2006 1st International Symposium on Pervasive Computing and Applications

A Real-time Streaming Media File Sharing Mechanism Based on P2P and SIP

Deguo Yang, Hui Wang, Yuhui Zhao, Yuan Gao
School of Information Science & Engineering Northeastern University,
110004 Shenyang, China
Yangdeguo2002@163.com

Abstract

P2P systems have high scalability, robustness and fault tolerance because of its no centralized server and the network self-organized itself. As the standard protocol for VoIP and NGN, SIP is a general protocol for establishing and controlling multimedia session. In this paper, we proposed a real-time streaming media sharing mechanism based on P2P and SIP, which uses Chord as the underlying distributed hash table (DHT). We use the advanced peer and piece selection algorithm to solve the real time media stream transmission. We have a redundancy design on the Tracker server to solve the Tracker server failure and bottleneck problem. The P2P combined with SIP real time stream transport mechanism supports user registration, media resource location, media streaming session establishment, and real-time media streaming playback online. Additionally, we give a simulation and test result.

Keywords: P2P, SIP, real time stream, file sharing.

1. Introduction

With the development of mobile wireless communication, the requirement of multimedia on mobile telephone and other handset has increased. Because the store space and bandwidth of this kind of equipment is limited, the suitable method is getting the multimedia file on line, playing while downloading. The other users of PC also prefer to watch the media file without waiting until the file is downloaded. It is necessary to develop a real time media system to transmit the media among the users.

Peer-to-peer (P2P) system is inherently scalable and reliable because of the lack of a single point of failure [1]. P2P system, in the purest form, has no concept of servers. All participants are peers and communicate in distributed to achieve a certain objective. Although Peer-to-peer media streaming system has received more and more attention, general peer-to-peer system only offer the function of file sharing, and the mode could not satisfy the demand of real time media stream transport. On the other hand, Session Initiation Protocol(SIP)[2] is an IETF standard for Voice over IP (VoIP). As SIP media application have increased in popularity, situation have emerged where centralized server are either inconvenient or undesirable. But as the IETF standard, SIP application is popular in communications systems. Using the open SIP standard helps us meet our requirement for compatibility and reuse the open source stacks and the application can be integrated with the tremendous number of existing SIP terminal.

In this paper, we design the P2P integrated with SIP real time media stream transport mechanism (named as P2P-SIP in the following). P2P-SIP combines the SIP family of IETF standards for VoIP(Voice Over Internet Protocol) with the Self Organizing properties of a Distributed Hash Table (DHT) P2P mechanism. The result is a standard-based, decentralized online media sharing system. Our design allows for compatibility with and reuse of existing SIP network elements. The P2P overlay is built using a DHT created through exchanging SIP messages, which are typically well-supported by existing firewalls, and therefore retain compatibility with the current network infrastructure. The uses OTS_{p2p} and DAC_{p2p} [3] based media stream hand out algorithm to solve real time media transport problem.

The main contributes of this work are:

Creating a fully distributed, P2P and SIP based real-time multimedia file sharing system.

In the design of system, using advanced media streaming hand out mechanism to support playback the media while downloading.

Add the redundancy Tracker server to solve the Tracker bottleneck and failure problem.

2. Background and Related Work

2.1 P2P and SIP

The fundamental principle behind peer-to-peer (P2P) architectures is that each and every member of the network has equal importance in the transactions that take place on the network, and that these nodes communicate with each other to accomplish tasks. The best known use of P2P technology is file sharing system. But the ordinary peer-to-peer file sharing system is not suitable for the real time media transport for the lack of optimized peers and pieces mechanism for the real time stream media. Such as BitTorrent[5], although its peers selection policy effectively prevent the free rider[4], it limits the overall throughout and reduces the peer

bandwidth utilization rate of the system, which is a important defect for the real time stream transport. Another peer-to-peer live multimedia stream system is Chainsaw[6]. Although Chainsaw is pull-based and effectively utilizes the source uplink bandwidth, but it has not the pieces optimal selection policy, and is not scalable.

