International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 02 Issue: 04 | July-2015 www.irjet.net p-ISSN: 2395-0072

Content Adaptation Framework for Servers in Peer To Peer Streaming System

Sreeja S
MTECH Lourdes Matha College of Science and Technology
Trivandrum,India

Abstract — To limit the crash against users demand for smooth video, clear audio, and performance levels specified and guaranteed by contract quality, an innovative approach to face these challenges in streaming media is considered. Here a new idea of boosting the capacity of seed servers to serve more receivers in peer to peer data streaming systems is focused .These servers complement the limited upload capacity offered by peers. The peer requests for a data segment is handled by the server or another peer with a seeding capacity of any finite number with a local cache attached in each peer, which enable the peer to temporarily store the data once requested, so it can be directly fetched by some other node near to the peer without accessing the server there by improving the performance of rendering data. The capacity of the cache in each peer can be designed based on popularity of the segment in cache. Once the peers are cached the peer , data segment request are handled by performing a distributed hash table search strategy, and seed servers boost the capacity of each peer based on utility to cost factor computed each time till it exceeds the seeding capacity. Apart from this selfish peers connected in system can be traced to check for unfaithful peers. This system efficiently allocates the peer resources there by considering the server bandwidth constraints.

Index Terms: *Cache capacity, peer-to-peer (P2P) networks, selfish users, video-on-demand (VoD).*

1. INTRODUCTION

Broadband Internet access is becoming popular in these years, which drives the increase of the number of users who are interested in media streaming services. The streaming services can be divided into two categories: Live streaming and video-on demand (VoD) streaming. In live streaming, all peers watch the same video at a time without knowing which frame to be played next. On the contrary, in VoD streaming, most of peers watch different videos or different parts of the same video. Moreover, if the video interested has been played once, the next frame to be played can be predicted with high probability.

For implementing VoD services, a client-server (C-S) approach can be a first choice but it is inappropriate in supporting many users as the server's upload bandwidth is linearly increased with the number of concurrent users. So the implementation is tried by multicast routing for live streaming and a content distribution network (CDN) for VoD streaming. However, the multicast routing is not practical due to the lack of IP multicast capability at routers. Also, the CDN suffers from the problem of scalability because its deployment cost increases with the number of concurrent users. Hence, in recent years, media streaming services are trying to be implemented by using peer-to-peer (P2P) approaches. They can be classified into two categories: Tree-based and mesh-based. In tree-based approaches, an overlay tree is generated among peers in advance and a peer pushes packets received from a parent towards children.

In mesh-based approaches, a peer connects to some other peers randomly to create a neighbor relationship topology which results in an overlay mesh. Then, the peer can pull packets it needs from its neighbors. In [1], it is shown that the mesh-based approaches perform better than the tree-based ones. The mesh-based approaches can be classified into two types according to the number of videos interested at a time: Single video approach and multiple video approach. In the single video approach, a single video is requested by several peers at a time.

Accordingly if the server has V videos, the problem becomes V sub-problems, i.e., one sub-problem for each video [2]. The advantage of the single video approach is limited due to its difficulty in supporting peers' various

demands. For instance, if the number of concurrently demanded videos is n different ones, the server should transmit at least n videos at a time Moreover, if a peer cannot find another peer watching the same video from its neighbors or some two users watch the same video but different parts, each peer should be served by the server directly.

To overcome the limitations of single video approaches, multiple video approaches have been proposed recently. In these approaches, a peer stores videos that have been watched before. When a peer wants to watch a video, it first contacts its neighbours to find it from their caches. Hence, the upload burden of the server can be shared by many other peers. But the previous works based on the multiple video approach have not considered the constraint of the server upload bandwidth which is crucial to solve the scalability problem. Moreover, the fairness in terms of the upload and download amounts has not been studied either, which gives room for selfish peers to ride the system freely [3].This motivates to design some methods to detect and punish such free-riders.

