2011 International Conference on Computer Science and Information Technology (ICCSIT 2011)
IPCSIT vol. 51 (2012) © (2012) IACSIT Press, Singapore
DOI: 10.7763/IPCSIT.2012.V51.82

Design and implementation of a streaming media transmission system

Xiaoyu Ma[+] and Rui Jin

Department of Computer Science And Engineering, Henan Institute Of Engineering, , No 1, Zhongshan North Road, longhu conuty, Xinzheng, 451191, China

Abstract. The transmission strategy of the streaming media is very important in streaming media transmission system, unreasonable transmission method of streaming media will greatly waste the network resources. In order to rationally use the network resources, and make the transmission of streaming media more fluent, this paper proposes a design scheme of the streaming media transmission system, and describes the design of the general structure,the buffer and the scheduling algorithm, last, describes the RTP transmission of the streaming media and the implementation of the buffer in detail.The new designed streaming media transmission system can improve the transmission efficiency, and reduce the waste of the network resources.

Keywords: Streaming Media, Transmission System, Design And Implementation

1. Introduction

The rapid development and popularity of Internet provides the strong market momentum for the development of the streaming media business, the streaming media business is becoming increasingly popular. Streaming media technology is widely used in multimedia news publishing, live broadcast online, advertising online, e-commerce, video on demand, remote education, telemedicine, Internet radio, real-time video conference and other aspects of the Internet Information Services [1]. The application of the streaming media technology will bring great changes on the network information exchange and have a profound impact on the people's work and life.

The streaming media technology is not a single technology, it is the organic combination of network technology and video / audio technology. The implementation of streaming media technology on the network, needs to solve the problems on the streaming media production, publishing, transmission and playing, and so on. The transmission platform which the streaming media is dependent on is IP network, because the connectionless packet forwarding mechanism of IP network is designed for the sudden data transmission, it does not apply to the transmission of continuous media stream. In order to transmit the effective and high-quality video streams in Internet, we need to solve many problems, this paper mainly solves the problem of the streaming media transmission.

2. The design of the streaming media transmission system

The rapid development and popularity of Internet provides the strong market momentum for the development of the streaming media business, the streaming media business is becoming increasingly popular. Streaming media technology is widely used in multimedia news publishing, live broadcast online, advertising online, e-commerce, video on demand, remote education, telemedicine, Internet radio, real-time video conference and other aspects of the Internet Information Services [1]. The application of the streaming media technology will bring great changes on the network information exchange and have a profound impact on the people's work and life.

[+] Corresponding author. Tel.: +00-86-371-68723925; fax: +00-86-371-68723925
E-mail address: hnzzmxy@yahoo.com.cn

2.1 The design of the streaming media transmission system's general structure

In MixCast streaming media transmission system, when the node broadcast a program, it first connect the node of which tracker can get available services at random, when the streaming media system begins to play, or reconnect the node for failure , there is a buffer delay, and the effective organizational strategy of data transmission should ensure that the buffer delay is small as much as possible [2] .

Through the study of general transmission, in the streaming media transmission system designed by this article, uses the RTP protocol which uses UDP protocol in the bottom layer to transmit data information, uses RTCP to control packet information, uses HTTP protocol to interact with Web server, video player interacts with data sources by using HTTP and MMS protocol, as is shown in Fig. 1.

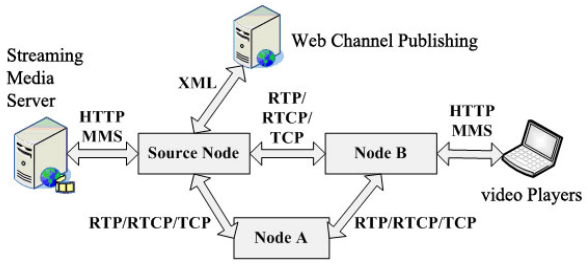

Fig. 1. network protocols graph of streaming media transmission system

Web server uses XML technology, the node communicates with web server by using HTTP protocol; use HTTP/MMS protocol to communicate between media source and the player, the node gets streaming media data from the local cache or media source ,and then calls the video player to play.

