Smart-Fit: Peer-to-Peer Topology Construction Strategy for Live Video Streaming towards Minimal Delay

Chun-Hao Chang, Peng-Jung Wu, Chung-Nan Lee
Department of Computer Science, National Sun Yat-sen University, Kaohsiung, Taiwan
M953040034@student.nsysu.edu.tw, wupl@cse.nsysu.edu.tw, cnlee@cse.nsysu.edu.tw

Abstract*-Due to the fast growing bandwidth of Internet users, the P2P live video streaming on the Internet is feasible today. In this research, we proposed a topology construction method: Smart-Fit, towards minimizing the transmission delay between users and video server. The concept is based on minimizing the hop counts between server and users, in further to reduce the delay. The proposed method builds a predicted ideal template. Then the topology is constructed according to the template. Simulation results show the proposed method successfully reduces the hop count and the transmission delay between users and server. Moreover, due to the reduction of hop count, the packet loss rate is also reduced.*

Keywords: P2P, Topology, Construction, Delay.

1. Introduction

The existing P2P topology construction method for peer-to-peer streaming can be categorized towards two groups: random-neighbor and tree-based. Random-neighbor approach such as Coolstreming [1] and GridMedia [2], are robust to the dynamic nature of peers. The random-neighbor approach distributes data to the random neighbors, with either "push" or "pull" modes and improves peers utilization efficiency according to their local information; consequently, the choosing of neighbor is not global optimum for a specific metric, for example, the average delay from server to peers. Outreach [3] is proposed by Small et al., a topology construction method reduces server bandwidth costs.

The tree-based approach such as Splitstream [4] and CoopNet [5], have been proposed. ZIGZAG [6] uses hierarchical clustering to manage topology while a system is in a large scale. However, the algorithms are complicated and may not be accommodated for high-transient environment.

In contrast to the benefits of the P2P architecture, most of the existing studies fall short of addressing the overlay topology construction for a live streaming environment towards minimizing video delay between server and peers. In a real time broadcasting session, such as sports event or financial information, the algorithms to construct such topology are necessary.

In this paper, a topology construction strategy, Smart-Fit, is proposed. The general objective is to construct an overlay topology with minimal delay and packet loss rate by reducing the peers' average peer hop count to video server. Our contributions are the following. First, we present the framework that minimizes the peers' average peer hop count in an ideal environment with complete knowledge of all participating peers. Second, based on the perception of the framework, we present the Smart-Fit algorithm to construction the topology. Since the topology with strictly minimum average hop count is not possible because of the transient nature of peers, we build the topology towards a goal template. The template represents the predicted minimum average hop count topology after ΔT time. Finally, by minimizing the average hop count, the packet delay and the packet loss rate are reduced significantly.

The rest of the paper is organized as follows. Section 2 introduces the preliminaries and definitions. Section 3 presents the related strategies. Section 4 presents the proposed method. Section 5 brings the experimental results and the conclusions are drawn in Section 6.

2. Preliminaries and Definitions

In a binary tree structure having two levels as shown in Figure 1, peers in level one (peer 1 and peer 2) obtain all streaming data directly from server, and the peers in level two obtain the streaming data from peers in level one. As long as the system grows up, the depth of the tree increases, which introduces network delay. Network delay is the time taken for a packet to traverse the network. When a server broadcasts a streaming packet, peer 1 receives the packet first and then forwards to peer 4. The time peer 4 receives the packet equals the time peer 1 receives the packet plus network delay between them. It is easy to understand that the peer who has larger hop count from the server suffers larger network delay.

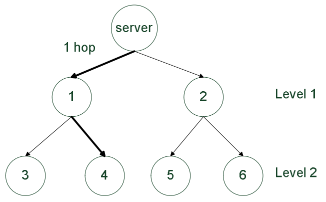

Figure 1. An example of the binary tree.

