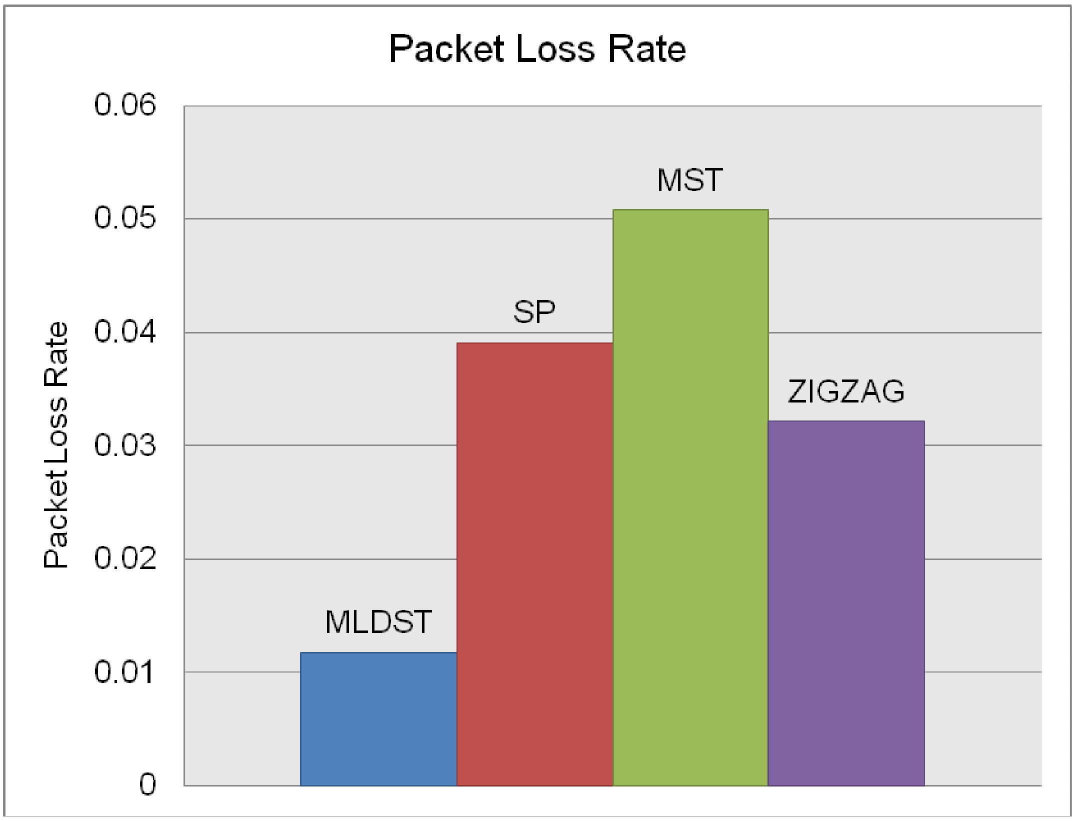

Figure 12 Packet loss rate measured at the end node of virtual Loss Diameter

Figure 12 shows the packet loss rate of each tree. We can see clearly that MLDST shows less packet loss than others and thus can provide higher PSNR as illustrated in Figure 11.