The interacting control information between each node is transmitted through the TCP protocol, to ensure that the control information can be received accurately. Thereinto,Control information includes joining and leaving of nodes, updating of neighbor nodes, Keep-alive,etc.

The real media data is transmitted by using RTP protocol. At the sending end,packet is encapsulated by RTP, added whith the time stamp and sequence number, and then transmitted data in the UDP layer, At the receiving end, processing in the reverse order. RTCP control packets transmit data through UDP / TCP and IP.

2.2 Design of buffer

In the system, After the data source node obtains the video data, it will split the data stream into fragments of the same size, and be responsible for marking each fragment with serial number, adds them to the circular queue buffer; but the common node receives the media data fragments through network, then adds them to the circular queue [3].

2.2.1 The two-level cache technology based on the content of streaming media

Setting the two-level buffer in the node of receiving end (open up a block of storage space in the computer memory of the peer node for storing streaming media data) is shown in Fig. 2, set two buffers at the receiving end of the streaming media data, They are the primary buffer(transmission scheduling cache) and the secondary buffer(playing cache).

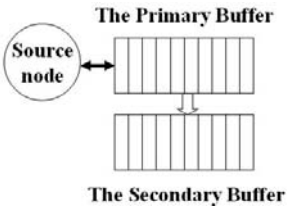

Fig. 2. Design drawing for two-level buffer

The primary buffer is the common buffer, in the system, the streaming media's transmitting and scheduling are operated in the buffer. In the real-time play process in which the node is receiving data, the primary buffer obtains data by the source node, the secondary buffer gets packets from the primary buffer for playing, meanwhile, the primary buffer continues to acquire data. The secondary buffer is playing buffer, after obtains data, it pushes the data to the video players according to the real-time play order and the video frame

play speed of the streaming media data. In the primary buffer, when the rate of getting data is bigger than or equal to the rate required for the players' real-time play, normally play, otherwise, start to pull data.

2.2.2 The two-dimensional circular buffer

The time when the packet reaches the receiving end is different according to the difference of the actual network's delay and quality, there often occurs the circumstances of receiving the disordered packets. The role of the primary buffer is to arrange the blocks which arrive disorderly into sequential and continuous blocks, and output to the play buffer, and perform the transmitting and scheduling with the neighbor node [4]. A good block buffer structure can simplify the design of algorithm, reduce the memory fragments and improve the execution efficiency. This paper proposes a circular queue buffer structure to buffer part of data to compensate for the effects of delay and jitter.

The circular queue buffer structure proposed in this paper plays a very important role in the whole algorithm. The specific structure is: use each big block as the unit to store data, the total number of blocks is total_count, each big block contains n small packets, as is shown in Fig. 3. A queque head pointer:head, a queque tail pointer: tail, when the head and the tail are coincident, the queque may be full, also may be empty, use "full" to mark, if "full" is true, represents that the queue is full, otherwise, represents the queue is empty. The buffer is marked by a two-dimensional array, Seq and Num is used to position a small packet.

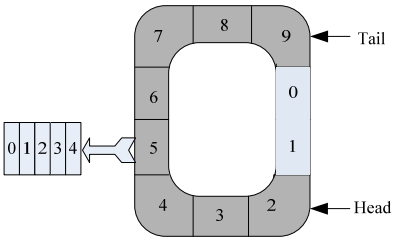

Fig. 3. The circular queue buffer structure diagram

As is shown in Figure 3, the circular queue loop_queue is made up of numbers, thereinto, 0,1,2 …… are the subscripts, stores the valid data in 2,3,4,5,6,7,8,9 between the tail and the head, and the remaining unit is empty. When the new block reaches the buffer, and is stored in the buffer, the head point moves forward, and pushes the tail data of the queue.