2.1. Definition of Average Hop Count

Let the peer who is connected to server as parent be called a "level 1 peer", q_1, which has the distance one hop count to the server; in the same way, a peer who is connected to a "level n-1 peer" as parent is called a "level n peer", q_n, which has the distance n hop counts to the server. Suppose there are n levels in the current session. Let the set of level n peer Q_n, be: $Q_n=\{q_n, q_n, \ldots, q_n\}$. Let the number of peer in the set Q_n, be m_n. Then the peers' average hop count (distance) to the server, D, is defined as:

$$D = \frac{(m_1 \times 1 + m_2 \times 2 + \ldots + m_n \times n)}{m_1 + m_2 + \ldots + m_n} = \frac{\sum_{i=1}^{n}(m_i \times i)}{\sum_{i=1}^{n} m_i}$$

2.2. Problem Statement

Since the peer's hop count to server is directly related to delay, based on a given topology Y, the goal is to build a topology after a duration ΔT, without moving the existing peers in Y, such that the global peers' average hop count to server is minimum. Suppose that there are n types of different uplink peers, without losses of generality, these n types of peer have the same willing to enter the session; thus, each type of peer has different arrival rate. Let λ_i be the arrival rate of type i peer. And every participating peer has a departure rate μ. Let N_i be the number of type i peer in the current session and Y be the current session topology. The peers' average hop count (distance) to server is D. The problem statement is: Given Y, $\{N_1, N_2, \ldots N_n\}$, $\{\lambda_1, \lambda_2, \ldots \lambda_n\}$, μ, ΔT, to find a topology Y' after ΔT, without moving existing peers in Y, such that D is minimized.

3. Related Strategies

In the Smart-Fit, we seek to construct the best topology towards minimizing the average hop count between the server and participating peers. To design such a superior topology, we first take insight of the characteristic of some existing and straightforward strategies previously proposed.

3.1. Strategy: Random

One simple and possible approach to arrange peers in the topology is to connect them randomly. When a new peer joins the session, it randomly connects to a participating peer, including server, who has sufficient upload bandwidth to forward the streaming.

Figure 2 shows an example of the random topology. The black node is the streaming server. The white peers are the participating peers. The number inside a peer represents the max number of children it can support. In reality, different peers have different uplink bandwidths, so the number of each peer is not quite the same. The index besides a peer represents the session entry order.

This kind of strategy is simple, flexible, and easy to implement. However, the random strategy does not take any metric into optimization. In Figure 2, the minimum hop count objective is clearly not achieved.

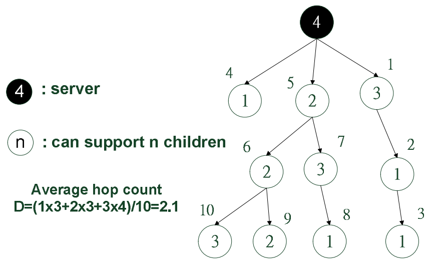

Figure 2. An example of the random topology.

3.2. Strategy: Best-Place First

Since the goal is to achieve minimum hop count between streaming server and participating peers, one straightforward strategy is to assign new arrival peer to the closest position to the server.

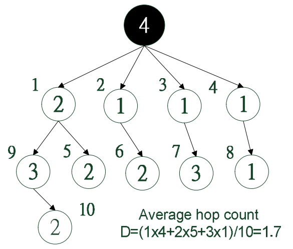

Figure 3. An example of the best-place first topology.

Figure 3 shows and example of best-place first strategy. When a new peer arrives, it first contacts the server. The server first checks if it has available bandwidth. If yes, the new joining peer connects to the server. Else, the server picks the peer who has the smallest distance to it and has enough available bandwidth as the parent of the new joining peer. If there are peers having same distance to server, the server picks randomly one from them. The new joining peer then connects to the assigned parent.

3.3. Framework of Minimum Hop Count

Although the Best-Place First assigns the closest position to server for each new peer, it does not guarantee the overall topology having a minimum average distance; it might fall into the trap of local optimum. Figure 4 shows a smaller average distance topology compared to Figure 3. The problem is the Best-Place First inserts peer only based on the current information. If some topology awareness mechanisms are added and the behaviors of peers are predicted, the system may insert peers in a more efficiency way.

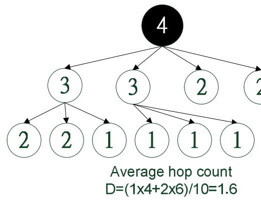

Figure 4. A better arrangement than the Best-Place First in Figure 3.

