Content Delivery Architectures for Live Video Streaming: Hybrid CDN-P2P as the best option

Melika Meskovic
BH Telecom, Joint Stock Company
Sarajevo, Bosnia and Herzegovina
melika.meskovic@bhtelecom.ba

Himzo Bajric
BH Telecom, Joint Stock Company
Sarajevo, Bosnia and Herzegovina
himzo.bajric@bhtelecom.ba

Mladen Kos
Faculty of Electrical Engineering and Computing
Zagreb, Croatia
mladen.kos@fer.hr

Abstract: **In the last 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), Peer to Peer (P2P) network or hybrid CDN-P2P system. In this paper, we analyse a hybrid solution for real-time streaming: hybrid CDN-P2P mechanism that takes the best of both CDN and P2P content delivery architectures. By adapting the best of breed of both worlds (CDN and P2P), we detail a hybrid model which, thorough quantative analysis, shows the benefits of merging the two architectures. Through simulation, we will show that a hybrid CDN-P2P approach is a much more stable platform whilst at the same time a very cost effective approach in providing a live streaming service to the masses.**

Keywords – live streaming; CDN; P2P; hybrid CDN-P2P

I. INTRODUCTION

Live video streaming has long been projected as the "killer-application" for the Internet. While this expectation has been in effect for several years now, only in recent years with the deployment of increased bandwidth at the end user's side has this promise finally turned into reality (e.g., [14, 15, 16]).

CDN and P2P networks have been most accepted technologies in use for delivering streaming content today. In a CDN scenario, the streaming content is delivered to sibling CDN servers that are placed in various georaphical regions and used to reduce the overall load on the streaming source. When a client requests for the streaming content, the CDN server closest to that client will deliver the stream and not the CDN server acting as the main source of the stream. This architecture (CDN) is caracterized by very high bandwidth capacity and huge disk space thus making it the best option to provide the highest quality stream. However, the economical factors, such as cost of hardware, and the complexities with the scalability make this model less popular.

Multiple other solutions have been found to reduce the number of deployed servers [2][4][5], thus trying to overcome the economical factors of CDN to some extent. However, the provided service on those solutions does not match the quality of service provided by the pure CDN architecture.

On the other hand, P2P streaming network [7], [8] introduces a concept of a completley decentralized system. Once the content is received, a peer automatically becomes a source of the stream to other peers. An increase of active peers increases the quality of the stream delivery since the model is based on peer abundant bandwidth utilization. In order to get the quality of service that is provided by the CDN architecture, a P2P architecture would require a huge number of participating peers.

This brings us to an evaluation of a hybrid CDN-P2P solution, which is highly recommended in order to eliminate all the weaknesses of those two original architectures. By using this type of architecture, we can have a cost-effective streaming system. This type of delivery system (hybrid CDN-P2P architecture) benefits from the advantages of two technologies: the use of CDN servers assures the best quality of streaming service and use of the P2P network reduces the price of system thus resulting in a cost-performance content delivery service.

In a given hybrid CDN-P2P architecture, a CDN server usually acts as a component which assures the availability of requested resource (in our case a live video stream) and the speed of transmitting the requested resource.

In contrast, a peer is not only a request component but is also a support function to the CDN server in the process of content delivery to other peers. A large number of well-organized peers can reduce the server load significantly.

In this document, in Sections 2 and 3, we will cover the basic infrastructure characteristics for both CDN and P2P networks taking a look at their functionality and overall benefits as well as the shortcomings. In Section 4, we will then introduce our new proposed architecture that is made up of a hybrid solution taking both previous architectures as the basis of our new hybrid cdn-p2p system. Section 5 will cover the methodology based on which we plan to simulate the proposed architecture. This section will be followed by analysis of our results, and finally, a conclusion.

