Enhancing the Performance of Multi-Source Streaming System Using the MCMCF-Based Algorithm

Wei-Cheng Xiao, Po-Lin Chou, Tsai-Lin Hsu, and Cheng-Fu Chou

Abstract — *We present the implementation and evaluation of an MPEG-4 multi-source streaming system, in which each receiver can simultaneously retrieve different parts of a streaming object. Unlike the traditional single-source streaming system that may have poor perceived quality at the receiver site due to the network congestion or the heavy-loaded server, our multi-source system aims to maximize the usage of available resources on both end-hosts and the network to provide better quality for the users. In order to achieve this objective, our system employs three key components: a) select appropriate servers, b) distribute load to those selected servers to maximize the overall throughput, and c) avoid self-congestion between different object flows that are requested by different users at the same time. We apply a minimum-cost multicommodity flow (MCMCF) [6] based approach to deal with the above issues.*

We have extensively evaluated the performance of our system and other different streaming systems through the simulation. The simulation results show that our system can achieve better end-to-end throughput than greedy-based or min-cost-flow-based streaming systems does.[1]

Index Terms — Multimedia Streaming, Multi-Source, Peer-to-peer systems, MCMCF, Mathematical programming / optimization

I. INTRODUCTION

Lately, broadband access has successfully drawn considerable public attentions and earned the wide adaptations in industries and markets as well. Nowadays, more and more people use streaming to obtain data; thanks to the growing network bandwidth, users now can be served with greatly improved quality of data through the Internet. A streaming system provides end users with a good means to obtaining their requested streaming data, such as online radio, live programs, and on-demand video, by retrieving the stream from streaming server(s) coordinated by some real-time streaming protocols, like MMS and RTSP. However, these UDP-based protocols, unlike TCP, are not reliable. Thus, when considering the quality of the transmission of media streams, preventing the packets from loss is as important as choosing the path with higher available bandwidth.

A. Multi-source streaming system

In most of the cases, when a user makes a request for an audio or video stream, he sends this request to only one selected server, and gets the stream data only from the same server later on. But when a network congestion happens on the client-to-server path, or when the selected sever becomes over-loaded, the client is very likely to suffer a very poor service quality of the requested object.

Besides the single-source streaming system mentioned above, a multi-source streaming system can be an alternative choice. Unlike the single-source, a multi-source streaming system allows clients to send requests simultaneously to several stream servers for a data stream, construct the connections with selected capable servers, and then start fetching data concurrently. Putting it in a simple way, a multi-source streaming system tries to make the best use of all the possible and available stream servers and considers path diversity in the network. Provided with both of two factors, the offered quality for users would greatly improve.

With the rapid growth of the P2P network, some well-designed systems, such as Share[2], BitTorrent [11], eMule [12], and eDonkey [13], have concentrated on how to speed up the downloading rate. In these systems, each file is split into pieces, and users can download different pieces of their requested file from different peers in parallel. This scheme is quite similar to the inherent properties of the multi-source system. Therefore, the multi-source streaming system is also applicable onto the P2P networks, with a few changes for the real-time requirements.

B. Contributions and features

To design an efficient multi-source streaming system, there are several challenging issues to notice: a) how to select the appropriate servers, b) how to maximize the overall throughput when loads are distributed among selected servers, and c) how to avoid self-congestion between different concurrent object flows. In order to solve this problem, we divide it into stages to discuss. First, we

[1] This work was partially supported by the National Science Council and the Ministry of Education of ROC under the contract No. NSC92-2622-E-002-002 and 89E-FA06-2-4-8.

W.C. Xiao, P.L. Chou, T.L Hsu, and C.F. Chou are with the Computer Science and Information Engineering Department, National Taiwan University, Taipei, 106 Taiwan. Emails: {garry, sinacloud, alyn, ccf}@cmlab.csie.ntu.edu.tw.

[2] Share is the most popular P2P system in Japan.