Packets in the buffer have three circumstances: packets which have been played, packets which are playing and packets which have not played. When the data comes in, the cache opens up the location of the entire large packets for storing packets which are splitted. Therefore, the data in packets is arranged according to the play order of the streaming media.

2.2.3 Replacement algorithm

When the buffer is full, we need to use the cache replacement policy to replace packets which have been stored by the newly added packets. Because the streaming media files are sequential files, in the streaming media transmission system, the buffer arranges the arriving data according to the order structure which is disordered. Therefore, the streaming media transmission system uses the replacement algorithm of the cache queue, culls the end packets, re-utilizes the vacant buffer space to store the follow-up contents which have not played [5].

2.3 The design of the scheduling algorithm

Packets decide which packets scheduling requests to be sent to the corresponding neighbor node according to the contents of the node and the available bandwidth resources on all the transmission paths, these packets to be scheduled only include those packets whose current state is unrequested or requested but not received, its purpose is to maximize the quality of nodes' service, meanwhile, to meet the bandwidth constraints of all network transmission paths and the contents constraints of each node's cached data.

In the streaming media service, a media service streaming session of the receiving node usually contains multiple supply nodes, the sender sends the different data segments, but the media data is continuous,

therefore, the data segments are ultimately reduced to a continuous media stream by recepters. Then, how to perform the effective and reasonable scheduling for each sender's data transmission is a challenging problem.

The factors which data transmission scheduling needs to considere:

- Network bandwidth between the supply node and the receiving node;
- Media data segment which Each supply node has obtained, and can be provided to the receiving node;
- The length of time from the current time to the time when the scheduled target data segment is to be played back.

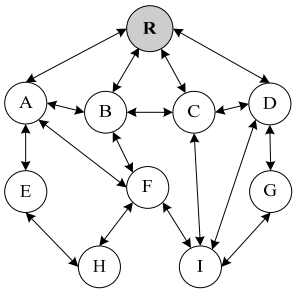

Fig. 4. Data interaction diagram in the streaming media system

As is shown in Fig. 4, the transmission direction of data between the nodes is not fixed, a node exchanges data with other node according to the situation of their caches, so the node needs to know the contents of the cached data with each other [6]. Therefore, use the hybrid streaming media scheduling algorithm based on the priority of play order, exchange data only between the nodes with a neighbor relationship.

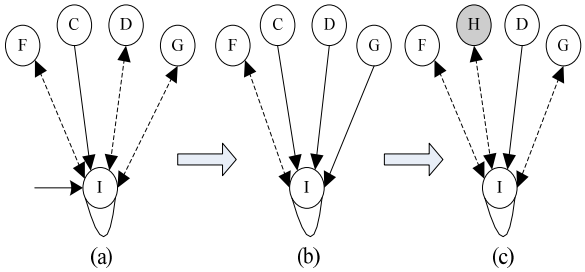

Fig. 5. Data scheduling diagram of the streaming media transmission system

The scheduling strategy of the combination of PUSH and PULL is shown in Fig. 5, when the node I is added, the set S {F, C, D, G} is the parent node which is randomly selected for the system, select a node of larger bandwidth as its PUSH parent node (C). At this time, C is its data parent node, and every second other neighbor node interacts its own BM with the node I, when the neighbor node receives the BM information, if find that self-BM has the vacant data in BM which is sent, then send it. Every A second, perform the statistics for the vacancy of local host and the required IP, cyclically pull all the vacant packets. When the packets which are not played in buffer are significantly reduced, the other neighbor node temporarily acts as its data parent node until the buffer is stable, then picks a new node as its parent node (D), at the same time, if node C exits the system because of the unknown reasons, then it will select a new parent node (H) through the neighbor node F [7].