1.1 Characteristics of P2P Media Streaming System

Characteristics of a peer-to-peer media streaming system: the first three are shared by all peer-to- peer systems, while the last one is unique in peer-topeer media streaming systems:
(1) A peer-to-peer media streaming system is self growing. With requesting peers later becoming supplying peers, the system's total capacity will be amplified: the more peers it serves, the larger the capacity it will have.
(2) A peer-to-peer media streaming system is server less. A peer is not supposed to exhibit server-like behaviour, such as opening a large number of simultaneous connections.
(3) Peers are heterogeneous in their out-bound bandwidth contribution to the system. This heterogeneity may be caused either by different access networks connecting the peers, or by different willingness of the peers to contribute.
(4) The supplying-peer/requesting-peer relation is typically many-to-one, instead of one-to-one as in the general peer-to-peer system. Since the out-bound bandwidth offered by a supplying peer may be less than the original playback rate of the media data, it is necessary to involve multiple supplying peers in one real-time streaming session.

2.RELATED WORKS

The approaches for mesh-based P2P VoD streaming systems can be divided into the single and multiple video approaches. A basically used protocol is BitTorrent. BitTorrent enables to reproduce a file by exchanging segments with neighbors. As the segments for a video may be downloaded in out-of-order, the video cannot be played until the download is complete.

A.Vlavianos, M. Iliofotou, and M. Faloutsos, entitled "BiToS: Enhancing BitTorrent for supporting streaming applications,"[4] modifies its segment selection mechanism to choose segments close to the current played one with high priority. This enables a video to be played while downloading.

C. Dana, D. Li, D. Harrison, and C.-N. Chuah, entitled "BASS: BitTorrent assisted streaming system for video-on-demand,"[5] Another P2P VoD streaming system based on BitTorrent where peers exchange segments with each other using BitTorrent. Meanwhile, peers download segments in order from the media server, skipping the segments that have been already downloaded by BitTorrent.

S. Annapureddy, S. Guha, C. Gkantsidis, D. Gunawardena, and P. Rodriguez, "Is high-quality VoD feasible using P2P swarming?" [6], a P2P VoD streaming system is proposed to support high-quality videos by employing network coding, segment scheduling, and peer-matching algorithms. Network coding replaces the complicated segment selection mechanism in BiToS.

N. Vratonjic, P. Gupta, N. Knezevic, D. Kostic, and A. Rowstron entitled, "Enabling DVD-like features in P2P video-on-demand systems,"[7] , all the segments of a video are replicated by overlaying in advance, and distributed hash table DHT is used to locate and download the segments. It supports digital versatile disk (DVD)- like fast forward and random search functionalities. In the single video approach, however, the server upload bandwidth should be proportional to the number of concurrently requested videos, which causes the scalability problem. To overcome this problem, multiple video approaches have been proposed

In the multiple video approach, the server shares the upload burden with peers. L. Ying and A Basu, entitled "pcVOD: Internet peer-to-peer video-on-demand with storage caching on peers,"[8] is the first scheme that uses

the peer cache. When a peer wants to download a video, it first contacts a tracker to get the address of a peer who caches the video. Then, it can download the video from that peer. In order to reduce the server load and utilize each peer's upstream bandwidth more, some replication methods have been proposed

K. Suh, C. Diot, J. Kurose, L. Massoulie, C. Neumann, D. Towsley, and M. Valleo, "Push-to-peer video-on-demand system: Design and evaluation,"[9] uses a content placement algorithm to improve content availability and use each peer's uplink bandwidth more in a controlled environment.

K. Graffi, S. Kaune, K. Pussep, A. Kovacevic, and R. Steinmetz, "Load balancing for multimedia streaming in heterogeneous peer-to-peer systems,"[10] DHT-based replication method was proposed to achieve load-balancing among peers. Each peer's upload capacity and current load are considered to meet the needs of consumers and providers

Y. Zhou, T. Z. J. Fu, and D. M. Chiu, entitled "Statistical modeling and analysis of P2P replication to support VoD service,"[11] describes the relationship between the number of peers, the number of videos, and the cache capacity at each peer assuming stationary popularity of the videos in the system was formulated.