II. CONTENT DELIVERY NETWORK (CDN) ARCHITECTURE

The Content Distribution Network (CDN) is the most used technology for real-time content distribution. Grouped into sets of dedicated servers, CDN servers have a very large bandwidth and a huge capacity of storage. This enabled them to deliver data to a large amount of users. These servers are often organized at a hierarchic structure and are placed in multiple locations over multiple backbones. There are three kinds of servers in a CDN group set:

 ISBN: 978-1-61208-192-2

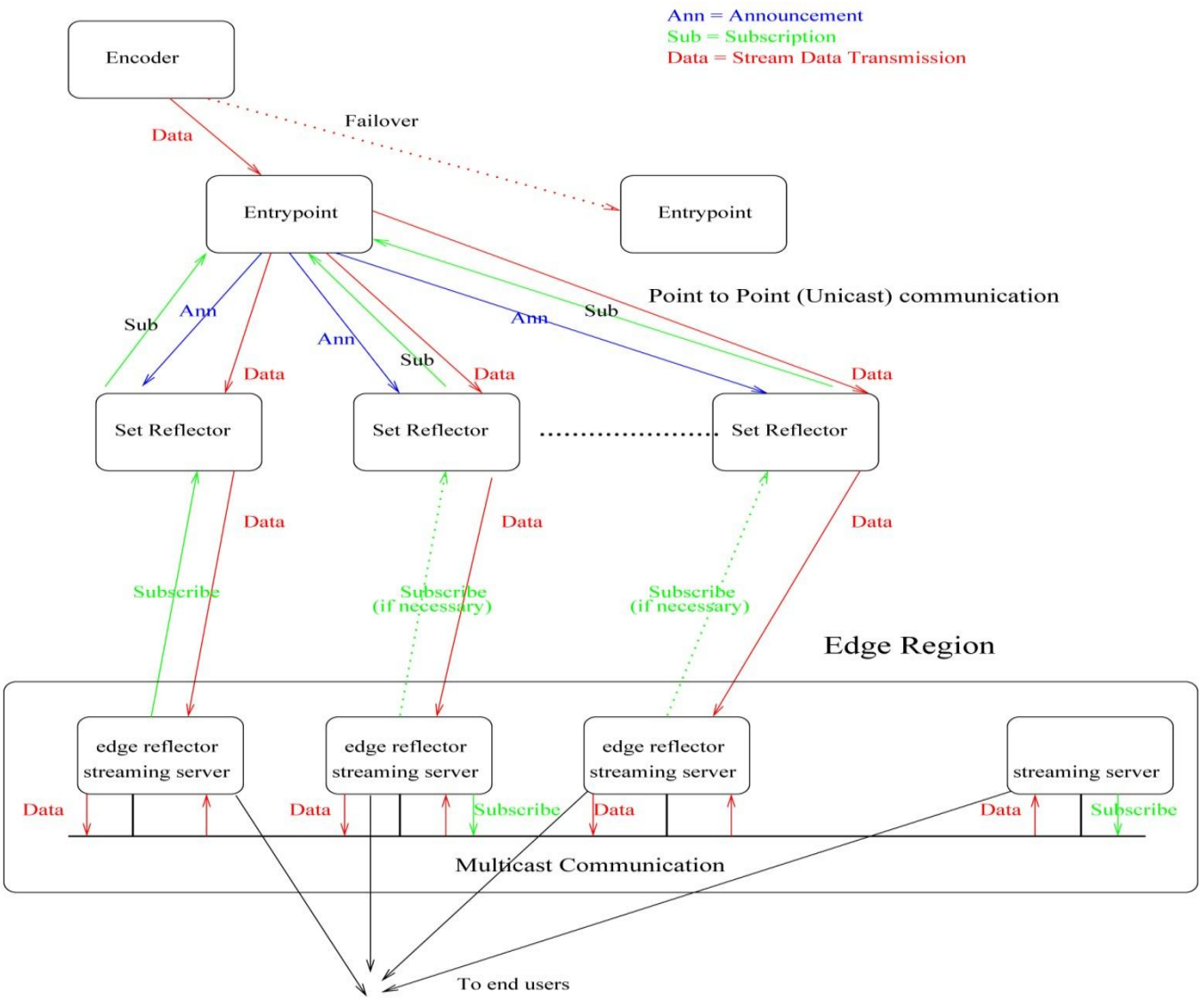

Figure 1. Akamai content delivery network [2]

- Encoder Server - gets and converts media from media source into small chunks
- Transport Server - distributes data in the network
- Edge server - transfers media to end-user

As an example, Figure 1 illustrates the architecture of a Akamai CDN.