From a simple view point of the data structure: assuming there are a fixed number of nodes with different maximum out-degree, building these nodes into a tree structure with minimum tree height is making the largest out-degree node as the root and connecting the remaining nodes in a descend way with respect to their out-degree. That is, making the larger out-degree peer at the higher level of the tree. The following examples briefly explain the view. Assuming there are one node with maximum out-degree three, one node with maximum out-degree two, and two nodes with maximum out-degree one. The tree structure builds with these nodes having the minimum tree height is the leftmost topology in Figure 5. No other methods build the tree with smaller tree height, neither the middle one nor the rightmost one in Figure5. Inversely, putting the smaller out-degree nodes at higher level of a tree makes the tree the highest, like the rightmost one.

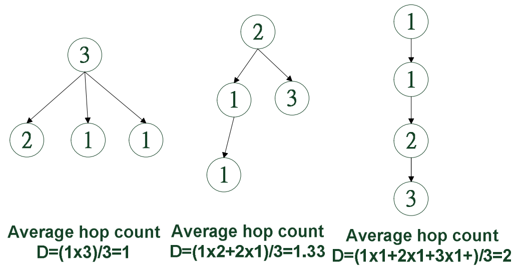

Figure 5. Framework of min. hop count.

3.4. The Placement Dilemma

In reality, the peers' behaviors are dynamic. So it is impossible to make the topology always having minimum average hop counts. The dynamic of peers brings insertion dilemma as illustrated in Figure 6(a). Suppose there are three types of peers who can support three, two, and one children respectively. When there is a moderate type of peer entering, it's hard to decide where it should connect to. From the view point of locally optimal, the new peer should connect to the server for reaching the minimum tree height. However, if the next entering peer is the strong uplink type, like Figure 6(b), the strong uplink peer is forced to connect to the lower level than the moderate peer, violates the objective of placing stronger uplink peer at higher level of the topology. The other way is to connect it as level two peer, reserving server bandwidth for future stronger uplink peer. Because the dynamic of peer's behavior, the next entering peer may be a weak one, like Figure 6(c). This makes the reservation in vain and causes the topology grows meaninglessly higher.

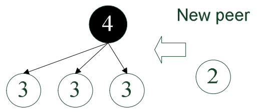

Figure 6(a). The insertion dilemma.

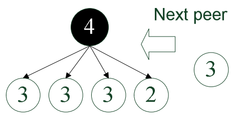

Figure 6(b). Dilemma case 1.

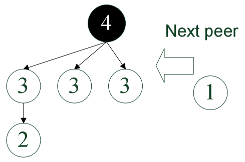

Figure 6(c). Dilemma case 2.

4. The Proposed Method

Because of the transient nature of peers makes the topology always having minimum average hop counts impractical. Based on the uplink bandwidth statistic of the real world peers [7], the proposed Smart-Fit strategy uses current information and the predicting model to build a sample template after *ΔT*, which represents the topology with minimum average hop counts to server. Smart-Fit inserts new peer with respect to the template. Once the peers' behaviors match the predicting template, then the topology has the minimum average hop counts after *ΔT*.

4.1. Building Template

The template is built based on the known information: the arrival rate of each type of peer $\{\lambda_1, \lambda_2, \ldots \lambda_n\}$, the number of each type peer in the current session $\{N_1, N_2, \ldots N_n\}$, peer departure rate μ, and ΔT. The model below builds the template after ΔT.

The predicted number of arrival peer of type i, Ai, i=1 to n, during ΔT is:

$$A_i = \Delta T \times \lambda_i$$

The predicted number of leaving peer of type i, Li, i=1 to n, during ΔT is:

$$L_i = \Delta T \times N_i \times \mu$$

Therefore, the predicted changing number of type i peer, P_i, i=1 to n, during ΔT is:

$$P_i = A_i - L_i = (\Delta T \times \lambda_i) - (\Delta T \times N_i \times \mu)$$

Because there are n types of peers, the predicted changing number of total peers after ΔT is:

$$\sum_{i=1}^{n} P_i = \sum_{i=1}^{n} ((\Delta T \times \lambda_i) - (\Delta T \times N_i \times \mu))$$

After the predicted changing number of each type of peer is calculated, the template after ΔT can be built. The template is built in a top-down manner, since the high uplink peers need to be placed at the top of the topology; it starts with the strongest type n peer. The predicted changing number of type n peer is P_n, because they are expected to but haven't come yet, so the predicted changing peer is called the "virtual peer". These P_n "virtual peers" need to be placed into the template. The process is simple: according to the available bandwidth in current template, place these P_n virtual peers as close to server as possible. And after all the P_n virtual peers are placed into the template, it takes turn to the second strongest uplink virtual peer, type n-1 with P_n-1 virtual peers. The process continues until all types of predicted virtual peers are inserted to the template.

