The Mathematical Models for Different Start Video Broadcasting

Hathairat Ketmaneechairat
King Mongkut's University of Technology North Bangkok
Bangkok, Thailand
e-mail: hathairatk@kmutnb.ac.th

Phoemphun Oothongsap
King Mongkut's University of Technology North Bangkok
Bangkok, Thailand
e-mail: phoempn@kmutnb.ac.th

Anirach Mingkhwan
King Mongkut's University of Technology North Bangkok
Bangkok, Thailand
e-mail: anirach@ieee.org

***Abstract*—The different start video broadcasting is a new approach to improve the service of P2P video streaming applications. This approach allows unpunctual users to view broadcast programs from the beginning during server broadcast time. This paper proposes the non-cluster and cluster model for different start video broadcasting. The buffer management and mathematical model are proposed to estimate the performance of non-cluster and cluster model. These models are based on an application layer MESH network. These models are composed of five processes: peer join/leave, peer exchange information, peer selection, buffer organization and segment scheduling. The proposed models are simulated and verified by using NS-2. Moreover, the mathematical model is proposed to evaluate the performance metrics of server load, peer load, control message and buffer size. The results show that (i) the unpunctual users with different joining time are able to view the first video frame, (ii) the video server load is reduced drastically, (iii) the peer load is also reduced, (iv) the number of control messages exchanged between the nodes is reduced, and (v) the buffer size is constant. Moreover, the performance of cluster model is better than the non-cluster model.**

Keywords-Peer-to-Peer (P2P); IPTV; live video streamin; video on demand; Different start video broadcasting;

I. INTRODUCTION

Peer-To-Peer applications (P2P) have become very popular among Internet users. P2P technologies offer obvious advantages over content delivery network or content distribution network (CDN). P2P technologies improve system scalability with low implementation costs. P2P content delivery is an important technique for commercial systems such as IPTV. There are a lot of popular P2P file-sharing systems that support downloading such as Napster [2], Gnutella [3], Kazaa [4], BitTorrent [5], and eDonkey [6]. The main area of usage is P2P-based file sharing systems, like BitTorrent. Unlike traditional client-server architectures, peers in the network act as both client (leech) and server (seed). A peer not only downloads file from the network, but also uploads the downloaded file to other users in the network. Parts of the files are exchanged over direct connections between the peers. To enhance the system scalability and reduce the cost, several P2P video streaming applications have been published by using P2P technologies for the streaming of video and audio contents. P2P technologies are provided content distribution services for live video streaming and video-on-demand (VoD). CoolStreaming [7], PPStream [8], Sopcast [9], UUSee [10], and PPLive [11], are demonstrated by the huge popularity of P2P video streaming applications. These works [7, 8, 9, 10, 11] cause unpleasant problems. The first problem is that a far away connection increases network traffic and thus decreases network resource utilization. The second problem is a heavy tracker load. These problems can be delineated by using a hierarchical architecture as explained in [12]. In [12, 13, 14, 15], the cluster concepts for P2P systems are introduced.

For the live video streaming, live video contents are disseminated to all users in real-time. Hence, all users in the system can watch the same part of the stream at the same time. If users join the program later on, they will miss the beginning of the stream. The advantage of live video streaming is that the users can watch video stream almost immediately, without having to download an all file. The disadvantage of live video streaming is that the quality is limited by available bandwidth of each node and when the number of users is large, a server has limited bandwidth to support all users.

For the video-on-demand, the users can watch the video stream anywhere at any time. Multiple users may watch the same movie at the different playback times. The advantage of video-on-demand is higher quality. The drawback of video-on-demand is the users have to store the whole file. The result is a large buffer size.

Besides these two categories, there is another application that takes advantages of the live video streaming and the video-on-demand characteristics. This application is called the different start video broadcasting [1, 16, 17, 18]. The unpunctual users can watch the video stream from the beginning during server broadcast time. By mixing a Peer-to-Peer download concept with a live broadcasting one, a new node can find users who have the needed parts of the stream, and can use them as sources for download.

For the example, there is a game of FIFA World Cup which is start at 3 PM. and the game is end at 5 PM. A big amount of viewers will connect to the network and select the channel of the game. When the game starts at 3 PM, the viewers can load and view the game in real-time. After the game has started for 15 minutes, a new viewer decides to join the stream. The new viewer will have 2 choices: (i) view the game as the server broadcast or (ii) view the game from the beginning. With the first choice, this broadcast will feature live video streaming while the second choice will employ different start video broadcasting by mixture live streaming and video-on-demand features.

To support the different start video broadcasting applications, the non-cluster and cluster model are proposed. The non-cluster model is composed of five steps: peer join/leave, peer exchange information, peer selection, buffer organization and segment scheduling. The cluster model consists of eight steps: peer join, super node selection, backup-node selection, peer exchange information, peer selection, buffer organization, segment scheduling and leaving peer. The buffer management is organized as data buffer, buffer map and sliding window. The data buffer is divided into three parts: playback buffer (old chunks), display buffer (fresh chunks) and future buffer (future chunks). The Mesh-based architecture is used to exchange data between users. For the non-cluster model, there is only one tracker. For the cluster model, there are multiple trackers. The tracker is used to keep a list of all peers. In this paper, the mathematical model is used to determine starting delay, buffer size, peer list search and server load. The probability of download chunks and peer selection are proposed. The efficiency of the different start video broadcasting can be estimate. The proposed model is simulated and verified by NS-2. The results are affected by vary performance metrics, server load, peer load and the number of control messages. The performance of the non-cluster and the cluster model are compared.

The remainder of this paper is organized as follows: Section II describes related works, including the overview of P2P video streaming and P2P clustering. The non-cluster and cluster system design are illustrated in Section III and Section IV. The mathematical model is proposed in Section V. Section VI shows the experimental results. Finally, conclusion and future work are presented in Section VII.

II. RELATED WORK

This section describes the review of literatures regarding to the study. The challenge in P2P video streaming application is designed to be scalable and efficient for realtime streaming service on the Internet. The application supports live, video-on-demand (VoD), and both live and VoD streaming services. The application is aware of the format, the required bandwidth, the structure of the content delivery, and allows the content to be played smoothly during the delivery. There are many P2P video streaming application development of efficient and robust P2P content delivery network (CDN).

CoolStreaming or DONet [7] is a P2P live video streaming application for only one channel. There is no static streaming topology. Every node in the network can be a video-source which produces the content for neighbor nodes. Every node acts as an origin node keeping all of video segments. An original node is a single point of failure when it leaves. The departure nodes and dead nodes do not send any control messages. This may be the cause of packet loss.

PPStream [8] is a widely popular peer-to-peer (P2P) Internet TV application and it employs P2P video streaming network software that is similarly to BitTorrent. It can broadcast TV programs stably and smoothly to broadband users. Compared to traditional stream media, PPStream adopts P2P-streaming technology and supports full-scale visit with tens of thousands of users online. PPStream transfers data mainly using TCP and a few UDP packets. There are at least four types of nodes on the PPStream network: a unique channel list server, trackers, peer list servers and media chunk sharers. When the user launches the PPStream Client, the software will automatically connect to the channel list server to update the channel lists. After the user selects the channel to watch, the user has to wait several or tens of seconds for playing. During this period, the client will firstly request the peer list server for some other users watching the same channel. Each user is identified by its IP address and the listening port. The client will try to connect these peers to download data chunks.