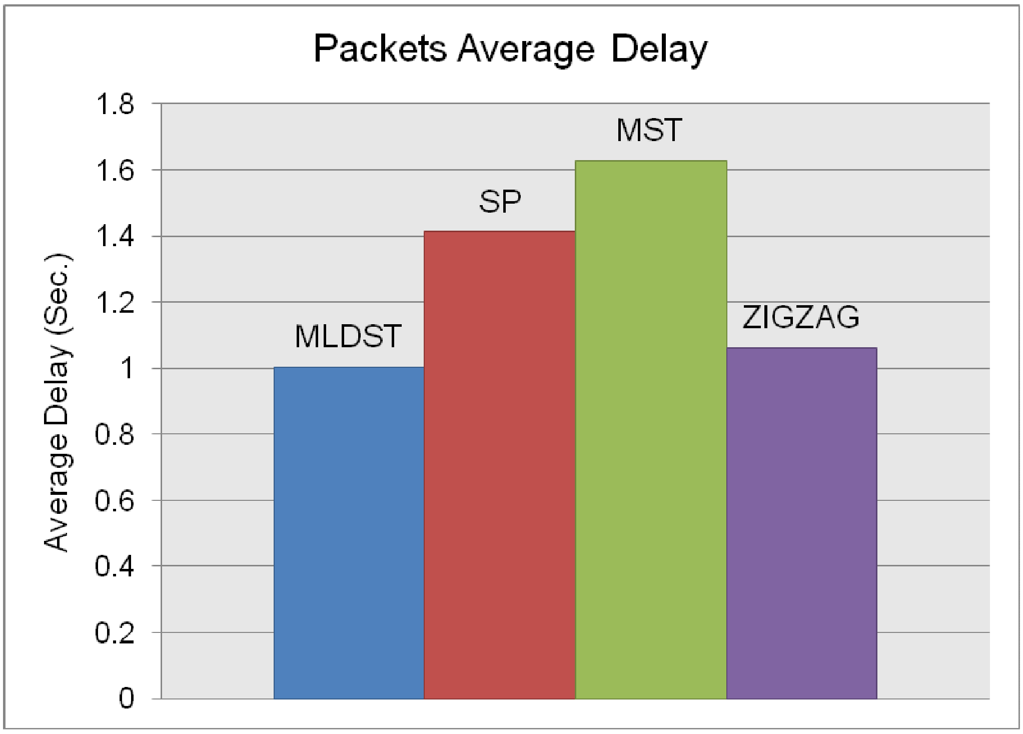

Figure 13 Average packet delay measured at the end node of Delay Diameter

Figure 13 depicts the average delay of each tree at the far end node of the Delay Diameter. Our solution has lowest accumulated delay due to the balanced node degree and the delay constraints.

Since we only take delay as a constraint rather than an objective, a minor superiority in delay is acceptable.