B. Tan and L. Massoulie, entitled "Optimal content placement for peer-to-peer video-on-demand systems," [12] describes a content placement strategy to maximize peers' uplink bandwidth in P2P VoD systems was proposed.

J. M. Dyaberi, K. Kannan, and V. S. Pai, entitled "Storage optimization for a peer to- peer video-on-demand network,"[13] describes a mechanism for pre-seeding of VoD segments onto the set-top boxes was proposed to minimize uplink traffic in a cable ISP environment. The pre-seeded contents are distributed to other set-top boxes within the ISP using a conventional P2P protocol

W.-P. Yiu, X. Jin, and S.-H. Chan, entitled "VMesh: Distributed segment storage for peer-to-peer interactive video streaming," [14] describes a scheme where each peer uses a part of its local storage to cache videos.

2.1 Motivation

In mesh-based approaches, a peer connects to some other peers randomly to create a neighbor relationship topology which results in an overlay mesh. Then, the peer can pull packets it needs from its neighbors The mesh-based approaches can be classified into two types according to the number of videos interested at a time: Single video approach and multiple video approach. In the single video approach, a single video is requested by several peers at a time. If a peer cannot find another peer watching the same video from its neighbors or some two users watch the same video but different parts, each peer should be served by the server directly.

To overcome the limitations of single video approaches, multiple video approaches have been proposed. In these approaches, a peer stores videos that have been watched before. When a peer wants to watch a video, it first contacts its neighbors to find it from their caches. Hence, the upload burden of the server can be shared by many other peers.

The multiple video approaches have not considered the constraint of the server upload bandwidth which is crucial to solve the scalability problem. Moreover, the fairness in terms of the upload and download amounts has not been studied either, which gives room for selfish peers to ride the system freely .This motivates to design some methods to detect and punish such free-riders.

2.2 PROBLEM DEFINITION

Seed servers have finite capacity, and this finite capacity needs to be optimally allocated to requesting peers such that high quality video is delivered to all peers is considered. The peers need to be cached in order to buffer the received video layers ,so that it can share the segment it has to its requested partners there by improving the system wide utility and hence the rendered quality. A multi-layer scalable video stream is considered, which can be encoded once and can support a wide range of heterogeneous clients, who can decode it. In addition, heterogeneous clients receiving different layers can still share common layers and participate in the same overlay network, leading to a larger pool of resources. Furthermore, scalable coding has lower overhead compared to other coding techniques. Here a P2P streaming systems that: (i) deploy seed servers to complement and boost the capacity contributed by peers, and (ii) serve scalable video streams to support a wide range of heterogeneous receivers are considered.

2.3 PROBLEM FORMULATION

CCPCAN is different from other methods since it meets the constraint of the server upload bandwidth and tries to balance the upload and download amounts of each peer. If the unbalance is allowed without any restriction, a selfish peer may abuse the network to download much more compared to the amount it uploads.

3. EXISTING SYSTEM

The system is used for avoiding scalability problem. It also avoids bandwidth overhead. Here at the time of downloading the video can be displayed on the screen and the time lagging can be reduced. In the proposed system enhanced cache replacement algorithm is used for cache replacement. Enhanced DHT based search can be used for handling searching particular video segment .Reputation and Monitoring System is used for handling selfish peers.

3.1 Structure of the System

The video server contains V distinct videos, each of which is encoded at a bit rate R. Only the video server can upload a video. When a video is uploaded its IP address is set to that of sever IP & status is set to 0. Uploaded videos are stored in a folder named pecan Videos at video server. The length of each video is L(seconds). Each video is divided into several seg-ments each of which has the same length (bytes). Each seg-ment is identified by

- VideoID
- SegNum

N denotes the number of peers. Only registered users can access the system. Videos uploaded by the video server are listed at the users. When a peer wants to watch a video it makes a request to the system and the participating peers are searched for the requested video. The video search is imple-mented by using Distributed Hash Table. Each peer caches the video it watched at its local storage to send when requested by some other peers. While streaming a video a peer can serve a request for that video from other peers. If a requested video is not cached at any peer, that request is served directly by the video server. A video cached at local storage is deleted manually.

