Server Selection Schemes Considering Node Status For a Fault-Tolerant Streaming Service on a Peer-to-Peer Network

Hyunjoo Kim*, Sooyong Kang**, and Heon Y. Yeom*

Abstract: Peer-to-Peer (P2P) networks are attracting considerable research interest because of their scalability and high performance relative to cost. One of the important services on a P2P network is the streaming service. However, because each node in the P2P network is autonomous, it is difficult to provide a stable streaming service on the network. Therefore, for a stable streaming service on the P2P network, a fault-tolerant scheme must be provided. In this paper, we propose two new node selection schemes, Playback Node First (PNF) and Playback Node First with Prefetching (PNF-P) that can be used for a service migration-based fault-tolerant streaming service. The proposed schemes exploit the fact that the failure probability of a node currently being served is lower than that of a node not being served. Simulation results show that the proposed schemes outperform traditional node selection schemes.

Keywords: Peer-to-peer network, Fault-tolerant streaming services

1. Introduction

As the demand for streaming service on the Internet is increasing and the quantity of streaming contents is growing rapidly, it has become difficult to provide a stable streaming service using traditional client/server-based architecture. The new network service model, Peer-to-Peer (P2P) service, addresses this problem. The P2P service is a form of reciprocal network service where each node in the P2P network acts as both a client and a server.

However, while the P2P service can address the problem of client/server service architecture, i.e., load concentration on the server, it is difficult to provide guaranteed quality of service (QoS) because each node in a P2P network is autonomous. In particular, because the streaming service has real-time characteristics, QoS degradation due to node failures is critical. Therefore, for streaming service on a P2P network, a fault-tolerant scheme that can cope with node/link failures must be developed. Previous works on fault tolerance for streaming service in P2P networks can be classified into: 1) data recovery using redundant data, 2) reallocation of the data sending rate from server nodes, 3) special encoding for graceful quality degradation, and 4) service migration.

Before a client node sends a service request to other nodes, it is necessary to select server nodes from many candidate nodes that have the desired object stored, and because the number of nodes participating in a P2P network is large and each node is not only autonomous but all may have different outbound bandwidths, the QoS depends mainly on the node selection scheme. Therefore, it is necessary to devise a good node selection scheme for a streaming service on a P2P network.

In this paper, we propose two new node selection schemes, Playback Node First (PNF) and Playback Node First with Prefetching (PNF-P) that decrease the probability of node failure, and therefore, increase the probability of providing a stable streaming service. The proposed schemes exploit the fact that the failure probability of a node currently being served is lower than that of a node not being served. We evaluated the performance of the proposed schemes when used in a service migration-based fault-tolerant streaming service model.

The organization of this paper is as follows. In Section 2, related works are described, and in Section 3, the models for P2P streaming service are presented. In Section 4, we describe the traditional and proposed server selection schemes. We will show various simulation results that reveal the performance of the proposed framework in Section 5, and present our conclusions in Section 6.

2. Related Work

In previous works, there are two kinds of P2P networks. The first establishes logical relations between nodes to provide service and manage the network. It mainly focuses on the contents distribution from origin node to all the participating nodes. vTrails[8], Allcast[9], SpreadIt[10], CoopNet[11], Narada[12], Nice[13], Zigzag[14] are examples. Narada[12] is for multi-sender multi-receiver streaming applications. It maintains a mesh among peers and establishes a tree when

Manuscript received February 22, 2006; accepted March 3, 2006.
Corresponding author: Sooyong Kang

* School of Computer Science and Engineering, Seoul National University, Seoul, 151-742, Korea ({hjkim, yeom}@dcslab.snu.ac.kr)
** Department of Computer Science Education, Hanyang University, Seoul, 133-791, Korea (sykang@hanyang.ac.kr)

a sender transmits data to receivers. However, Narada is appropriate for a small P2P network. To extend it to a large P2P network, Nice[13] uses the multi-layer hierarchical clusters. Nice uses the head to forward content to its subordinates, thus incurring a high bottleneck. To address this problem, Zigzag[14] maintains the administrative organization representing the logical relationship and the multicast tree representing the physical relationship among peers. The head in Zigzag forwards content to foreign members rather than own members. The second does not establish logical relations between the nodes. It mainly focuses on file sharing. Napster and Gnutella are examples.

