

A novel Hybrid CDN-P2P mechanism

For effective real-time media streaming

Duyen Hoa HA
Université Pierre et Marie Curie
4 Place Jussieu
telephone: +33 6 23 51 01 83
tyt_g207@yahoo.com

Thomas Silverton
Université Pierre et Marie Curie
4 Place Jussieu
Thomas.Silverton@lip6.fr

Olivier FOURMAUX
Université Pierre et Marie Curie
4 Place Jussieu
Olivier.Fourmaux@lip6.fr

ABSTRACT

In the early years, with the help of high-speed and broadband networking, the content delivery service has been grown up widely. There are a lot of providers for online streaming via Content Delivery Network (CDN), P2P network or hybrid CDN-P2P system.

In this paper, we introduce one of new hybrid solution for real time streaming: novel hybrid CDN-P2P mechanism. Different from others, our solution works on the application level by effective management of playing buffer at the peer-side. By divide the playing buffer into 2 parts, we can profit all the advantages of CDN servers and P2P network: the performance of CDN servers and the cheap cost of using peers to distribute media content.

Keywords

Real time streaming, hybrid P2P-CDN, PeerSim simulator, streaming buffer manager

1. INTRODUCTION

There are two technologies of delivery content most used: by using CDN and by using P2P network. In the CDN architecture, the content is first sent to distributed CDN servers which are placed in many regions. Whenever a user need a content, a server-load balancing will search for closest server that have this content and forward the request to that server. Because of a huge capacity of disk space and large bandwidth of all the servers in the network, this architect gives the very good quality of service. However, CDN servers are usually expensive and difficult to deploy and maintain.

Several solutions have been found to reduce the number of deployed servers [2] [4] [5]. However, the provided service still lacks of quality.

On the other hand, P2P streaming network [7] [8] acts like a de-centralized system. After receiving the content, a peer should become a source of that content to other peers who request this. The higher quantity of active peers, the better delivery service works. Hence, this architect needs a huge number of participation and their availabilities.

Therefore, a hybrid CDN-P2P solution is highly recommended to eliminate all the weak of those two original technologies. By using this architecture, we can have a cost-effective streaming system. This type of delivery system (hybrid CDN-P2P architecture) benefits the advantages of two technologies: use of CDN server assures the best quality of streaming service and use of P2P network reduces the price of system. Because of that, we will have a cost-performance content delivery service.

In a given hybrid CDN-P2P architecture, a CDN server is usually acts as a component to assure the availability of resource and the speed of data transaction. In contrast, a peer is not only a request component but also it supports the CDN server to delivery content, a huge number of good organized peers can reduce a lot of server load. In fact, the hybrid CDN-P2P architecture for best content delivery has been researched in several years [9] [10] [12]. However, those works just make the implication at the ISP-side.

In this paper, we propose a new hybrid solution which integrates both CDN and P2P technologies for live/real time streaming. This solution is based on the effective management of playing buffer at the peer-side to best equilibrate the bandwidth used between CDN side and P2P side. The main idea is to divide the playing buffer into two parts: CDN priority part and P2P priority part. During the playback time, lacked packets in the CDN priority part will be received from CDN servers, and to the contrary, lacked packets in the P2P priority part will be received from other peers.

The advantages of the proposed mechanism are two folds. First, this mechanism will help to reduce the playback time by using CDN servers to get immediately some parts of the needed content during the playback process. Indeed, Live streaming is time sensitive and the playback delays are crucial to get a smooth playback quality. Moreover, the use of P2P technology will help to alleviate the cost for a Content Provider. The consumed bandwidth to deliver the content is not only provided by the CDN but also by all the peers that want to get the content.

Different from other hybrid CDN-P2P systems (which we will discuss in section 2.3), our mechanism works on application level so it is easier to deploy and do not need any modification at ISP's side.

The rest of this document is organized as follow: we present related works of media streaming service in section 2. We introduce in section 3 our mechanism of manager playing buffer. In section 4, we present the simulator we use to validate our new mechanism and in section 5 we present the results of our solution. Finally, we discuss and conclude this work in section 6.

2. RELATED WORKS

2.1 Content Delivery Network

The Content Distribution Network (CDN) is the most used technology for real time content distribution. CDN servers are set of dedicated servers which have a very large bandwidth and a huge capacity of storage so that they can deliver data to large amount of users over the internet. These servers are often organized at a hierarchic structure and are placed in multiple locations over multiple backbones. We call CDN servers as distributed. There are three kinds of servers: server to get and convert media from media source to small chunks (encoder server), server to distribute data in the network (transport server) and edge server to transfer media to end-user (edge server) [2].

Figure 1 illustrates architecture of a content delivery network of Akamai. An encoder server is usually close to the content resource (television channels, radio channels ...) to get the content fastest while the transport server must have a very large capacity of storage because it has to store many distributed data. The rest of CDN network are edge server which is closest to end user. In a hierarchic architecture, an edge server is a leaf which manager its end user. There are also several tracker servers to balance server-load between servers in the network. Sometime, we can use any given server to do this task.