SIP has emerged as the standard protocol for VoIP. SIP supports the initiation, modification, and termination of media sessions between user agents. Although some systems use other standards or proprietary mechanisms such as H.323, the majority of new VoIP development uses SIP. SIP with SIMPLE extensions is also extensively used by existing IM clients, for example in Microsoft's MSN messenger. Using the open SIP standard helps us meet our requirement for compatibility, and allow the system to leverage open source stacks[1]. Although the applications of SIP are increasingly popular communications systems for private, corporate, and academic purpose, some of the SIP applications face some deficiency. For example, companies concerned about intellectual property typically ban the use of current, centralized services, instead deploying their own in-house services. Under some condition, completely distributed system may be the best choice.

2.2 Distributed Hash Table (DHT) Systems: Chord

To improve the peer and resource locating efficiency, most systems locate resource by a Distributed Hash Table (DHT). The Chord[7] system is one particular DHT algorithm. In this system, every node have a Node-ID which is get by hashing the IP address and port of the node; and every resource has a Resource-ID, which is obtained by hashing some keyword or value that uniquely identifies the resource. The node is responsible for storing all resources that have Resource-IDs near the node's Node ID. Chord uses a ring-type structure for the nodes in the overlay. In this structure, if the hash has 2^n bits in the range, each node will keep a finger table of pointers at most *n* other nodes. The *ith* entry in the finger table contains a pointer to a node at least 2^i units away in the hash space. These highest finger table entries thus point to a range 1/2 of the way across the hash space, the next 1/4, and so on, and the smallest entry points to a range only 1 away in the hash space. Searching in Chord is accomplished by sending messages to the node in the finger table that is closest to the destination address. That neighbor will have finer resolution detail about the area and can route the message closer to the desired node. This process is repeated until the message reaches the node responsible for the destination.

2.3 Real-time media streaming transport

To send or receive a live media over the Internet or Intranet, you need to be able to receive and transmit media streams in real-time. Although there have been significant research efforts in real time media stream deliver during the past two years, such as Real-Time Transport Protocol(RTP) and Real-time Transport Control Protocol(RTCP), but peer-to-peer media real-time streaming system has received less attention. The character of its play-while-downloading mode of peer-to-peer system is different from the ordinary one. In [3], the algorithms OTS_{p2p} and DAC_{p2p} are proposed. The algorithm OTS_{p2p} offers an optimized piece choice policy which computes the media data assignment for each peer-to-peer streaming session. The assignment will lead to the minimum buffering delay experienced by the requesting peer. After computing the media data assignment, it will initiate the peer-to-peer streaming session by notifying each participating supplying peer of the corresponding assignment. The algorithm DAC_{p2p} offers a peer selection policy which can rapidly amplify the peer-to-peer system capacity. It chooses the peers on the uploading bandwidth and the decision is made in a probabilistic fashion.

3. P2P-SIP Mechanism Design

Our design has no central servers and nodes communicate directly with one another, there are some differences between our approach and traditional SIP. A peer connects to a few other peers in the overlay network and use SIP message to search file message and locate nodes. Nodes act both as UAs and Proxies simultaneously. When it gets message and searches file it acts as UA; and when it maintains and records the file message and response the request node, it acts as proxies and replace the functionality of SIP registrars and proxies. Each node is responsible for those roles for some portion of the overlay.

We use the SIP REGISTER message to pass the overlay information between nodes, as well as the original use of sending user location information to a registrar (peer). We use INVITE message with SDP information to request suitable user to transfer needed peer information and media pieces. If a new node joins the system, a number of REGISTER messages should be exchanged. At first, the node is assigned a Node-ID created by hashing (using SHA-1) the real IP address and appending a port. Then send a REGISTER message to a node which is responsible for the region, and insert itself into the overlay. Once the new node has joined the overlay, it will be responsible for storing information (user registrations) associated with the portion of the overlay mapped to that calculated Node-ID, and transfer the information for the region from the node presently storing that data which should belong to the new joining node.

There is an example of a node join in Figure 1. When a new node wishes to join the overlay it must locate some node already in the overlay, referred to as a bootstrap node. Presently this node is located via out-of-band mechanisms. The joining node calculates its Node-ID, 143, in our example, and sends it in a REGISTER to the bootstrap node, which has Node-ID 143 (1).