Leung[1,2,3] used Reed-Solomon Erasure (RSE) correcting code for data recovery. Although such schemes can cope with a predefined number of node failures, they require additional network bandwidth for redundant data. Thinh[4] proposed a dynamic transmission rate adaptation scheme where multiple server nodes change data transmission rates to the client node in order to minimize packet loss. This scheme distributes the transmission rate of a failed node among the active server nodes, and because it assumes that the sum of the bandwidths of each node is larger than the necessary bandwidth for a normal playback rate, it is impossible to continue service when the aggregated bandwidth becomes smaller than the playback rate because of the multiple node failures. Coopnet[11] proposed Multiple Description Coding (MDC), which enables graceful QoS degradation when packet loss occurs because of node/link failure or link state change. This scheme encodes a continuous media object into multiple independently decodable substreams so that the client node can continue playback despite node/link failures with degraded quality using only the substreams received. Spreadit[10] used a service migration scheme that finds a new server node to replace the failed node. However, because this scheme finds a new server node in the application level multicast tree from the root node, a considerable amount of time is required to find the new server node. Zigzag[14] can recover node failures gracefully and quickly by separating foreign head for forwarding content and own head for organizing clusters. However, Zigzag focuses on live media streaming that has one source peer, and thus, on-demand streaming using multi-sender requires another solution for node failures.

Authors considered video-on-demand streaming and live streaming differently in most previous works because the two types of streaming have fundamental differences. End-to-end delay is more important to live streaming than VoD streaming because the shorter end-to-end delay makes the stream more live. On the contrary, in VoD streaming, it is more important to deliver the whole video to the new joining user. [6,7,10,12,13,14,15,16] are approaches for live streaming in P2P and [1,2,3,5,17,18] are for VoD streaming in P2P. Most approaches for live streaming use application multicast trees which are focused on reducing end-to-end delay and being constructed efficiently whenever a client joins or leaves. When application multicast trees are used for VoD streaming, other additional streams are needed at the beginning to deliver the video to a late client. For example, a patching stream for the beginning part and a base stream for the current part like in P2Cast[18]. A server or another early client can deliver the beginning part to a late client. P2VoD[17] is the similar work to P2Cast, however, it eliminates the use of a patching stream by introducing the concept of generation and a caching scheme. The caching scheme allows a group of clients, arriving to the system at different times, to store the same video content in the prefix of their buffers. Such a group forms a generation. When a late client comes in, it is served by only one stream from an early client. If the early client which serves to the late client leaves during the service, one of the group members replaces it and resumes the service.

In this paper, a client finds candidate server nodes which have(stored) the requested video object and selects a few server nodes from among them. Hence, we focus on VoD streaming service.

3. Peer-to-Peer Service Model

In a P2P network each node acts as both a client and a server node. A client node receives data for playback from one or more server nodes. A server node can provide services to multiple client nodes within the predefined outbound bandwidth. Therefore, a node can receive services from multiple server nodes and can provide services to multiple client nodes, simultaneously. Assuming no node/link failures, the service-requesting node should 1) *search*: find nodes that have the required object, 2) *node select*: receive available outbound bandwidth information from those nodes and select a suitable number of server nodes that will actually provide the service, and 3) *schedule*: assign data segments for transmission to the selected nodes. Because the search is beyond the scope of this paper, we assume that the client node already knows all the nodes that have the required object. We also assume that media objects are encoded into constant bit rate data and are sequences of data segments of equal size.

Every node can act as both a client and a server for a P2P streaming service. When a service request arrives from a client node, the server node first advises the client node of the available outbound bandwidth. Upon receiving the bandwidth information, the client node informs the server node whether it is selected as a server node. If selected, the server node provides the service to the client, and if not, it sends state update information whenever the available outbound bandwidth changes to the client.

After selecting the initial server nodes, the client node requests data transmission and assigns segments to each selected server node, selects a new server node when a fault signal arrives from the buffer monitor, and sends a request for data transmission to the selected new server node. Therefore, the server node should have: 1) a node selector that chooses server nodes from the many nodes that have the demanded object; 2) a segment handler that assigns data segments to each server node for transmission; 3) a service manager that requests data transmission from the server nodes and sends a service finish request to the node that should cease data transmission because of node/link failure or link state change; and 4) a state manager that receives state update information from nodes that are not currently selected as a server node but can be selected later if a server node fails, and transfers that information to the node selector.