Once the end-user requests for a content, the tracker server will detect the edge server which has the needed content and closest to this user. Then all the request of media will be transferred to that edge server. Whenever an edge server can not provide the content, it will hand-over the request to another edge server or it get it-self content from network. This task of organization is done by server-load balancing which use one or more "layer 4-7 switches". For example, in the system of content distribution of Akamai, they use DNS forwarding mechanism to redirect request come from clients to equilibrate server-load between CDN servers and to make the content distribution more effective.

In a CDN network, to enhance the quality and the reliability, provider can also use some fault tolerant server or backup server [11] to assure that there is always response of a given request and there is no sudden break data transfer. This is one of reason why a CDN network is usually cost too much but can have a best quality streaming service.

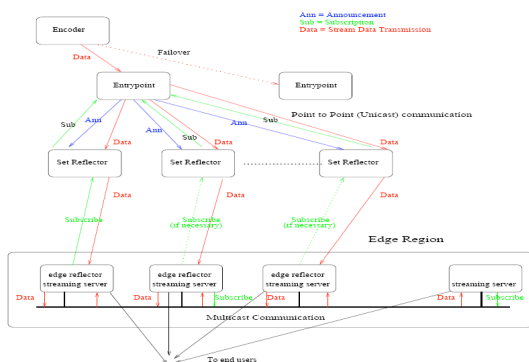


Figure 1: Akamai content delivery network [2]

2.2 P2P streaming

Nowadays, we use even P2P architecture to delivery media content [6] [8]. In a system of media streaming via peer-to-peer network, there is also a media server to get media from media

source and distribute to the network so that this content is distributed by all the participants (called *audiences*).

A mesh-pull P2P live streaming architect often has three major components:

- The streaming peer node includes streaming engine and media player
- The channel streaming server converts the media content into small chunks, each chunk composed by some piece.
- The tracker server provides streaming channel, peer and chunk information for each peer node to join the network.

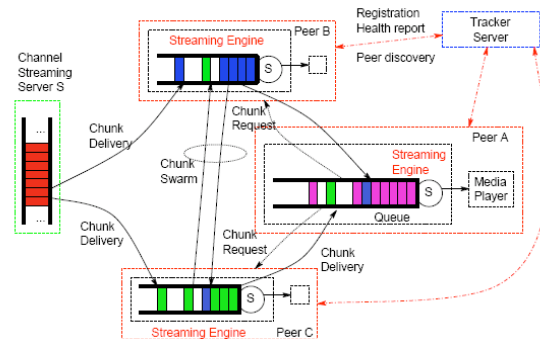


Figure 2: P2P streaming process [8]

When a peer joins the streaming network, it first downloads the list of distributed channels. Then, after select a channel to play, it is registered in the track server. From now, like other peers already registered to the same channel, it participates in the process of streaming the media. During the media playing process, this peer downloads list of pieces available in others peers to know which is the best peer to response the request of lack pieces in his playing buffer. P2P streaming network works in the de-centralized mode, however, there are also some server called tracker server to store peers and channels information. A tracker server can be a normal computer which has a limited capacity of storage but a fast internet connection. That is because information to store is in sample format (text) so that server does not need a lot of space to store them; in contrast, it needs a high speed internet connection to send this information as fast as possible.

The most popular P2P live streaming as we have known is PPLive. It is reported that PPLive supported 1,480,000 audiences viewing a live play at the same time with 1 PC serer and 10Mbps bandwidth. The method they have used is the mechanism of rejection old chunk in playing buffer.

It is clear that P2P live streaming network can support a large number of participants but we can not guarantee a demand of high quality media streaming service.

2.3 Hybrid CDN-P2P architecture

Both the two above technologies have their advantages and disadvantages. A CDN can assure the quality of service by using distributed CDN servers with high bandwidth and large capacity of storage. But these servers often cost too much. In contrast, a P2P Live streaming system is much cheaper but the speed of media streaming depends on the number of joined peers and their availability of content resource, internet connection. PPLive can be

used in cases which need to serve a huge number of audiences in the same time; however, they can not assure the quality of service and can not serve a special requirement of high definition content. Therefore, hybrid CDN-P2P architecture is indispensable to have the best solution for content streaming, in particular for live streaming service.

Using hybrid architecture CDN-P2P for data streaming service has been proposed by many researchers [1] [9] [10] [12]. These enterprises of data streaming: Akamai, Verisign, CacheLogic, Grid Network, Joost have deployed their own CDN-P2P services as well for several years. The principle idea is to equilibrate load of CDN and P2P network.

The best known is the streaming service using CDN-P2P hybrid architect launched by Akamai. In their system, they use CDN servers and P2P network to distribute separately content. During the streaming session, their tracker server always checks the load-balance between CDN servers and P2P network. This server will do a “hand over process” to balancing the load of two content resources (CDN servers and P2P network). The figure 3 demonstrates this process.