Sopcast, UUSee and PPLive are channel-based systems, which provide a lot of different video streams on different channels. So each of the application networks need at least one media encoding server, where the video streams are created and stored, and a well known channel server where the clients can get information about available programs [10, 11, 19, 20, 21].

Sopcast [19] has a set of root servers, which maintains the information what peer is available for what channel. Sometimes also peer lists are exchanged between the peers. The most important difference of Sopcast is the usage of UDP as transport protocol [20]. This leads to fast packet transmission but also causes a lot of overhead for control. The usage of an external media player and a second buffer are very inefficient and lead to a huge start-up delay.

UUSee provides the videos by several dedicated streaming servers, so that there is no single point of failure and the video streaming quality especially the playback continuity is improved. The TCP protocol is used to communicate with all peers, exchange the buffer map, measure the round trip time (RTT) and estimate the throughput [10]. If a huge number of peers try to join the same channel in the network at short time duration, a noticeable influence on the network performance has been recognized.

PPLive [22] uses different methods to exchange information about the availability of channels or movies, chunks and pieces. A distributed hash table (DHT) is used to assign dedicated movies to dedicated trackers and to achieve a load balancing [23]. On the other side, PPLive tries to improve its playback quality at the expensive of the network architecture. A locality mechanism, which prefers physically near peers (e.g., of the same ISP) is implemented, but peers with high bandwidth are preferred. This may lead to a bad network performance also for other participants.

Most of these works [7, 8, 9, 10, 11, 19, 20, 21] have drawbacks related with low bandwidth utilization, high delay and a single point of failure. Thus to improve the performance of content distribution, the peers can be grouped in clusters. Many peers clustering approaches are proposed as the following:

The hierarchical architecture to group peers into clusters called CBT is proposed in [12]. The CBT has two novel algorithms: a peer joining algorithm and a super-peer selection algorithm. The proximity measurements of the RTT value and the TTL value between a pair of peer and

super-peer are used. The CBT system improves the performance and scalability, and can be used to build a large-scale BitTorrent-like P2P overlay network.

A novel super node overlay based on information exchange called SOBIE is proposed in [13]. The main contributions are to select the super nodes by considering the aggregation of not only the delay and distance, but also the information exchange frequency, exchange time and query similarity. The SOBIE is guaranteed the matching between the physical network and logical network. Moreover, the SOBIE has small-world characteristic to improve the efficiency and robustness.

The super node selection problem for Peer-to-Peer applications is presented in [14]. Three super nodes selection protocols for overlay P2P networks are proposed: SOLE, PoPCorn and H2O. An integrated approach to the super node selection problem built on strong graph theoretic foundations and guided by realistic applications, can benefit the Peer-to-Peer community through cross-fertilization of ideas and sharing of protocols.

An effective real-time Peer-to-Peer streaming system for the mobile environment is proposed in [15]. The peers are grouped into clusters according to their proximity using RTT values between peers as criteria for the cluster selection. The cluster leaders are using to help a service discovery server. The partial streams help to utilizing the upload capacity with finer granularity than just per one original stream. This is beneficial in mobile environments where bandwidth is scarce.

A cluster model for different start video broadcasting is proposed in [1]. The peers are grouped into cluster according to join time of each node and availability of first chunk. The cluster model consists of five processes: peer joining, super node selection, backup-node selection, download paths and leaving node process. The performance of the cluster model will be compared with the one of non-cluster model.

III. NON-CLUSTER SYSTEM DESIGN

This section introduces the concept of non-cluster system architecture and non-cluster system design. When a new node joins and wants to download chunks from the peers in the non-cluster model for different start video broadcasting, the tracker will send the random list of peers to a new node.

A. Non-Cluster-Based System Architecure

The non-cluster system architecture is composed of one server, only one tracker and normal nodes. The server is a node that provides all chunks of the live video stream. The global tracker is known by all nodes and maintains the list of all nodes in the network. The normal nodes are downloader and uploader. When a new node join in the network to used the video streaming, it will contact with tracker. The tracker will reply the random list of peers to a new node. The new node will exchange buffer map with the list of peers and selects peer neighbor to download the video streaming.
An overview of the non-cluster based system model is shown in Figure 1.

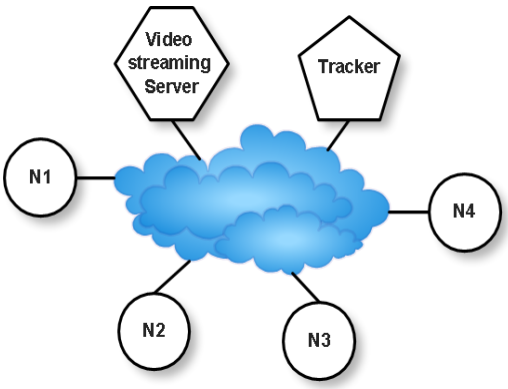

Figure 1. The non-cluster model for different start video broadcasting.

B. Non-Cluster-Based System Design

The method of the non-cluster system design consists of five stages as follow: (i) peer join/leave; (ii) peer exchange information; (iii) peer selection; (vi) buffer organization; (v) segment scheduling.

1) Peer join/leave: When a new peer joins the network and wants to use a video streaming, it sends a request message to the well known tracker server. This tracker maintains the list of all peers, which are currently streaming the same video channel, and the actual playback time of these nodes. After the tracker received the request message, the new peer is added in the peer list of the tracker and a list of nodes that are watching the same video stream at the same time is prepared. If this list is longer than a predefined value, a random subset is generated and sent to the new node. After receiving the list of peers, the new peer will exchange buffer maps with all peers in the list. With several buffer maps from different peers, the new peer can select partner peers and request video segments from them. As for leaving peers, they can leave the network at anytime.

2) Peer exchange information: A peer needs to know the availability of chunks on all of the known peers. This information is exchanged by BitTorrent-like buffer maps (BM) and HAVE or DONTHAVE messages. Since the buffer size of each node can be smaller than the whole video, parts of the video may be deleted. Hence, DONTHAVE messages had to be introduced additionally. They are used to inform neighbored nodes about the deletion of chunks. Initially, a new node creates a connection to all nodes that are reported by the tracker and requests BMs of these nodes. The replied BMs contain information about all chunks that are available on the sending node and are stored on the receiving node. When a node downloaded a new chunk or had to delete a chunk (due to restricted buffer size), it informs all connected nodes by sending a HAVE or a DONTHAVE message. The HAVE and DONTHAVE message should to be used because it may not be the cause of packet loss. Each of these messages contains the number of the new or deleted chunk. The messages are used by all receivers to update the BMs of the sending node.