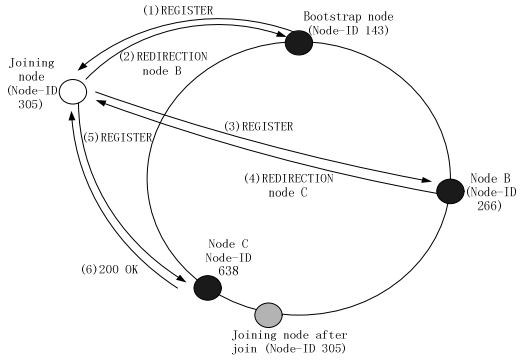

Fig. 1. An example of a new node joining the overlay

Assuming that the bootstrap node is not the node currently responsible for this region, it responds with information about the nodes it knows nearest to where the joining node will be placed in the overlay—in this case, Node B, with Node-ID 266. This information is passed back in our new headers within a SIP 302 Moved Temporarily response (2). The joining node repeats the process, using this nearer node as the new bootstrap node (3-4). Finally, the joining node reaches the node that is currently responsible for maintaining the appropriate section of the overlay, In this case Node C, with Node-ID 638. Node C responds with a SIP 200 OK response including detailed information about nearby neighbors in the headers (5-6), allowing the joining node to insert itself into the overlay. Further messages, not shown, are sent between the joining node and the node previously responsible to exchange the actual information (registrations) in the overlay the new node must store. Additionally, the new node will send messages to other nodes in the overlay to update their finger tables. At this point, the node has joined the overlay and node registration is complete.

3.2 File publishing and pieces distribution

To start a media file deployment, a static file is stored on a node according to Chord architecture. The file contains information about the file, its length, name, and hashing information and the information which nodes are downloading it and downloading what part of it, and the downloaders' IP addresses and ports. If a node wants to publish content, it hashes the key word of the content, and finds the node which should be responsible for the content according to the Chord peer-to-peer structure. The node which is responsible for the content will store the static file.

When a file is published on the overlay network, the node which stores the file information will acts as a Tracker. The Tracker is a manager of the file on the node, it receives from all downloaders and giving them a random lists of peers. The downloaders periodically checking in with the Tracker to keep it informed of their progress, and the Tracker offers the information of the file information.

For the purposes of playing while downloading with best effort, the seed file is partitioned into fixed-size pieces which are all the same length except for possibly the last one which may be truncated. Piece length is almost always a power of two, most commonly 2^8= 256K. The seed computes an SHA1 hash for each piece and distributes these hash values to each receiver so that the latter can verify the integrity of the pieces it receives later on. Each piece is further divided into blocks, typically of size 16Kbytes. A receiver can download blocks within the same piece from multiple peers simultaneously. However, a node can relay blocks of a piece to other nodes only after it receives the entire piece and successfully verifies its integrity.

3.3. Peer strategy and piece selection

Once a node wants to download a file, it will hash the resource file key word, and get a file ID. According to the file ID, The node searches its finger table and sends the request information to the overlay network. If the file is not existed, the node whose node ID is just equal or follows the file ID will return the response that the request file is not exist. If the File does exist, the Tracker on the node who is responsible the file will return the request node a file information including the nodes random lists table with which the request node can download the file from. Then the request node chooses the node from the list in some policy to download the file pieces which it needs.

3.3.1 Pieces selection. Once a BitTorrent client establishes connections with its potential peers, it needs to determine which pieces to download from which peers, based on the knowledge of the set of file pieces available in all it peers. As shown in Figure A, different assign policy leads to different buffering delays.

In the Figure 2(a), the requesting peer is P_r; and the supplying peer are P_{s1}, P_{s2}, P_{s3}, P_{s4}. with out-bound bandwidth of $R_0/2$, $R_0/4$, $R_0/8$, and $R_0/8$ respectively. In assignment Ⅰ， P_{s1} is assigned media data segments 8k, 8k+1, 8k+2, 8k+3 (k=0,1,2,3…), $P_{s2}$is assigned 8k+4, 8k+5; P_{s3} is assigned segments 8K+6; P_{s4} is assigned segments 8k+7.the start time is 5ót. However, if assignment Ⅱ is used the buffering delay will be reduced to 4ót.