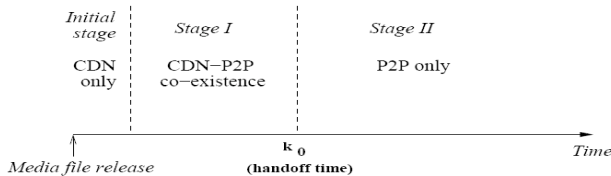


Figure 3: Hand-over between CDN and P2P network [1]

Different from those providers, we propose a new hybrid CDN-P2P mechanism which acts on the effective management of playing buffer at the client-side. Our motivation of this solution is: data slots in buffer which is near the play side must be ready before the scheduled playing times and data slots in the rest of buffer can be filled later. Therefore, we consider the part which is near the play side has more priority of filling data and others have less priority.

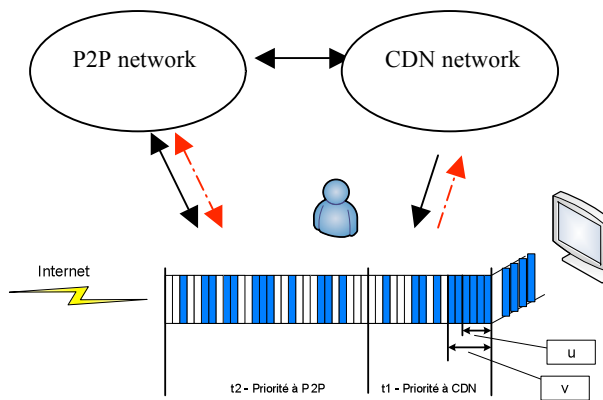


Figure 4: streaming process

From that, we simply treat the playing buffer as two parts: CDN server’s priority part and priority part for peers in P2P network. The CDN server priority part is the part of buffer whose data slot must be received as soon as possible to assure the media playback

in time. In the other hands, the P2P network priority is a part of buffer whose data slot can be received later.

Furthermore, in the streaming network, CDN server is always ready of resource and of higher transaction speed than a peer is. Hence, during a media streaming session, all the requests in the CDN server priority part will be transmitted to CDN servers to assure the scheduled playback time. Request of the rest of playing buffer will be transmitted to other peers.

In this system, we do not need the server-load balancing to equilibrate CDN and P2P network but we use peers in network to involve to this task.

Detail of this mechanism is described in the section 3.

3. BUFFER MANAGEMENT IN CDN-P2P HYBRID SYSTEM

As we have already mentioned above, our solution is based on the effective management of playing buffer. We now introduce our new organized buffer. In this section, we will present our new playing buffer map and present how this buffer works during a media streaming session then describe why this solution can make a cost-effective streaming system.

3.1 Buffer map

Based on structure of P2P streaming architect, we propose a new mechanism for integration of P2P and CDN to make the cost-effective real time media distribution via internet.

As we have mentioned above, our hybrid solution is done by the effective management of playing buffer at the peer-side. We now describe the buffer map used:

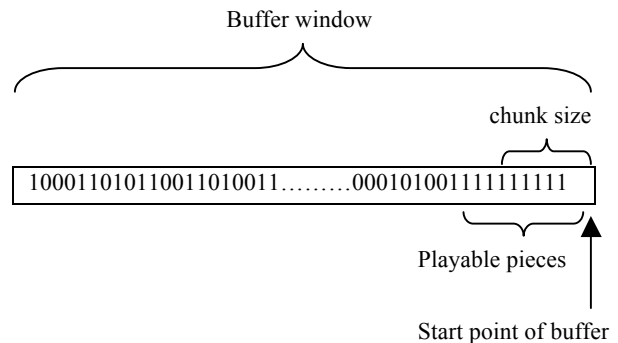


Figure 5: playing buffer map

In a streaming network, the media data is usually organised, transmitted and cached as a unit called *chunk*. *Chunk* is usually uploaded to network by a media server, each with a sequence number so that they can be played in correct ordered by a media player. In our system, for flexible transmission of data, we divide chunk into smaller units called pieces. Then, each chunk has the same number of ordered pieces. A media player can only play out a chunk if all the pieces of that chunk are sent to the media player.

The peer’s local buffer has a length of L piece. That means at any given time, a peer can stores up to L pieces of data. Figure 2 present a buffer map. A buffer map is a presentation of a buffer window which contains data in a unit of *piece*. Each slot of buffer map is equal to a slot of buffer window, value 1 in buffer map

mean that the slot correspondent in buffer window is filled with data, value 0 mean that data is not received.

We call the beginning point of media starting to play is “**start point**”. Hence, the playable media called “**v**” is the numbers of contiguous pieces from the beginning of buffer. A section of continuous data becomes playable if its length is more than the length of a chunk which can be played on media player in each scheduled time. We named the chunk size “**u**” the number of slot of media data. After each play out, the buffer is moved forward **u** slot, in other words, the value of start point is increased by **u**.