0-7803-8920-4/05/$20.00 © IEEE

transform the whole network topology into a connected graph; here a topology consists of the sources, receivers, transit nodes, and links. Second, we assign the capacity of each edge in the graph with the same capacity value of the corresponding links in the network. In addition, we can measure the average packet loss rate on each link applying the proposed method in [10], and continue to compute the cost of its corresponding edge with the cost function defined in section IV.B and again each streaming connection among the clients and sources are given same as its corresponding flows in the graph. With the capacity constraints and costs of each link in the graph, we can formulate the service selection problem into a minimum-cost multi-commodity[3] flow (MCMCF) [6] problem.

Through applying MCMCF, we can derive an optimized solution that satisfies the bandwidth constrains and minimizes the overall cost; that is, minimizes the overall packet loss rate in the network, which implies maximizing the overall performance. Moreover, due to property of multi-commodity in MCMCF, the performance optimization is also reachable when multiple streaming objects are delivered concurrently on the network. Since the MCMCF problem is linear, we can solve it very efficiently by using the well-known tool CPLEX [5], and then quickly adapt the service selection to the varying network condition.

We have evaluated the performance of our MCMCF-based streaming system on the ns2 [4] simulator. In the simulation, we compare our system with other two commonly used systems, i.e. min-cost-flow-based and greedy-based. From the simulation results, we observe that the MCMCF-based system always achieves higher throughput, given different values of link capacities and loss rates in various network conditions.

In addition, a MCMCF-based implementation on the multi-source streaming system has also been done. In our implementation, the clients can select some better servers from all server candidates to connect, retrieve a complementary set of parts that form the original stream, and then reconstruct the original stream for playback.

C. Organization of this paper

The rest of this paper is organized as follows. In Section II, we review the related work on the streaming system. In Section III, we give the details of the implementation of our multi-source streaming system. In Section IV, we present our MCMCF-based multi-source streaming system and go through the details about service selection problem. We evaluate the performance of our system and compare it to the greedy-based and min-cost-flow-based systems in section V. Finally, we conclude our paper and give the future works in section VI and VII.

[3] In this paper, we use the terms "commodity" and "streaming object" interchangeably to represent the video/audio stream requested by a client.