We adopt an optimized media data assignment algorithm OTS_{p2p} to reduce the buffering delay. Let *Ro*

denotes the playback rate of the media data. We assume that each requesting peer P_r will be able to set aside all in-bound bandwidth of $R_{in}(P_r) \geqslant R_0$ to receive the streaming service. And the out-bandwidth $R_{out}(P_s)$ offered by the supplier P_s has one of the following values, R_0 $R_0/2$, $R_0/2^2$… $R_0/2^n$. According to the out-bound bandwidth, the peers are classified in to N classes.

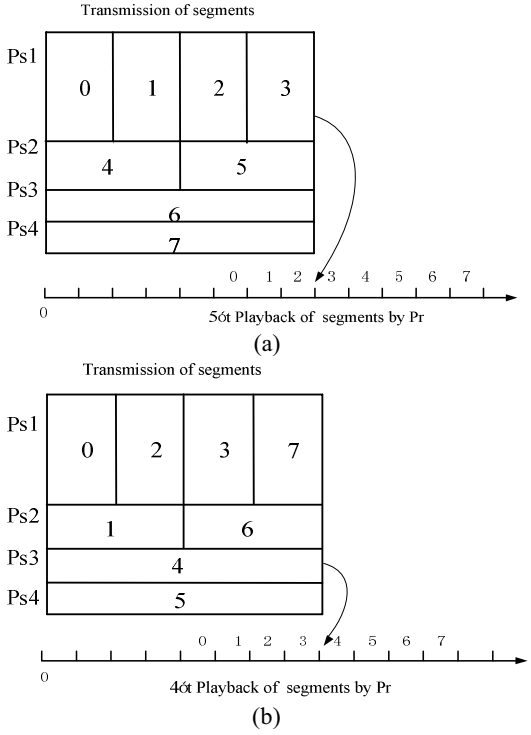

Fig. 2. different assignment leads to different buffering delays

The peer who is willing to offer out-bound bandwidth $R_0/2^n$ is called *class-n* peer. For example, a peer which can offer out-bound bandwidth $R_0/2^2$ is called *class-2* peer. To satisfy the demand of playing while downloading, the goal is ensure a continuous playback with minim buffering delay at P_r. We define the buffer delay as the time interval between the start of media data segment and the start of playback at P_r. Based on the assumption above, for a requesting peer P_r and a set of supply peers P_{s1}, P_{s2}… P_{sm}, if every piece buffering time is δt , and $R_{in}(P_r) = \sum_{i=1}^{m} R_{out}(P_s^i) \geq R_0$ ($1 \leq i \leq m$), then Algorithm OTS_{p2p} will compute an optimal media data assignment, which achieves the minimum buffering delay in the consequent peer-to-peer streaming session. The minimum buffering delay is $T_{buf}^{min} = m * \delta t$ .The pseudo-code of algorithm can be found in[3].

3.3.2. Peer strategy. Peer strategy has a great impact on local performance. To join a session, a node first contacts the Tracker to obtain a list of randomly chosen peers that are currently in the session. Then it establishes new socket connections with these peers until the number of connected peers reaches some threshold, currently 40. In the mean while, it can start accepting connection requests from other peers, but will stop accepting new connection requests when the number of connected peer reaches another threshold, currently 80. A client will asks the Tracker for a new list of peers either every some time，generally 30 minutes, or when the number of neighboring nodes reaches the minimum, currently 20.