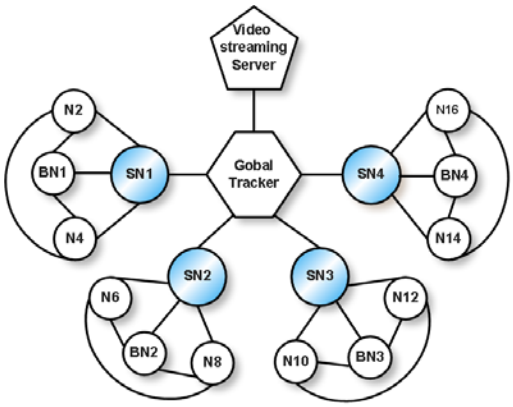

Figure 2. The cluster model for different start video broadcasting.

3) Peer selection: When the new peer knows the buffer maps of other peers, it has to select some of them to download parts of the stream. In opposite to BitTorrent, choking algorithms are not used. We figured out that these concepts leads to disruptions of the video stream. Instead, a node looks for all known peers that have the needed chunk and selects a source randomly.

4) Buffer organization: For the different start video broadcasting, each node has a buffer to store the video chunks. The length of buffer is smaller than video streaming file. The buffer of each node is organized into three parts: data buffer, buffer map and sliding window. The data buffer is used to store video frames. The buffer map is a bit vector representing the information of available segments on a node. Each node exchanges its buffer map with its partners periodically. From buffer map information, the peer will decide which partner nodes are used to fetch required segments. If there is more than one partner having the same segments, the peer node will randomly select the partners or select the partners with minimum delay or maximum bandwidth. Besides buffer map, each node needs to have a sliding window which is used to store a number of displaying segments. From this buffer organization, the video segments will be displayed continuously, and the starting delay of each node will be bounded. In this work, the circular buffer is used as buffer management. The buffer data is divided into three parts: playback buffer, displaying buffer, and future buffer as follows [1, 16, 17, 18].

- The playback buffer (old chunks) is used to buffer data stream for a certain period of time before playing the stream. The number of frames in playback buffer is calculated from the delay called playback delay between the sending and receiving peer. The playback delay between each peer is random since the mesh-based architecture is used. For simplicity, the playback delay is defined as a maximum delay bound in this group of users in this particular network. Thus, every peer will have the same playback delay.

- The displaying buffer (fresh chunks) is used to store data that will be viewed by users. This buffer is designed by using a sliding window. The frame in the beginning for buffer is the displaying frame and the next frame is the next frame in the window will be viewed in the next minutes.

- The future buffer (future chunks) is used to receive new frames. The new frames are received from other peers or partners by using sequential or rarest-first scheduling.

5) Segment scheduling: A peer client has chosen some peers to exchange segments, it is necessary to select these segments. This should be transmitted first and to select one peer client or more peer client as a source for this chunk, based on the knowledge of segment available in all its peers. The important information to calculate the chunk scheduling is the playback time. If a part of a video stream has already been played, the chunks which have to be played next time must be available in the local buffer. Two of the most important concepts are the sequential download and rarest-first download. The sequential download always chooses the chunk which is closest to the playback time. The rarest-first downloads the chunk which is available on the smallest number of peer clients and takes longer time to download. This is usually the newest segment. In this work, sequential download is used to download the segment of each peer. Several threads will be created to fill the different segment from each peer.

IV. CLUSTERING SYSTEM DESIGN

This section introduces the concept of cluster system architecture and cluster-based system design. When a new node joins and wants to download chunks from the peers in the cluster model for different start video broadcasting, the global tracker has to decide which cluster and super node will be joined.

A. Cluster-Based System Architecure

The cluster system is composed of server, global tracker (GT), super node (SN) or local tracker (LT), backup-node (BN) and normal nodes (NN). The server is a node that shares all chunks of a live video stream. The global tracker is known by all nodes and maintains the list of all super nodes. The super node acts as a local tracker keeping the list of all nodes in the cluster. All super nodes are connected with global tracker to synchronize the lists of all nodes in the cluster. The super node, normal node and backup-node are all downloader and uploader. The cluster means that the grouping of node partnerships according to their network proximity. The proximity is measured by using the join time of each node or availability of first chunk. The clustering is used to control the traffic streams within a P2P system and additionally helps to decrease the load of the server and global tracker. Based on the non-cluster model for the different start video broadcasting presented in [1, 16, 17, 18], the behavior and algorithms (peer exchange information, peer selection, buffer organization, segment scheduling) of the nodes are not changed but extended by a logical clustering mechanism. The clustering is realized by the separation of nodes, super nodes, backup-nodes and local trackers.

B. Cluster-Based System Design

The method of cluster system design consist of eight stages as follow: (i) peer join; (ii) super node selection; (iii) backup-node selection; (iv) leaving peer; (v) peer exchange

information; (vi) peer selection; (vii) buffer organization; (viii) segment scheduling.

1) Peer join: When a new node joins, it will contact with the global tracker to ask for the cluster and super node. Therefore, the peer joining algorithm has 2 important phases: connect to global tracker and connect to local tracker. For the first phase, all nodes know the address of the global tracker. When a new node contacts with the global tracker and asks for the first chunk. The global tracker will contact SN of each cluster to search for nodes having the first available chunk. The global tracker then selects the cluster which has the maximum number of nodes containing the first chunk. The new node gets the address of the local tracker (SN) and registers there. For the second phase, the new node contacts with the local tracker. The local tracker returns a random list of neighbor peers in the same cluster to the new node. The new node receives a random list of neighbor peers and sends the message to exchange buffer maps with neighbor peers. The new node selects neighbor peers to download chunks.

2) Super node selection: When the first node joins, the global tracker will set the first node to be a super node and local tracker of the first cluster. The cluster size is limit to C nodes. After the global tracker received joining message from a new node, it will check the cluster size to select appropriate cluster. The global tracker will verify the member size of the selected cluster. If the size of the selected cluster is less than C, the address of SN in that selected cluster will be send to the joining node. If the size of the selected cluster is full (equals C), a new cluster is created. The global tracker will split a node that have first chunk available in the old cluster to be a SN for a new cluster. If there is no cluster in the system, the first cluster is created and the first joining node will be a SN of the first cluster.

3) Backup-node selection: If the size of cluster is full (equals C), the BN will be selected from all normal nodes. The backup-node keeps a list of all peers in the cluster by contact with SN. When the SN leaves from the cluster, a BN will be a new SN. The BN will receive the list of all peers in the cluster from SN. The BN can be selected by three different methods as follow.

- Select the node joining the cluster after the first node. (the second joining node, 2nd)
- Select the node joining the middle of the group. (the $\frac{C^{th}}{2}$ node)
- Select the lasted node that joining the group. (the Cth node)

For the first method, all nodes in the cluster will have an equal chance to be a SN and BN. The drawback of this approach is a frequent SN and BN selection. The second method selects a new SN and BN not often and works well. The third method selects a new super node not often but may cause packet losses. In this paper, the second method is implemented in the simulation.

