

國立交通大學

網路工程研究所

碩 士 論 文

基於DHT應用層群播之可信賴

同儕式多串流機制



Dependable Peer-to-Peer Multi-Streaming Using

DHT-based Application Level Multicast

研 究 生：何韋呈

指 導 教 授：王國禎 教授

中 華 民 國 九 十 七 年 六 月

基於 DHT 應用層群播之可信賴同儕式多串流機制

Dependable Peer-to-Peer Multi-Streaming Using
DHT-based Application Level Multicast

研究生：何韋呈

Student：Wei-Cheng He

指導教授：王國禎

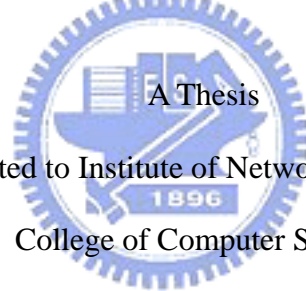
Advisor：Kuo Chen Wang

國立交通大學

資訊學院

網路工程研究所

碩士論文



Submitted to Institute of Network Engineering

College of Computer Science

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2007

Hsinchu, Taiwan, Republic of China

中華民國九十七年六月

基於DHT應用層群播之可信賴

同儕式多串流機制

學生：何韋呈 指導教授：王國禎 博士

國立交通大學 資訊學院 網路工程研究所



近年來，同儕網路上的影音串流應用已愈來愈普及。然而，此類網路高度不穩定的節點狀況仍舊存在，此乃是導至傳輸率下降與接收端影音串流品質不穩定的重要因素。本論文中，我們提出一個在同儕網路上以多串流來傳送封包的機制，稱為 ConStream，以改善這個問題。在這個機制中，整個待傳的內容將被看成由多個 multi-stripe 所組成，並且每個 multi-stripe 將被配給多棵群播樹來同時傳送串流封包。在這些串流中，我們會加入校驗封包 (parity packets)，以使得接收端能夠回復傳送過程所遺失的封包。除此之外，在我們所提出的 ConStream 方法，每個串流的傳輸速度(delivery rate)相對於其它方法的單一串流將能有效的減半，每

個節點對於串流傳送的負擔也將因此降低。模擬結果顯示，我們的方法相較於 Scribe-PRM 與 SplitStream，在不同程度的節點斷線率 (MTTFs, mean time to failures) 下，皆具有較高的傳輸率 (delivery ratio)。



Dependable Peer-to-Peer Multi-Streaming Using DHT-based Application Level Multicast

Student: Wei-Cheng He Advisor: Dr. Kuochen Wang

Institute of Computer Science and Engineering
National Chiao Tung University

Abstract

Streaming applications in P2P networks are more and more popular recently. However, high degree of churning in peer populations is a common problem, and it would result in a low delivery ratio and instable quality of received multimedia. In this thesis, we propose a P2P multi-streaming scheme, called *ConStream*, to improve this problem. In our scheme, the whole content is composed of *multi-stripes*, and each of the multi-stripes will be distributed to multiple multicast trees and be forwarded concurrently. Also, we insert parity packets into streams so that clients can recover lost data. Furthermore, the delivery rate of each stream in our *ConStream* is only half of the original single streaming. That is, the burden of each node is lower than those nodes using single streaming to forward. Simulation results show that the proposed *ConStream* has a higher delivery ratio than Scribe-PRM and SplitStream under various MTTFs (mean time to failures).

Acknowledgements

Many people have helped me with this thesis. I deeply appreciate my thesis advisor, Dr. Kuo-chen Wang, for his intensive advice and instruction. I would like to thank all the classmates in *Mobile Computing and Broadband Networking Laboratory (MBL)* for their invaluable assistance and suggestions. The support by the National Science Council under Grants NSC96-2628-E-009-140-MY3 and NSC96-2628-E-002-138-MY3 is also gratefully acknowledged.

Finally, I thank my family for their endless love and support.



Contents

Abstract (in Chinese)

Abstract (in English)

Acknowledgementsiv

List of Figuresvii

List of Tablesviii

Chapter 1 Introduction 1

Chapter 2 Background & Related Work.....3

2.1 Distributed Hash Tables (DHTs) 3

2.2 IP Multicast and Application Level Multicast..... 5

2.3 Streaming Algorithms..... 6

Chapter 3 Design Approach..... 11

Chapter 4 Simulation Results and Discussion..... 18

4.1 Simulation Scenarios 18

4.2 Simulation Results..... 19



Chapter 5 Conclusion24

5.1 Concluding Remarks24

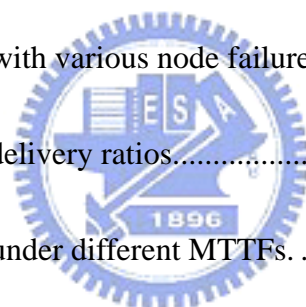
5.2 Future Work24

Bibliography.....25



List of Figures

Figure 1. Architecture of P2P multimedia streaming based on DHT.....	10
Figure 2. Concept of stripe and multi-stripe.	12
Figure 3. Deliver the content via stripes.	13
Figure 4. Deliver the content via multi-stripe.	13
Figure 5. Concurrent streams during the time interval of multi-stripe.	14
Figure 6. Conception of indegree and forwarding capacity.	16
Figure 7. Delivery ratios with various node failure rates.....	21
Figure 8. Distribution of delivery ratios.....	21
Figure 9. Delivery ratios under different MTTFs.	22
Figure 10. Variation of absolute delays among received packets.	23
Figure 11. Absolute delays by each streaming.....	23



List of Tables

Table 1. Comparison of related P2P streaming algorithms.....9



Chapter 1

Introduction

File sharing in peer-to-peer (P2P) today is well known, and several applications such as eMule, BT, etc., are using P2P technology. Multimedia streaming using P2P software like PPStream is targeted for P2P applications where contents must be delivered in a real-time manner. Real-time P2P applications, such as P2P live-media, have higher constraints than P2P file sharing applications, because the content requested by clients must be delivered and played simultaneously. There are several typical algorithms used in multimedia streaming for real time delivering, and we will introduce them in Chapter 2.