3.2 Distributed Hash Table Based Search

The lookup function is implemented using a hash table. Sys-tem searches for peers that caches desired video using a DHT. Videos are uploaded by video server. When a video is added its videoID and serverIP is added to the DHT. First request for a video is served directly by the server. While a peer streams a video V, ip address corresponds to V is changed to ip address of that peer. When a new request comes for V, video server returns the ip address corresponds to V in DHT. Now a peer, whose ip matches the ip address returned by video server, will serve the request.

When Peer A requests for video V, it first finds a Peer B that caches V through DHT in video server. Now Peer A streams V from Peer B's cache and in DHT ip address corresponds to V becomes the ip of Peer B. When a new request comes from Peer C for V it is served by Peer B and in DHT ip became that of Peer C and so on. When streaming is completed ip address corresponds to video V is reset to serverIP. Hence the next request for V will be directly served by the video server.

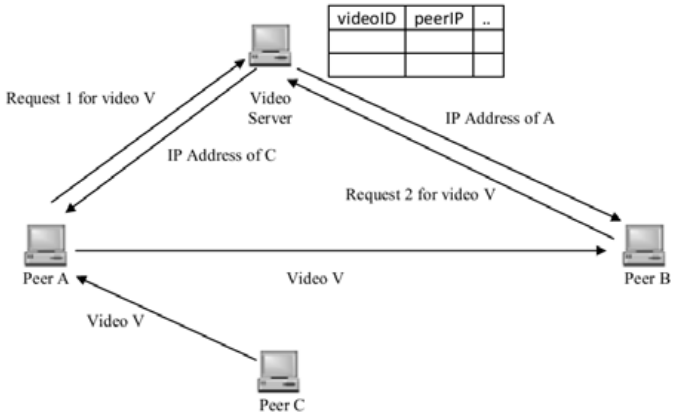

Fig. 1. Steps for Video Search & Cache Replacement

Conventional routing cache systems store destination IP addresses in their cache directory. In this paper a routing cache technique is proposed that stores the most recently used route prefixes, instead of IP addresses, to achieve a significantly smaller cache size. A nesting prefix is partially represented in this cache by its minimal expansions. Such expanded pre-fixes are obtained using an incremental technique without any modifications to the routing table. Consequently, cache works with most of the common route

lookup algorithms and efficiently maintains coherency with the routing table.

3.3 Upload Load and Fairness

As more peers request a video its request rate increases as a result its popularity increases. If M videos are there, each video will have different popularity. A peer that caches more popular segments receives more requests which results in the use of larger upload bandwidth. So upload load for each peer may differ.

In CCPCAN whenever a video is requested, a new cache is created for that video at the requesting peer. So the number of cache for a video is proportional to its popularity. Each peer acts as a temporary server and serves the next request to the video which is being downloaded by it, at that moment. Whenever a new peer streams a video, the ip address corresponds to that video is changed to that of the new peer in DHT. So a new request must be served by the last served peer or last client becomes the new server. So the upload load will be the same for each peer regardless of the popularity of the video.

If the cache capacity for each peer is the same, the same amount of segments is uploaded by each peer onto each other. Peers generating low segment request rates upload more seg-ments than they download and vice versa. The peers with high request rates should upload more segments to achieve the fairness between upload and download amounts, hence they should be with more cache capacity.

Here a peer's cache capacity is proportional to the rate of video request it makes. While Peer A downloads video V, a request for V by Peer B must be served by Peer A. Now the next request for V by Peer C must be served by Peer B and so on. Each peer can upload a video V to some other peer - which requests for V- while downloading V in parallel. So upload and download amounts at each peer will be the same and hence fairness can be achieved.