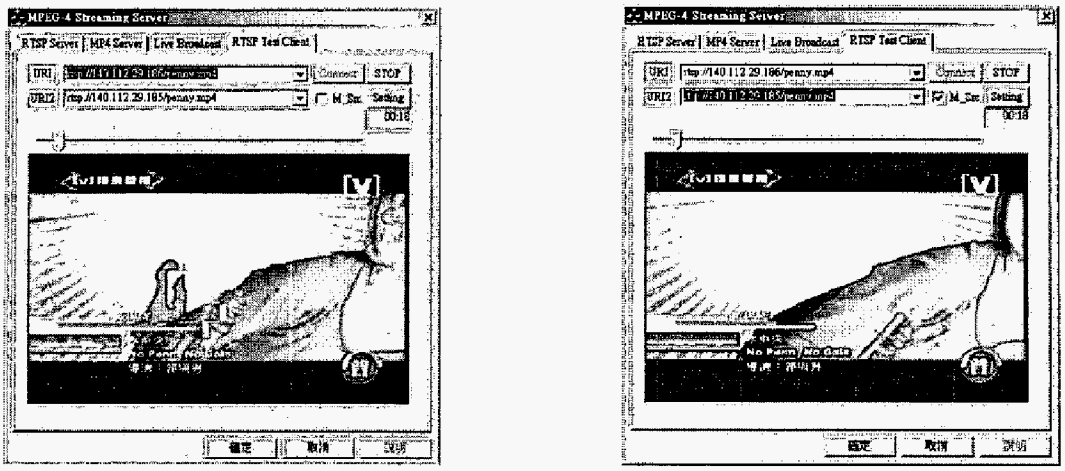
II. RELATED WORK

During the last few years, a large number of interesting discussions on media-stream are becoming popular and various significant researches on media streaming are broadly studied in many aspects. For simplicity, first, we classify all the proposed media-stream schemes into two categories, a) *networking* and b) *video/audio processing,* on different perspectives. The methods in the first category urge to increase the probability of the successful decoding video/audio stream using specific techniques like FEC (Forward Error Correction) or MDC (Multiple Description Coding) [14]. In FEC, to bear a better tolerance to packet loss rate of the network, every original file requires the processing from certain specific encoding schemes, such as the Reed-Solomon code or the Tornado [16] code. In MDC, the audio/video signal is encoded into separate streams so that on receiving any subset of these streams, the receiver is still capable to decode the received stream data into the original signal yet with some distortion.

On the other hand, video frames sometimes may be either lost due to the unreliability of UDP channel or caused by the bandwidth deficiency during the delivery of a video stream over a resource-constrained network. In such circumstances, the policy of frame transmission becomes very important. For example, results of different decision policy of frame dropping selection may vary greatly because I, P, and B frames are of distinctive relative importance. Usually, it is found that choosing I, P frames last dropped may keep the perceived video quality better. There are papers ([15]) dealing such kind of issues.

Our proposed multi-source and multi-commodity streaming system belongs to the second category, *networking*, which mostly focuses on the network topology, load distribution, path selection, etc. Besides, we realize these two categories, i.e., signal processing and networking aspects, are orthogonal and work together well without conflicts. Therefore, we can combine two mechanisms for cooperation to achieve the compound gains. This supports our system to be capable to cooperate with any already established audio/video processing system, take advantage of the great research contribution of former great proposed scheme, which is another contribution. In the following paragraph, we give some related work on this category.

Research work on real-time streaming has been widely studied in many aspects by former people in past years. However, many researches are only in considerations of cases with single receiver or single commodity, e.g., PROMISE [3], ZIGZAG [8], DONet [9], and the system proposed in [7], which assuming both single-receiver and single-commodity are sustained. Moreover, a system's sending rate assignment is based on the end-to-end information rather than the underlying topology. The system in PROMISE focuses on multiple sources with regard of topology-aware service selection, which all



(a) Without multi-source (b) With multi-source

Fig 1. Video quality under bandwidth constrained network

resembles our scheme. Unfortunately, it does not account for multi-receiver and multi-commodity, and is very likely to suffer from the sender selection and rate assignment at the high cost of required time. In ZIGZAG [8], it deploys a hierarchical tree-based architecture for media stream to improve scalability, but its algorithm is only compatible to the single-sender topology.

III. Design of the Multi-Source Streaming System

A. System overview

In a multi-source streaming system, a client may need to get different parts of a streaming object from different servers. To utilize the resources efficiently, each server sends a number of **non-overlap** parts of video/audio data, proportional to the available bandwidth, to the client. Namely, we would like to adaptively determine the sending rate of the sender according to the network condition at that time. Fig. 2 illustrates an example of the overview of a 2-source streaming system.

We follow the MPEG-4 standard to implement our multi-source streaming system. To meet the decoding requirements for the stream at the least cost of client's buffer space, the unit size for load assignment to each sender should be an audio/video frame at least, or a unit of group of pictures (GOP), e.g., we use 2 GOPs as a unit in our system. Moreover, to synchronize different parts of video/audio streams for successful playback, we add an extra field, called frame index, into the modified RTP header.

We have developed the software for our multi-source streaming system, whose interface is shown in Fig. 1. In Fig. 1(a), the video is fetched from only one source; while in Fig. 1(b), we fetch the same video from two sources concurrently. We fetch the video stream under the bandwidth constrained network, and we can see that in Fig. 1(b) the received video suffers from frame loss and has worse quality than in Fig. 1(a).