Multicast is an efficient mechanism to support group communication applications. It decouples the size of the receiver set from the amount of states kept at any single node and potentially avoids redundant communications in the network. Multicast mechanisms can be classified into *network-layer multicast* and *application-layer multicast*. One of the most important challenges of application layer multicast is its ability to handle high degree of transiency inherent to their environments. A good indication of transiency is the peers' median *session time*, where session time is defined as the time between when a peer joins and leaves the

network [23].

In this thesis we will propose a multi-streaming scheme based on the structure of SplitStream [13], a typical DHT-based cooperative multicast system. In the DHT-based system, we can distribute the forwarding load among all participants and have low system dependency on any particular node.

The rest of this thesis is structured as follows. In Chapter 2, we will introduce the background infrastructure and some related work. In Chapter 3, the design approach will be illustrated and in Chapter 4, simulation results will be discussed.



Chapter 2

Background & Related Work

In order to set the stage for the following discussion, this section provides an overview of the underlying protocols upon which our approach builds and some streaming algorithms that have addressed related topics on application level multicast.

2.1 Distributed Hash Tables (DHTs)

DHT-based decentralized distributed systems provide lookup service that the idea is similar to a hash table: pairs in the form of $(name, value)$ are stored in a DHT, and any participating node can efficiently retrieve the value associated with a given name. The workload of maintaining these tables is distributed among the nodes, so that a change in the set of participants causes a minimal amount of disruption. Now some known P2P tools, like BT and eMule, have successfully added DHTs as one of their connect modes. The basic idea of using DHTs is to equally share indexes stored in the server originally among peers, and each peer has a unique ID associated to a local address to its managing data. When a query arrives, the local peer will check its index to find out the mapped data or the address closest to data to forward. Although the DHT successfully solves the drawback of load in centralized systems, it has some

troubles for a user to query the data he/she needs. Frequently, a user sends a query only with partial information but the hash key is unique mapped to the original data; thus it is hard to find out what he/she needs. Therefore, in [2],[3],[4], the authors introduced some methods to improve the DHT and have some saving grace for us.

Pastry: Every peer in Pastry [10], a sample of DHT, is assigned a random unique 128-bit node identifier (*nodeId*). NodeIds are uniformly distributed in the circular identifier space formed by all possible identifiers. Pastry routes messages to the node with the numerically closest nodeId. In order to route messages, each node maintains a routing table including entries that some nodeIds associate with. A message is routed to a node whose nodeId matches at least one digit longer than the current node's nodeId with the message key in the prefix. If no such node exists, the message is routed to a node whose nodeId matches as long as the current node's nodeId with the message key in the prefix, but the node is numerically closer to the key. Therefore, each node maintains a leaf set and a neighborhood set. The leaf set contains n nodes which are numerically closest to the local node's nodeId, and the neighborhood set consists of nodes which are closest to the local node's nodeId based on a proximity metric. During routing, Pastry takes a consistent mapping from keys to overlay nodes by persistently searching intermediate nodes for successful message delivery.

2.2 IP Multicast and Application Level Multicast

IP multicast is implemented in the IP layer and it's not so far widely deployed because of its drawbacks. Firstly, it needs routers to maintain the members of each multicast group. Secondly, its reliability and security cannot be guaranteed. Thirdly, IP multicast calls for changes at the infrastructural level instead of pure software, which slows down the pace of deployment, and it is extremely difficult to work efficiently by a large scale [6].

In contract to IP multicast, application level multicast uses only end hosts to construct an overlay network that offers multicast functionalities. Thus, two of the main disadvantages in IP multicast can be improved in application level multicast.

Firstly, it does not need to maintain routers. Secondly, it is flexible for all interfaces of the network to be used in the application layer. Although application level multicast seems better than IP multicast, it still has some drawbacks. For example, the mapping of the overlay to the network topology may be suboptimal. Some ideas of application level multicast were proposed in [5], [6], [7], in which each efficient multicast method suitable to general internet has its own feature.

Scribe: The Scribe [11] relies on the underlying DHT substrate routing table for

data forwarding and builds upon Pastry to support applications that demand a large number of multicast groups. Any multicast group may consist of a subset of all nodes in the Pastry network and is assigned a random ID (*topicId*). The multicast tree for a group is formed by those forwarding nodes that Pastry routes from each group member to the root, identified by the *topicId*. Messages are then multicast from the root using reverse path forwarding [16]. Additionally, Scribe also enables higher level protocols to specify policies, for example, to assign the outdegree (forwarding capacity) parameter value [13], [17].

2.3 Streaming Algorithms



Existing streaming work on P2P networks can be classified into categories according to how the content is disseminated: *tree-based*, *mesh-based*, and *forest-based* [1]. Firstly, the tree-based algorithm is that the source and interior nodes will offer extensive bandwidth for leaves, but the leaves just receiving data. Due to this character, such an infrastructure may be not suitable for P2P streaming because only partial interior nodes carry large load of forwarding multicast messages. Secondly, the mesh-based algorithm is that each client maintains the partnership and periodically exchanges data availability information with a set of partners.

CoolStream [14], a representation of mesh-based streaming, provides a buffer map to check what a peer is lack and then request from other peers. Finally, the forest-based algorithm strips the content across a forest of interior-node-disjoint multicast trees on its overlay network. SplitStream, a representation of forest-based streaming, is introduced as follows.

SplitStream: Considering tree-based protocols are unsuitable to cooperative environments, in [13], the authors proposed to split the multicast content into k stripes and each stripe is multicast by a separate multicast tree, where an interior node in one tree is a leaf node in all others. By those disjoint trees, the forwarding load will be distributed among participating peers, and the system will be more robust to node failures by reducing the dependency on any node. In order to ensure the system being feasible, inbound bandwidth consumption is controlled; a peer joins at most as many stripes as its bandwidth capacity permits. The SplitStream implemented relies on Scribe [11], an application level group communication protocol built on Pastry [10], a DHT based, structured peer-to-peer routing protocol.

CoopNet: Like SplitStream, CoopNet [20] splits the multicast content across different trees for improving load sharing and resilience. In addition, CoopNet relies on a centralized organization protocol and adopts MDC (Multiple Description Coding)

to provide redundancy in data.

AMSS: In [18], the authors proposed a model named asynchronous multi-source streaming (AMSS). In AMSS, each of the source peers is responsible to transmit different packets to each of the requesting leaf peers. Also, it attaches some parity packets in each stream, so as to improve tolerance of faulty source peers and packets lost. Additionally, contents peers must exchange control packets to each other to detect whether every other contents peer is active or dormant.