We consider these notions:

- **L**: length of buffer in number of piece that the buffer can store in same time.
- **L_{CDN}**: length of buffer part of playing buffer which is reserved to CDN servers.
- **L_{P2P}**: length of buffer part of playing buffer which is reserved to peers
- **v**: playable length: number of continuous pieces from the beginning of buffer
- **u**: chunk size: number of piece in each play out.
- **p**: start point. This value determines the beginning of stream that this peer joins to the streaming. It is measured by the sequence of next chunk to be played. $p = 0$ if a peer starts from the beginning of media stream.
- **N**: number of chunk maximum that a playing buffer can store. Furthermore, as playing buffer can buffer up to **L** piece, we have then:

$$N = \left\lceil \frac{L}{u} \right\rceil$$



3.2 Buffer management – streaming process

The playing buffer is now treated as two parts: part which is reserved to CDN servers (has length L_{CDN}) and part which reserved to peers (has length L_{P2P}). We have a buffer map with two parts too. (Ref. figure 4). The streaming process of media content is:

- Consider that our peers have been registered to the list of playing content A.
- At any instant, the media player of this peer checks its buffer to know the current buffer map. Then it will send this map to others peers. By that way, the latest buffer information of peers in the network is distributed to each other. This work is done frequently to make sure that each peer knows the newest buffer map of other peers. Otherwise, information distributed in network would be useless.
- During the streaming, in parallel with sending buffer information to neighbors, the media player query positions of lacked pieces in order to know lack pieces in buffer and to know to whom it will send the request of each piece. If a lacked piece is in the priority buffer part of CDN, it will send the request of this piece to

CDN servers. Otherwise, the request will be sent to others peers who have that piece available in their buffer. The idea of this division is: there is a part of buffer which must fulfill pieces in shortest time in order to play out media in time. So, all request of piece went from this section must be responded as fast as possible. Of course, there is a part of buffer which we can fulfill pieces slower.

Table 1: Table captions should be placed above the table

Definition	Description
CDN buffer	Buffer part of peer which is more priority to CDN servers
P2P buffer	Buffer part of peer which is more priority to others peers
piece	A packet of data (video or audio)
piece size	Size of piece in number of byte
chunk	A group of continuous packets could be played on peer’s application
chunk size	Number of continuous packets in a chunk
	Data request
	Data transfer

- The rest of buffer is reserved to P2P network because the schedule play times of these media chunk are later than whom in CDN priority part so they do not need to receive pieces so early. By using this architect, we can profit the CDN servers to keep the playing plan in time. Moreover, the P2P network also much reduce charge of CDN. As we can see, CDN servers just have to fulfill some lack pieces because almost all slots in the CDN part have been filled by P2P network before.
- Once the first **u** slot of buffer has been filled, we can play out the media. The play times are already scheduled depending on codec of media source. At these times, the media player checks the buffer map to know that if it can play out a chunk of media. If yes, the media data will be played on the peer machine and the media player moves forward the buffer map to next chunk. It means that the chunk which has been played will be taken out of buffer. Otherwise, play times are delayed and re-scheduled. We call the delay time **d**. Hence, d_i is the delay time of *i*-th chunk. The first chunk scheduled is the chunk from start point of the media stream. Our estimation is: $d_i = 0$ for every value of *i*. In others words, there is no delay in playing media.

3.3 P2P percentage in buffer map and requirement of tolerant

One of the most important things in our solution is the rate of P2P part and CDN part in the buffer map. We must choose the best rate so that we can profit the advantages of all the two components (CDN and P2P). The choice of P2P-CDN rate

depends on the number of CDN used in the system and number of peers participates into the streaming process. The percentage of CDN part is always much less than which of P2P. In general, this percentage must satisfy these conditions:

- Length of CDN part is always higher than length of a play out chunk. In others words, $L_{CDN} \geq u$
- At the same number of peers participated, percentage of CDN part when there are more CDN server is higher than in the case there are less CDN server.

Furthermore, to assure the quality and the reliability of streaming service, we have to use some hand-over works. This is necessary when a request can not be provided in one side, then it will be forwarded to other side. For example, a peer request for some pieces of content which is in the P2P part, it will send the list of request piece to other peers. But in some case, the peers who have those pieces can be unavailable later, so this request could never be resolved. Hence, for better speed of fulfill the buffer map, we can transfer this request to CDN servers even this is not their business.

4. SIMULATION

In order to evaluate the effectiveness of this solution, we have modified a simulator written in Java named "PeerSim" to simulate the network and to give some test case.

4.1 PeerSim simulator

PeerSim is a project open source under the license of GNU, written by Java. PeerSim is a peer-to-peer simulator and it has been designed to be both dynamic and scalable. The philosophy of PeerSim is to use a modular approach in order to easily re-use existing functions. That will much help developers freely to develop this project to the proper functions.