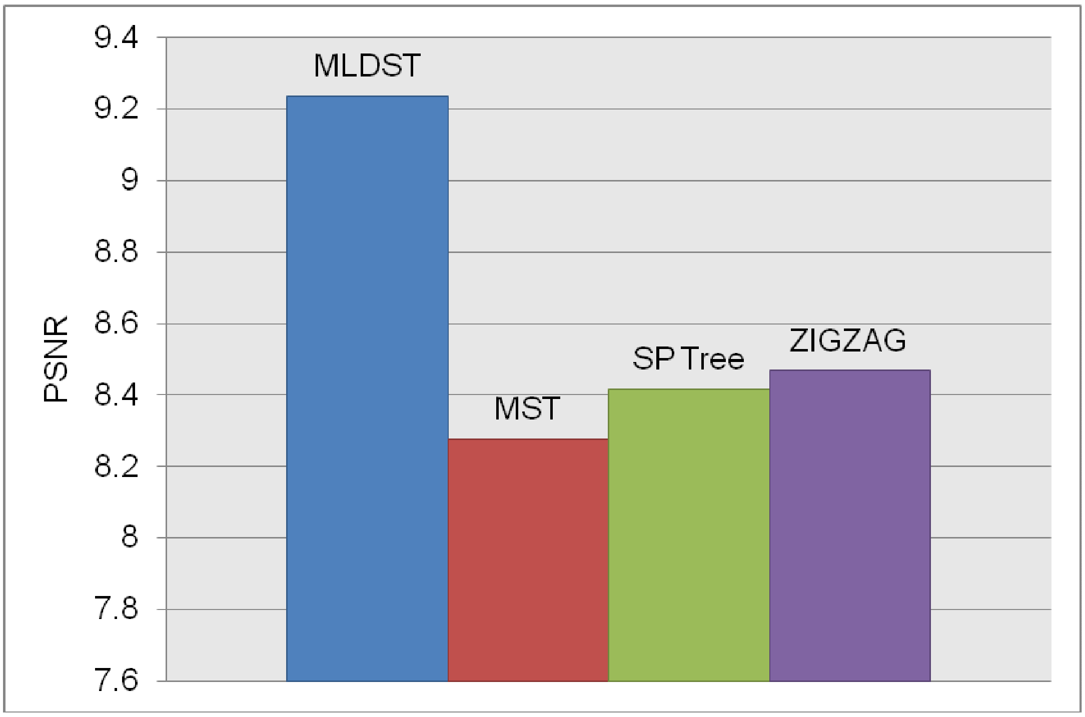

Figure 14 Average frame-wise PSNR

Figure 14 shows average PSNR measured at the far-end node of Delay Diameter of all trees. The value was calculated from dividing the sum of frame-wise PSNR by the total number of frames. MLDST had higher value than others.

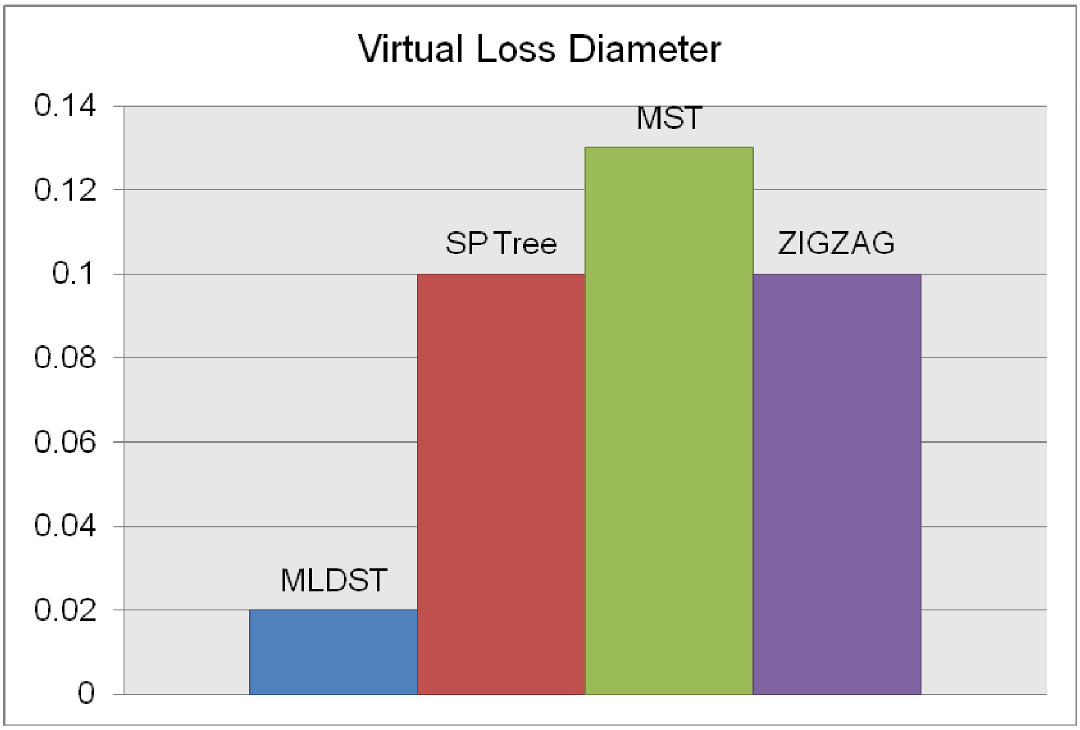

Figure 15 Virtual Loss Diameter

Since we avoid selecting links with high packet loss rate into the multicast tree, the loss diameter of MLDST is explicitly much smaller than others. The result showed in figure 15 further explains why our solution has better data quality.