PRM: Probabilistic Resilient Multicast (PRM) [22] is a data recovery scheme which can be applied to any application layer multicast for improving data delivery ratios. It used two techniques to recover data. Firstly, it utilizes randomized forwarding; each overlay node chooses a certain amount of other overlay nodes at random, and forwards data to each of them with a low probability. Secondly, it adopts a reactive mechanism called triggered NAKs to handle data losses due to link errors and network congestion. In order to compare it with our approach, we apply PRM to the Scribe application layer multicast protocol and name it as Scribe-PRM.

Additionally, in the research of overlay multicast, some protocols have also been proposed aiming at reliability and resilience [8], [9], [12]. Table 1 shows the comparison of the existing methods described above and the proposed ConStream.

Table 1. Comparison of related P2P streaming algorithms

Approach	Architecture	Data delivery	Recovery mechanism	Bandwidth overhead	Streaming mode
SplitStream [13]	Multiple trees	Application level multicast	None	Low	Single streaming
CoopNet [20]	Multiple trees	Application level multicast	MDC	Low	Single streaming
CoolStream [14]	Overlay mesh	Data-driven overlay	None	Medium	Buffer map
AMSS [18]	Multi-source	Asynchronous overlay streaming	Parity packet	High	Multi-source streaming
Scribe-PRM [22]	Single tree	Application level multicast	Proactive randomized forwarding	Low	Single streaming
ConStream (proposed)	Multiple trees	Application level multicast	Parity packet	Medium	Multi-streaming

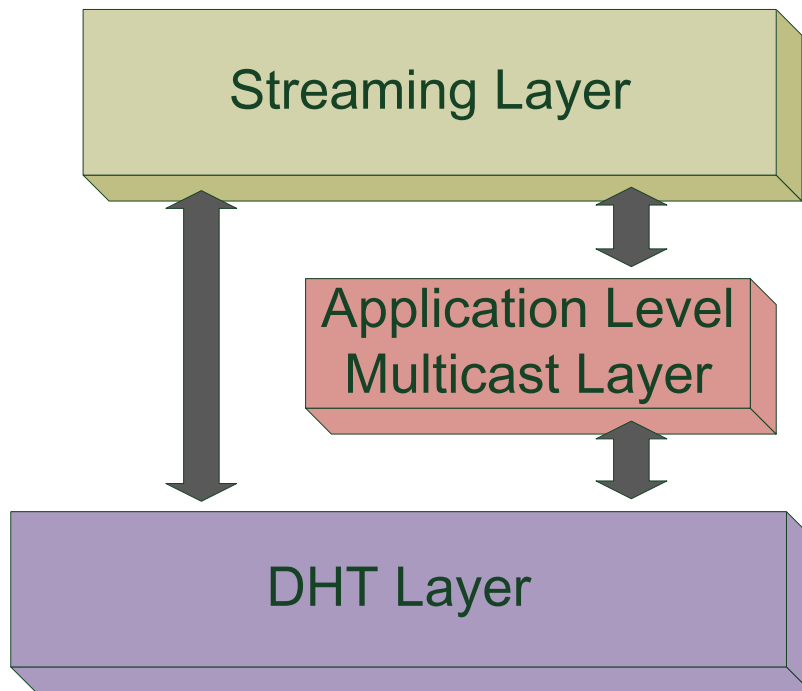


Figure 1. Architecture of P2P multimedia streaming based on DHT.



Figure 1 shows the architecture of P2P multimedia streaming based on DHTs. The streaming layer is the focus of our approach. In general, a stream is composed of segments. And the streaming layer is responsible to make the peers able to set some sorts of priority to the segments and to their partners. The application level multicast acting on the DHT layer is responsible to construct a multicast overlay network, like Scribe [11]. Finally, we used Pasty [10] as our DHT layer to implement our basic structure.

Chapter 3

Design Approach

In [15], it shows that the delivery ratio decreases when the transient condition becomes serious in P2P networks. We propose a multi-streaming scheme on the streaming layer, called *ConStream*, to resist this condition. The basic architecture of our proposed approach is like SplitStream, striping the content across a forest of interior-node-disjoint multicast trees that distributes the forwarding load among all participating peers. In SplitStream, a content source forwards the content via different stripes in a proper sequence, and the content length to forward a stripe each time is fixed. However, in our ConStream, we merge three stripes as a unit named multi-stripe and the total stripes in SplitStream can be viewed as several multi-stripes. The content in a multi-stripe will be divided into small pieces, and they will be forwarded by three multicast trees concurrently. Also we insert parity packets [18] into streams for error recovery.

There are some problems when several streams work concurrently. Firstly, if streams deliver contents via the same tree architecture, the forwarding capacity that senders (interior nodes) can offer will be shared by those streams. In addition, several streams a client received will be interrupted at the same time if any one of this client's

ancestors lefts. Secondly, if streams are delivered via different tree architectures, how we can do to prevent generating more load to the source or bigger delay jitter to clients.

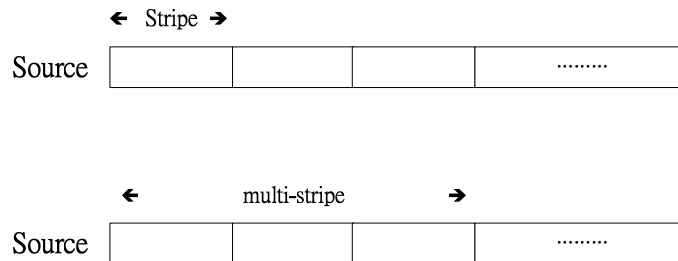


Figure 2. Concept of stripe and multi-stripe.

In Figure 2, it shows the difference between the traditional stripe and multi-stripe. We view the total contents as many fixed time segments, and stripes are used to deliver these segments in turn. Traditionally, the source delivers contents via one single tree during each stripe delivering, as illustrated in Figure 3. In multi-stripe, three stripes are grouped into a unit and contents are delivered by three stripes concurrently, as illustrated in Figure 4. During packets delivering in sequence via different streams, we insert parity packets into them. A parity packet is generated from the two latest transmitted packets. Once any interior node fails in one stream, the descendants of this streaming tree can recover the lost data by parity packets from the other two living streams during the time of repairing the failed tree. In Figure 4, the parity packet (5 6) is generated and inserted into streaming 1 after packet (5) and

packet (6) are delivered via streaming 2 and streaming 3, respectively. For a certain client, if data packet (5) is lost, this packet can be recovered by packet (6) and parity packet (5 6). Additionally, due to the characteristic of the forest of interior-node-disjoint [13], one faulty interior client just causes bad effect on one tree. Thus, our approach can work robustly unless one client loses more than one stream simultaneously, which is a situation that seldom happens.