3.4 Server Upload Bandwidth Constraint

When a peer requests for a video, the video size is calculated and an equal amount of local storage is taken as cache from that peer. Hence while streaming a video the cache capacity of a peer is adjusted adaptively. As a result the server's upload bandwidth constraint can be met. Each peer that caches a video can act as a server. So the server needs to handle only one peer when several peers request the same video at the same time. This will result in a constant packet transfer rate. In ex-isting systems each peer acts as a seed and starts data transfer. As the number of peers increases each node including the server behaves as seed for new coming load thus increasing the packet transfer rate of the server.

4 .PROPOSED SYSTEM

In PECAN[14] when a peer request for a video it will be made available in real time ie viewing while downloading. But it can't able to provide the downloading to a second peer for the same video same time. For that CCPCAN is used ,which uses a distribution system . The distribution system makes it possible simultaneous downloads for different peers. Downloading segments will be taken from peers just downloaded. So the downloading will be done progressively.

4.1 Structure Chart

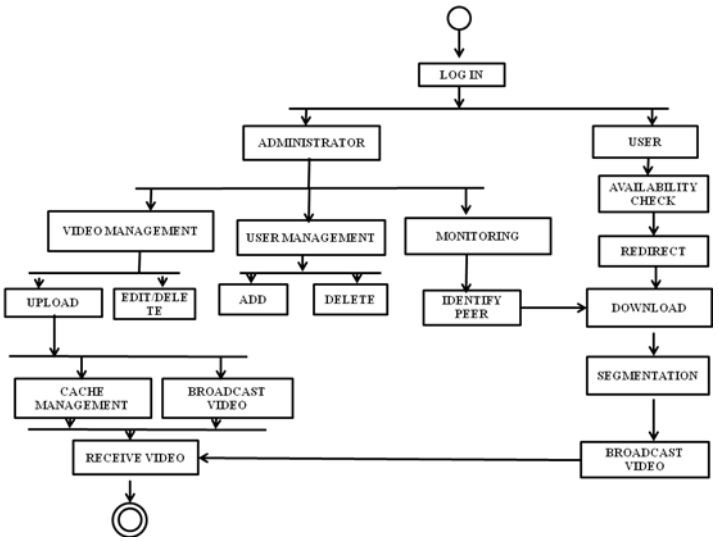

fig 2:Structure chart

4.2 MODULE DESCRIPTION

The system has six modules. They are

- Seed allocation
- Cache management
- Uploading

- Downloading
- Monitoring System
- Distribution System

4.2.1 Seed allocation module

Seed allocation module allocates resources of seed servers. These servers are key components of peer to peer streaming system. Here the seeding resources are allocated to the peer such that a system wide utility function is maximized. Utility function is the expected upload rate of peers as a function of demand and bandwidth.

Peers join the system by contacting the administrator .Administrator controls a number of seed servers. Administrators are also called as trackers, receives a periodic update reports from its peers informing it about available data and capacity. This enables the tracker to monitor its network and keep track of the set of active peers, their contributions and their data availability. Trackers allocate the resources of seed servers to peer's request. The tracker determines the request to be served by one of the seed servers and those to be forwarded to other peers in the system. Peers serve the lower layers of the video first in order to avoid starvation, a situation where some peers are not receiving lower layers while other peers are receiving higher quality .

Trackers queues the requests received from peers and allocate existing requests every few seconds. If there are 'k' requests in the queue each request 'req_k' is break into 'n_k' sub request. Serving each sub request has a cost for the seed server which is the sum of the bitrates of the requested layers. Sub requests are sorted in decreasing order of utility –to-cost ratio and are picked one by one as long as the total seeding capacity allows. A peer can only serve data to peers that are ta earlier segments. Video file is divided into short intervals called video segments. The number of layers received by a peer is assumed constant during a segment but may vary between consecutive segments. In addition to receiving from other peers a peer may request to receive the streams from seed servers.