Current version of PeerSim supports two simulation models: the cycle-based model and event-based model. The cycle-based model is based on a very simple scheduling algorithm so it is scalable and can simulate a network of 10^6 node. However, it has some limitations: user can not intervene to the simulation. Therefore, the event-based model is developed to perfect this simulator. In this model, user can modify behavior of node during the simulation so they can simulate a complex operation network. It is the event-base model that we are interested in because we can modify the behavior of node by event.

A life cycle of each simulation executed by PeerSim is:

- Import network configuration.
- Run the simulation and export necessary reports.
- Exit the simulation by some given condition

The network configuration contains information about network size (number of nodes), protocols used, node information, control objects, and some additional information depending on requirement of the simulation. Network configuration is imported to the simulation via an ASCII text file. Hence, it is easy to change the network configuration and run a simulation again without changing the source code and recompile.

After having studied this simulator, we found that by using PeerSim in event-based model, it is suitable to simulate and verify our proposed solution. However, it also requires some

modification of source code to adapt nodes in network to new behaviors. Our modifications are:

+ First, we categorize nodes into two kinds: nodes which are CDN servers and nodes which are peers. A CDN node has larger bandwidth than a peer node does. After that, we add buffer to all the nodes in the network. Size of the buffer is the number of slots which can store a piece during the playing time. Therefore, there is a different between buffer of CDN node and buffer of peer node. The buffer of CDN node has an unlimited capacity while the buffer of peer node has a limited capacity. Because we can suppose that each CDN server is always ready to provide all requested pieces of peer nodes if that request is for content that already be streamed by content source. Concerns the delay of sending and receiving request between CDN server and a peer, we consider it is like delay between peers

After that, each local peer's buffer is divided into two parts like we have already mentioned above. The percentage of P2P part in the buffer is defined later in the configuration file. In a real application, this parameter would be variable in order to adapt to number of CDN server and number of peer clients so that we can profit at maximum work-load of CDN servers.

Cause we use event-base model, we have changed the compartment of buffer so that in each time the buffer receive a responded piece, we will check the buffer to know if it can play out a chunk or not. If this chunk is scheduled to be played out first (depend it sequence number), the buffer will take it out of buffer then move the buffer map forward a length equal to length of a chunk (value u). Then, first u pieces of P2P part will be belong to CDN part and last u pieces of P2P part will empty to receive new pieces.

We do not use reports integrated in PeerSim but we add new reports to the simulator. Those reports help us to show up delay times of each simulation and to know about the process of requesting and sending piece in the networks.

In fact, there is maybe a moment that more than 1 chunk can play out in the same time. In real application, to play a chunk, it will take a little time depend on how length this chunk is. However, in our simulation, we do not take care of playing times but we compute just the delay than a scheduled time of each payout.

Detail of our test cases is discuss in next section – Test Parameters

4.2 Test bed

Major parameters of a network are:

- Network size (N): the number of nodes including CDN servers.
- Network protocol: protocol used in each node.
- Buffer length (l_{buffer}):
- Chunk size (l_{chunk})
- P2P percentage in a buffer (a)
- CDN bandwidth (b_{CDN})
- P2P bandwidth (b_{P2P})
- delay between nodes ($d_{transport}$)
- delay between CDN-source ($d_{CDN-source}$)

We have tried to simulate real-time streaming the video content of 400Kbps (or 50KB/s) which has a quality of o business video conference. In the simulation, we consider each piece of data has

5KB length (or 40Kb). Suppose that each play out is for 1 second of media. Hence, a chunk to play has a length of 400Kb.

CDN servers in our simulations have an upload capacity of 10Mbps. In real world, internet connection speed of peers are usually much variable, but we suppose that each peer in our network have a connection of 512Kbps. We consider that each node can buffers up to 20s of playing time. Our network uses also an unreliable transport to make it more reality. From that, we can than adjust the rate of lost packet.

Table 2: Test parameters

Name	Value used in simulation
Video codec	400Kbps
CDN bw	10Mbps
P2P bw	512Kbps
Lost packet	Random value: 0 – 20%
Delay between CDN-P2P	Random value: 20 – 1000ms
Delay between P2P-P2P	Random value: 20 - 1000ms
Delay between CDN-source	Random value: 0 - 15ms
Buffer length	20s of playing content
Network size	Depend purpose of test
Chunk size	10 pieces

After having taken a look at some peer-to-peer applications, we found that delays between peers are from 20ms to 1500ms [13]. Therefore, we apply this interval of delay to all the transaction, not only transactions between two peers but also transactions between a peer and a CDN server. Furthermore, it takes a little delay when CDN requests content from source so we define this delay ($d_{\text{CDN-source}}$) too. Hence, each transaction j -th has a random delay $d_j = d_{\text{CDN-source}_j} + d_{\text{transport}_j}$.