Analyzing the architecture from the stream source and all the way to the end user, we first have an encoder server which is by default closest to the stream source (i.e. video stream or a audio stream). This positioning enables the encoder server to get the content fastest as it needs to encode it before it is sent out to the rest of the network.

The next server in the CDN architecture is the transports server which is responsible for distribution/transporting the encoded content to the edge servers. The transport server must have a very large capacity of storage because it has to store a lot of encoded data.

The rest of CDN network is made up of edge servers which are the closest to end user. In a hierarchic architecture, an edge server is a leaf which manages its end user.

For performance optimization, there also can be several tracker servers which are used to balance the server load between all the servers in the network. Sometimes, any given server in the architecture, irrelevant of its original purpose, can be used to do this task provided that it has free resources available.

The process of obtaining the stream from the end user's side starts with a tracker server detecing the edge server that needs to be used which is closest to the originating user.

Once the edge server has been identified (i.e. tracked), all the requests of media will be transferred to that edge server. Whenever an edge server cannot provide the content, it will hand-over the request to another edge server. This task of

 ISBN: 978-1-61208-192-2

organization is done by server load balancing which uses one or more "layer 4-7 switches".

Taking Akmai as an example, a DNS forwarding mechanism is used to redirect request coming from clients in order to equilibrate server-load between CDN servers and to make the content distribution more effective.

To enhance the the quality and the reliability of the stream, a provider can also use some fault tolerant servers or backup servers [11] to assure that there is always response to any given request and that there is no sudden break of data transfer. This is one of the reasons why a CDN type network usually costs a lot. Hoever, this is also one of the factors that enable a CDN network to provide an unrivaled streaming quality.

III. PEER TO PEER (P2P) ARCHITECTURE

Whilst P2P architectures have been mainly used for file sharing in the past, today they are more and more used to delivery media content [6], [8]. In a P2P streaming network, we also have a media server which gets the stream from the source and distributes it to the network so that the content can be distributed by all the participants (referred to as audiences).

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

- Streaming peer node – this node includes a streaming engine and a media player
- Channel streaming server - converts the media content into smaller chunks with each chunk being composed by some piece
- The tracker server - provides the streaming channel, peer and chunk information for each peer node that joins the network.

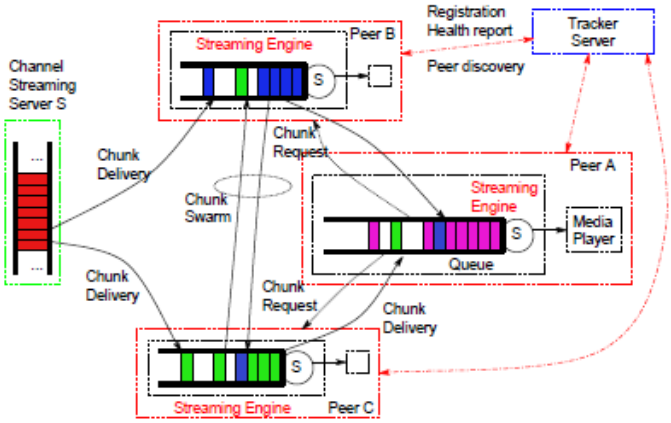

Figure 2. P2P streaming process [8]

On joining the network, a peer first downloads the list of distributed channels available on the network. After selecting a channel from the provided list, the node is registered in the tracking server. From that point onwards, like other peers that have already registered to the same channel, this node participates in the process of streaming the media to other existing nodes or new joining nodes. At the media playing stage process, the peer downloads the list of pieces available in other peers so that it will know which is the best peer that will respond the request of missing pieces in its playing buffer.