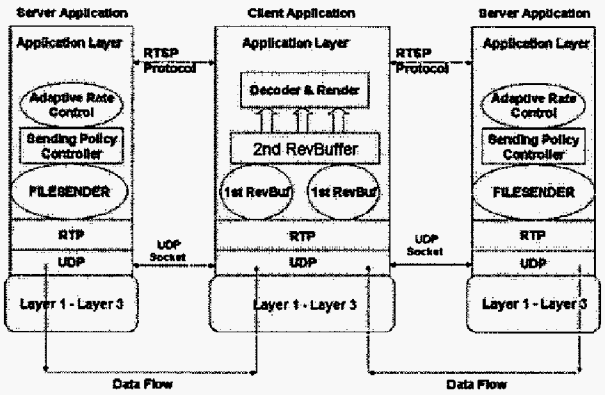


Fig 2. A diagram for the implementation of our system

B. The implementation of our streaming system

To implement in a multi-source scheme, there are some small adjustments needed in the original streaming protocol in our streaming system. First, we add extra fields in the header of the RTSP request packet. These fields are used to provide information to notify the server of its responsibility, like the streaming ratio. Second, we deploy multiple buffers at the client site to support multi-source service; a single buffer only stores the streaming data from a single server. In our simulation, this association between servers and their responsible buffer is constructed during the 'Setup' state of RTSP; once the server-buffer association is terminated, connected servers start up the transmission. Each time the amount of received data in a buffer is enough to reassemble back into a complete frame, this reassembled frame will be sent to the upper second buffer, shown in Fig 2. The function of the second buffer in the scheme is to manage the total reassembled complete frames from all the first buffers. Because frames from different servers are received out of order, we need to sort them back in sequence at the second

buffer according to their Frame Numbers. Then, the output sequence is fed to the Video/Audio decoder to the playback job afterwards.

The complete streaming procedures are as follows:

1. The client sends multiple requests to multiple servers and establishes the connections with them.
2. The client sets up the server-buffer association with each buffer for a single connected server.
3. The transmission starts. The client receives streaming data stored in first buffers.
4. All reassembled frames sent to the second buffer are sorted back in sequence according to Frame Number.
5. The sequenced output is fed to the decoder and the graphic device renders them.

IV. Service Selection Problem

A. The MCMCF problem

Before we talk about the details of the solution to the service selection problem, we first briefly review the MCMCF problem.

The MCMCF (Minimum-Cost Multi-commodity Flow) problem consists of a directed graph $G = (V, E)$, a set of links L, a set of commodities M, sources with limited supplies S, and sinks with demands R. Let b be the positive capacity function of links, and c to be the nonnegative cost function. Each flow f of commodity m from source s to sink r must abide by all the constraints. The cost of a flow f is defined as $\sum_{e \in p_f} f \cdot c_e$ where c_e is the cost of link e and p_f is the path to route flow f. Now, the problem can be formulated as to compute the set of flows F, which minimizes the total cost while satisfies the link capacity constraints, sink demands, and source supply upper bounds

To solve the above MCMCF problem, many algorithms and tools are suggested to implement; here we use a well-known tool – CPLEX in our system for solution, and we will not go through the details of this tool.

B. Solving the service selection problem

The service selection problem in a multi-source streaming system includes (a) how to select the proper servers, and (b) how to determine how much amount of data should be delivered from which server. Here we assume that the network information, i.e., the network topology and link properties, could be obtained via some end-to-end measurements. For example, the physical topology can be built by using a tool like traceroute. Traceroute can be performed in parallel on all the clients, and then these clients can communicate with each other to build the whole network topology. Including the available bandwidth and link loss rate, link properties can also be obtained using the scheme mentioned in PROMISE [3].

R	the set of receivers
M_r	the set of commodities requested by receiver r
F_m	the set of flows containing commodity m
L	the set of links
P_l	the loss rate of link l
B_l	the capacity of link l
C_l	the cost of link l
$S_{r,m}$	the set of senders to receiver r on commodity m
$BW_{l,f}$	the bandwidth on link l used by flow f
X_s	the sending rate of sender s
$Y_{r,m}$	the requested rate of receiver r on commodity m

Table I. Notations for the service selection problem

In order to formulate the service selection problem into a MCMCF problem, we describe the service selection problem in a mathematical way. Several related notations used in the following discussions are listed in Table I.