Figure 7 shows the example. At time now, there is an existing topology, and we wish to generate the template topology after ΔT, ΔT=10 seconds. Assume the statistic data shows there are two types of peer whose upload and download bandwidth is larger than the video rate; the type one peer takes account for 2/3 of the total peer and type two peer takes account for 1/3. Assume there are 0.6 peers entering the session every second on average and each peer has the mean service time of 50 seconds, equals the departure rate μ=0.02. Then after time period ΔT, it is expected to increase [(10×0.4)-(10×4×0.02)]=3 type one peers and [(10×0.2)-(10×2×0.02)]=2 type two peers in the session.

The dot-line nodes are virtual peers. The template is first built from inserting type two virtual peers. In this case, the closest available locations to server are two quotas at level two, so these two type two virtual peers are placed at level two. After all the type two virtual peers are inserted, it begins to insert the type one virtual peers. Now the closest available locations to server are at level three, so the type one peers are placed at level three. The built template is like Figure 7. The template indicates the quotas of each type of peer at different levels during ΔT. In this example, during ΔT, there are two quotas for type two peer at level 2, and three quotas for type one peer at level 3. After the template is built, the peer who joins during ΔT is placed according to the template

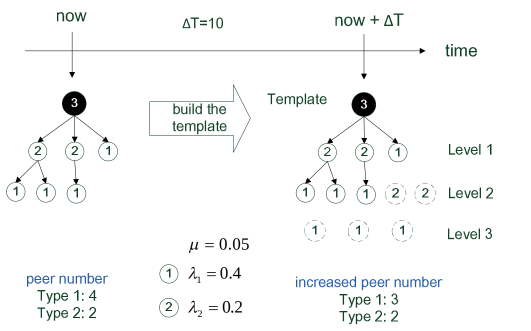

Figure 7. Building template.

4.2. Arrival and Departure of Peer

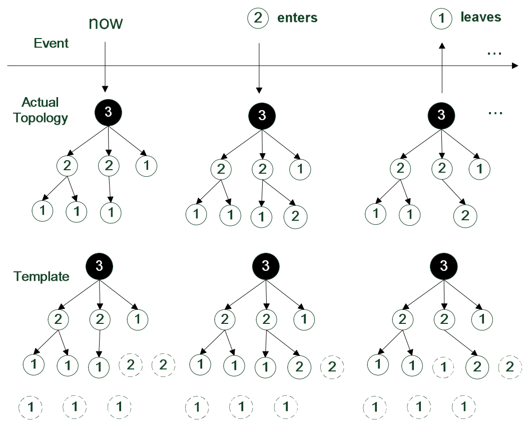

Figure 8. Arrival and departure of peers.

Figure 8 illustrates the process when a peer arrives and leaves. At time now, the template is already built. When a type two peer enters, it first checks the template for the closest type two virtual peer to server; it finds two quotas at level two so it is randomly connected to a level one peer. Then, a type two virtual peer in the template is replaced with the real peer and link. The joining process is done. Next, when the middle type one peer leaves, the real link in the template is removed and it is changed to a virtual peer. The process continues, if the peers behave just like predicted, then the peers in the topology has the minimum average hop count to server after ΔT. And after ΔT, the next new prediction is made and the new template is built, it follows the new template to insert the joining peer afterwards. We only care about placing each peer to the right level; connecting to which parent is not specifically restricted. The process repeats until the session is over.

4.3. The Deviations

Since the template is built from the prediction with statistic data, there is time when the peers' behaviors out of the prediction, for example, a certain type of peers arrive too many and exceed the predicting number in template. The Smart-Fit makes new prediction and new template at the moment deviation happens and builds the topology heading for the new goal. After the new template is made, the new peer would be able to insert to the topology again.

