An Optimized Topology Maintenance Framework for P2P Media Streaming

Rui Guo[1,3], Longshe Huo[2], Qiang Fu[2], Shuangjia Chen[1,3], and Wen Gao[1,2]

[1] Institute of Computing Technology, Chinese Academy of Sciences, 100080 Beijing, China
[2] Institute of Digital Media, Peking University, 100871 Beijing, China
[3] Graduate School, Chinese Academy of Sciences, 100039 Beijing, China
{rguo, lshuo, qfu, sjchen}@jdl.ac.cn, wgao@pku.edu.cn
http://idm.pku.edu.cn

Abstract. In this paper, we present an optimized topology maintenance framework used in AVStreamer P2P media streaming system. To help new incoming clients get an initial view of the whole network, a new weak hierarchy model is designed to share burdens among different servers. Based on this new model, several key techniques are adopted: node evaluation criteria which can help optimize the node-list cache, receiver-driven gossip which is used to update node-list cache from others more effectively, random walk algorithm which can make the network more random, and status polling which is used to ensure the aliveness of each node cached. Experimental results show that the proposed techniques are effective.

Keywords: Peer-to-Peer, media streaming, topology maintenance, weak hierarchy model, receiver-driven gossip.

1 Introduction

Recently, P2P (Peer-to-Peer) technique has been widely used in the realm of media streaming. As a distributed system, it can support a very large amount of users in the condition that few equipments and network bandwidth are needed at the server's side. However, there still exist some challenging problems. On the one hand, since the experiences of P2P related technique are almost originated from the researches of P2P file sharing, and there are lots of differences between media streaming and file sharing, therefore the experiences should be modified to fit this new realm; on the other hand, because of distribution, dynamic peer failures may lead to certain damages to the living peers, which further result in poor quality of service at the client side.

To address these issues, several P2P structures have been put forward, which are more or less different from structures in file sharing systems. Tree-based structure, such as PeerCast [1] etc., is one of them. The idea is from IP multicast, except that the users in IP multicast are more likely to be steady and similar. Such a structure is not suitable for scenarios where there is a high probability of node failures.

Next to the tree-based structure, some improvements are proposed, among which multi-connected structure is mentioned, such as P2PCast [2] etc. Though these structures still cannot solve all of the above problems, they make important progress in the field of P2P media streaming.

In [3], Zhang et al. proposed to use unstructured overlay network. From the view of topology maintenance, this paper gives some important points: (1) There is no need to maintain a certain structure; (2) multi-connection should be used to get data from other peers; (3) backup peers should be maintained to recover from failures of connected nodes. Such points have been accepted by most people, and have been implemented in lots of present live P2P media streaming systems [4]. But new problems appeared soon. Where to get the backup peers? How to decide which one should be preferred as a backup peer and which one should be preferred as a connected peer? What should the whole network look like? How to ensure the data transition is on the rails?

Based on previous works, recently we developed a system named AVStreamer [5], which is the first P2P media streaming system supporting the Chinese AVS standard (Audio Video coding Standard). This paper focuses mainly on the topology maintenance aspect of AVStreamer, and tries to answer the above questions encountered in the development of AVStreamer.

2 Framework of AVStreamer

As a distributed system, the framework design of a P2P streaming system from global view is very important. It is desirable to make new incoming nodes get robust service quickly and be scalable to provide service in a large extension. To meet these requirements, in our AVStreamer system four kinds of nodes are designed: Node-List Server (NLS), Normal-Client Node (NCN), Leased-Client Node (LCN), and Media Fetching Node (MFN). In this section, we first describe these nodes and their mechanisms respectively, and then propose a P2P topology maintenance model named weak hierarchy model based on them.

2.1 Node-List Server

The aim of designing Node-List Server (NLS) is to give the new incoming client node an initial list of nodes, so as to help them build their own view of the whole topology. To ensure that the initial node-list can serve various nodes and to avoid overload on some certain nodes, the capacity of the node-list cache on this server should be sufficient enough, and the contents of the cache should be updated periodically. To ensure the activeness of the cached nodes, we restrict them in two categories: (1) nodes which announce their activeness periodically to this server; (2) nodes which have just joined in the network. The criterion to determine which node should be stayed in cache lays in the capability of each node. When a node's capability has been exhausted, the Lease will be released. By using this criterion, it can be guaranteed that there are enough unsaturated nodes for new incoming nodes.