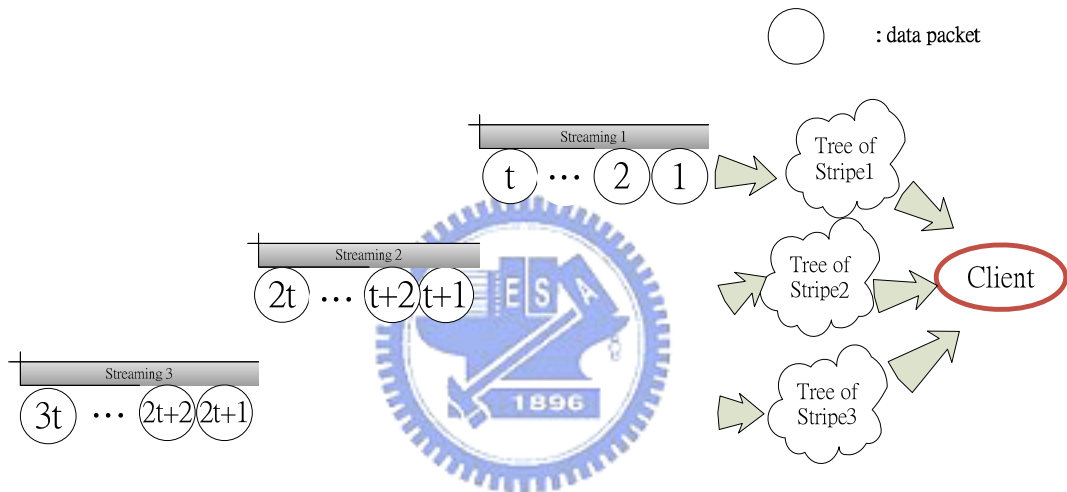


Figure 3. Deliver the content via stripes.

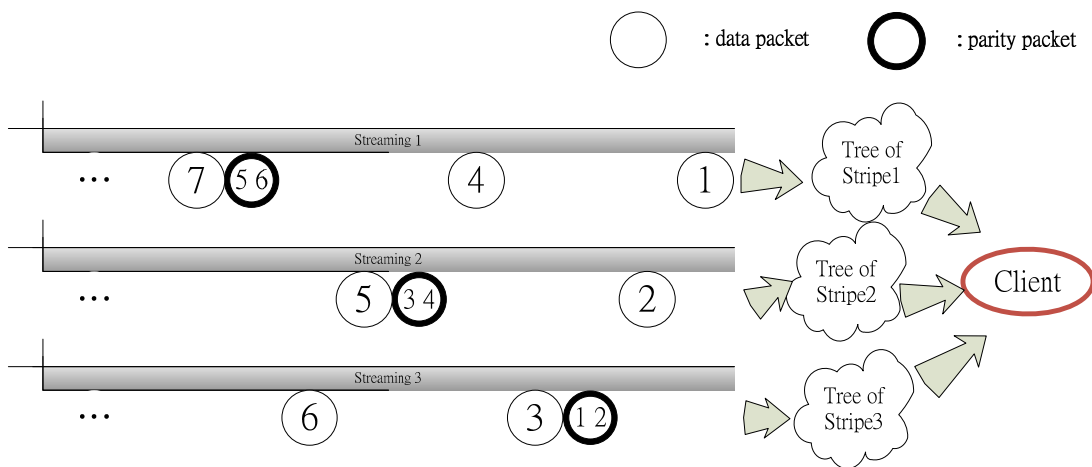


Figure 4. Deliver the content via multi-stripe.

Based on the characteristic of the forest of interior-node-disjoint, these concurrent streams will work on different tree structures and the forwarding capacity of each client won't be shared by more than one stream. In the view point of any node itself, among the trees, a node is an interior node in only one tree, and it is a leaf node in each of the other trees. In this way, one client receives the content via three different streams, but it plays as an interior node which is responsible to forward contents to no more than one stream. Figure 5 shows that nodes A, B, and C are interior nodes in stripe 1; nodes D and E are interior nodes in stripe 2; nodes F and G are interior nodes in stripe 3; node H just plays leaves in the multi-stripe. Here we can observe that the forwarding capacity of each node is just used for a single stream even if this node also receives from other distinct streams.

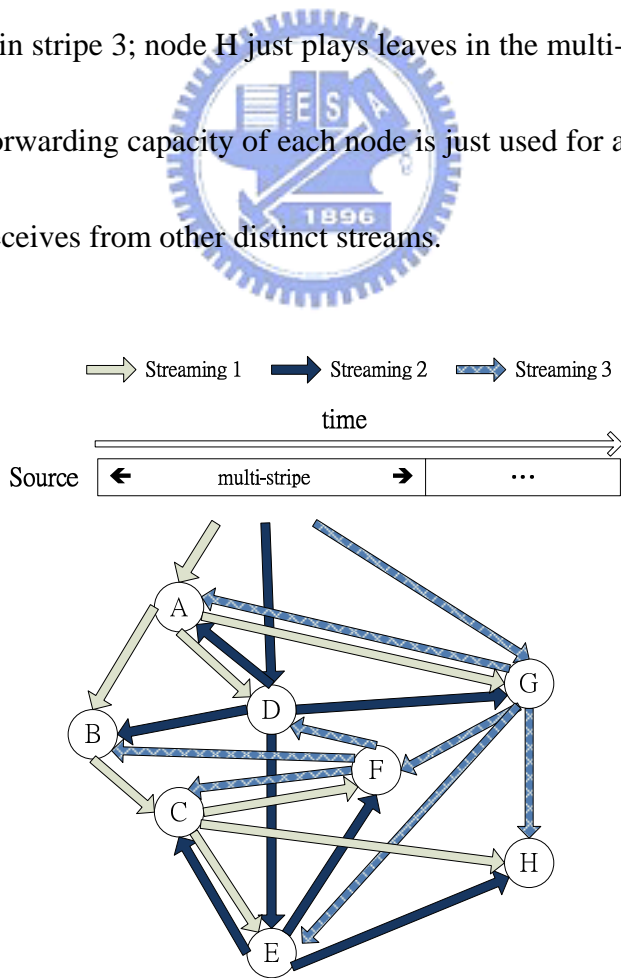


Figure 5. Concurrent streams during the time interval of multi-stripe.