On formulating the problem, first, we define the cost function to each link l as follows:

$$C_l = P_l \times 100 / B_l \qquad (1)$$

Specifically, with the cost function, we need to find out the sending rate of each sender considering minimizing the total cost

$$\sum_{r \in R} \sum_{m \in M_r} \sum_{f \in F_m} \sum_{l \in L} BW_{l,f} \times C_l \qquad (2)$$

and meeting the following constraints:

(1) Flow conservation on each node, except for the senders and receivers.

(2) The total sending rate of those senders who provide commodity m to receiver r has to be higher than or equal to the demand of receiver r on commodity m:

$$\sum_{s \in S_{r,m}} X_s \geq Y_{r,m} \qquad (3)$$

(3) All flows of all the commodities from the senders aggregated on l have to meet the bandwidth constraint of link l:

$$\sum_{r \in R} \sum_{m \in M_r} \sum_{f \in F_m} \sum_{l \in L} BW_{l,f} \leq B_l \qquad (4)$$

To solve the above MCMCF problem, we use the tool CPLEX [5], with which the linear MCMCF problem can be solved quickly.

V. Performance Evaluation

In this section, we use the simulator – ns2 [4] to compare our MCMCF-based system to other two different

multi-source streaming systems, greedy-based system and min-cost-based systems.

In the min-cost-based system, we use the same cost function as mentioned in section IV.B for sending rate assignment, but here the algorithm only considers on each local client site, without view of global network. Thus, the min-cost-based system focuses on optimizing the performance for some receiver only. For the greedy-based algorithm, the paths with more available bandwidth are chosen with priority, and the algorithm also runs by each client individually. Under the greedy-based model, receivers can do the service selection in a simple and fast fashion, but the overall performance may not be good. Moreover, both in the min-cost-based and the greedy-based system, the algorithms are run on the clients in random order.

A. Simulation setup

The simulation topology is illustrated in Fig. 3. There are two commodities provided. Each *Si* presents a sender with maximal sending rate, and *Ri* is its relative receiver with minimal requested rate, and each link is depicted with bandwidth in Mbps and link loss rate. For example, for sender S_2, it can offer 3MB/s of the commodity 1 and 2MB/s of commodity 2, and its receiver R_2 requires the playback rate of 3MB/s of commodity 1 only. In addition, due to free edition of CPLEX of software limitation for students, we only can run the simulation on such a 12-node topology.

In the first simulation, the overall average link loss rate is set to 0.25 as Fig. 3 shows. We change the multiple of link capacity with scales from 1.0 to 1.8 of the original environment and measure the overall throughput of the three systems. In the second simulation, however, the scale of link capacity keeps in 1, and the average link loss rate varies between 0.010 and 0.085, which is a reasonable accuracy for the Internet. Both of the simulations measure the performance on the metric of ratio of overall achieved throughput to the total demanded rate on the receiver sides. These simulations run for 10 minutes.

B. Simulation result

The first simulation result is shown in Fig. 4. We can observe that under the condition of limited link capacity, the MCMCF-based system performs better by almost a factor of 1.33 compared to the other two systems. This is because it selects senders and distributes loading in a good fashion, such that the self-congestion among different flows is avoided, and the overall packet loss rate is also minimized. In the instance of higher link capacity, the condition of self-congestion, both in greedy-based and min-cost-based systems, is not so severe as in the instance of lower link capacity, so the differences in throughput among the three systems are also diminished, but the MCMCF-based system still performs better than the other two systems.

Fig. 5 shows the result of the second simulation. We can see that even at high link loss rate, the MCMCF-based system still performs better, and in a general condition, its achieved throughput is about 20% higher than the other systems.