There are two types of fault - failure and state change. Failure means that there can be no more service between server node and client node because of server node shutdown or intermediate link failure. State change means that the actual data transmission rate from the server node to the client node becomes lower than the expected rate because of server overload or network congestion. In this paper, we cope with these in a unified manner.

Before starting playback, the client node prefetches data segments in buffers, with each buffer dedicated to a specific server node. The user can determine the prefetching quantities by configuring the buffering time (t_p) in the player. The quantity of data prefetched in a buffer is determined as $t_p \cdot r_i$, where r_i is the transmission rate of server *i*. Therefore, different prefetching quantities for each buffer can be defined to match the transmission rate of each server node, and because at least one segment should be transmitted to each buffer before playback starts, the transmission rate of each server node should be no smaller than s_d / t_p, where s_d is the segment size. Therefore, the client node has to exclude those nodes that have smaller available outbound bandwidth than s_d / t_p from the possible server nodes.

After prefetching data, during the buffering time, the client node starts playback. The volume of data in a buffer does not change significantly unless a fault occurs. However, if a fault occurs, either no more data arrives at the corresponding buffer (failure) or the rate of arrival decreases (state change) while the consumption rate does not change. Therefore, the amount of data in the buffer decreases as time passes. Thus, we can determine the occurrence of a fault by monitoring the volume of data in a buffer.

The client node starts to monitor each buffer when the playback begins. Let δ be the predicted time length from a fault occurrence to when a newly chosen server node starts service. There should be sufficient data for playback during δ in each buffer. Therefore, we can determine the fault detection threshold (θ_i) for *buffer(i)* from the following equation:

$$\theta_i = \frac{r_i}{s_d} \times \delta$$

When a fault is detected, the node selector chooses one or more replacement server nodes (R-servers) that will replace the failed node. To prevent the necessity of segment reassignment to all servers, the aggregated service rate of R-servers is the same as that of a failed node. The segments previously assigned to the failed node will be reassigned to the R-servers.

4. Server Selection Schemes

After searching candidate nodes that have the demanded object, the client node should select server nodes that will collectively provide the service. We considered two previous schemes, Larger available outbound Bandwidth First(LBF) and Smaller available outbound Bandwidth First (SBF), and propose new server selection schemes, PNF and PNF-P.

4.1 Larger available outbound Bandwidth First (LBF)

In the LBF scheme, a node that can provide larger available outbound bandwidth is selected first. Therefore, the number of server nodes can be minimized and we can predict that the probability of a node/link failure will be small. However, the probability that a server node will fail, when using the LBF scheme, is the same as that when the server nodes are randomly selected. This means that the LBF scheme does not decrease the node failure probability. In addition, when a node failure occurs, the overhead to cope with the fault is large because the failed node (which has large available outbound bandwidth) must be replaced by multiple nodes (which may have smaller available outbound bandwidth). Therefore, the recovery time can be large.

4.2 Smaller available outbound Bandwidth First (SBF)

Using the SBF scheme, a node that can provide smaller available outbound bandwidth is selected first. Therefore, the number of total server nodes is maximized and this also maximizes the probability of a node/link failure. In addition, the SBF scheme cannot decrease the node failure probability itself. Therefore, the overall failure probability of SBF is larger than that of LBF. However, the overhead to recover from a failure is smaller than that of LBF because the outbound bandwidth of the failed node is generally smaller than that of other candidate nodes, which means that only one candidate node is sufficient to replace the failed node.