4) Leaving peer: If a node leaves from cluster, the local tracker will delete it from the list of peers. If the leaving node is a local tracker (SN), the backup-node will be a new local tracker (SN). If the last node leaves the cluster, the cluster is deleted. The local tracker always tells the global tracker about leaving nodes to synchronize the list of SN in the global tracker. The leaving-node process can be divided into three cases: the leaving of SN, BN and NN. For the first case, when the SN is leaving from the cluster, it sends flooding message to all nodes. The all nodes in the cluster will send keep-alive message to their SN. The SN sends keep-alive message to the global tracker. For the second case, the BN exchanges information periodically with the SN. If the BN leaves, it sends the message to inform the super node. For the last case, the NN can leave the network at anytime.

The peer exchange information, peer selection, buffer organization and segment scheduling of cluster model are similar to non-cluster model as described in Section III.

V. MATHEMATICAL MODEL

The mathematical model is used to determine starting delay, buffer size, peer list search and server load. The probability of peer download chunks and peer selection to download are proposed. The efficiency of the different start video broadcasting can be estimate.

A. Starting Delay Estimation

There are four types of delay: startup delay, starting delay, playback delay and delay. The startup delay, denoted T_{stu}, is the time that user supposes to wait until first frame arrives in the buffer. The playback delay, denoted T_{pb}, is the time that the user waited to fill chunk in the buffer until play smooth. The delay, denoted T_{join}, is an initial time according to the server time. Then the starting delay, denoted T_{sti}, is the total waiting time that user supposes to wait until displaying the first frame. The delay will be calculated as following.

The startup delay is depending on the number of neighbors that are used for content discovery and download, and the time used to exchange buffer maps. The startup delay can be considered in two cases, (1) only one available neighbor and (2) more than one neighbor.

1) Startup Delay (T_{stu}): The startup delay is mainly influenced by the network parameters like bandwidth and delay.

a) Only one neighbor

T_{stu} = Transmission delay + Propagation delay + Tracker time + Exchange buffer map time + Peer selection time.

$$T_{stu} = \sum_{i=1}^{n_{link}} \left(\frac{Packet\ \ Size}{Transmission\ \ Rate_i}\right) + \sum_{i=1}^{n_{link}} (Propagation\ Delay_i) \quad (1)$$

$$+\text{Tracker time} + \sum_{j=1}^{L} \sum_{i=1}^{n_{link}} \left(\frac{buffer\ map\ package\ size}{Transmission\ \ Rate_i}\right) + Peer\ selection\ time$$

Note: assume that queuing delay and processing delay is negligible.

n_{link} denoted the number of link from the sender to the receiver.

L denoted the number of peer in the list.

Transmission Rate denoted the number of bits per second.

Tracker time is the time that user contacts with tracker to receive the list of peers.

Exchange buffer map time is the time that the new comer exchanges buffer map with the peers in the received list.

Peer selection time is the waiting time that user is selected a peer to download the first chunk.

b) More than one neighbor

Since, there are several neighbors that there will be several paths. The first frame will receive from the neighbor that has the smallest delay. Thus P_i denoted a path is walk through nodes from a source (selected from the i^{th} neighbor) to a destination.

This case is calculated from the time that is necessary to download the first chunk. It depends on transmission delay and the propagation delay that is necessary for the sender to provide this packet.

$$T_{stu} = Min\left(Delay_{P_1}, Delay_{P_2}, \ldots, Delay_{P_k}\right) = Min\left(Delay_{P_i}\right) : i=1, 2, 3, \ldots, k \quad (2)$$

From Eq. (1), Let's $Delay_{P_i} = T_{stu}$

K is the number of neighbors.

P_i denoted path from the i^{th}neighbor.

2) Starting Delay (T_{sti}):

The starting delay is the total waiting time required to display the first frame. It relies on the join time, the playback time and the startup delay. The starting delay can be calculated as Eq. (3)

$$T_{sti} = Max(T_{join}, T_{pb}) + T_{stu} \quad (3)$$

B. Buffer Size Estimation

The buffer size can be estimated by using the join time or the release time. The unit of buffer size is defined as seconds because the waiting time for displaying video depends on the fill rate of node.

In case of few users, the delay has an impact on the different start video broadcasting viewing process, the peer joining late may not be able to view the whole stream. Then, the arrival condition is proposed. The arrival condition is used as maximum threshold of the delay of each peer. The arrival condition value is equal to Eq. (4).

$$\frac{N}{M} - T_{stu} \leq T_{pb} + T_{release} \quad (4)$$

Note: *N* is the total number of chunks.

M is the total number of nodes.

$T_{stu}$is the waiting time until the first chunk arrives in the buffer.

T_{pb} is the playback time.

$T_{release}$ is the release time.

In case of several users, the delay has no impact on the different start video broadcasting viewing process, the peer can view the whole video streaming.

If a join time is under the arrival condition, the new node can use the different start video broadcasting concept. If a join time is over the arrival condition, the new node will use the live-video streaming.

The buffer size estimation is calculated as following:

1) If the user joins under the arrival condition: The buffer size will be equated as in Eq. (5) and Eq. (6).

2) If the user joins over the arrival condition: The buffer size will be equated as in Eq. (6).

Note: The future time (T_{future}) is the maximum number of future chunks that receiver can receive in a unit of time, as shown in Eq. (7).

The release time, denoted $T_{release}$, is the time to wait until buffer is released.

In this paper, the buffer size is calculated from Eq. (6). The buffer size is constant.

$$Case\ 1 : Buffer\ size = Max\left(T_{join}, T_{pb}\right) + T_{future} \quad (5)$$

$$Case\ 2 : Buffer\ size = T_{release} + T_{pb} + T_{stu} + T_{future} \quad (6)$$

$$T_{future} = \frac{Link\ \ Speed\ \ (bandwidth)}{Fill\ \ Rate} \quad (7)$$

Since, the future time is used to receive new frames. The playback buffer and release buffer are constant. The fill rate usually refers to the number of chunks send to buffer per second.

C. Peer List Search

For the non-cluster model, the tracker uses the sequential search to find the list of peers in the peer list table. With the non-cluster model, the searching time of tracker is *O(M)* where *M* is the total number of nodes as shown in Eq. (8).

For cluster model, the global tracker employs the binary search to seek the proper super node in the peer list table. The local tracker employs the sequential search to seek the list of peers in the peer list table. The global tracker keeps a list of all peers and arrival time of each node. The cluster model reduces the searching time in the global tracker and local tracker. It groups peers into cluster according to joining time (arrival time) of each node. Let server starts broadcast at the time, $T = 0$ and ends at the time, $T = t$. The arrival time of each node will be referenced with the server broadcast time and sorted from minimum to maximum value. Each node will be grouped to each cluster according to its arrival time, as shown in Figure 3. Then, the number of clusters is in order of 2^X. The tracker will check the arrival time of each node and then assign the proper cluster to that particular node. Thus, the number of nodes in each cluster is a random number. Let *Ci* is the number of nodes in each cluster, *M* is the total number of nodes in the system and 2^X is the number of clusters. With this structure, the binary search is used to find the proper super node in GT. The searching time of GT is *O(log* 2^X*)*. The sequential search is

used to find the list of peers in LT. The searching time of LT is *O(C)*. Thus, the total searching time of cluster model is equal to is $O(\log 2^X + C)$ as shown in Eq. (9).