5. RESULT

In this section, we analyse the performance of streaming service in different scenarios. We simulated with the main parameters described in previous section and we change the number of peers participated, rate of packet lost.

5.1 First delay

First delay is a value of time which a media player must wait for from beginning of streaming process to play out the first chunk. In the media on demand network, first delay is not very important and we can tolerate it but in the real time streaming mode, first delay has a very important role in streaming process. Suppose that the streaming, lately does not have any delay time during the playback but the first delay is too much, so the content of media be played in the media player would be older and there's no meaning of real time service. For example, users joined into a football match streaming session, even there are not any interruption during the session but if the first delay is long, people would see a goal lately. In our simulations, we tried to evaluate the first delay in different configurations to know if we can have a best configuration which assures the smallest first delay.

As we can see in the figure 6, the first delay is usually longer than other delays. The reason is: at the beginning of streaming, all the peers in network do not have any data in buffer. In other words, all slots in buffer map are empty. Hence, it must wait for streaming process to fill in at least first l_{chunk} slots so that media

player can play out the first chunk. Furthermore, when a peer sends a request to other peers or to CDN server, it will receive (if possible) list of pieces responded randomly. For example, the peer named "A" need a list of piece {0, 1, 2, 3, 4, 5, 6, 10, 12, 14, 16, 18} to fill in the buffer, it sends this list to peers which have any of those piece. Because the bandwidth of a peer is limited; one peer can send only k pieces at one time then if $k < \text{size of list request}$, that peer would choose random k piece in his available response list to reply back to peer A. Hence, peer A will receive a chaotic response list from others peers. That is why, to fulfill all l_{chunk} pieces of first chunk (from start point), it can take a longer time.

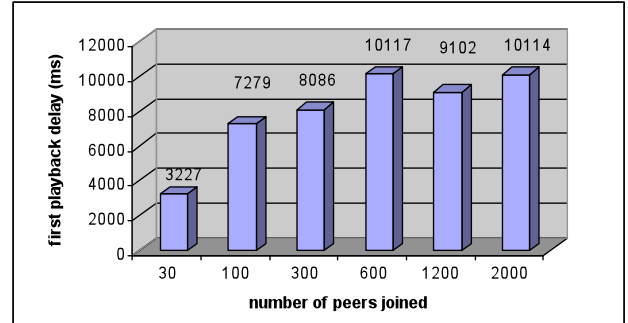


Figure 6: Playing delay of first 100 chunks

From the second chunk, the delay time is almost reduced because after each playback, the buffer map is moved forward so there are slot in P2P part which have data will be transfer to CDN part. Therefore, the CDN part can be fulfilled even much faster. This effect also proves the important impact of P2P part: prepares and reduces waiting time of playing buffer.

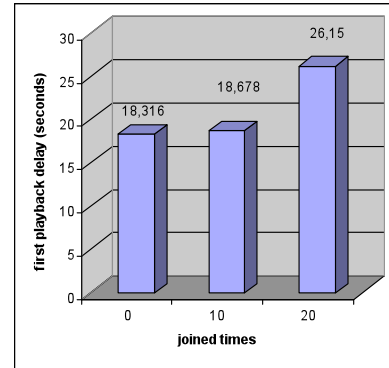


Figure 7: Playing delay in different joined times

However, there are also other moments that buffer must wait to play next chunk because in the simulator. The reason is the way we choose piece to send back each time is randomly like we have discussed above. There are properly pieces that could be received much longer than other. Hence, to play chunks which these pieces belong to, it takes longer time than other earlier chunks.

Joined times also has impact to the first delay of streaming process. In the figure above, we consider a peer join from beginning of stream. Now, we let audiences join at different time and see how first delay change in different nodes. We then take a test with 1000 nodes. The results show that, a peer which joins a streaming later would wait longer to play first chunk.

Additionally, network size can have impact to first delay too. If the number of joined peers in the streaming increases, the first delay of a given peer who joined from the beginning of streaming could be longer. In real live streaming application, audiences usually join in at same time from the beginning of a live streaming session, for example to view a live football match. Therefore, we must reduce the first delay so that the streaming has a meaning of “real time”.

5.2 Buffer map

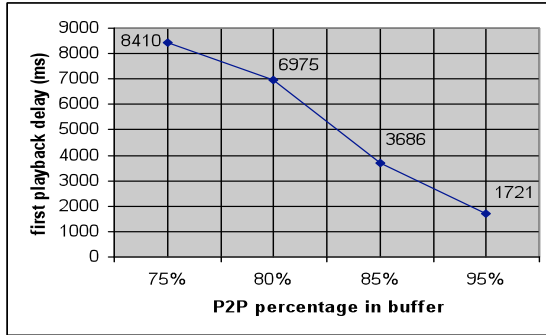


Figure 8: First playback delay in different %P2P (Network with 100 nodes)

The modification of P2P percentage can lead to the word load of CDN server and P2P peers. It is clear that if the percentage of CDN part is large, CDN servers load is increased too; hence for best quality service, we need more servers or each server has to have large internet connection. Therefore, in each given network, we must adjust this proportion to the best value.