Given a set of file block from its peers, a BitTorrent node needs to determine which request to service and which to ignore. In BitTorrent, it employs a fairness policy to prevent free riders. The policy is that every client chooses to serve four peers that have contributed the most uploading bandwidth to it. With the increase of the heterogeneous degree in the network, in the BitTorrent peer-to-peer selection policy, some uplink bandwidth may be wasted[8]. It is obvious that the admission which favors higher-class requesting peer will lead to faster amplification of the peer-to-peer system capacity, and will ultimately benefit requesting peers of all class. It is especially important for the real time stream media system. But the policy should not starve the lower-class request peers. It should ensure that the higher the out-bound bandwidth pledged by a requesting peer, the greater the possibility that it will be admitted. This will create an incentive to encourage requesting peers to contribute its truly available out-bound to the system. In this system, the solution is a distributed admission control protocol DAC_{p2p}[3]. To the requesting peer list, each supplying peer P_s maintain an admission probability vector<P_r[1], P_r[2]… P_r[N]>. P_r[i] ($1 \leqslant i \leqslant N$) will be applied to class-i requesting peers. Supposed P_s itself is a class-k peer, then the values in the probability vector of P_s is determined as follow: for $1 \leqslant i \leqslant k$, we initialize P_r[i]=1.0, and for $k<i \leqslant N$, we initialize $P_r[i] = 1/2^{i-k}$. if P_s has been idle, then its probability vector will be updated after some time, if $P_r[i]<1.0$, for each $K<i \leqslant N$, $P_r[i] = P_r[i] \times 2$. the simulation in the section 5 shows the policy is available.

4. The robustness design

Because the Tracker is the only way for the peer to discovery other peers, and the Tracker is run on an ordinary node. The central coordination presents the single point failure of the entire system. Whenever the Tracker shut off at any time, it will result in system temporary failure; with the number of requesting peer increase, the Tracker will become the bottleneck of the system. To solve the problem, we have a redundancy design on the Tracker server. We have another copy of the Tracker information on the other node, generally the node whose node ID is just next to the Tracker. When the client number is over 1000, the Tracker will redirect the request information to the backup Tracker server,

and the redundancy Tracker will provide the information which the requesting peers want. When the Tracker is shut off, the redundancy Tracker will become the really Tracker, and find another node to be the redundancy Tracker server.

5. Result and analysis

We simulate the P2P-SIP media streaming system with p2psim. In first experiment, we contrast the system with the pure BitTorrent file sharing system on the rate of bandwidth utilization; then we verify the validity of piece selection policy in the decrease media transport latency time; At last, we test the change of system temporary failure rate brought by the redundancy Tracker. In the simulation, the uplink bandwidth of seed is set 2000Kbps; the size of sharing file is 256MB; the size of each file piece is initialized 256KB; the size of each block is initialized 16KB; the number of receiver nodes is 400; the number of neighbors each node has is 40; and the maximal number of concurrent upload connection per node is 5.

We contrast overall efficiency between the general BitTorrent system and the P2P-SIP system. We define the overall throughout ratio between BitTorrent and P2P-SIP as the evaluation parameter(formula(1)).

$$Ratio = \frac{Throughout_{BT}}{Throughout_{p2p-sip}} \qquad (1)$$

Figure 3 shows the overall throughout ratio between BitTorrent and P2P-SIP with the number of node changing in three different Networks. From the figure, we can draw a conclusion that the overall throughout of the P2P-SIP is obviously higher than that of BitTorrent.

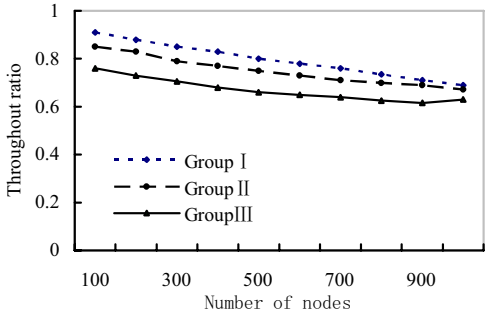

Fig.3. Overall throughout ratio change between BitTorrent and P2P-SIP with the number of node changing in three different Networks.

The higher bandwidth utilization efficiency of P2P-SIP is benefit from the peer selection policy. Because the peer in P2P-SIP chooses the peer with more uplink bandwidth to upload media data, the overall throughout of the system is improved.

Figure 4 shows the influence brought by the different piece selection. It is obvious that the P2P-SIP with OTS_{p2p} piece selection policy has less buffering delay time than that of non-OTS_{p2p}. The system with P2P-SIP has average 200ms buffering delay less than the non-OTS_{p2p}. The reason is because the OTS_{p2p} algorithm can compute an optimal media assignment, which achieves the minimum buffering delay in the consequent peer-to-peer streaming session.