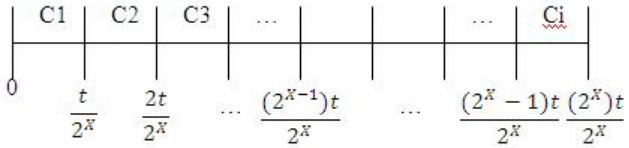

Figure 3. Even time line.

$$Non\text{-}Cluster\ Model = O(M) \quad (8)$$

$$Cluster\ Model = O(\log 2^X + C) \quad (9)$$

D. Server Load

For the non-cluster model for different start video broadcasting, the server load has two scenarios. In the first scenario, all users join at the same time like the server and start to download directly from the server only. In the second scenario, all users join over the arrival condition time, the all users will use the live-video streaming and download from server. From Eq. (10) is worst case of the non-cluster model. The server load is highest. When all peers can not download chunks from peer neighbors, the all peers will use the live-video streaming from server. The worst case of the non-cluster model is occur, when $peer_i$ join over 25 chunks of $peer_{i-1}$. (The value of 25 chunks is calculated from $T_{release} + T_{pb}$, this value is guaranteed that the new node can download available of first chunk). Assume that the video contents have N chunks. N is divisible by 25. The maximum number of peers is $\frac{N}{25}$. The maximum number of chunks that has been sent by server is equal to $\sum(25 + 50 + 75 + \cdots + N)$ or $25\sum(1 + 2 + 3 + \cdots + M)$. The best case of the non-cluster model is shown in Eq. (11). The server supports only one node in the non-cluster model.

For the cluster model for different start video broadcasting, the server load has two cases: worst case and best case. The worst case is the server supports equal to maximum number of super nodes follow from Eq. (12). The best case is the server supports only one super node follow from Eq. (13). Then, the server load can be calculated in Eq. (10), Eq. (11), Eq. (12) and Eq. (13).

Non-clustering with more than one node contacting server (Worst Case):

$$SL_{NC} = \frac{25xM}{2}(M + 1) \quad (10)$$

Non-clustering with only one node contacting server (Best Case):

$$SL_{NC} = 1\ x\ N \quad (11)$$

Clustering with more than one super node contacting server (Worst Case):

$$SL_C = SP\ x\ N \quad (12)$$

Clustering with only one super node contacting server (Best Case):

TABLE I. NOTATIONS AND DEFINITIONS OF DIFFERENT START VIDEO BROADCASTING

Notation	Definition
$\Omega = \{1,2, ...,N\}$	All currently available chunks across the entire network.
$N = \|\|\Omega\|\|$	The number of all currently available chunks.
K	Assume that all peers have exactly K neighbors.
M	The total number of nodes.
S	The total number of peers in the session.
l^{ext}	The buffer length of each node, $l^{ext} \leq N$ $l^{ext} = l^{old} + l^{fresh} + l^{future}$ or $l^{ext} = l_{xi} + l^{future}$
l^{old}	The number of old chunks in the buffer that waiting for overwritten, $l^{old} < l^{ext}$
l^{fresh}	The number of fresh chunks waiting for play in the buffer, $l^{fresh} < l^{ext}$
l^{future}	The number of chunks scheduled to download, they are not in the buffer, $l^{future} < l^{ext}$
l_{xi}	The length of the upload chunks, $l_{xi} = l_{xi}^{old} + l_{xi}^{fresh}$
x_i	The ith peer's conceptual view is downloader.
x_j	The first piece in the jth peer's conceptual view is $x_j + 1$ The jth peer's conceptual view is uploader. Assume x_j is uniformly distributed over [0, N]
$\Omega j = \{x_j + 1, x_j + 2, ..., x_j + l_{xj}\}$	All upload chunks in the jth peer's buffer, $\|\|\Omega j\|\| = l_{xi}$, $\Omega = \cup\ \Omega j$ $1 \leq j \leq S$
$\Omega' j = \{x_j + l_j + 1, x_j + l_j + 2, ..., x_j + l_{xj}^{ext}\}$	The chunks scheduled to download by the jth peer, $\|\|\Omega' j\|\| = l_{xj}^{future}$
η	The efficiency of different start video broadcasting,defined as the probability of a peer having at least one chunk interested to at least one of her neighbors. $\eta = 1 - (1 - \Pr\{\Omega j \cap \Omega' i \neq \emptyset\})^k$

$$SL_C = 1xN \quad (13)$$

Note: SL_{NC} denote the server load of non-cluster.
SL_C denote the server load of cluster.
M is the total number of nodes.
N is the total number of chunks.
SP is the number of super nodes.

E. Probability of peer selection to download

The probability of peer selection to download for the different start video broadcasting, the peer neighbor is important to get needed chunks. The downloader peer has to select the peer neighbors to download chunks. Therefore, a possible metric to evaluate the performance of peer selection for the different start video broadcasting is the Binomial probability distribution of a peer being in the downloading status as shown in Eq. (14) to Eq. (15).

$$Pr\{Selecting\ K\ peers\} = \binom{L}{K} P^K (1-P)^{L-K} \quad (14)$$

P = *Probability of the neighbor having at least one interested chunk*

$$= \sum_{i=1}^{l_{xi}} P(having\ i\ chunks)$$

$$Pr = \binom{L}{K} \left(\frac{l_{xi}(l_{xi}+1)}{2N} \right)^K \left(1 - \frac{l_{xi}(l_{xi}+1)}{2N} \right)^{L-K} \quad (15)$$

Note : L denotes the number of peers in the peer list.
K denotes the number of peer neighbors.
N denotes the total number of chunks.
l_{xi} denotes the length of upload chunks.
P denotes the probability of the neighbor having at least one interesting chunk.

F. Probability of download chunks

The probability of download chunks for the different start video broadcasting is from the concept in [24] that the new peer has to download chunks from the peer neighbors. When a new peer joins in the network, it will contact with the tracker. The tracker replies the list of peers. A new node will exchange buffer map with peers in the list. Then, it select peer and starts to download chunks. Since a new peer is downloading and uploading at the same time. This model will start the analysis by focusing on only two peers in the neighborhood first for peer i (x_i) and peer j (x_j). Peer j is downloading and uploading. Peer i is downloading only. Suppose at a given time t_0, there are totally N chunks available across the entire network, denoted by the set $\Omega = \{1, 2, ..., N\}$. The all chunks is uploading in the j^{th} peer's buffer as $\Omega j = \{x_j + 1, x_j + 2, ..., x_j + l_{xj}\}$, $j = 1, 2, ..., S$, where l_{xj} is the buffer length of upload chunk's peer j, l_{xj} is equal $l_{xj}^{old} + l_{xj}^{fresh}$. Obviously, each peer holds only a subset of Ω, namely, $0 \le x_j \le N - l_{xj}$, $j = 1, 2, ..., S$ and $\Omega = \Omega_1 \cup \Omega_2 \cup ... \cup \Omega_S$, where S is the total number of peers in the session. x_j is independent discrete random variable following a uniform distribution over [0, N].