The server load and bandwidth needed in our approach will increase due to adding parity packets, but the required delivery rate in each stream will be moderated. The forwarding load of each interior node in a multi-stripe will be 1.5 times the forwarding load in a traditional stripe, because each parity packet generated is accompanying with two data packets. However, considering the time interval of multi-stripe that increases 3 times, the delivery rate of each stream in multi-stripe is only half of the delivery rate of the stripe. The results are that packets in Figure 4 are sparser than packets in Figure 3. The delivery rate is defined as the amount of packets delivered over a time unit. In this way, the burden on peers will be efficiently reduced during delivery time although the total forwarding load increases. Therefore, it's a feasible solution to ease up the burden on those peers with not enough outgoing bandwidth.

According to [13], a node in the overlay network will contribute its bandwidth for data forwarding. As shown in Figure 6, the forwarding capacity of a node is defined as the number of its receivers, and indegree is defined as the number of distinct stripes that a node requests. For the feasibility of forest construction in SplitStream, it needs to satisfy the following two conditions. (1) *Feasible condition*: if forest construction is feasible, the sum of the desired indegrees of all nodes cannot exceed the sum of the forwarding capacities of all nodes. (2) *Sufficient condition*: for

the above feasible condition to hold, each node's forwarding capacity must exceed its desired indegree [13].

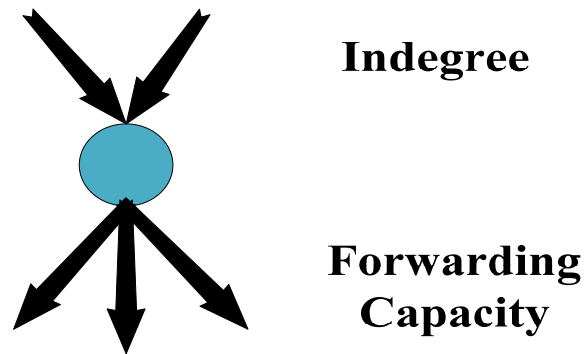
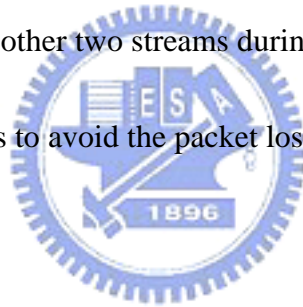


Figure 6. Conception of indegree and forwarding capacity.

Following the above conditions, the more forwarding capacity a client contributes the more indegree a client can receive. These two conditions emphasize fairness and are used to improve performance. And since our proposed architecture is based on SplitStream, we can also make our model ideal by satisfying the feasible condition and sufficient condition. According to our improved architecture, clients can contribute more capacity to promote efficiency in a system and to easily satisfy the need of indegree, because their original load is lowered by decreasing the delivery rate.

In our ConStream, we merge three stripes as a multi-stripe and take it as a unit to forward the content. But why not merge four, five, or six as a multi-stripe? When

streams are delivered concurrently, a client will receive contents from different paths, and the delay time of these data will be more variant than that of just from one path. For example, in Figure 5, node F receives contents form streaming 1, streaming 2, and streaming 3 concurrently, but the hops of stream paths from the source to node F are 4, 3, and 2, respectively. This situation will result in unsteady quality of media contents played in real time. To reduce such a jitter problem, we merge only three stripes and we also base on the slowest stream to start the real time multimedia playing during multi-stripe delivering. However, in this way, it will cause the client buffer being occupied by the data from the other two streams during waiting for the slowest stream. So, we increase clients' buffers to avoid the packet loss problem.



Chapter 4

Simulation Results and Discussion

Based on the architecture in Figure 1, we used FreePastry (version 2.0_03) [19] to implement the DHT layer (Pastry), application layer multicast (Scribe), and streaming layer (SplitStream). And then we implement our proposed ConStream scheme on this structure. In addition, we implemented Scribe-PRM, which we are going to compare with, by applying PRM to Scribe to significantly augment the data delivery ratio of Scribe.

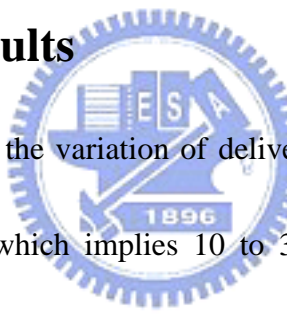


4.1 Simulation Scenarios

Our environment was built on a Pastry overlay with 1000 nodes and built a forest structure with all overlay nodes. The *delivery ratio* is defined as the ratio of packets a client received to total delivered packets by the source. *Repair time* is determined primarily by SplitStream's failure detection period, which triggers a tree repair when no heartbeats or data packets have been received for 30 seconds [13]. Besides, the randomized forwarding probability p in Scribe-PRM is chosen to be 0.05 [22]. We will measure the delivery ratio under transient conditions, and the parameters for the degree of transiency include *mean time to failure* (MTTF) [15] and *node failure rate*

[22]. We assume that the delivery ratio of each peer is 100% under non-transient environments. On the other hand, we measure the *absolute delay* of sequential packets on a certain client to examine *delay jitter*. The absolute delay is defined as the elapsed time of a packet from source to destination. In addition, the environment we used to measure the absolute delay is established with non-transient condition because the transient condition may cause more unstable factors in the measurement.

4.2 Simulation Results

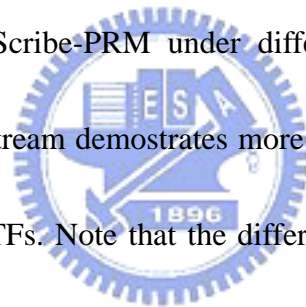


In Figure 7, we examine the variation of delivery ratios under the node failure rate between 1% and 30% (which implies 10 to 300 simultaneous failures in the overlay with 1000 nodes) with a fixed MTTF of 120 seconds. The result shows that the descending rates of the delivery ratio with the increase of the node failure rate for ConStream and SplitStream are similar, but Scribe-PRM has a lower descending rate. Our ConStream performs better than Scribe-PRM under the node failure rate less than 20%. And in reality, 20% of the node failure rate is too high in usual Internet standards.