Even though P2P streaming network works in a de-centralized mode, there are also servers called „trackers" that store peers and channel information. A tracker server can be a regular computer which has a limited capacity of storage but also has a fast internet connection. That is possible since the information that needs to be stored is in simple text format which makes the requirement for hige disk space mute. In contrast, the tracker server must have a high speed internet connection so that it can send the peer and channels list as fast as possible.

PPLive is the most popular P2P streaming architecture today. It has been reported that a PPLive supported 1,480,000 nodes viewing a live media stream at the same time with 1 PC server and 10Mbps bandwidth.

IV. HYBRID CDN-P2P ARCHITECTURE

Both of the two above mentioned 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 cannot assure the quality of service and cannot 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.

A. *System Overview*

In this section, we describe the general architecture of the proposed system. As shown in Figure 1, there are three major components: (1) Management Center (MC) comprising the DNS-based Global Server Load Balance (GSLB) system, content management and configuration system, and monitoring and billing systems; (2) cache servers, referred to as Service Nodes (SN) that deliver video contents from content providers to end users; (3) end hosts which may either be legacy clients, which directly obtain the stream from the edge servers or LiveSky-enabled clients, which can additionally engage in P2P transfers.

System Management: The Management Center (MC) is responsible for efficient control and monitoring of the proposed system. The DNS-based GSLB system in the MC redirects user requests to the nearest, lightly loaded server [3].

The MC distributes configurations to the SNs using XML messages; these messages use incremental updates to reduce the communication overhead. The configuration

 ISBN: 978-1-61208-192-2

parameters include channel information, source information, operating strategies etc.

B. CDN Overlay

The SNs are organized into several tiers, with Tier0 being closest to the content source and Tiern−1 closest to the end users as shown in Figure 3. We refer to the SNs in Tiern−1 as edge SNs since they are directly responsible for serving end users. The SNs in the remaining tiers are core SNs since their primary responsibility is to act as a distribution overlay to deliver the content to the edge SNs. This hierarchical arrangement is typical of many CDN infrastructures to effectively magnify the total system capacity, reduce the load at the content source, and also leverage the benefits of caching requested contents in higher layers.

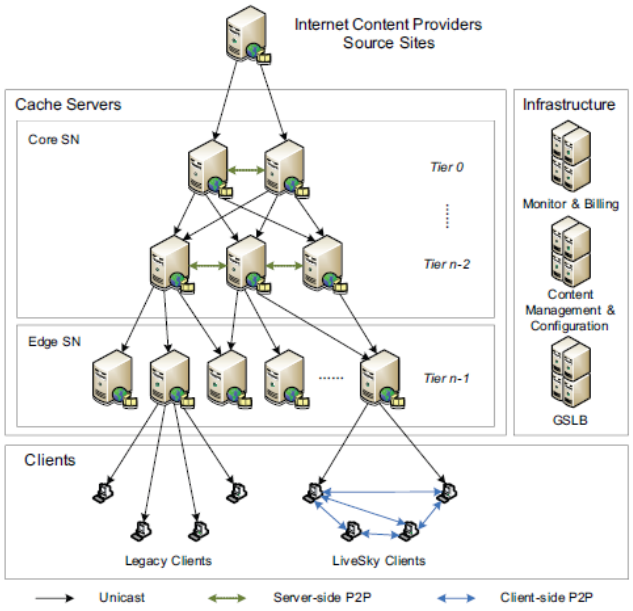

Figure 3. System architecture

Each SN is allocated a unique ID. When SNi boots up, it sends a "alive" message to the MC. The MC then broadcasts the alive message to other SNs. A different SNj can obtain the attributes of SNi (e.g., IP address and TCP port information) from the MC to establish a TCP connection with SNi if necessary.

The server-side distribution mechanism is largely tree-based. However, in order to provide greater reliability in the presence of node or network failures, we allow each SN to retrieve the content either from SNs higher up in the hierarchy (i.e., a lower numbered tier) or from peer SNs in the same tier. Since the edge SNs are responsible for serving end users, they are typically heavily loaded and we disable peering between SNs in the edge tier.