Seed capacity Allocation Algorithm

```
SCA(K,n[],b[][],C)

//K=number of request, C:seed Capacity
//n[k]=number of sub request in k^th requests
// b[k][j],c[k][j]:utility and cost of sub request(k,j)
//x[]=number of subrequest to serve from request #k;
//z=total utility gained.
  x[] ←CreateArrayOfZeroes(K)
  z←0
  S[]←all sub-request(k,j)
Sort S[] in decreasing order of utility-to-cost ratio
For(k,j)∈ S[] do
    If x[k]> j then
      Continue
   Cost← c[k][j] –c[k][x[k]]
   Utility←b[k][j]-b[k[x[k]]
 If cost<=C then
    C←C-cost
   z←z+utility
    x[k]←j
done
return x[],z.
```

4.2.2 Cache management module

Cache management module give importance to cache capacity required for peers and the replacement of cache policy .This is to be followed in case of full cache. Each peer request segments it want to watch. The segment search is implemented. If a peer wants segment 'j' that is not cached at any peer it should be requested from the server.

Each peer caches the segment it watches at its local cache and sends when requested by some other peers. When a new segment needs to be cached if there is no storage space available then an existing cache should be deleted to create a memory space. When peer 'i' request segment 'j' ,it can find peer 'k' which manages the list of peers that store segment 'j' peer 'k' picks up a $peer_1$ from the list. When a peer stores a segment it may cache some other segments of the same video with high probability.
If a peer 'i' played segment 'j' received from $peer_1$ it is reasonable to search $peer_1$ first for some other segments of the same video. Two case needs to be considered in deciding segment length. First is the case that the segment length exceeds the cache capacity of a peer. A possible problem of using a longer segment occurs when a peer watches a part of it and stops watching it or jumps to some other segments. In this case the peer is not able to cache the corresponding segment completely. Second is the case that the segment length is too short incur much overhead

because of too many search requests. So the segment length should be decided considering these 0together.

Tracker maintains per each [peer, video] pair a list of keeping track of the layers that the peer holds from each segment of the video. The number of entries in this list is the number of video segments and the peers watching and seeding others.

4.2.3 Uploading module

Uploading videos are performed by the administrator based on upload bandwidth. Each peers are provided with certain upload bandwidth. Administrator adds new videos into the video list from peers .New videos can also be stored which are not available in any of its peers. Administrator provides video id and tags for each video for identification. Since peers upload bandwidth is typically less than its entire upload bandwidth for uploading pieces of video being downloaded. A peer that is watching any stream can seed one or more video files. T_{seed} is the average effective seeding time for each file . The peers are expected to use their upload bandwidth for serving lower layers first and also as many layers as they can upload. This is to avoid starvation of some peers in the system while also getting some higher quality layers distributed in the network. If the upload bandwidth is higher than the total bitrate of the demanded layers the peer is able to serve more than its download stream rate.

4.2.4 Downloading module

Downloading is performed by the user .For each user request the videos are downloaded in parallel from different peers. The peers that have entire file is called seeds and peers that only have parts of the file called downloader and are still downloading the rest of the video. A tracker maintains information about the peers participating. New peers wanting to download a file are directed to a tracker, which provides each new peer with the identity of a random set of participating peers. Each peer typically establishes persistent connections with a large set of peers consisting of peers identified by the tracker as well as by other peers to which the peer is connected. The peer maintains detailed information about which pieces the other peer have.

Videos are stored as segments in each peer. When user requests a particular video it will be selected from different peers as segments and are downloaded to user buffer. A video stream can be encoded into several layers. Each layer are encoded and provided with a layer number for identification.
Each encoded layer will be further divided into segments each with one unit play back time. A layered segment will be identified by two identifiers layer number and sequence number. The layer number start from 1 and the segment number gives the time instance when to play the segment. For example LS_{23} means 3rd segment in layer 2.The encoded layers are downloaded and stored at the user buffer only after receiving all the lower layers higher layers are decoded and played. Downloading rate and video details of current video will be provided to the user. Decoded segments are viewed by the user without completing the full video download .Downloading also considers the buffer capacity .All the folders of user are displayed for knowing the available storage space, where the downloaded video will be saved. If their is no available space then cache replacement takes place by deleting less frequently used videos. The downloaded video will be available at the video list so that the user can then be a seed to other peers requesting the currently downloaded video.