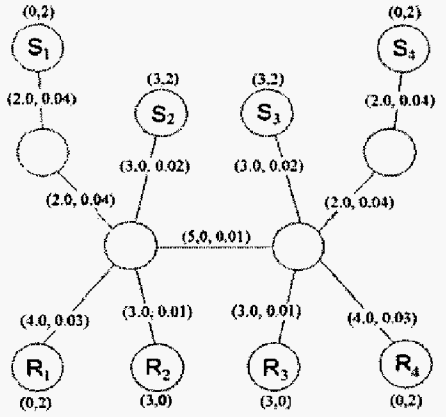


Fig 3. The simulation Topology

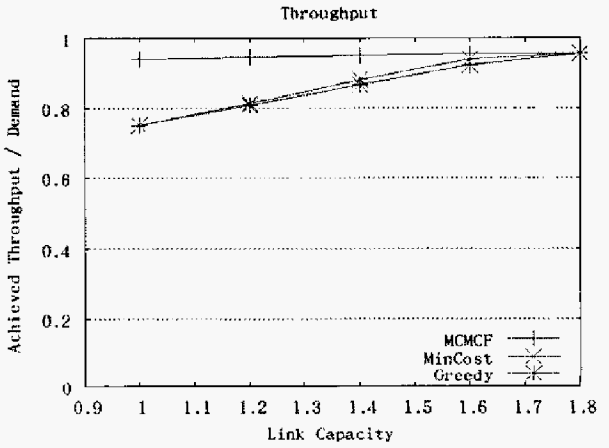


Fig 4. Comparison of average packet loss rate on different link capacity

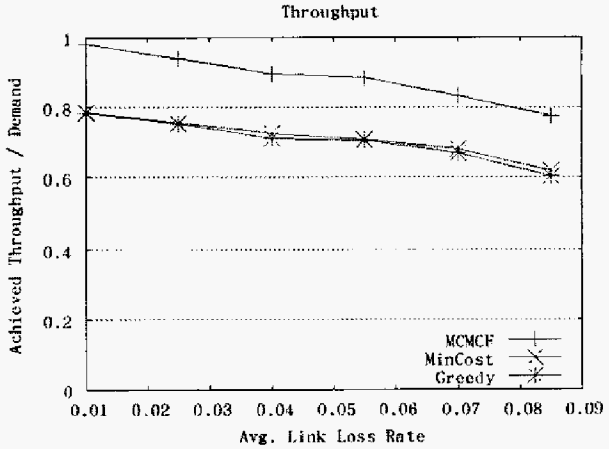


Fig 5. Comparison of average packet loss rate on different link loss rate

From all above, we can conclude that the MCMCF-based system achieves the maximum throughput under bandwidth-limited and loose-link networks; in other words, it leads to the best performance.

VI. Conclusions

In this paper, we presented a system scheme for the media-stream system to better utilize the network resources, and suggested the concept of the multi-source streaming system, which allows users concurrently fetch a streaming object from multiple sources in the network. In order to maximize the total network throughput, we offered a MCMCF-based algorithm for service selection. The MCMCF-based algorithm achieves the optimized performance with the following steps: a) select appropriate servers for a demand, b) distribute the loads among selected servers, and c) avoid self-congestion between different object flows that are requested in parallel by different users. In the simulation evaluation, we compared our system to the greedy-based and min-cost-flow based systems, and results proved that our system greatly outperforms other two systems, given with different link capacities and loss rates under various network models. In addition, we have also implemented our MCMCF-based multi-source streaming system and given the design details in this paper.