The scheduling algorithm is as the following

The Node acquires the set N which includes the randomly selected nodes, takes the node for which the bandwidth is fit as its parent node, uses the PUSH model, the node passively accepts data;

Assuming that C are all nodes of the set N (C include the node and all its neighbor nodes), in C, each node stores its own BM and all the neighbor nodes' BM. retrieve its own buffer every a certain time, compare its own BM information with that of neighbor node, get the information which its neighbor node has, but it has not, cyclically comes down on its neighbor node for the vacant data. the number of fragments which the data buffer can store is labeled as N, repeat the following steps:

Step 1: Assuming that the current retrieving serial number is S. View whether the data fragments which are corresponding to the serial number S in its own buffer exist. If exist, then need no request, make the current retrieving serial number plus 1, and return to Step 3. Otherwise, enter Step 2);

Step 2: View whether the data fragments which are corresponding to the serial number S in its own buffer have been requested, and the recent request time from now is no more than a certain time period, such as within two seconds, then need no request, make the current retrieving serial number plus 1, return to step 2. Otherwise, enter step 3);

Step 3: If do not receive the data fragments corresponding to the serial number S, there are two cases. First: have not sent the request of the data fragments. Second: have sent the request, but do not return data within the prescribed time. In the second case, need to increase the number of the peer connections' missing packets which have sent the request, continue to the next step;

Step 4: If the connection C is not empty, then mark that the fragment has been requested in the buffer, and note the requested time and the requested connection C0. Set the corresponding requesting BM position of C0 equal to 1;

when the system receives data fragment, if the packets is received within the time range, then adds it into the buffer queue. Otherwise discards it. If the fragment received is the next fragment which is to be received, then pushes the continuous fragment to the decoding module and waits for decoding and playing, and updates the next fragment to be received. If the update is complete, then returns to step 2.

3. The Implementation of the streaming media transmission system

The rapid development and popularity of Internet provides the strong market momentum for the development of the streaming media business, the streaming media business is becoming increasingly popular. Streaming media technology is widely used in multimedia news publishing, live broadcast online, advertising online, e-commerce, video on demand, remote education, telemedicine, Internet radio, real-time video conference and other aspects of the Internet Information Services [1]. The application of the streaming media technology will bring great changes on the network information exchange and have a profound impact on the people's work and life.

3.1 Compilation of JRTPLib

JRTPLib is an open source RTP library, JThread is an open source thread class, JRTPLib and JThread jointly use, can automatically get the data in the background. If there is no JThread , JRTPLib can also pass the compilation (we need to change the parameter RTP_SUPPORT_THREAD in rtpconfig_win.h file), but if do not add JThread in JRTPLib, we will need program to periodically call the relevant function to get the data. The following are the steps of JRTPLib compilation [8]:

Step1. Release JRTPLib -3.6.0, JThread-1.2.1.zip, release the JThread-1.2.1 under JRTPLib -3.6.0 /;

Step2. Add JThread.h and jmutex.h to JRTPLib- 3.6.0/src;

Step3. Procedures which contain JThread.h and jmutex.h in all files included in JRTPLib are changed to # include "JThread / jmutex.h";

Step4. For debugs of JThread and JRTPLib project,we can choose Debug Multithreaded DLL in run-time library;

Step5. Compile JThread-1.2.1 and JRTPLib-3.6.0, generate JThread.lib and JRTPLib-3.6.0.lib.

Steps of compilation and execution in system:

Step1.Create a new project, add MixCast system files and all the head files of JRTPLib.dsw project to the new project.

Step2. Copy JRTPLib.lib and JThread.lib generated by compiling to the system directory : C:\Program Files\Microsoft Visual Studio\VC98\Lib;

Step3. tells the compiler to use the lib,and adds the following sentences in the front part of the files :

#pragma comment(lib, "JRTPLib.lib")

#pragma comment(lib, "JThread.lib")

Step4. when compiling, likewise,we can also choose the item: Debug Multithreaded DLL(for debug);

Step5. Compile ,execute.

3.2 Sending the streaming media data

The main procedure of sending the streaming media data is :gets the IP address and prot of the receiving end,creates RTP session, specifies the receiving end of RTP packets, sets the defalt parameters of RTP session, and sends the streaming media data.