In addition, we examine the distribution of delivery ratios in each algorithm under the MTTF of 120 seconds and the node failure rate of 10% (which implies 100

simultaneous failures in the overlay with 1000 nodes). And we randomly chose 100 nodes from this overlay and the result is shown in Figure 8. The delivery ratio in SplitStream or Scribe-PRM is about 60% ~ 100%. The delivery ratio in ConStream is about 80% ~ 100%, which is denser than the other two. As a result, we observed that nodes in ConStream will receive stable contents even under transient condition.

Assuming that the actual internet environment of node failure rate is 5% (which is still high in usual Internet standards) [22], we examined the delivery ratio under different MTTFs. From simulation results in Figure 9, ConStream has higher delivery ratio than SplitStream and Scribe-PRM under different MTTFs. Additionally, we found that the curve of ConStream demonstrates more steady trend than the other two with respect to different MTTFs. Note that the difference of delivery ratios between ConStream and Scribe-PRM (or SplitStream) increases as MTTF decreases. For example, when MTTF is between 180 and 60 seconds, the variances of delivery ratios for ConStream is about 0.02, while the variances for Scribe-PRM and SplitStream are about 0.05 and 0.06, respectively.



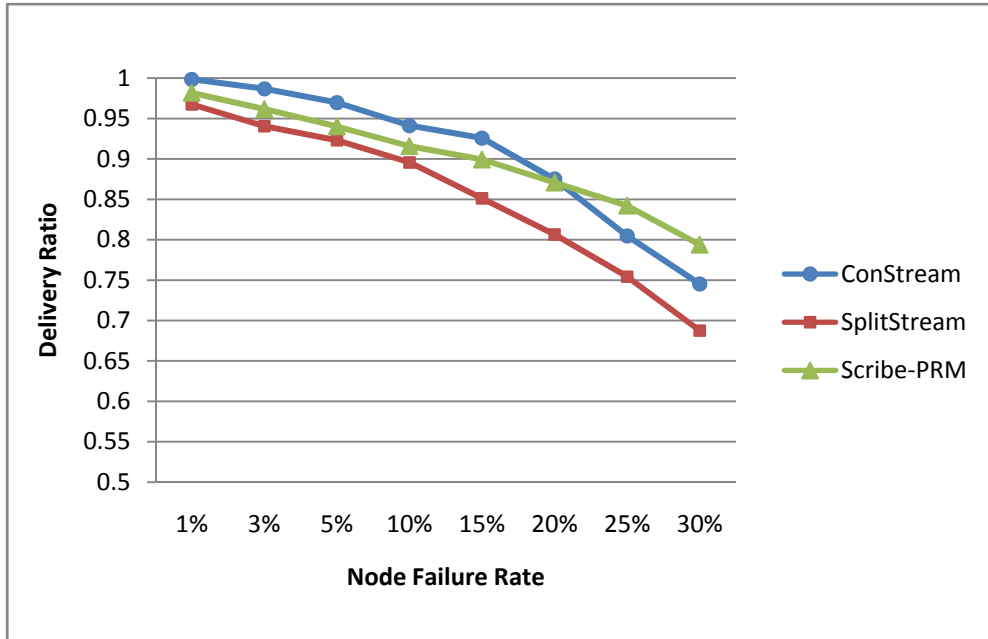


Figure 7. Delivery ratios with various node failure rates

under a fixed MTTF (120 seconds).

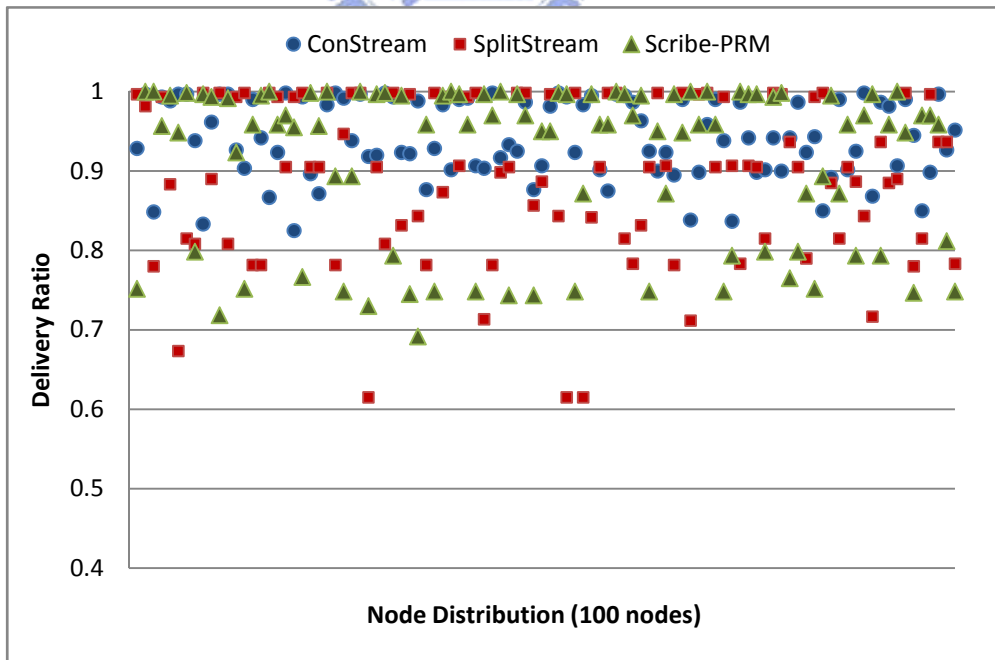


Figure 8. Distribution of delivery ratios.