In order to reach the effective streaming service, we tried to adjust the percentage of P2P part and CDN part to profit the contributions of CDN servers and peers in P2P network. The rate of P2P percentage is very important in our solution. Because, the destination of a request depend on what part the piece in. Therefore, if we give more length to the CDN part, we can not profit all the help of P2P network and if we do not install a lot of servers; the transfer throughput will be exceeded. In contrast, if the P2P part is too large, the performance of streaming service could be reduced. As in our experiences, in all case, the best percentage of P2P part is from 85% to 95% depend on proportion between CDN servers and peer nodes in the network and the relation between u and v . When the number of peers participated to streaming is large, the value of L_{CDN} would decrease down to u so that streaming process is faster.

In our simulations discussed above, a network with 1000 peers and 1 CDN can stream without playback delay when the P2P percentage is from 90%-95%. This value can be a little different when we apply this solution on a real streaming application.

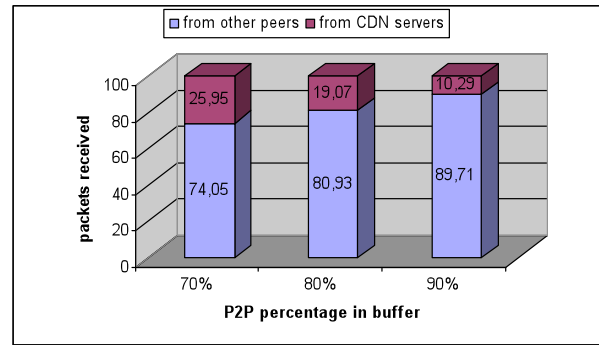


Figure 9: CDN and P2P load balance

CDN streaming performance

We now analyze the result to know the capacity of streaming service and to know how much CDN servers can help.

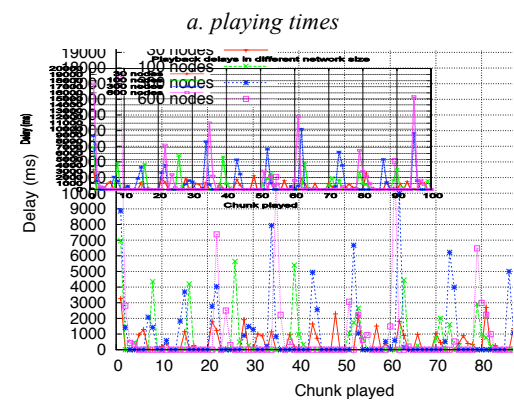
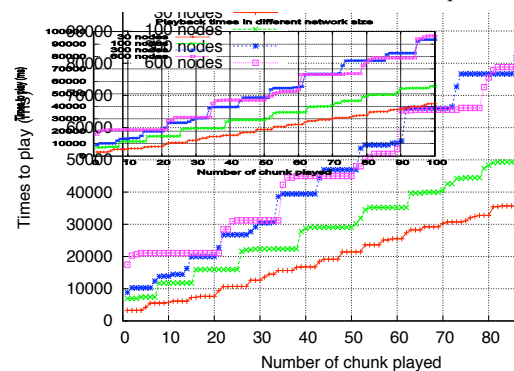


Figure 10: content playing of first 100 chunks

First, we consider a network without lost. We executed the simulation with different number of joined peers and we found that with the given configuration, one CDN can serve up to 2000 peers streaming at same time. The figure 11 demonstrates playing time of first 100 media chunk when the number of joined peers changes. It is clear that when the number of peers increases playing speed is slower. It means user would have to wait longer when the size of network is increased. However, from 300 nodes, we do not see big different.

The figure 11 proves the benefits of using CDN servers. We can see that the help of CDN in this streaming service is a lot. After studying report files of a simulation with 100 nodes, we found that when the number of CDN increases two times, number of

received pieces increases 1,4 ÷ 1,5 times. That means the delay would decrease 30%.

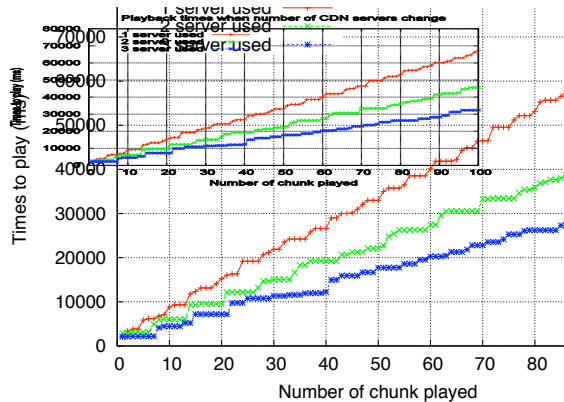


Figure 11: Impact of CDN on playing delay

5.3 Lost packet and its influences

During the streaming of media, sometimes packet can be lost because of bad quality of connection or due to the sudden corruption of a peer. A packet lost rate depends on internet connection of a peer and depend on how Internet providers serve. For example, if a peer is a user of an ADSL connection, the packet lost rate can be small or very small, about 0.1%. Otherwise, if that peer is a user of a dialup connection, the packet lost rate can be up to 20%.