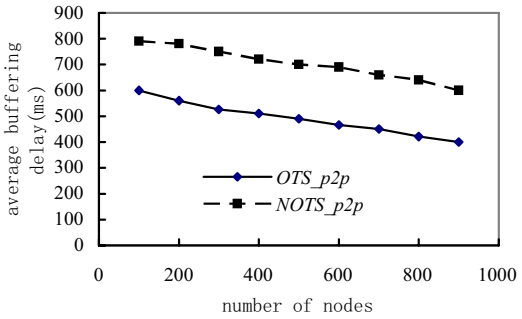

Fig.4. Different influence brought by the piece selection.

At last, we test the Tracker failure rate under the situation of with redundancy Tracker and without redundancy Tracker. Figure 5 shows the tracker have no failure rate with in some time. The period unit we test is 3 hours. We can see that the failure is improved with the redundancy, and the average Tracker failure rate is lower than 19 percent, while without the redundancy Tracker, the average Tracker failure rate is over 30 percent.

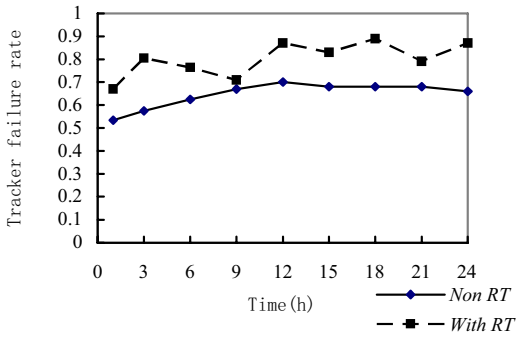

Fig.5. The change of Tracker failure rate without redundancy and with redundancy Tracker

6. Conclusions and future work

SIP is expected to be the communication standard in the future, and the peer to peer media stream system

becomes more and more popular. In this paper, we combine the SIP family of IETF standards with Self Organizing properties of a Distributed Hash Table (DHT) P2P mechanism. To achieve the goal of playing while downloading, we apply the OTS_{p2p} media assignment algorithm, and the DAC_{p2p} peer selection mechanism. To ensure the robustness of the system on the Tracker bottleneck problem, we choose another node as the backup Tracker. The simulation shows the P2P-SIP system can meets the need for the distributed real time media communication without the central server.

The future work includes the encoded technique which will recover the original file in the face of block loss and remove the last block problem. The Multiple Description Coding(MDC) which is used to encode the multimedia data to accommodate a set of heterogeneous client, will be the suitable choice. Another problem to be solved is the fairness, i.e. without decreasing the overall throughout, how to let the peers who contribute the Networks most have the highest priority and discourage free riders.

References

1. David A. Bryan, Bruce B. Lowekamp, and Cullen Jennings, "SOSIMPLE: A Serverless, Standards-based, P2P SIP Communication System" *Proceedings of the 2005 International Workshop on Advanced Architectures and Algorithms for Internet Delivery and Applications* (AAA-IDEA 2005),2005.
2. Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A.,Peterson, J., Sparks, R., Handley, M. and E. Schooler "SIP: Session Initiation Protoco", RFC 3261, 2002.
3. Dongyan Xu, Mohamed Hefeeda, Susanne E. Hambrusch, Bharat K. Bhargava "On Peer-to-Peer Media Streaming". ICDCS, 2002, 363-371
4. E. Adar, B. A. Huberman "Free Riding on Gnutella" First Monday, (2000), 5(10).
5. B. Cohen "Incentives Build Robustness in BitTorrent". *The 1st Workshop on Economics of P2P Systems, Berkeley, CA,* 2003.
6. Vinay Pai, Kapil Kumar, Karthik Tamilmani, Vinay Sambamurthy, Alexander E. Mohr "Chainsaw: Eliminating Trees from Overlay Multicast". *4th International Workshop on Peer-to-Peer Systems. February,* 2005.
7. Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, Hari Balakrishnan "Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications". *IEEE/ACM Transactions on Networking, Vol. 11, No. 1,* 2003. 17-32.
8. Gang Wu, Tzi-cker Chiueh "How Efficient is BitTorrent?". Thirteenth Annual Multimedia Computing and Networking (MMCN'06), San Jose, CA., January, 2006.