Edge SNs handle client requests and obtain the required contents from the core SNs. Requests from edge SNs are forwarded up the hierarchy until they find a node that has the desired content. To minimize the load at the content source, only Tier0 SNs retrieve content directly from it. The goal of the server-side overlay is efficient data distribution with some measures to guard against some node failures and network delays. As the CDN nodes have high availability and are stable, a tree-based overlay with additional peer edges satisfies the goals of providing reliable, yet efficient data transmission.

C. System operation

A client first obtains the URL for the live stream from the content source (e.g., http://domainname/live1). The GSLB component of the CDN takes into account the client location, the edge SN location, and the edge SN loads to find a suitable edge SN for this client. The client is then redirected to this edge SN using traditional DNS-based redirection techniques [3].

Each edge SN serves multiple roles. First, it acts as a regular server for legacy clients. Second, it serves as a tracker for the P2P operation to bootstrap new clients with candidate peers. Third, it acts as a seed for the P2P operation for the proposed system-enabled clients assigned to it. The edge SNs are pre-configured with some decision logic that decides if a new proposed system-enabled client should be served in CDN-mode or if they should be redirected to the P2P overlay. Finally, the edge SN is used for some optimizations in the P2P operation. Note that the P2P overlays are localized on a per-edge SN basis; i.e., the peers with which a LiveSky enabled node communicates in the P2P mechanism are also assigned to the same edge SN as this node. We discuss these last two roles in more detail in the next section.

D. Client distribution

Legacy Clients: As discussed earlier, there are two types of clients: legacy clients which receive contents directly from the edge SNs and LiveSky enabled clients which can either receive contents from the edge SNs or additionally use P2P mechanisms. An important distinction between the legacy and LiveSky clients is that the LiveSky clients can access a higher quality video stream whereas the legacy clients may only be able to access a lower quality stream. This incentivizes users to install the LiveSky client software and encourages widespread adoption. In our experience, we find that typically more than 50% of users have adopted LiveSky.

LiveSky's P2P Mechanism: Recent proposals [17, 18] demonstrate that a hybrid approach combining the multi-tree [19, 16] and mesh [20, 21] schemes achieves both efficient delivery and robustness to churn. We adopt a similar scheme in the proposed system. The video stream is a single bit-rate encoding (i.e., we do not use any layered coding) and is separated into several sub-streams according to the stream frame id. For example, if the video is divided into six sub-streams, substream0 consists of frames 0, 6, 12, 18, . . . substream1 consists of frames 1, 7, 13, 19, . . . and so on. Peers are organized in a tree-based overlay on a per sub-stream basis. This ensures that all nodes contribute some

 ISBN: 978-1-61208-192-2

upload bandwidth. Additionally, in order to be robust to network or node failures, peers also use a mesh-style pull mechanism to retrieve missing frames for continuous playback.

V. SIMULATION

In order to evaluate the effectiveness of hybrid CDN-P2P solution, we have simulated the network architecture and have given a certain test case.

The NS2 Simulator has been chosen in order to test our scenario. The NS2 is a discrete event simulator targeted at networking research. NS provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. This simulator is used in the simulation of routing protocols, among others, and is heavily used in ad-hoc networking research, and supports popular network protocols, offering simulation results for wired and wireless networks alike [22].

Before we started our simlation, we had to categorize the nodes into two types. The chosen types for the simlation were

- CDN Server type - nodes that act as CDN servers
- Peer types – nodes that act as Peers

Once we have completed the categorization, we introduced a buffer to all the nodes in the network. The size of the buffer used indicates the number of slots that can store a piece of transmitted data during a playback time. Keeping that in mind, all nodes that have been categorized as CDN servers would have unlimited capacity for the buffer size since while the nodes categorizes as Peer types would have limited capacity for the buffer size assigned.

We assume that CDN server nodes are always ready to immediatley transmit, when requested by the peer nodes, any piece of stream that has been already received by the stream source without going back to the stream source.

For transmission between the CDN server node and a peer node, we will assume the same delay as between two peer nodes.

After that has been completed, each local peer's buffer is then divided into two parts as already mentioned above. The percentage of P2P part in the buffer is defined by the configuration file.