However, there still exists a serious problem: since new nodes are likely to be more unsaturated [6], the topology constructed using this method may become long and narrow, and the probability of division is higher than that of a totally random topology. To address this problem, we introduce two mechanisms to restrain the new incoming nodes. First, new nodes are refused to be used as Leased node according to a certain probability, so as to let more old nodes be used in the node-list. Second, random-walk based technique is introduced, which will be explained in detail in Section 3.3.

2.2 Normal-Client Node

Each client-side user who enjoys P2P media streaming service is classified as Normal-Client Node (NCN). In each NCN, there also exists a node-list cache. Connected-peers (partners) are selected from the cached node list. In the next section, we will introduce several mechanisms, such as receiver-driven gossip, random walk and lease, to maintain the cached node list. Besides, an evaluation criterion should be used to select partners from the cached node list.

2.3 Leased-Client Node

Leased-Client Nodes (LCNs) are the nodes who send lease messages to NLS periodically to announce their aliveness. From the view of topology, these nodes are connected directly to the NLS, and are a part of the backup nodes which might be given to new incoming nodes as contents of their initial cached-node list. Each LCN sends a message to keep its lease on NLS periodically. If leased time has passed and no re-lease message has arrived, then this node will be removed from the NLS's node-list cache. NLS can confirm the aliveness of LCNs or gather topology information from them. Besides, NLS can redirect new incoming nodes to a LCN to lighten its own burden.

LCN is developed from NCN. There are two chances for a NCN to become a LCN: (1) when a node joins the overlay network; (2) when a node's node-list cache is empty and turns to NLS for help.

2.4 Media-Fetching Node

Media-Fetching Node (MFN) is used to get media data from the streaming server, and then distribute them in the P2P network. The data receiving status of MFN should be sent to its partners, so as the partners can request media data from it.

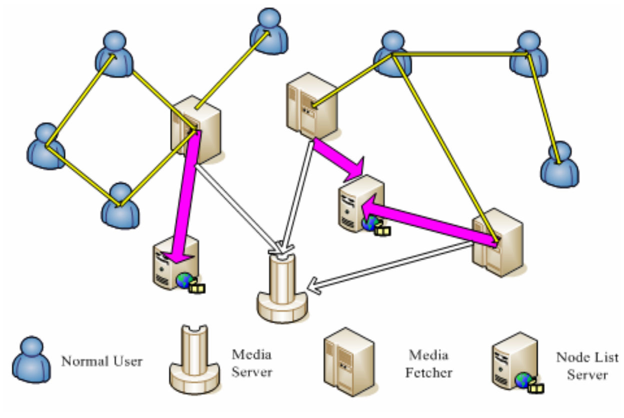

Fig. 1. How NLS, MFN, LCN and NCN are connected

In a practical system, MFN and NLS can be installed separately. Such a design has a salient advantage, i.e., it provides more selections for implementing the system. For example, we can deploy a MFN for each ISP, thus clients from every ISP can get better service, while only one NLS is installed in the whole network. As shown in Fig. 1, one NLS can provide services for different MFNs.

2.5 Weak Hierarchy Model

In AVStreamer, when a new client node joins the network, it should be given a node list to help it to get the initial view of the whole network. Therefore NLS is designed to deal with this problem. If we want to give an optimized node list, the NLS must track the records of all the nodes in the system, and extensive computations are needed to decide which ones should be the best. It may become the bottleneck both for performance and scalability, and is vulnerable to be attacked.

To address the above problem, we propose to use Leased Client Node (LCN). The advantages of using LCNs can be summarized as follows: (1) the cost of maintaining the node list to be sent and the computation of deciding which nodes should be sent can be shared with the NLS. (2) LCNs are selected from NCNs by capability, and are given greater responsibilities. This makes the whole network more efficient. (3) LCNs can also be used to gather the information about the whole overlay network. (4) For an existing NCN, what a LCN is responsible for is not critical. If a LCN becomes fail, no existing NCNs will be affected, thus there is enough time to recover a new LCN.