4.3 Playback Node First (PNF)

If we know whether a node is currently being served or not, and when a node is currently being served how much time is left before the service will be completed, then we can decrease the failure probability of a server node. Because the probability that a node currently being served will fail in the course of service is relatively small, it will

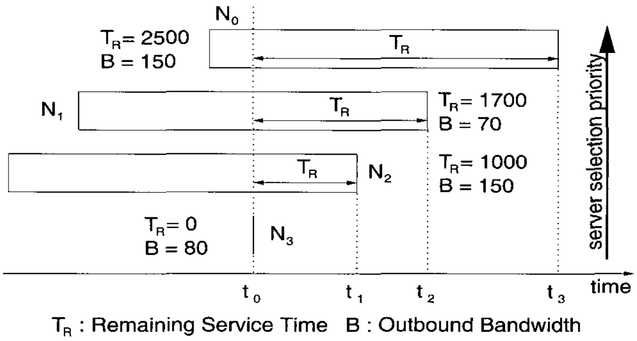


Fig. 1. Playback Node First (PNF) scheme

decrease the overall node failure probability when we select the node that is currently being served first. The Playback Node First (PNF) scheme originated from this idea.

The PNF scheme is based on the LBF scheme. However, when there are nodes that are currently being served, they are selected first. Nodes that have recently started playback are more likely to be selected, which means that any node that has the longer remaining playback time is selected first. If there are nodes that have the demanded object and are playing another object or that do not have the demanded object but are playing it, then the nodes can provide service of the object, and therefore, they can be server nodes. If there are no nodes that satisfy these conditions, server nodes are selected by the LBF scheme. Therefore, the server nodes that are selected by the PNF scheme have a lower probability of failure than those selected by the LBF or SBF schemes.

Fig. 1 shows an example of the PNF scheme. N_0, N_1, and N_2 represent candidate nodes that are currently being served, and N_4 is a candidate node that is not under service.. Assume that a new client node requests service at time t_0 and the bit rate of the requested object is 300 Kbps. Then, N_0, N_1, and N_2 will be selected as server nodes for the new client. Subsequently, it is likely that the new service will be stable until time t_1 when the service for N_2 finishes. From that time, the probability of node failure increases. However, it is still smaller than that of the LBF or SBF scheme because N_0 and N_1 are still currently being served. When the playback time reaches t_3, the failure probability of each node becomes the same as that for LBF or SBF. However, although N_2 fails after t_1 because the PNF scheme selects a new server node for replacement using the playback node first rule, the newly selected node can also be predicted to be stable during its service time.

4.4 Playback Node First with Prefetching (PNF-P)

The PNF scheme can decrease overall node failure probability by decreasing the failure probability of each node. However, because the node failure probability increases after the service of the server node finishes, we propose an enhanced scheme, Playback Node First with Prefetching (PNF-P), which prefetches future data in advance into the storage of the client node. Using the PNF-P scheme, a client node receives the future data from one or more server nodes selected for prefetching. The server nodes for prefetching transmit data in the reverse direction, from the end of the object, and when the playback reaches the point where the remaining data are already prefetched to the local storage, the client node sends a service finish message to all server nodes and uses local data. The server nodes for prefetching are also selected using the PNF scheme.

Table 1. Simulation parameters

Parameter	Value
number of nodes	1500
number of initial servers	20
outbound bandwidth	30
number of objects	20
object playback rate	300 ~ 100 Kbps
segment size	10 Kbits
object size	3600 seconds
buffering time	5 seconds
predicted migration time	2 seconds
request arrival rate	exponential distr. with mean of 0.012
network congestion interval	10000 seconds
network congestion duration	50 seconds
normal state migration time	exponential distr. with mean of 0.1 sec.
congestion state migration	exponential distr. with mean of 1 sec.
time	6250 ~ 20000 seconds
Mean Time Between Failures	
(node/link)	86400 seconds (24 hours)
simulation time	

5. Simulation

In this section, we present and discuss various simulation results. The parameters used for the simulation are shown in Table 1.