The chunks are scheduled to download in the j^{th} peer's buffer as $\Omega' j = \{x_j + l_j + 1, x_j + l_j + 2, ..., x_j + l_{xj}^{ext}\}$, where l_{xj}^{ext} is equal l_{xj}^{future}. On the other hand, at *time* = t_0, for the i^{th} peer's buffer, the chunks scheduled to download are $\Omega' i = \{x_i + 1, x_i + 2, ..., x_i + l_{xi}^{ext}\}$. The i^{th} peer is interested in chunk's peer j if and only if $\Omega j \cap \Omega' i \neq \emptyset$. This condition can be simplified to $X_j - l_{xi}^{ext} + 1 \le X_i \le X_j - 1$ as show in Figure 4. The list of all notations and definitions are shown in Table 1.

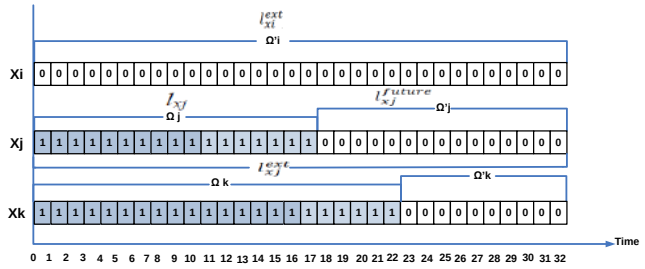

Figure 4. Download and upload chunks of each peer.

In the different start video broadcasting, the share resources are peers' buffer and bandwidth. All participating peers contribute their uploading bandwidth to increase the overall system throughput. Therefore, a possible metric to evaluate the performance of download chunks for different start video broadcasting is the probability of a peer being in the downloading status as shown in Eq. (16).

$$Pr = \{i \text{ is interested in at least one of } j\text{'s pieces}\}$$

$$Pr = \{\Omega j \cap \Omega' i \neq \emptyset\}$$

$$Pr = \{X_j - l_{xi}^{ext} + 1 \le X_i \le X_j - 1\} \quad (16)$$

Then, this equation $Pr\{X_j - l_{xi}^{ext} + 1 \le X_i \le X_j - 1\}$ will proof under the assumption that x_i and x_j are independent discrete random variables following a independent and identically distributed (i.i.d.) as shown in Eq. (17) and Eq. (18).

The probability of download chunks can be divided into two conditions: $N - l_{xi} \ge l_{xi}^{ext}$ and $N - l_{xi} < l_{xi}^{ext}$. The probability of download chunks is as follows (the whole proof will be shown in the appendix):

Condition 1 : when $N - l_{xi} \ge l_{xi}^{ext}$

$$\Pr\left(X_j - l_{xi}^{ext} + 1 \le X_i \le X_j - 1\right)$$

$$= \sum_{X_j=0}^{l_{xi}^{ext}-1} Pr\left(X_j\right) Pr\left(0 \le X_i \le X_j - 1\right)$$

$$+ \sum_{X_j=l_{xi}^{ext}}^{N-l_{xi}} Pr\left(X_j\right) Pr\left(X_j - l_{xi}^{ext} + 1 \le X_i \le X_j - 1\right)$$

$$= \sum_{X_j=1}^{l_{xi}^{ext}-1} \frac{1}{N - l_{xi} + 1} \times \frac{X_j - 1}{N - l_{xi} + 1}$$

$$+ \sum_{X_j=l_{xi}^{ext}}^{N-l_{xi}} \frac{1}{N - l_{xi} + 1} \times \frac{X_j - 1 - (X_j - l_{xi}^{ext} + 1)}{N - l_{xi} + 1}$$

$$= \frac{(l_{xi}^{ext} - 2)\left(2\,(N - l_{xi} + 1) - l_{xi}^{ext} - 1\right)}{2(N - l_{xi} + 1)^2} \quad (17)$$

Condition 2 : when $N - l_{xi} < l_{xi}^{ext}$

$$\Pr\left(X_j - l_{xi}^{ext} + 1 \le X_i \le X_j - 1\right)$$

$$= \sum_{X_j=0}^{N-l_{xi}} Pr\left(X_j\right) Pr\left(0 \le X_i \le X_j - 1\right)$$

$$= \sum_{X_j=1}^{N-l_{xi}} \frac{1}{N-l_{xi}+1} \times \frac{X_j}{N-l_{xi}+1}$$
$$= \frac{N-l_{xi}}{2(N-l_{xi}+1)} \quad (18)$$

Combination two cases together,

$$Pr\ (X_j - l_{xi}^{ext} + 1 \le X_i \le X_j - 1)$$
$$\begin{cases} = \frac{(l_{xi}^{ext}-2)\,(2\,(N-l_{xi}+1)-l_{xi}^{ext}-1)}{2(N-l_{xi}+1)^2} & ; \text{when } N\text{-}l_{xi} \ge l_{xi}^{ext} \\ = \frac{N-l_{xi}}{2(N-l_{xi}+1)} & ; \text{when } N\text{-}l_{xi} < l_{xi}^{ext} \end{cases}$$

G. Efficiency of the system

The system efficiency [24] of the different start video broadcasting can be considered from Eq. (17) and Eq. (18). The Eq. (18) is not selected for different start video broadcasting because the buffer size of each node is bigger than the number of chunks. For the different start video broadcasting, the buffer size of each node is less than the number of chunks. Therefore, the Eq. (17) is selected for different start video broadcasting. The Eq. (17) can be transformed to a more compact form. To calculate efficiency of the different start video broadcasting is shown in Eq. (19).

$$\text{Efficiency } (\eta) = \Pr\{\text{for arbitrary peer } i,\ i \text{ is in downloading status}\}$$
$$= 1\text{- Pr}\{\text{all } k \text{ neighbors of } i \text{ are not interested in } i\text{'s pieces}\}$$
$$= 1\text{- Pr}\{\text{peer } j, \text{ an arbitrary neighbor of } i, \text{ is not interested in } i\text{'s pieces}\}^k$$
$$= 1\text{- (1 - Pr}\{i \text{ is interested in at least one of } j\text{'s pieces}\})^k$$
$$= 1\text{- (1 - Pr}\{\ \Omega j \cap \Omega' i \neq \emptyset\ \})^k$$
$$= 1\text{– (1 – Pr}\{X_j - l_{xi}^{ext} + 1 \le X_i \le X_j - 1\})^k$$

$$\eta = 1\text{– (1– }[\frac{(l_{xi}^{ext}-2)\,(2\,(N-l_{xi}+1)-l_{xi}^{ext}-1)}{2(N-l_{xi}+1)^2}]^k)$$
$$\eta = 1\text{– (1– }[\frac{(l_{xi}^{ext}-2)(2\alpha-l_{xi}^{ext}-1)}{2\alpha^2}]^k) \quad (19)$$

where $\alpha = \frac{N-l_{xi}+1}{l_{xi}^{ext}} > 1$.