This model is similar as but also has notable difference from traditional Hierarchy Model [7], since LCNs are only used when new incoming users join the overlay network, and from NCN's view there are no differences between LCNs and themselves, therefore we call this model weak hierarchy model.

3 Proposed Key Techniques

For P2P media streaming, to make the whole system more robust and effective, partners and node-list cache play a very important role. Partners of a node are the neighbor nodes which directly connect to and exchange data with it. Node-list cache contains backup partners preparing for the fails or updates of current partners. Whenever partners and cached nodes are optimized, nodes can get better service from the overlay and the whole system can be more robust and steady. Aiming at this requirement, first we need a mechanism to get an overview of the status of the new nodes, and then use a criterion to select from them. In AVStreamer, we use receiver-driven gossip to exchange node-lists between nodes to help nodes get more information about new nodes. We use our node evaluation criteria to pick up partners and merge the exchanged node-list into nodes' own node-list cache and make the cache optimized. We also use random walk to make the overlay more random and steady, and status-polling to ensure the aliveness of the nodes cached. As a whole, these four techniques can be combined together and make the topology in AVStreamer system effective and robust.

3.1 Node Evaluation

In order to optimize the topology of a P2P media streaming system, some performance standards should be built first. While developing AVStreamer, we summarized four criteria to evaluate its topology maintenance system. First, the topology should be random enough, so that organization in which no peer is directly or indirectly connected with data provider has little chance to appear. Random topology also makes the network data transmission steady and robust in case of topology changes. Second, the importance of a peer should be affected strongly by its capability. If a peer has strong capability (such as abundant bandwidth, etc.), then it should take more responsibility. If a peer has poor capability, then it must not play an important role, otherwise it may become a bottleneck of the whole overlay network. Third, application-layer topology should be strongly affected by network-layer topology. If the application-layer topology is consistent with the network-layer topology, throughput between any two nodes can be maximized; as a result, the resource of the whole network can also be utilized more efficiently. Finally, the cost for topology maintenance should be as small as possible.

Some of the above criteria are conflict with each other. For example, if the topology is totally random, then the second and the third criterion may not be fit; if the application-layer topology is determined by peers' capacity and the network-layer topology, then the first criterion cannot be fit. Therefore a balance should be achieved among them.

To reach the balance, we need a standard to evaluate different nodes to which a node may connect. In AVStreamer, the following strategy is used: when comparing two nodes in cache, the popularities of them are considered first; if the difference is not big enough, then their capabilities are taken into account; if the difference of capability is rare, then the freshness of the nodes is considered.

There are two reasons for choosing the above strategy. On the one hand, they reflect a node's quality in some degree. In AVStreamer, popularity is decided by the times a node is recommended to others. If a node has been recommended more times, it is less tended to be recommended again, unless this node has more capability. Freshness is determined by a node's latest message time-stamp. Since a node's aliveness cannot be ensured all the time, thus the freshness is an important feature to guess whether the node is still alive. Capability is determined by the announced capability in periodically exchanged messages. Nodes exchange each others' capability at a certain short interval. Because recent node's capability is close to its current capability, it can be used to play the same role. On the other hand, they are easy to implement. If we want to confirm the exact aliveness, popularity, or current capability of a node, then messages will be transmitted more frequently and very high bandwidth will be cost. To avoid these costs, only freshness and capability at a certain time of a node should be used.

3.2 Receiver-Driven Gossip

In order to maintain and optimize a node' local cached node-list, it needs to get new nodes periodically and use the ones which are fit for it to replace the old unfit ones in the cache. Although each node has its own node-list cache, a node can get new node information from other nodes.

Gossip based dissemination protocol seems to be helpful to solve this problem and can provide good scalability and reliability properties [8][9][3]. However, traditional gossip protocol is not fit for our scene, since traditional gossip is sender-driven. In AVStreamer, a node needn't to transfer its node list to every one in the network, because not all of the receivers need the information. Besides, if all the nodes send its message to every one in the network periodically, there will be a very high traffic load over the network.