The system definites rtpsend.h class, preside over the sending of RTP packets.

```
rtpSend* rtp = new rtpSend;
rtp->rtpSendInit(port);    //RTP sending initialization
rtp->rtpAddDest(port,chanMgr->channel->ChIpaddr[]);  //Add the IP of sending nodes
chunk1.write(rtpsend,rawPack.pack[i].data);  //Sending
```

Above all, generates instances of RTP Session class, call the method Create () to initialize it. Set the proper timestamp units (call the SetTimestampUnit method of RTP Session class), and set up the destination address of data sending, RTP protocol allows multiple destination addresses to exist in the same session, we can accomplish the address's adding, deleting, and clearing by calling the method AddDestination (), DeleteDestination () and ClearDestinations () in RTP Session class. After specified all the destination addresses, calls the method SendPacket () in RTP Session class, sends the streaming media data to all the destination addresses

3.3 Receiving the streaming media data

The main procedure of receiving the streaming media data is : gets the port number specified by users,creates RTP session, sets the receiving mode, receives RTP packets, indexes the RTP packets source, gets RTP datagram, deletes RTP datagram.

RTP datagram has three receiving modes (RECEIVEMODE_ALL,RECEIVEMODE_IGNORESOME, RECEIVEMODE_ACCEPTSOME), each receiving mode concretely specifies which incoming RTP packets will be accepted. We can set the receiving mode by calling the method SetReceiveMode () in the RTP Session class, this system uses the default receiving mode of RECEIVEMODE_ALL, all the incoming RTP packets will be accepted.

The system definites rtprevcv.h class, preside over the receiving of RTP packets.

```
ch->rtprev->rtpRevInit(port);   //RTP receiving initialization
ch->rtprev->rtpAddSour(port,SourceIp);     // Add the IP of the source nodes
packet = sess.GetNextPacket();     //Get RTP datagram
```

3.4 The implementation of multicast transmission

In MixCast system, Each node in the application layer multicast is many to many relationship. That is, a node may receive data from multiple nodes simultaneously, and likewise, a node may also send packets to more than one node. This is controlled by each node.

3.4.1 The transmission of a single node to multiple nodes

UDP Multiplexing makes RTP protocol support for multi-point delivery, can meet the requirements of streaming media session between multiple points.we can add the multicast address to the sending list of RTP session by calling the function AddDestination() in JRTPLib, the key codes of implementation are as follows:

```
dw = m_sess.AddDestination(ulIP,usPort);      //Add multicast address to the sending list of RTP session
m_sess.SendPacket();       // Function of sending packets
```

3.4.2 The transmission of multiple nodes to a single node

Use multithread to transmit packets.A pare of transmitting nodes take up a thread, when a node links to the parent node, a new thread will start;when the node exited,and the thread will be ended.

```
stream function
{
while(1) m_sess.SendPacket(); //Sending packets
}
thread1.func =stream;
sys->startThread(&thread1);//start a thread
sys->endThread(&thread1);// end a thread
```

3.5 The implementation of the buffer

3.5.1 The implementation policy for the replacement algorithm

If the cache is full, then need to delete the media object which is the first one stored in the buffer, release the cache space for the new media object until there is enough space to store the new media object. The implementation code is as the following:

```
While (writePos> MAX_PACKETS)
// writePos is the serial number of the packet which is being written in the buffer
{
packets[writePos%MAX_PACKETS] = pack;
// The new media object replaces the media objects of the buffer head
lastPos=writePos;
// The serial number of the packet which is written at the most recent time
firstPos=writePos- MAX_PACKETS;
// ensure that each time the stream objects in the buffer are the recent continuous packets
writePos++;
// The serial number of the packet which is written in the buffer plus 1
}
```