VI. Experimental Results

This section describes the experimental setups, the experimental results and evaluates the performance of non-cluster and cluster model for the different start video broadcasting. The discrete event simulator NS-2 [25, 26, 27] is used to create network topology. Network simulation (NS-2) is such an open source simulation tool that operates on the UNIX-based operating systems.

A. Simulation Setup

1) Non-Cluster Model

The experimental setup of the non-cluster model for different start video broadcasting creates one video media server. The server generates a live video streaming. The video stream length is 64 Mbytes, the size of each chunk is 64 Kbytes and the number of chunks is 1024. The video stream bitrate is 512 Kbps. Hence, each chunk represents exactly 1 second of video. The playback buffer size and release buffer size are set to 10 sec and 15 sec, respectively. The number of nodes varies from 50 to 200 nodes and the number of peer neighbors varies from 2 to 16. The delay of each physical network link equals to 2 ms and the bandwidth of each link is set to 2 Mbps. The video play rate is 1 chunk/1 sec. The random joining time of each node depends on the arrival condition from Eq. (4). The random joining time of each node is varied as 18.48, 8.24, 4.83 and 3.12, respectively. The startup delay (T_{stu}) is 3 sec (This value is estimated from simulation). The buffer size of each node equals 32 sec.

2) Cluster Model

The experimental setup of the cluster model for different start video broadcasting creates one video media server. The server generates a live video streaming. The video stream length is 64 Mbytes, the size of each chunk is 64 Kbytes and the number of chunk is 1024 chunks. The video stream bit rate is 512 Kbps. Hence, each chunk represents exactly 1 second of video. The playback buffer size and release buffer size are set to 10 sec and 15 sec, respectively. The number of nodes equals to 200 nodes and the number of neighbors varies from 2 to 16. The number of clusters is varied as 2 (100 nodes), 4 (50 nodes), 8 (25 nodes) and 16 (13 Nodes), respectively. The delay of each physical network link equals to 2 ms and the bandwidth of each link is set to 2 Mbps. The video play rate is 1 chunk/1 sec. The random joining time of each node depends on the arrival condition from Eq. (4). The random joining time of each node equals to 2.12. The startup delay (T_{stu}) is 3 sec (This value is estimated from simulation). The buffer size of each node equals 32 sec.

B. Simulation Results

1) Non-Cluster Model

The non-cluster model is implemented to evaluate the performance of different start video broadcasting. The performance is considered from varies the number of nodes and the number of peer neighbors. The performance metrics of the non-cluster model for different start video broadcasting is follows:

a) Server load

The result of chunks that are downloaded directly from the server is shown in Figure 5. The chunks download from server is plotted for varies number of nodes and number of neighbors. The number of chunks that are downloaded from the server represents the server load. The x-axis is the number of nodes and the y-axis is the number of chunks downloaded from the server. The result shows that the number of nodes and the number of peer neighbors have impact on sever load. If the number of nodes and the number of neighbors are increased, the server load is reduced.

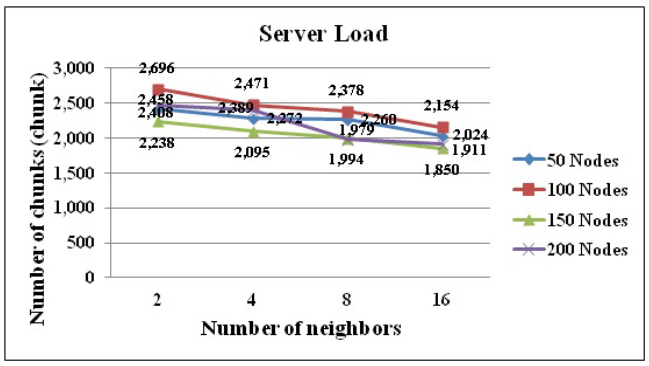

Figure 5. The server load of non-cluster model for different start video broadcating.

b) Peer load

The result of the average number of chunks downloaded from one peer for non-cluster model is shown in Figure 6. The chunks downloaded from one peer is plotted for varies the number of nodes and the number of peer neighbors. The number of chunks that are downloaded from a peer represents the peer load. The x-axis is the number of nodes and the y-axis represents the number of chunks downloaded from one peer. The result shows that the number of nodes has impact on the peer load but the number of peer neighbors has no impact on the peer load. If the number of nodes is increased, the peer load is increased. The peer load is better performance because one peer serves chunks less than 1,024 chunks. (The peer load of system is increased follow the number of nodes is increased)

c) The number of control message

The control messages are used for communication between nodes. TCP is used as transport protocol for control messages. UDP is used for the exchange of video chunks. The result of control message for non-cluster model is shown in Figure 7. The number of TCP control messages of each network used to exchange information between nodes. The x-axis represents the number of nodes and the y-axis represents the number of control messages between all nodes. The result shows that the number of nodes and the number of neighbors have impact on the number of control messages. If the number of nodes and the number of neighbors are increased, the number of control messages grows accordingly.

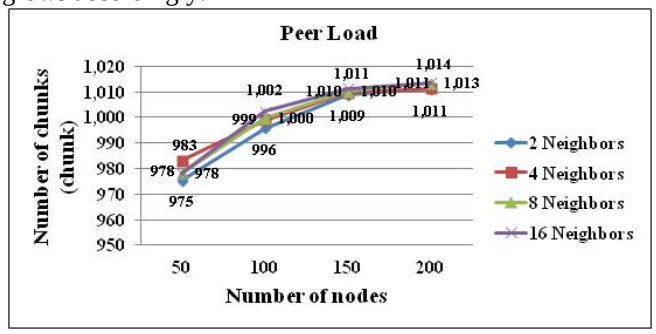

Figure 6. The peer load of non-cluster model for different start video broadcating.

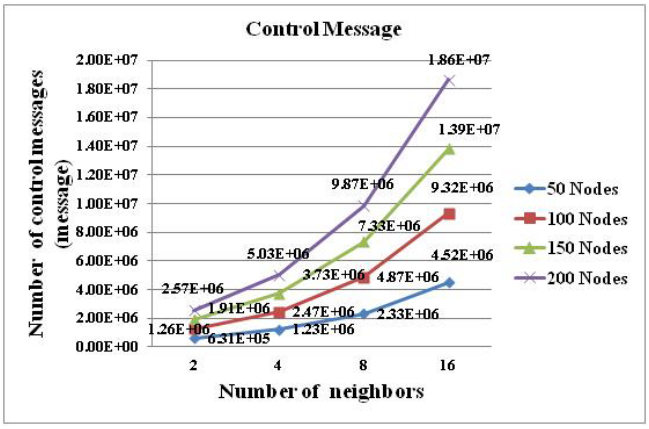

Figure 7. The control message of non-cluster model for different start video broadcating.