The initial servers for a client are those nodes that have the demanded object. We fixed the number of initial server nodes at 20. Therefore, initially, the client should select server nodes from those 20 nodes and the remaining nodes become candidate nodes. The predefined outbound bandwidth to provide services to other nodes varies from 30 Kbps to 100 Kbps with 10 Kbps increments and they are uniformly assigned to all nodes in the P2P network. The predicted migration time is the predicted time from a fault occurrence to when a newly selected server node starts service. The value is used to detect node failure or link state change. The initial buffering time is five seconds, but this does not affect the performance of the system, provided that it is larger than the predicted migration time, which was set as two seconds. We assumed that each node requests an average of one streaming service per day. Therefore, the average request arrival rate is (1/86400)× 1000~0.012. When we considered the link state change we assumed that each path from a server node to the client node is congested following an exponential distribution with mean interval 10,000 seconds and the congested state lasts for 50 seconds. For normal state networks, the actual migration time follows an exponential distribution with a mean of 0.1 seconds, and for a congested network it

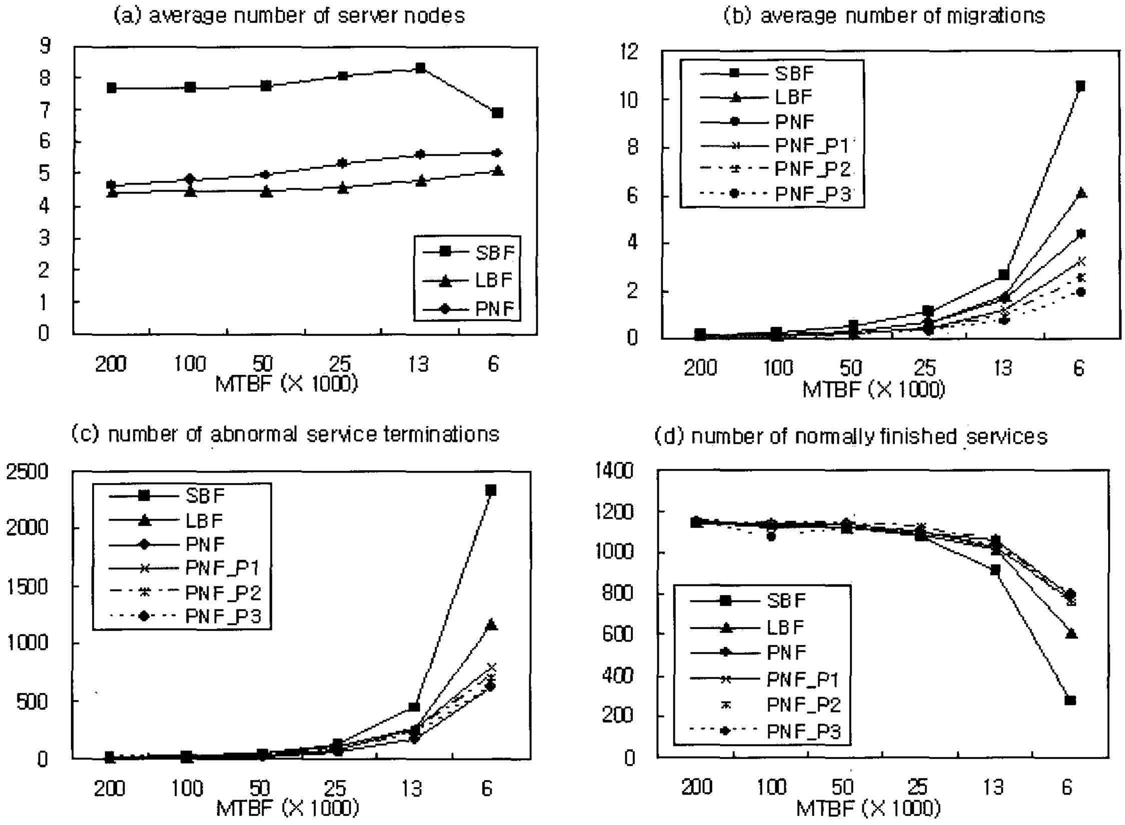


Fig. 2. Simulation results

follows an exponential distribution with a mean of one second. We limited the number of clients that are served concurrently in the P2P network to 50.