5. Experimental Results

In this simulation, the video rate is 256kbps and the server's upload bandwidth is 10Mbps. There are 5 different types of peer, which is set according to the statistic data mentioned in [1]. From type one peer to type five peer, they have upload bandwidth of 4096Kbps, 2048Kbps, 1024 Kbps, 512Kbps and 256Kbps respectively. The arrival rate of type tree peer is 0.5 and the others are 0.125. The service time of participating peers are random variable with exponential distribution with mean 1800 seconds. The peers' arrival behaviors are random variable follow the exponential distribution with their respective arrival rate. In the ns-2 simulations, we use the GT-ITM Generator to create 600 nodes transit-stub graph as the underlying network topology. Amount them are 24 transit nodes and 576 stub nodes. The 24 transit nodes distributed in 4 transit domains with 6 transit nodes in each one. Each transit node connects three stub domains and there are 8 stub nodes in each one. The participating peers are randomly connected to the stub nodes. There is only server with no participating peer at the session startup time. Each simulation runs 10 times and the results are the average of it.

Figure 9 shows the results of peers' average hop count to server with different strategies. Random strategy performs the worst because it doesn't take any metric into optimization. During the early period of the session, the Best-Place First and the Smart-Fit ΔT=50 perform better. However, when the time goes on, the Best-Place First always connects the joining peer to the highest level of the topology ignoring the global objective, leaving low upload bandwidth peers at the high level of the topology, causing the performance to drop down. When the system approaches the stable state, the Best-Place First performs worse than any of the Smart-Fit strategies.

Smart-Fit performs the best when predicting period ΔT=300 and ΔT=500, that is, predicting the future peers' behavior and making new template as the topology growing goal every 300 to 500 seconds. The small ΔT causing the predict period too short, can't precisely predict the trend of the peer in a macro scope way, making a result toward the local optimal, just like the Best-Place First. However, when the ΔT is too large, the fault tolerance of the system becomes bigger, this means when the topology grows to a wrong way, it takes longer time to be aware of and makes new goal and correction, which also brings negative effect to the topology. The Smart-Fit reduces 17% of hop count compared to the Best-Place First.

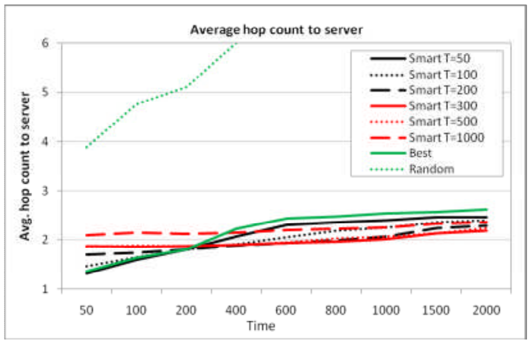

Figure 9. Average hop count to server.

Figure 10 shows the result of the average packet delay. The smaller hop count means smaller delay. The result of this simulation matches the trend in previous section. Among them, the Smart-Fit still performs best and reduces the packet delay up to 19% compared to the Best-Place First.

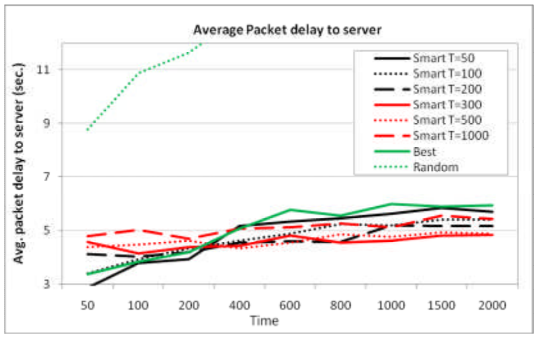

Figure 10. Avg. packet delay to server.

The packet loss rate is directly related to the hop count of peers; the more links and hops a packet traverses from the source to the destination, the higher the loss probability of the packet is.

Figure 11 shows the average packet loss rate from video server to each participating peer. In this simulation, the packet loss probability among each link is a random variable with uniform distribution between mean 0 and 0.05. The results show that the Smart-Fit not only reduces the delay of the packet transmission but also reduces the packet loss rate by minimizing the hop count between peers. The Smart-Fit reduces the packet loss rate up to 10% compared to the Best-Place First.