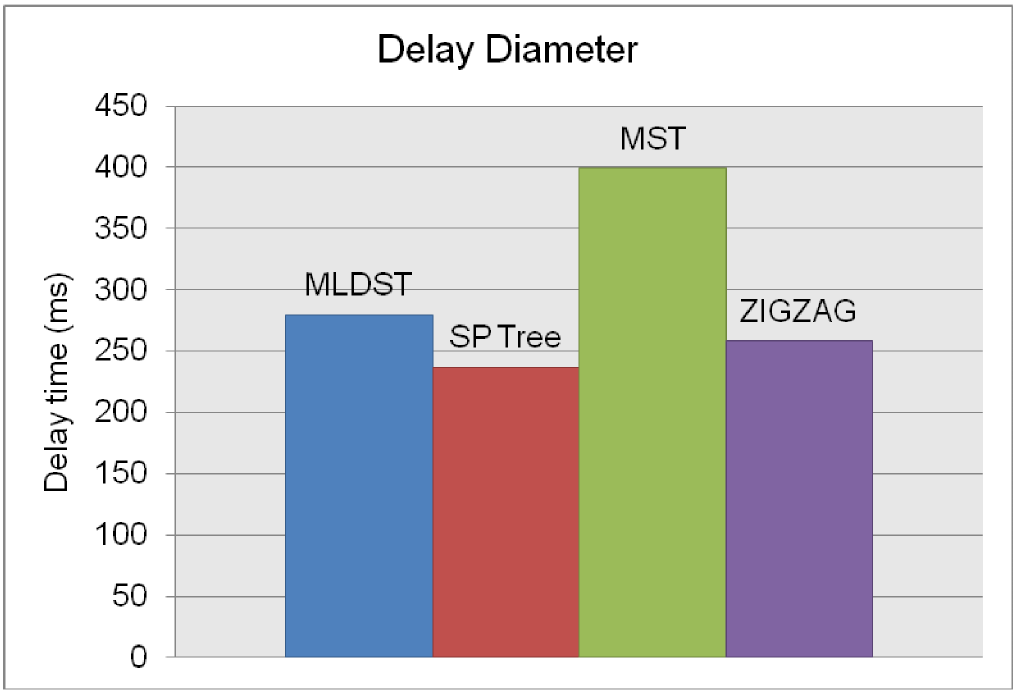

Figure 16 Delay Diameter

Figure 16 shows the Delay Diameter of each participating tree. Note that we take delay as a constraint, higher delay is allowed, as long as the value does not exceed the bound.

V、Conclusion

Streaming services has been becoming more and more popular in recent years. The reason can be traced back to the explosive growth of Internet. Along with the prolification of streaming services, many issues showed up, such as the poor quality of service, long transmission delay and frequent disconnection. Many solutions have already been proposed to resolve these issues. They can be roughly classified into two categories, say, Mesh-pull and Tree-push. Mesh-pull systems divide transmitting video into clips. User nodes send requests to neighbors and download video clips from them if positive responses are received. Overhead and extra delays are incurred from large amount of control messages. As a result, Mesh-pull systems seem not appropriate for those services that demand short transmission delay.

Hereby we choose to take Tree-push based approach to model the problem. It avoids the heavy traffic of control messages and thus reduces transmission delay. We model the problem into a MLDST that minimize the worst case packet loss rate while bound delay and node degree. This problem is then proved to be NP-Complete. We propose Heuristic MLDST by modifying the single-source shortest-path algorithm, Dijkstra’s algorithm, by bounding the Delay Diameter and node degree. Proposed solution depicts a spanning tree with minimum virtual Loss Diameter under delay and node degree constraints. Through several experiment evaluations on PlanetLab, we showed that our solution outperforms other tree construction algorithms in video quality.

The issue of peer churn and membership changes is ignored in our research and will be studied in the future.

Reference

[1] Marc Bui, Franck Butelle, and Christian Lavault, "A Distributed Algorithm for Constructing a Minimum Diameter Spanning Tree", Journal of Parallel and Distributed Computing, Vol. 64, No. 5, May, 2004, pp. 571-577.

[2] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, Atul Singh, "SplitStream: High-Bandwidth Multicast in Cooperative

Environments", Proceedings of the 19th ACM Symposium on Operating Systems Principles, 2003, pp. 298-313.

[3] M. R. GAREY and D. S. JOHNSON, W. H. Freeman, "Computers and intractability: A guide to the theory of NP-completeness", W. H. Freeman and Company, 1979.