Fig. 2 shows the simulation results. As the Mean Time Between Failures (MTBF) becomes larger, less failures occur. Fig. 2-(a) shows the average number of server nodes for each service. As can be seen from the figure, SBF uses more server nodes than LBF because the SBF selects nodes that have the smallest available bandwidth first, whereas the LBF selects nodes with the largest available bandwidth first. In the case of SBF, as the MTBF value becomes smaller, more failures occur, and the average number of server nodes is increased because there can be a server that does not use the whole available bandwidth and has bandwidth left over. For example, consider a failed server with a transmission rate of 70 Kbps and two server nodes, N_1 and N_2, which have data rates of 40 Kbps and 50 Kbps, respectively. Then, N_1 serves with the 40 Kbps data rate and N_2 is sufficient to serve with 30 Kbps data rate. Therefore, as the MTBF value decreases, the number of server nodes increases. However, when the MTBF value is too small, the number of available nodes becomes too small, which means that there are a small number of nodes with small available outbound bandwidth. Therefore, two or more server nodes with small transmission rate can be replaced with one node with large available outbound bandwidth when they fail, which decreases the average number of servers. On the contrary, because the number of nodes with large available bandwidth also decreases as the MTBF value decreases, the average available bandwidth of server nodes selected by LBF decreases, which increases the number of server nodes for a client using LBF, and because the PNF scheme is based on the LBF scheme, the number of server nodes is similar to that of the LBF scheme. However, because the PNF scheme disobeys the LBF rule when there are nodes currently being served, it uses slightly more server nodes than LBF.

Fig. 2-(b) shows the average number of service migrations caused by a node failure or network congestion. When a server node fails, the service from the failed node should migrate to another server node selected for replacement. From the figure, the number of migrations increases as the MTBF decreases, and because the PNF scheme guarantees almost no failures during the playback time of the server nodes, the number of migrations is smaller than for the LBF and SBF schemes. For the PNF-P scheme, additional server nodes are used for prefetching. However, because a client can prefetch part of the demanded object in advance and use local data instead of receiving data from server nodes when the playback reaches the time when the remaining data are already stored in the local storage, the actual service time decreases. Therefore, the average number of node failures also decreases. In the figure, PNF-P1, PNF-P2, and PNF-P3 use one, two, and three server nodes for prefetching, respectively. As the number of server nodes for prefetching increases, the actual service time decreases, and therefore, the average number of node failures also decreases.

In Fig. 2-(c) and (d) are the number of abnormal service terminations and the number of normally finished services, respectively. When a server node fails and there are insufficient candidate nodes to replace the failed node, the service cannot continue and playback is terminated. This is called an abnormal service termination. As more failures occur, the number of candidate nodes decreases and the probability of abnormal service termination increases because of the lack of suitable candidate nodes. Since the PNF scheme has a smaller number of node failures, as shown in Fig. 2-(b), the number of abnormal service terminations is smaller than that for LBF or SBF. Correspondingly, the number of normally finished services is larger than that for LBF or SBF, and because the PNF-P scheme uses more server nodes than the original PNF scheme, the number of abnormal service terminations for PNF-P is slightly larger than that of the original PNF scheme, and, correspondingly, the number of normally finished services for PNF-P is slightly smaller than that of the original PNF scheme.

6. Conclusion

To provide a stable streaming service on a Peer-to-Peer network, a fault tolerance scheme must be devised to deal with node failures. One of the traditional fault tolerance schemes is a service migration from the failed node to a new node. When using a service migration-based fault-tolerant scheme, it is important to select relatively stable nodes as server nodes. In this paper, we proposed new node selection schemes for a service migration-based fault-tolerant streaming service on Peer-to-Peer networks. The proposed schemes, Playback Node First (PNF) and Playback Node First with Prefetching (PNF-P), are based on the fact that the failure probability of a node currently being served is lower than that of a node not being served. The proposed schemes decrease the average number of node failures, and therefore, increase the probability of providing stable streaming service. From the simulation results, we found that the proposed schemes decrease the number of node failures, and therefore, provide a more stable streaming service on Peer-to-Peer networks.

References

[1] W. T. Leung and J. Y. B. Lee, "A server-less architecture for building scalable, reliable and cost-effective video-on-demand systems," Proc. of Internet2 Workshop on Collaborative Computing in Higher Education: Peer-to-Peer and Beyond, 2002.

[2] Jack Y. B. Lee and W. T. Leung, "Study of a Server-less Architecture for Video-on-Demand Applications," Proc. of the IEEE International conference on Multimedia and Expo., 2002.

[3] Jack Y. B. Lee and W. T. Leung, "Design and Analysis of a Fault-Tolerant Mechanism for a Server-less Video-on-Demand System," Proc. of the International conference on Parallel and Distributed Systems, 2002.