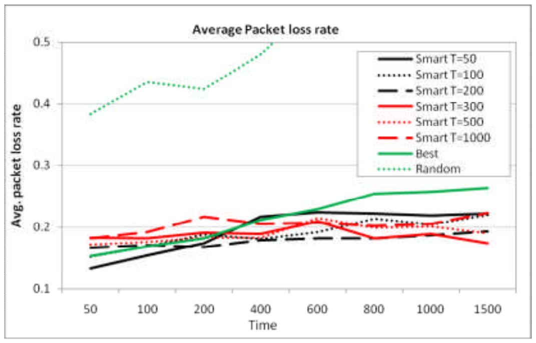

Figure 11. Packet loss rate.

The Smart-Fit strategy arranges new arrival peers according to a template which is built with current peers and future predicted peers. Thus, there is a possibility that the Smart-Fit assigns a peer to connect a certain level parent that haven't enter yet. In this case, the peer has to wait for the correct parent to enter, starting the data transmission. Figure 12 shows the waiting percentage of the total peers. With the proper setting of the ΔT, about 4% to 6% percent of the peers suffer a waiting period. Figure 13 shows the average waiting time of them. Again, with the proper setting of ΔT, the peers who suffer the waiting period only need to wait about 2 to 4 seconds for parent. Both of the waiting percentage and the waiting time are considered pretty low.

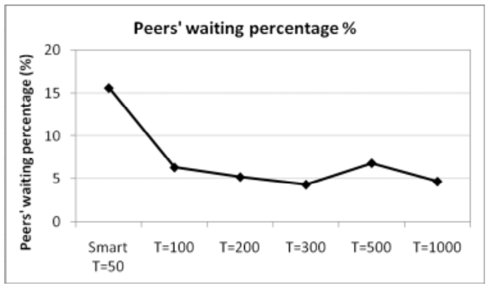

Figure 12. The waiting percentage.

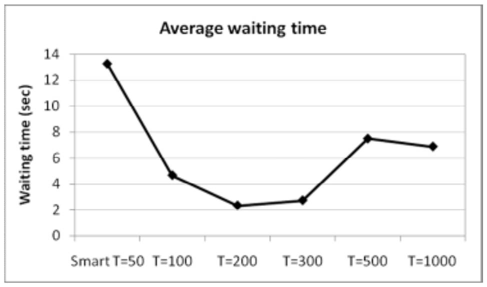

Figure 13. Average waiting time.

6. Conclusions

In this paper, the Smart-Fit, a strategy of building P2P topology for live media streaming toward minimizing the delay between broadcasting server and each participating peers is proposed. The other straightforward methods are compared. Simulation results show that among them the Smart-Fit looks the most promising. The Smart-Fit reduces packet delay up to 19% compared to the Best-Place First and reduces packet loss rate up to 10% when the network condition is poor.

References

[1] X. Zhang, J. Liu, B. Li and T. P. Yum, "CoolStreaming/DONet: A DataDriven Overlay Network for Efficient Live Media Streaming," in Proceedings of the IEEE INFOCOM'05 Conference, pp.2102-2111, Mar. 2005.

[2] M. Zhang, L. Zhao, J. L. Y. Tang and S. Yang, "GridMedia: A Multi-Sender Based Peer-to-Peer Multicast System for video streaming," in Proceedings of the IEEE ICME'05, pp.614-617, Jul. 2005.

[3] T. Small, B. Li, and B. Liang, "Outreach: Peer-to-Peer Topology Construction towards Minimized Server Bandwidth Costs," IEEE Journal on Selected Areas in Communications, vol. 25, no. 1, pp.35-45, Jan. 2007.

[4] M. Castro, P. Druschel, A.-M. Kermarrec, A. Rowstron and A. Singh, "SplitStream: High-bandwidth Content Distribution in Cooperative Environments," in Proceedings of the 2nd International Workshop on Peer-to-Peer Systems, vol. 2735, pp. 292-303, Feb. 2003.

[5] V. Padmanabhan, H. Wang, P. Chou and K. Sprianifkulchai, "Distributing Streaming Media Content using Cooperative Networking," in Proceedings of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video, pp. 177-186, May. 2002.

[6] D. A. Tran, K. A. Hua and T. T. Do, "A Peer-to-Peer Architecture for Media Streaming," IEEE Journal on Selected Ares in Communications, vol. 22, no. 1, pp. 121-133, Jan. 2004.

[7] S. Saroiu, P. K. Gummadi and S. D. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," in Proceedings of the Multimedia Computing and Networking, Jan. 2002.