In a non-simulated environment (i.e. in an application environment), this parameter would actually be variable and able to adapt to the changing number of CDN servers and peer clients which would profit from the maximum work load of CDN servers.

Major parameters of our simulated 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 ($d_{CDN\text{-}source}$)

We have tried to simulate real-time streaming the video content of 400Kbps (or 50KB/s) which has a quality of a 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.

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 ($d_{CDN\text{-}source}$) too. Hence, each transaction j-th has a random delay $d_j = d_{CDN\text{-}source\_j} + d_{transport\_j}$.

TABLE 1 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

VI. RESULT

We simulated with the main parameters described in previous section and we changed the number of peers participated, rate of packet lost.

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 is no meaning of real-time service. For example, users joined into a football match streaming session, even there are not any interruption during

 ISBN: 978-1-61208-192-2

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 4, 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 lchunk 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 < 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 lchunk pieces of first chunk (from start point), it can take a longer time.

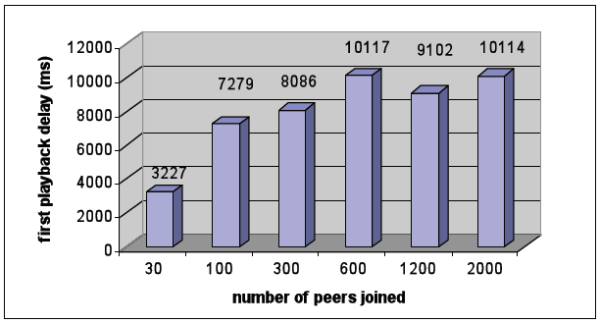

Figure 4. Playing delay of first 100 chunks

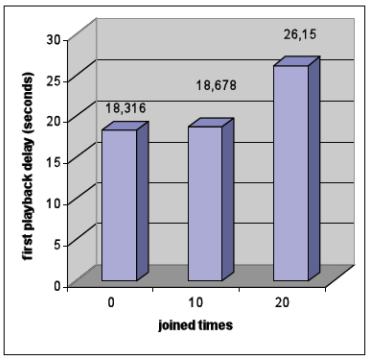

Figure 5. Playing delay in different joined times

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.

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".

VII. CONCLUSION

By combining the best of both approaches in delivering live streaming content to end users (the CDN architecture and P2P architecture) we were able to show that a hybrid model based on the above mentioned architectures can provide a much better service than either architecture individually.

Having setup a simulation environment in NS2 simulator, we tested different cases that all had the same QoS parameters which were used to evaluate performance of the overall network. We tested the setup environment using 1000 nodes and different join times (which may not be the case for live-streaming as usually everyone would join at approximately the same time).

By concentrating on the most important factor of live-streaming service provisioning (i.e. the playout delay), we have shown that in our simulation, with the number of joining peers varying all the time and whilst the QoS factors are being fulfilled, the first playback delay's performance on a network peeks at approximately 600 peers. At this point, the QoS parameters are still met and past this point the first playout delay does not cross over 11000 ms even at a point of 2000 peers joining.

This shows us that a hybrid CDN-P2P approach is a much more stable platform whilst at the same time a very cost effective approach in providing a live streaming service to the masses.

REFERENCES

[1] Dongyan Xu, Sunil Suresh Kulkarni, Catherine Rosenberg, and Heung-Keung Chai, "A CDN-P2P hybrid architecture for cost-effective streaming media distribution," Computer Networks, vol. 44, pp. 353-382, 2004.