4.2.5 Monitoring System

The monitoring system traces all the selfish and non selfish peers. Selfish peers are also called as unfaithful peers. They obey the system protocol .It doesn't respond to the server requests .Unfaithful peers may not transmit segments they have cached when requested by other peers. Monitoring system also traces all the peers connected to the network , peers that have videos with them, peers that have videos by not connected ,peers without video ,all the registered users etc.. Then stores their details including video details for further references.

4.2.6 Distribution System

In normal P2P streaming systems when a client requests a video, it will be downloaded from the server or another peer based on availability .while the client ,requesting the video is being downloaded no other client can download the video from the first client .Only after the downloading is completed any other client can download and play it.But using the peer cache Adaptation mechanism this can be made possible .In this system when a client request and start downloading the segments of a particular video another client/peer can also download and view the segments progressively Also that while downloading IP address of the source of video also changes .ie if client A request a video V available in the server S , IP address of

the video source will be IP_S . So client A will be downloading from the server S .But if another client B also requests for the same video V ,it will be downloaded from the client A other than S .At that time the the IP address of the source video will changes to IP_A in client B, which means that the segments of the downloading video are available in client A also. So the server overload can be reduced.

5. PERFORMANCE ANALYSIS

5.1 Server Bandwidth Rate

CCPCAN is evaluated in real time and a performance analysis is done. CCPCAN is compared with BASS. For streaming a video of size 699Mb, BASS takes 1 hr (actually it is 8 hrs, since number of seed is low set it as 1 hour) while CCPCAN takes only 15mins (with 3 peers). The server bandwidth usage of PVP compared to BASS is evaluated. The server bandwidth usage based on the number of nodes and packet transfer rate (kb/s) is depicted. In CCPCAN, when number of nodes increase the packet transfer rate is constant because each node will behave as a temporary server for data transfer so the server need to handle only one node at time. But in other system, the each node will act as a seed and start the data transfer. If the number of nodes in-creases, all nodes including the server behave the seed for new coming load. So the packet transfer rate of server will increase while the number of nodes increases.

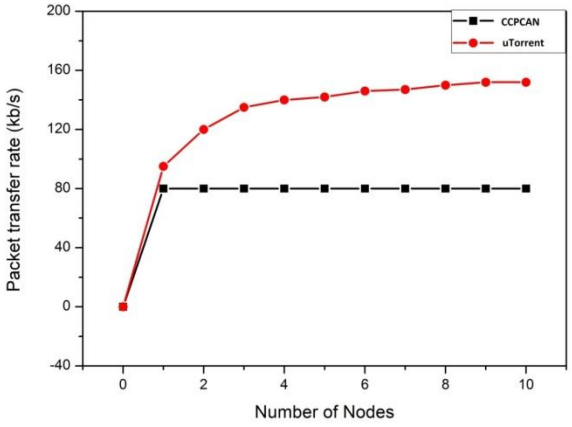

Fig 3. Comparison of packet transfer rate

5.2 Query Accessibility Rate

Here the query accessibility rate of CCPCAN and BASS is evaluated on the based on the number of nodes and data accessibility time(s). Here in CCPCAN initial value for data accessibility is an constant value because the time required to identity the local server. But in BASS this initial value is higher than CCPAN because it required to collect all the available peers in their network. When the number of nodes increases the data accessibility time is suddenly decrease for CCPCAN due to the easily availability of server. But in BASS the data accessibility time almost maintain the same rate, because all the time it re-quires initialing all peers for accessing data.

From this performance evaluation it is proved that CCPCAN provides more downloading speed when compared to BASS. Start-up delay to start watching a video is also reduced in CCPCAN.