[4] T. P. Nguyen and A. Zakhor, "Distributed video streaming over internet," Proc. of the Multimedia Computing and Networking, 2002.

[5] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," Proc. of NOSSDAV, 2002.

[6] H. Deshpande, M. Bawa, and H. Garcia-Molina, "Streaming live media over a peer-to-peer network," Tech. Rep. TR 2001-30, Stanford Database Group, 2001.

[7] D. Xu, M. Hefeeda, S. Hambrusch, and B. Bhargava, "On peer-to-peer media streaming," Proc. of International Conference on Distributed Computing Systems, 2002.

[8] vTrais, http://www.vtrails.com

[9] Allcast, http://www.allcast.com

[10] H. Deshpande, M. Bawa, and H. Garcia-Molina, "Streaming live media over a peer-to-peer networks," Tech. Rep. TR2001-30, Stanford Database Group, 2001.

[11] Venkata N. Padmanabhan and Helen J. Wang and Philip A. Chou and Kunwadee Sripanidkulchai, "Distributing Streaming Media Content Using Cooperative Networking," Proc. of NOSSDAV 2002

[12] Yang-Hua Chu, Sanjay G. Rao and Hui Zhang, "A case for end system multicast," ACM SIGMETRICS 2000.

[13] S. Banerjee, Bobby Bhattacharjee and C. Kommareddy, "Scalable application layer multicast," ACM SIGCOMM 2002.

[14] Duc A. Tran, Kien A. Hua and Tai T. Do, "Zigzag: An efficient peer-to-peer scheme for media streaming," Proc. of IEEE INFOCOM 2003.

[15] Xinyan Zhang, Jiang-chuan Lin, Bo Li, Tak-Shing and Yum P., "Coolstreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming," Proc. of IEEE INFOCOM 2005

[16] Padmanabhan V.N., Wang H.J. and Chou, P.A., "Resilient peer-to-peer streaming," Proc. of IEEE International Conference on Network Protocols, 2003

[17] Do, T.T., Hua, K.A. and Tantaoui, M.A., "P2VoD: providing fault tolerant video-on-demand streaming in peer-to-peer environment," Proc. of IEEE International Conference on Communications Society, 2004

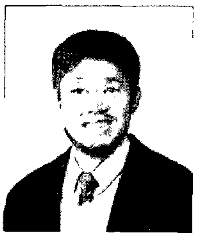
[18] Yang Guo, Kyoungwon Suh, Jim Kurose and Don Towsely, "P2Cast: Peer-to-peer Patching Scheme for VoD Service," WWW 2003

Hyunjoo Kim

She received her BS in physics from Ewha Womens University and MS degree in computer science from Seoul National University, Seoul, Korea in 1996, 2002 respectively. She was with Ssangyong Information and Communications Corp., a system integration company, from 1996 to 1999. Currently she is a PhD candidate at Seoul National University. Her research interests include multimedia, P2P, grid systems, distributed systems, etc.

Sooyong Kang

He received his B.S. degree in mathematics and the M.S. and Ph.D degrees in computer science, all from Seoul National University (SNU), Seoul, Korea, in 1996, 1998 and 2002, respectively. He was then a Postdoctoral Researcher in the School of Computer Science and Engineering, SNU. He is now with the Department of Computer Science Education, Hanyang University, Seoul. His research interests include multimedia systems, especially multimedia data transmission system and multimedia storage system. He is also interested in the distributed computing systems, Grid computing and file systems for nonvolatile memories including NAND flash, PRAM, FRAM and MRAM.



Heon Y. Yeom

He is a Professor with the Department of Computer Science and Engineering, Seoul National University. He received his BS degree in computer science from Seoul National University in 1984 and received the MS and PhD degree in computer science from Texas A&M University in 1986 and 1992, respectively. From 1986 to 1990, he worked with Texas Transportation Institute as a Systems Analyst and from 1992 to 1993, he was with Samsung Data Systems as a Research Scientist. He joined the Department of Computer Science, Seoul National University in 1993, where he currently teaches and researches on distributed systems, multimedia systems and transaction processing, etc. His research interests includes distributed computing systems, Grid systems and e-Science.