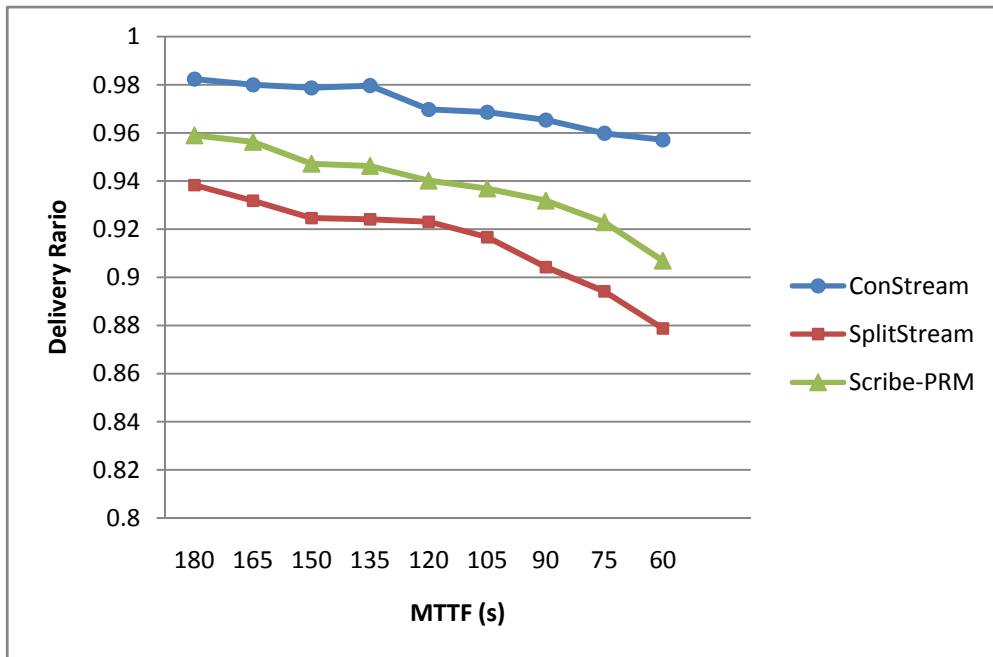
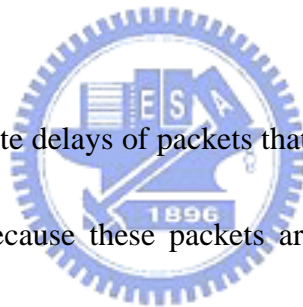


Figure 9. Delivery ratios under different MTTFs.



In ConStream, the absolute delays of packets that a client receives from the same sequence may be unstable because these packets are delivered via different paths.

In Figure 10, it shows packets with wide delay variation received by one client in a certain time interval. In addition, we classified these packets according to the streams they originally belong to, as shown in Figure 11. The absolute delays in multi-stripe of our ConStream is less stable, and we have proposed an improved solution, introduced in Chapter 3, for each client, in order to handle this unfavorable condition.

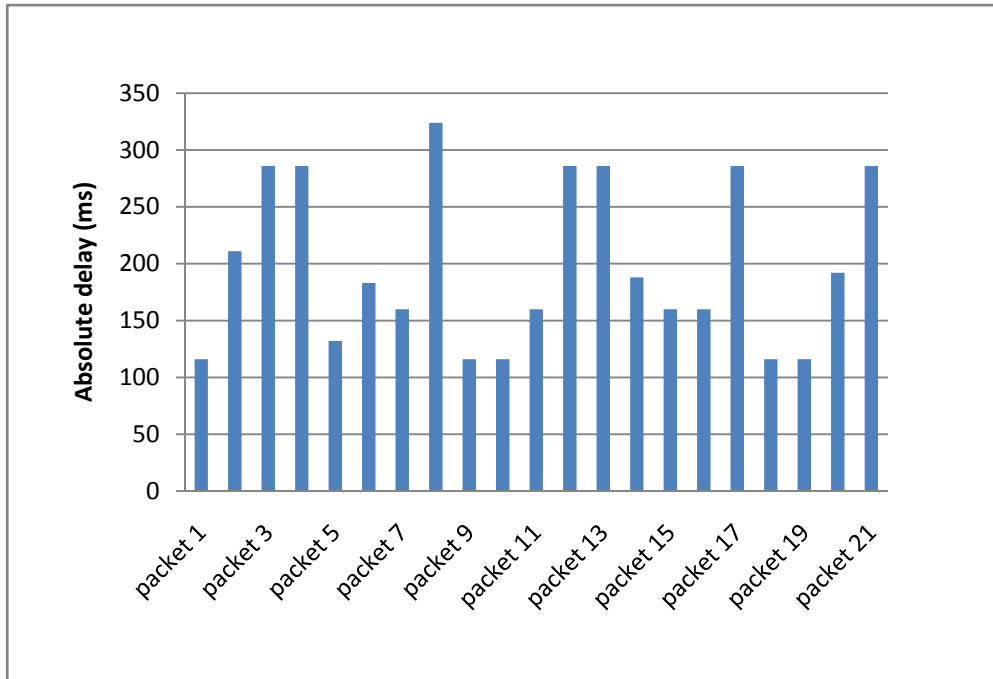


Figure 10. Variation of absolute delays among received packets.

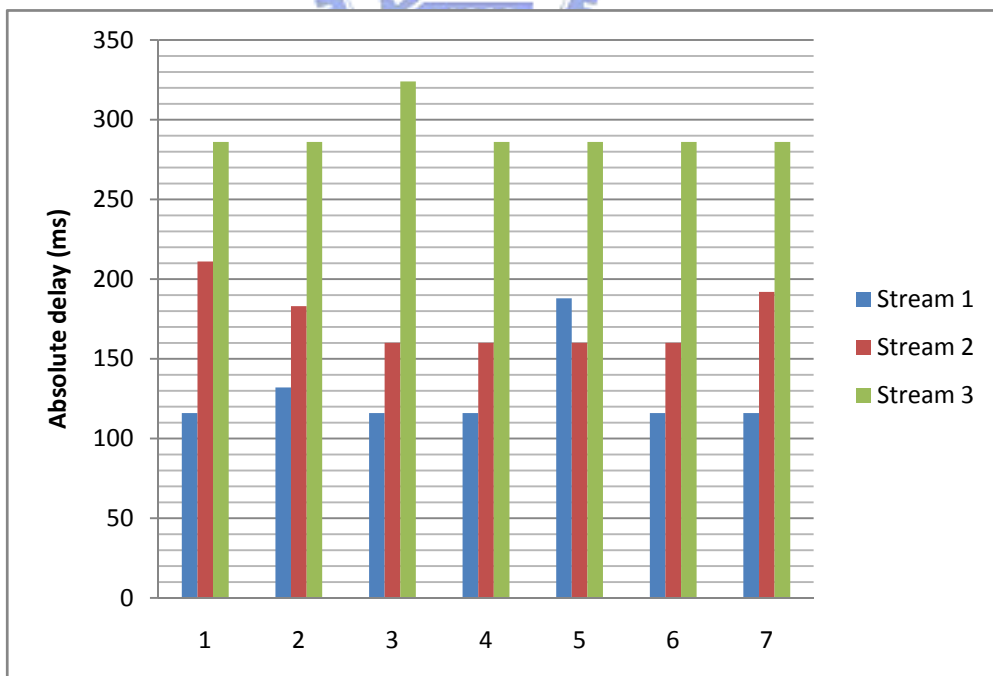


Figure 11. Absolute delays by each streaming.