[4] Yu-Hsuang Guo, John K. Zao, Wen-Hsiao Peng, Lin-Shung Huang, Fang-Po Kuo, Che-Min Lin, "Trickle: Resilient Real-Time Video Multicasting for Dynamic Peers with Limited or Asymmetric Network Connectivity", Proceedings of 8th IEEE International Symposium on Multimedia, December, 2006.

[5] Jan-Ming Ho, D.T. Lee, Chia-Hsiang Chang, and C.K. Wong, "Bounded-Diameter Minimum Diameter Spanning Trees and Related Problems", Proceedings of the 5th Annual Symposium on Computational Geometry, 1989, pp. 276-282.

[6] Jan-Ming Ho, D. T. Lee, Chia-Hsiang Chang and C. K. Wong, "Minimum Diameter Spanning Trees and Related Problems", SIAM Journal of Computing, Vol. 20, No. 5, 1991, pp. 987-997.

[7] Xiao-Jun Hei, Yong Liu, Keith W. Ross, "IPTV over P2P streaming networks: The mesh-pull approach", IEEE Communications Magazine, Vol. 46, No. 2, February 2008, pp. 86-92.

[8] Hung-Chang Hsiao and Chih-Peng He, "A Tree-Based Peer-to-Peer Network with Quality Guarantees", IEEE Transactions on Parallel and Distributed Systems, Vol. 19, No. 8, August 2008, pp. 1099-1110.

[9] X. Jia, "A Distributed Algorithm of Delay Bounded Multicast Routing for Multimedia Applications in Wide Area Networks", Proceedings of 6th International Conference on Computer Communications and Networks, September, 1997, pp. 208-213.

[10] K. Kalapriya, R. Venkatesh Babu, S. K. Nandy, "Streaming Stored Playback

Video Over A Peer-to-Peer Network", Proceedings of IEEE International Conference on Communications, Paris, Vol. 3, June 2004, pp. 1298-1302.

[11] Bo Li, Hao Yin, "Peer-to-Peer Live Video Streaming on the Internet: Issues, Existing Approaches, and Challenges", IEEE Communications Magazine, Vol. 45, No. 6, June, 2007, pp. 94-99.

[12] Jiang-Chuan Liu, Sanjay G. Rao, Bo Li and Hui Zhang, "Opportunities and Challenges of Peer-to-Peer Internet Video Broadcast", Proceedings of the IEEE, Vol. 96, No. 1, January, 2008, pp. 11-24.

[13] Vinay Pai, Kapil Kumar, Tamilmani, Vinay Sambamurthy, Alexander E. Mohr, "Chainsaw: Eliminating Trees from Overlay Multicast", Proceedings of 4th International Workshop on Peer-to-Peer Systems, February, 2005.

[14] Mohit Singh, Lap Chi Lau, "Approximating Minimum Bounded Degree Spanning Trees to within One of Optimal", Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June, 2007, pp.661-670.

[15] Duc A. Tran, Kien A. Hua, Tai Do, "ZIGZAG: An efficient peer-to-peer scheme for media streaming", Proceedings of 22nd IEEE INFOCOM Conference, Vol. 2, 2003, pp. 1283-1292.

[16] DEB Veeravalli, JB Weissman, "A Robust Spanning Tree Topology for Data Collection and Dissemination in Distributed Environments", IEEE Transactions on Parallel and Distributed Systems, Vol. 18, No. 5, May, 2007, pp. 608-620.

[17] Vidhyashankar Venkataraman, Kaouru Yoshida, Paul Francis, "Chunkyspread: Heterogeneous Unstructured Tree-Based Peer-to-Peer Multicast", Proceedings of the 14th IEEE International Conference on Network Protocols, 2006, pp. 2-11.

[18] Aggelos Vlavianos, Marios Iliofotous and Michalis Faloutsos, "BiTos: Enhancing BitTorrent for Supporting Streaming Applications", Proceedings of 25th IEEE

International Conference on Computer Communications, 2006, pp. 1-6.