Based on this, we make an important improvement in AVStreamer: a receiver is not passively waiting for node-lists, but actively asking for them when it thinks it is necessary. In this way, nodes are no longer responsible for sending its own node-list to others actively, but only sending on demand. We call this mechanism receiver-driven gossip.

There are several advantages for receiver-driven gossip. (1) No broadcasting is involved. If gossip is driven by sender, every time the status of the sender is changed, a message should be send through the network. But for receiver-driven gossip, one message may contain several pieces of information which show all the changes during query interval. (2) It is more efficient. Once a message is only transmitted between a pair of peers, lower bandwidth is required. Also, the receiver asks for new peers only when it thinks its own node-list cache needs to be improved. It never receives messages which it is not interested in. (3) Node information can be transmitted from peers to peers, spread to certain scope, just like the traditional gossip protocols. For example, if *A* is a part of the node-list cache on *host_a*, when *host_b* queries *host_a* for node-lists, *A* might be sent to *host_b*. Next time when *host_c* query *host_b*'s node-lists, *A* also might be sent to *host_c*. (4) Newest nodes can always be used to query new nodes to make the diffusion of cached nodes fast enough.

By using receiver-driven gossip, a node can get to know other nodes which are currently far away from it. Its local node-list can be more and more random and fit for itself.

3.3 Random Walk

To make the topology more random, we introduced random walk [6] into the implementation of AVStreamer. In our design, we use it to help a peer to select its partners following some certain rules. When a node has cost more than half of its capability, it begins to use a probability which is inverse proportional to its current capability to decide whether to accept this new incoming node. If the new node is not accepted, it is redirected to another node, and this node makes the same decision repeatedly. Like this process, the joining message will also be redirected until one node decides to accept it or become out of time.

In our system, random walk plays an important role in several aspects. (1) The method can reduce the burden of LCNs and avoid them to be saturated too soon, because LCNs can redirect requests to other nodes. 2) If one partner is get by redirecting, this partner is tend to be far away from original node topologically. When original nodes exchange node-list with this partner, it can get more new nodes far away. The whole overlay network will be more random, too.

3.4 Status Polling

In AVStreamer, we adopted status polling to ensure the aliveness of nodes in local node-list cache. When a node is inserting into one's cached node-list, the inserting time is recorded. Whenever a data transmission occurs, the inserting time will be updated to the current time. If the interval between current time and the inserting time is longer than a certain value, a status-query message will be sent to make sure the node is still alive. Some other information about the node's current status is wanted, too.

By using status polling, the aliveness of cached nodes is ensured in a certain extent. If a node is selected as partner, the probability of success is increased.

4 Experimental Results

To evaluate the performance of the proposed techniques, we test AVStreamer in our laboratory. In our experiment, 50 PCs are involved, each PC runs 2~8 instances of AVStreamer during the test. For each instance, limitation of the size of node-list cache is set to 8, limitation of the size of partners is set to 5, limitation of the size of active connected partners is set to 2. We organized 4 groups of experiments. Whole sizes of the overlay network are different between each experimental group.

Table 1. Average hops to media-fetching node in different size of network

Total size	100	200	300	400
Average Hop	5.8	6.7	7.3	7.6

4.1 Average Hops to Media-Fetching Nodes

If a overlay network is totally random, its diameter should be proportioned to log(N), where N is the size of the network. A random network can be more robust to topology changes. Besides, the average data transmission delay can be reduced. In our experiments, we measure average hop to MFNs instead, because the average diameter is not easy to get. MFNs are the source of data transmission. The distance between MFNs and NCNs show the semi-diameter of the network to some extents.

As shown in Table 1, we can see that the average hop to media fetching nodes is close to being logarithmic to the size of the network. It should be because of random walk mechanism, and our node evaluation criteria.

4.2 Node Information Diffusing

Table 2 shows the average amount of different nodes that a client can get by using receiving-driven gossip. Member exchanging is done at interval of 40s, each exchange queries 1/3 of the size of node-list cache nodes from other peers.