Chapter 5

Conclusion

5.1 Concluding Remarks

We have presented a P2P multi-streaming scheme, ConStream, which have a feature of a high delivery ratio. And in this scheme, with the parity packet and the multi-stripe techniques, the lost data caused by transient peers in P2P networks can be reduced, and the tree recovery time can be tolerated as well. However, clients' buffers that are used to cache contents temporarily need to be increased to resolve variable delay times between streams. Although the bandwidth consumed by each stream will increase for parity packets, the load on peers will be efficiently reduced during delivery time because of the reduced delivery rate of streams.

5.2 Future Work

We will implement the proposed ConStream in an actual internet environment, and experimental results will be evaluated to justify our simulation results.

Bibliography

- [1] <http://mmdays.wordpress.com/2007/04/19/p2p-tv/>
- [2] M. Balazinska, H. Balakrishnan, and D. Karger, “INS/Twine: A scalable peer-to-peer architecture for intentional resource discovery,” in *Proc. ICPC*, pp. 195–210, August 2002.
- [3] M. Harren, J. Hellerstein, R. Huebsch, B. Loo, S. Shenker, and I. Stoica, “Complex queries in dht-based peer-to-peer networks,” in *Proc. IPTPS*, pp. 242-250, March 2002.
- [4] L. Garces-Erice, P.A. Felber, E.W. Biersack, G. Urvoy-Keller, K.W. Ross, “Data indexing in peer-to-peer DHT networks,” in *Proc. ICDCS*, pp. 200-208, 2004.
- [5] C. Luo, J. Li, and S. Li, "DigiMetro – An Application-Level Multicast System for Multi-party Video Conferencing," in *Proc. IEEE Globecom*, Vol. 2, pp. 982-987, November 2004.
- [6] Z. Liu, G. He, Zhi Liu, “Metrino - an application-level multicast system for real-time video communication,” in *Proc. WiCOM*, Vol 2, pp. 1311–1316, Sept. 2005.
- [7] C.L. Chan, S.Y. Huang, H.H. Chen, W.H. Tung, J.S. Wang, “An Application-Level Multicast Framework for Large Scale VOD Services,” in *Proc. ICPADS*, pp. 98–104, 2005.
- [8] V.N. Padmanabhan, H.J. Wang, P.A. Chou, “Resilient peer-to-peer streaming,” in *Proc. IEEE ICNP*, pp. 16–27, Nov 2003.
- [9] D.A. Tran, K.A. Hua, and T. Do, “ZIGZAG: An efficient peer-to-peer scheme for media streaming,” in *Proc. IEEE INFOCOM*, pp. 1283–1292, April 2003.
- [10] A. Rowstron, P. Druschel, “Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems,” in *Proc. IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pp. 329–350, November 2001.
- [11] M. Castro, P. Druschel, A.M. Kermarrec, and A. Rowstron., “Scribe: A large-scale and decentralized application level multicast infrastructure,” *IEEE Journal on Selected Areas in Communication*, Vol. 20, pp. 1489-1499, October 2002.
- [12] M. Yang and Z. Fei, “A proactive approach to reconstructing overlay multicast trees,” in *Proc. IEEE INFOCOM*, Vol. 4, pp. 2743-2753, march 2004.
- [13] M. Castro, P. Druschel, A.M. Kermarrec, A. Nandi, A. Rowstron, A. Singh, “Splitstream: High-bandwidth content distribution in a cooperative environment,” in *Proc. IPTPS’03*, pp. 292-303, February 2003.

- [14] X. Zhang, J. Liu, B. Li, Y.-S.P. Yum, "CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming," in *Proc. IEEE INFOCOM*, Vol. 3, pp. 2102-2111, March 2005.
- [15] S. Birrer, F.E. Bustamante, "The feasibility of DHT-based streaming multicast," in *Proc. MASCOTS*, pp. 288-298, Sept. 2005.
- [16] Y.K. Dalal and R.M. Metcalfe, "Reverse path forwarding of broadcast packets," *Communication of the ACM*, Vol. 21, pp. 1040-1048, December 1978.
- [17] A.R. Bharambe, S.G. Rao, V.N. Padmanabhan, S. Seshan, and H. Zhang, "The impact of heterogeneous bandwidth constraints on DHT-based multicast protocols," in *Proc. IPTPS*, pp. 115-126, February 2005.
- [18] S. Itaya, T. Enokido, M. Takizawa, A. Yamada, "A scalable multimedia streaming model based-on multi-source streaming concept," in *Proc. International Conference on Parallel and Distributed systems*, Vol. 1, pp. 15-21, July 2005.
- [19] http://freepastry.rice.edu/FreePastry/README-2.0_03.html
- [20] P.A. Chou, V.N. Padmanabhan, and H.J. Wang, "Resilient peer-to-peer streaming," in *Proc. IEEE International Conference on Network Protocols*, pp. 16-27, March 2003.
- [21] T. Xiangrout, S. Datta, "Building multicast trees for multimedia streaming in heterogeneous P2P networks," in *Proc. Systems Communications*, pp. 141-146, July 2005.
- [22] S. Banerjee, S. Lee, B. Bhattacharjee, A. Srinivassan, "Resilient multicast using overlays," *IEEE/ACM Transactions on Networking*, pp. 237-248, April 2006.
- [23] S. Birrer, F.E. Bustamante, "Resilient peer-to-peer multicast without the cost," in *Proc. SPIE*, Vol. 5680, pp.113-120, January 2005.