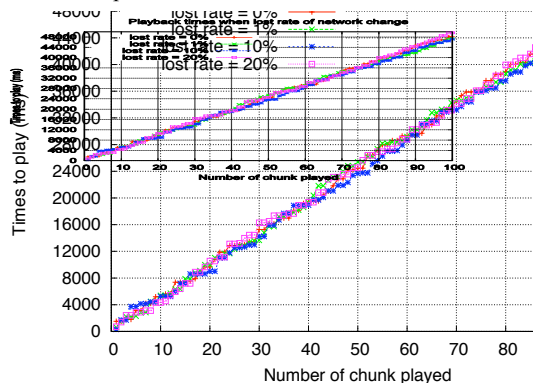


Figure 12: lost packet influences

As the PeerSim can simulate even the lost packet by using an unreliable network transfer protocol, we add connections rates of lost packet. Then, we studied the simulation in different value of packet lost. We found that, when the rate of lost packet is increased, playing delay is increased too but there are not many different. Especially, when the network size is over 300 nodes, lost packet almost does not influent to playing delay. The reason is when the number of nodes is increased; the resource of a given packet is greater. In other words, the availability of a given packet is always higher. So when a peer is corrupted, all pieces that it must response to some given request can be retrieved at other peers. Furthermore, a CDN is always ready to support peer network when it become unavailable.

6. CONCLUSION

By using a new mechanism of management of playing buffer at peer-side, we can create new cost-effective real-time streaming system to distribute content over the Internet. The results of our

simulations confirm that our solution can much improve the performance of a hybrid CDN P2P content distribution network.

However, because of the limitation of a simulator, we can only simulate a network with several thousand nodes. Furthermore, for best streaming service, we have to adjust the P2P percentage to best value but in this paper, we just introduce our estimation. Therefore, we consider our future works are to find out the formula to calculate this value and to evaluate this solution with very large networks of up to million nodes.

7. REFERENCES

- [1] *A CDN-P2P hybrid architecture for cost-effective streaming media distribution* - Dongyan Xu, Sunil Suresh Kulkarni, Catherine Rosenberg, Heung-Keung Chai
- [2] *A transport layer for live streaming in a content delivery network* - Leonidas Kontothanassis, Ramesh Sitaramant, Joel Wein, Duke Hong, Robert Kleinberg, Brian Mancuso, David Shaw, Daniel Stodolsky – PROCEEDINGS OF IEEE
- [3] *Large-scale cooperative caching and application-level multicast in multimedia content delivery networks* - Jian Ni, Yale University and Danny H. K. Tsang, HKUST, Proc. IEEE INFOCOM 2005
- [4] *A Case for Peering of Content Delivery Networks* - Rajkumar Buyya, Al-Mukaddim Khan Pathan, James Broberg, Zahir Tari, Distributed System Online, IEEE Oct 2006
- [5] *Globally Distributed Content Delivery* - John Dille, Bruce Maggs, Jay Parikh, Harald Prokop, Ramesh Sitaraman, and Bill Weihl - Akamai Technologies – SEP 2002
- [6] *On Next-Generation Telco-Managed P2P TV Architectures* - Meeyoung Cha, Pablo Rodriguez, Sue Moony, Jon Crowcroft - Telefonica Research, Barcelona KAIST, Korea University of Cambridge, UK
- [7] *Measuring P2P IPTV Systems* – Thomas Silverston, Olivier Fourmaux–Université Pierre et Marie Curie - NOSSDAV 07
- [8] *A Measurement Study of a Large-Scale P2P IPTV System* - Xiaojun Hei, Chao Liang, Jian Liangy, Yong Liu and Keith W. Ross - IEEE transactions on multimedia - 2007
- [9] *Hierarchical Content Routing in Large-Scale Multimedia Content Delivery Network* - Jian Ni, Danny H. K. Tsang, Ivan S. H. Yeung, Xiaojun Hei – IEEE PROCEEDING '03
- [10] *Hybrid content delivery network and peer-to-peer network* – Afergan Michael, Leighton Thomson, Parikh Jay – WO '08
- [11] *Large-Scale Cooperative Caching and Application-Level Multicast in Multimedia Content Delivery Networks* - Jian Ni, Yale University and Danny H. K. Tsang, HKUST - IEEE Communications Magazine - MAY 2005
- [12] *Understanding Hybrid CDN-P2P: Why Limelight Needs its Own Red Swoosh* - Cheng Huang, Angela Wang, Jin Li, Keith W. Ross - Nossdav '08
- [13] <http://www.bittorrent.com/>