3.5.2 The implementation of the buffer

The implementation code is as the following:

```
chunk2.read(in);//receive packets
for( int i=0;i<chunk2.pack_num;i++)
{
if(count1==0)  //judge whether the packet is the firt one
{//open up the buffer, and put the head packet in it
MemoryStream mem(ch->headPack.data,sizeof(ch->headPack.data));
ch->rawData.writePacket(ch->headPack,true);
}
else
{// judge whether the head packet arrives, if true, then put it in the proper position. otherwise, open up the buffer
unsigned int LastestPos=ch->rawData.getLatestPos();
if(chunk2.seq<=LastestPos)
{
MemoryStream mem(pack_temp.data,sizeof(pack_temp.data))
ch->rawData.writePacket(ch->dataPack,true);//stored in the play buffer
}
}
}
```

4. Implementation of multicast transmission

In MixCast system, Each node in the application layer multicast is many to many relationship. That is, a node may receive data from multiple nodes simultaneously, and likewise, a node may also send packets to more than one node. This is controlled by each node [9].

4.1 Implementation of one to many relationship

UDP Multiplexing makes RTP protocol support for multi-point delivery, can meet the requirements of streaming media session between multiple points.we can add the multicast address to the sending list of RTP session by calling the function AddDestination() in JRTPLib, the key codes of implementation are as follows:

```
dw = m_sess.AddDestination(ulIP,usPort);      //Add multicast address to the sending list of RTP session
m_sess.SendPacket();       // Function of sending packets
```

4.2 Implementation of many to one relationship

Use multithread to transmit packets.A pare of transmitting nodes take up a thread, when a node links to the parent node, a new thread will start;when the node exited,and the thread will be ended.

```
stream function
{
while(1) m_sess.SendPacket(); //Sending packets
```

```
}
thread1.func =stream;
sys->startThread(&thread1);//start a thread
sys->endThread(&thread1);// end a thread
```

5. Concluding remarks

This paper analyzed factors of impacting the packet of size,and proposed a design scheme of adaptive packet, and analyzed the transmission performance of the execute of the design scheme in the actual monitoring system, According to the experimental results, we can see that the design of the new adaptive packet can improve the efficiency and quality of the video transmission.

6. References

[1] Xinyan Zhang, JC Liu, Bo Li, and Tak-Shing Peter Ynm. CoolStreaming/ DOnet:A data- driven overlay network for efficient live media streaming [C].In Proceedings of IEEE INFOCOM, pp.68–73, Mar, 2010.

[2] S.Deering. Host Extensions for IP Multicasting[S].RFC 1112, IETF, pp. 112–116,Apr, 2010.

[3] Liao X, Jin H, Liu Y, Ni L, Deng D. Anysee: Peer-to-Peer Live Streaming[C]. IEEE INFOCOM, pp. 271–350, Feb, 2010.

[4] R. Frederick, Blue Coat Systems Inc., V. Jacobson. RTP: A Transport Protocol for Real-Time Applications[S].RFC3550, IETF, pp. 65–68, Jun, 2010.

[5] Li Fan, Pei Cao, Jussara Almerida. Summary cache: A scalable wide-area web cache sharing protocol [J].IEEE/ACM Trans Networkin, pp. 281–293, Aug, 2010.

[6] R.Wooster and M.Abrams. Proxy Caching the Estimates Page Load Delays [C].In the 6th International World Wide Web Conference, Santa Clara,CA, pp. 977-986, Apr, 2010.

[7] Meng Zhang, Jian-Guang Luo, Li Zhao.A Peer-to-Peer Network for Live Media Streaming-Using a Push-Pull Approach[C]. In ACM Multimedia, pp. 361–372, May, 2010.

[8] FRANCIS P. Yoid: extending the multicast internet architecture [EB/OL]. http://www.aciri.org/yoid, pp. 126–130, Oct, 2010.

[9] BANERJEE S, BHATTACHARJEE B, KOMMAREDDY C. Scalable application layer multicast [J]. ACM SIGCOMM Computer Communication Review, pp. 205–217, Oct, 2010.