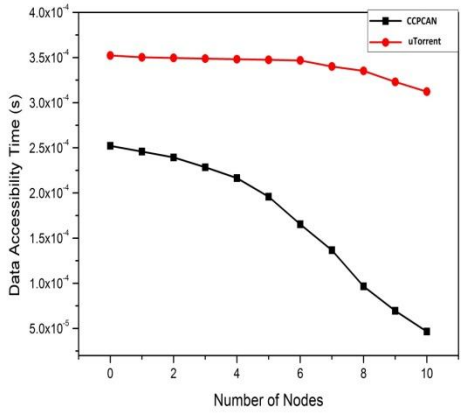

Fig 4. Comparison of data accessibility time

6. CONCLUSION

CCPCAN is a novel cache adaption method proposed for P2P VoD streaming systems to meet server's upload bandwidth constraint. Server bandwidth usage is a constant at some Packet transfer rate in the proposed method. Fairness between up-load and download amounts at each peer can also be achieved.

REFERENCES

[1] N.Magharei and R. Rejaie, "Mesh or mmultiple-tree: A comparative study of live P2P streaming approaches," in *Proc. IEEE INFOCOM*, Apr. 2007.

[2] C. Huang, J. Li, and K. W. Ross, "Can Internet video-on-demand be profitable?" in *Proc. ACM SIGCOMM*, Aug. 2007.

[3] V. Gopalakrishnan, B. Bhattacharjee, K. K. Ramakrishnan, R. Jana, and D. Srivastava, "CPM: Adaptive video-on-demand with cooperative peer assists and multicast," in *Proc. IEEE INFOCOM*, Apr. 2009.

[4] A. Vlavianos, M. Iliofotou, and M. Faloutsos, "BiToS: Enhancing BitTorrent for supporting streaming applications," in *Proc. IEEE Global Internet Symp.*, Apr. 2006.

[5] C. Dana, D. Li, D. Harrison, and C.-N. Chuah, "BASS: BitTorrent assisted streaming system for video-on-demand," in *Proc. IEEEMMSP*, Oct. 2005.

[6]S. Annapureddy, S. Guha, C. Gkantsidis, D. Gunawardena, and P. Rodriguez, "Is high-quality VoD feasible using P2P swarming?" in *Proc. WWW*, May 2007.

[7] N. Vratonjic, P. Gupta, N. Knezevic, D. Kostic, and A. Rowstron, "Enabling DVD-like features in P2P video-on-demand systems," in *Proc. ACM P2P-TV*, Aug. 2007.

[8] L. Ying and A Basu, "pcVOD: Internet peer-to-peer video-on-demand with storage caching on peers," in *Proc. DMS*, Sept. 2005.

[9]K. Suh, C. Diot, J. Kurose, L. Massoulie, C. Neumann, D. Towsley, and M. Valleo, "Push-to-peer video-on-demand system: Design and evaluation," IEEE J. Sel. Areas Commun., vol. 25, no. 9, pp. 1706–1716, Dec. 2007.

[10]K. Graffi, S. Kaune, K. Pussep, A. Kovacevic, and R. Steinmetz, "Load balancing for multimedia streaming in heterogeneous peer-to-peer systems," in Proc. NOSSDAV, May 2008

[11] Y. Zhou, T. Z. J. Fu, and D. M. Chiu, "Statistical modeling and analysis of P2P replication to support VoD service," in Proc. IEEE INFOCOM, Apr. 2011.

[12]B. Tan and L. Massoulie, "Optimal content placement for peer-to-peer video-on-demand systems," in Proc. IEEE INFOCOM, Apr. 2011.

[13] J. M. Dyaberi, K. Kannan, and V. S. Pai, "Storage optimization for a peerto-peer video-on-demand network," in Proc. ACM MMSys, Feb. 2010

W.-P. Yiu, X. Jin, and S.-H. Chan, "VMesh: Distributed segment storage for peer-to-peer interactive video streaming," IEEE J. Sel. Areas Commun., vol. 25, no. 9, pp. 1717–1731, Dec. 2007.

[15]Jongtack Kim and Saewoong Bahk, "PECAN: Peer Cache Adaptation for Peer-to-Peer Video on-Demand Streaming",IEEE journal of communications and networks, vol. 14, no. 3, june 2012.