Table 2. Average amount of different nodes a client has got after 20 minutes

Total size	100	200	300	400
Average nodes get	72	77	81	83

From this table, we can see that, by using receiver-driven gossip, each node has the chance to know lots of other nodes in the network. That is helpful to recovering from partner failures.

4.3 Node Degree Distribution

In the whole network, there should be enough unsaturated nodes for the new incoming client, but for each individual, a node should have enough partners to make the data transmitting steadier, therefore we give the node degree distribution of each experiment.

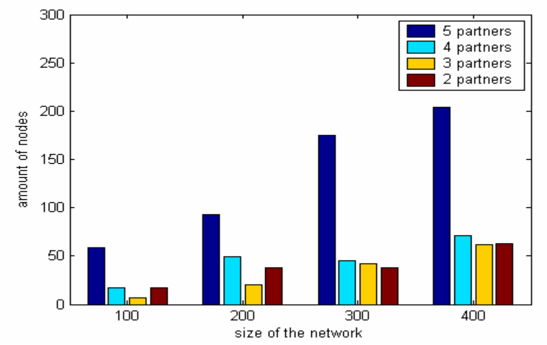

Fig. 2. Node degree distribution

As shown in Fig. 2, we can see almost half of the nodes are unsaturated, which means when new node joins the network, there are nearly half nodes available to be connected to them. The reason results from the difference between the limitation of the active connected partners and the limitation of the total partners.

4.4 Bandwidth Cost

We test the bandwidth costs of four experiments. The results shown in Table 3 tell us the bandwidth cost is rare, and it increases very slowly when the size of network become larger. This is because all the messages are transmitted locally.

Table 3. Average bandwidth cost in 20 minutes

Network Size	100	200	300	400
Average download bitrate(kbps)	1.1	1.2	1.3	1.3

5 Conclusions

In this paper, we presented the design of an optimized topology maintenance system in AVStreamer, and discussed the new weak hierarchy model. This model is more effective, it shares burdens from Server to help new incoming client to get an initial view of the network. We also discuss some key techniques used in AVStreamer, such as how to evaluate nodes to help optimizing the node-list cache, using receiver-driven gossip to help fetching new nodes from others more effectively, using random walk to redirect nodes' burden and make the network more random, and using status polling to ensure the aliveness of each node. Experimental results show that the proposed techniques are effective.

Acknowledgments. This work was performed in Peking University, and was partially supported by the Key Technologies R&D Programme of China under grant No. 2006BAH02A10, the China Postdoctoral Science Foundation under grant No. 20060400001, and the"985" construction project of Peking University.

References

[1] Deshpande, H., Bawa, M., Garcia-Molina, H.: Streaming Live Media over a Peer-to-Peer Network. Stanford database group technical report (2001-20) (August 2001)
[2] Nicolosi, A., Annapureddy, S.: P2PCAST: A Peer-to-Peer Multicast Scheme for Streaming Data. In: Proceedings of First IRIS Student Workshop August 2003. Cambridge, Massachusetts (2003)
[3] Zhang, X., Liu, J., Li, B., Yum, T.S.P.: Coolstreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming. In: IEEE INFOCOM, Miami, FL, USA (2005)
[4] Gao, W., Huo, L., Fu, Q.: Recent Advances in Peer-to-Peer Media Streaming Systems. China Comminications.5, 52–57 (2006)
[5] Huo, L.: Study on key techniques of media streaming over the internet, Ph.D. dissertation, Graduate University of Chinese Academy of Sciences (2006)
[6] Gkantsidis, C., Mihail, M., Saberi, A.: Random Walks in Peer-to-Peer Networks. In: IEEE INFOCOM. HongKong, China (2004)
[7] Yang, B., Garcia-Molina, H.: Designing a Super-Peer Network. In: Int'l Conf. on Data Engineering, Washington, US (2003)
[8] Ganesh, A.J., Kermarrec, A.M., Massoulie, L.: Scamp: Peer-to-peer lightweight membership service for large-scale group communication. In: Workshop on Networked Group Communications, London, UK (2001)
[9] Jelasity, M., Babaoglu, O.: T-Man: Gossip-based overlay topology management. Engineering Self-Organising Applications. Utrecht, The Netherlands (2005)