 ISBN: 978-1-61208-192-2

[2] Leonidas Kontothanassis, Ramesh Sitaramant, Joel Wein, Duke Hong, Robert Kleinberg, Brian Mancuso, David Shaw, and Daniel Stodolsky, "A transport layer for live streaming in a content delivery network," Proceedings of IEEE, vol. 92, issue 9, pp. 1408 – 1419, September 2004.
[3] ChinaCache CDN Principle, http://en.chinacache.com/index.php/products-and-services/streaming.html., December, 2011.
[4] Rajkumar Buyya, Al-Mukaddim Khan Pathan, James Broberg, and Zahir Tari, "A Case for Peering of Content Delivery Networks," IEEE Distributed System Online, vol. 7, issue 10, pp. 3 – 3, October 2006.
[5] John Dilley, Bruce Maggs, Jay Parikh, Harald Prokop, Ramesh Sitaraman, and Bill Weihl, "Globally Distributed Content Delivery," IEEE Internet Computing, vol. 6, issue 5, pp. 50-58, September 2002.
[6] Meeyoung Cha, Pablo Rodriguez, Sue Moony, and Jon Crowcroft, "On Next-Generation Telco-Managed P2P TV Architectures," Proceedings of the 7th International Conference on Peer-to-peer systems (IPTPS'08), pp. 5-5, 2008.
[7] Thomas Silverston and Olivier Fourmaux, "Measuring P2P IPTV Systems," Traffic, Retrieved from http://www.nossdav.org/2007/program.html, 2007.
[8] Xiaojun Hei, Chao Liang, Jian Liangy, Yong Liu, and Keith W. Ross, "A Measurement Study of a Large-Scale P2P IPTV System" IEEE Transactions on multimedia, vol. 9, issue 8, pp. 1672 – 1687, December 2007.
[9] Jian Ni, Danny H. K. Tsang, Ivan S. H. Yeung, and Xiaojun Hei, "Hierarchical Content Routing in Large-Scale Multimedia Content Delivery Network," IEEE International Conference on Communications (ICC '03), vol. 2, pp. 854 – 859, May 2003
[10] Afergan Michael, Leighton Thomson, and Parikh Jay, "Hybrid content delivery network and peer-to-peer network," United States Patent Application 20080155061, June 2008.
[11] Jian Ni and Danny H. K. Tsang, "Large-scale cooperative caching and application-level multicast in multimedia content delivery networks," IEEE Communications Magazine, vol. 43, issue 5, pp. 98-105, May 2005
[12] Cheng Huang, Angela Wang, Jin Li, and Keith W. Ross, "Understanding Hybrid CDN-P2P: Why Limelight Needs its Own Red Swoosh," Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV '08) Germany, pp. 75-80, May 2008
[13] http://www.bittorrent.com/, December, 2011.
[14] Eric Franchi, "Live Streaming Continues Momentum With March Madness," http://www.mediapost.com/publications/article/101750/, November, 2011.
[15] Marshall Kirkpatrick, "The Numbers Are In, Live Video Online Is Blowing Up," http://www.readwriteweb.com/archives/live_video_big.php, November, 2011.
[16] Liz Gannes, "The Obama Inauguration Live Stream Stats," http://newteevee.com/2009/01/20/the-obama-inauguration-live-stream-stats/, November, 2011.
[17] Mengjie Zhang, Yaobin Lu, Ling Zhao, and Shuiqing Yang, "A Peer-to-Peer Network for Live Media Streaming – Using a Push-Pull Approach," Proceedings of ACM International Conference on Multimedia, pp. 287-290, 2005.
[18] Susu Xie, Bo Li, Gabriel Y. Keung, and Xinyan Zhang, "Coolstreaming: Design, Theory, and Practice," IEEE Transactions on Multimedia, 9(8), pp.1661–1671, May 2007.
[19] Vidhyashankar Venkataraman, Paul Francis, and John Calandrino, "Chunkyspread: Multi-tree Unstructured Peer-to-Peer Multicast," Proceedings of IPTPS, pp. 2-11, Februar 2006.
[20] Xinyian Zhang, Jiangchuan Liu, Bo Li, and Tak-Shing, Pete Yum, "CoolStreaming / DONet: A Data-driven Overlay Network for Peer-to-Peer Live Media Streaming," Proceedings of IEEE (INFOCOM 2005), pp. 2102-2111, March 2005.
[21] PPLive. http://www.pplive.com., December, 2011.
[22] http://www.isi.edu/nsnam/ns/, December, 2011.

 ISBN: 978-1-61208-192-2