VII. Future Work

Our multi-source streaming system can be extended in several directions. First, our system will further consider stability of the sources; that is, to the sources which tend to stay longer in the system are chosen first. This scheme also can improve by modifying the cost function according to individual application. Second, in this paper, the source list is pre-determined. If there are new sources joining the system during stream transmission, then the new coming sources will not be added into the source list. Therefore, we will add the function to handle this dynamic source joining, provide more resources to the clients, and then benefits the overall performance. Third, we will do large-scale testing and evaluate the performance on more metrics to show the advantages of using MCMCF for service selection. Finally, we may deploy the streaming system on some popular P2P networks, such as BitTorrent, with a few adjustments and refinements. With the inherent features of BitTorrent, peers can speed up fetching the stream objects, utilize the network resources, and redistribute the cost of upload from servers to all peers.

References

[1] H. Schulzrinne et. al., Internet Draft: Real Time Streaming Protocol (RTSP), Section 10 of RFC2026, February 16, 2004

[2] D.R. Liu, M.J. Shieh, Y.C. Lee, and W.C. Chen, "On the Design and Implementation of an MPEG-4 Scene Editor", ICCE 2000 Digest of Technical Papers.

[3] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava. PROMISE: Peer-to-Peer Media Streaming Using Collect Cast. ACM Multimedia 2003

[4] NS2: http://www.isi.edu/nsnam/ns/

[5] CPLEX: http://www.ilog.com/products/cplex

[6] J. Castro and N. Nabona, "An implementation of linear and nonlinear multicommodity network flows," European Journal of Operational Research, vol. 92, pp. 37-53, 1996.

[7] T.Nguyen and A. Zakhor. Distributed video streaming over Internet. ACM SIGCOMM 2003.

[8] D. Tran, K. Hua, and T. Do. ZIGZAG: An efficient peer-to-peer media streaming. IEEE ICDCS 2002.

[9] X. Zhang, J. Liu, B. Li, and T.P. Yum, CoolStreaming/DONet: A Data-Driven Overlay Network for Efficient Live Media Streaming. IEEE Infocom 2005.

[10] V. Padmanabhan, L. Qiu, and H. Wang. Server-based inference of Internet link lossiness. IEEE Infocom 2003.

[11] Bram Cohen. Incentives Build Robustness in BitTorrent. May 22, 2003. http://bitconjurer.org/BitTorrent

[12] eMule: http://www.emule-project.net

[13] eDonkey: http://www.edonkey2000.com/

[14] V.N. Padmanabhan, H.J. Wang, P.A. Chou, and K. Sripanidkulchai. Distributing Streaming Media Content Using Cooperative Networking. ACM/IEEE NOSSDAV 2002.

[15] Z.L. Zhang, S. Nelakuditi, R. Aggarwal, and R.P. Tsang. Efficient Selective Frame Discard Algorithms for Stored Video Delivery across Resource Constrained Networks. IEEE Infocom 1999.

[16] B. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. ACM SIGCOMM 1998.

Wei-Cheng Xiao received the B.D degree in Computer Science and Information Engineering from the National Taiwan University in summer, 2004. He is currently a Master student for the first year in the department of Computer Science and Information Engineering in National Taiwan University. His main interest is in the P2P network the streaming system.

Po-Lin Chou received the B.D degree in Computer Science and Information Engineering from the National Taiwan University. He is currently a Master student in the department of Computer Science and Information Engineering. His main interest is in Wireless Sensor Network and Ad Hoc Network.

Tsai-Lin Hsu received her B.S. degree in Computer Science and Information Engineering from National Taiwan University, in 2003. She went on to study for her M.S. in graduate school in Computer Science and Information Engineering, National Taiwan University in the same year. Now, she is a graduate student in her second year. Her research interest is networking, and the recent studying is on wireless Access Networks, especially Ad Hoc and Sensor Networks.

Cheng-Fu Chou received the M.S. and Ph.D. degree from University of Maryland, College Park, in 1999 and 2002, respectively. In 2002, he joined the Department Computer Science and Information Engineering, National Taiwan University. His current research interests are in wide-area network applications, distributed multimedia systems, heterogeneous wireless communication systems, and wireless sensor network and their performance evaluation.