2) Cluster Model

The cluster model is implemented to evaluate the performance of different start video broadcasting. The performance is considered from varies the number of clusters and the number of peer neighbors. The performance metrics of the cluster model for different start video broadcasting is follows:

a) Server load

The number of chunks downloaded directly from the server in a cluster model is shown in Figure 8. The chunks downloaded from a server are plotted for various the numbers of clusters and the number of neighbors. The chunks downloaded from server have effect for server load. The x-axis is the number of clusters and the y-axis is the number of chunks downloaded from server. The result shows that the number of clusters and the number of neighbors have impact on sever load. If the number of clusters and the number of neighbors are increased, the server load decreased equal to constant. It means that server serves only one peer. The other peers can download chunks from neighbor peers in the cluster.

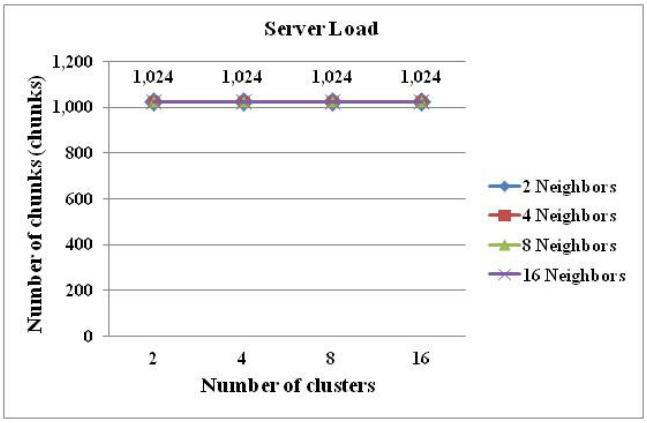

Figure 8. The server load of cluster model for different start video broadcating.

b) Peer load

The result of the number of chunks downloaded from one peer for non-cluster model is shown in Figure 9, the chunks downloaded from one peer is plotted for varies the number of clusters and the number of peer neighbors. The number of chunks that are downloaded from a peer represents the peer load. The x-axis is the number of clusters and the y-axis represents the number of chunks downloaded from one peer. The result shows that the number of clusters and the number of peer neighbors have impact on the load of one peer. If the number of clusters and the number of peer neighbors are increased, the peer load decreased equal to constant. The peer load is better performance because one peer serves chunks less than 1,024 chunks. (The peer load of system is increased follow the number of nodes is increased)

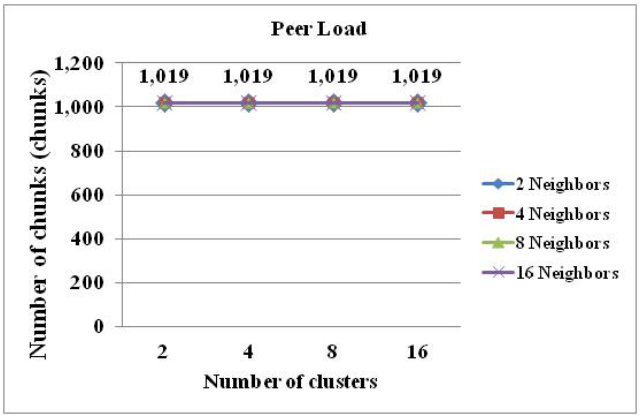

Figure 9. The peer load of cluster model for different start video broadcating.

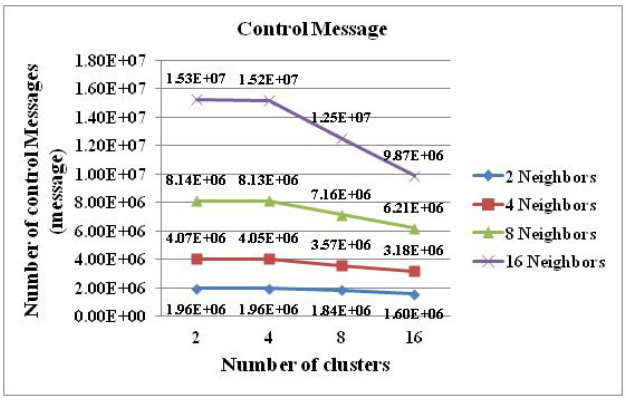

Figure 10. The control message of cluster model for different start video broadcating.

c) The number of control message

The control messages are used to communication between nodes. TCP is used as transport protocol for control messages. UDP is used for the exchange of video chunks. The result of control message for cluster model is shown in Figure 10. The number of TCP control messages of each network used to exchange information between nodes. The x-axis represents the number of clusters and the y-axis represents the number of control messages. The result shows that the number of clusters and the number of neighbors have impact with the control message. If the number of clusters is increased, the control message is decreased. If the number of neighbors is increased, the control message is increased.

VII. CONCLUSION AND FUTURE WORK

This paper proposed the non-cluster model and cluster model for different start broadcasting to improve the service of P2P video streaming. The different start video broadcasting used both characteristics of the live video streaming and the video-on-demand, created on an application layer Mesh network and the BitTorrent concept. The system design of non-cluster model and cluster model for different start video broadcasting is proposed. The non-cluster model for different start video broadcasting consists of five stages: (i) peer join/leave; (ii) peer exchange information; (iii) peer selection; (vi) buffer organization; (v) segment scheduling. The cluster model for different start video broadcasting consists of eight stages: (i) peer join; (ii) super node selection; (iii) backup-node selection; (iv) peer exchange information; (v) peer selection; (vi) buffer organization; (vii) segment scheduling; and (viii) leaving peer. The system model is evaluated using the mathematical model such as starting delay, buffer size, peer list search, server load, probability of peer selection to download, probability of download chunks and system efficiency. The performance of the non-cluster model and the cluster model for different start video broadcasting are compared. As a result, the performance of the cluster model is better than non-cluster model. The cluster model can improve the utilization of available bandwidths for upload and download, reduces the server load, reduces peer load and reduces the number of control messages. The unpunctual users can view the video stream from the beginning during server broadcast time. Furthermore, the tracker traffic, dynamic node joins and leaving nodes have to be implemented. The backup-node selection method has to be reconsidered. The different start video broadcasting should be investigated on mobile network environment.

REFERENCES

[1] H. Ketmaneechairat, P. Oothongsap and A. Mingkhwan. "Peer Clustering System for Different Start Video Broadcasting." Proceeding of The sixth International Conference on Digital Telecommunications (ICDT 2011) : 116-122.

[2] Napster [online], available at http://www.napster.com [Accessed: Aug. 10, 2010]

[3] Gnutella [online], available at http://www.gnutella.com [Accessed: Aug. 10, 2010]

[4] Kazaa [online], available at http://www.kazaa.com [Accessed: Aug. 10, 2010]

[5] BitTorrent [online], available at http://www.bittorrent.com [Accessed: Aug. 10, 2010]

[6] eDonkey[online], available at http://en.wikipedia.org/wiki/EDonkey 2000 [Accessed: Aug. 10, 2010]

[7] X. Zhang, et al. "CoolStreaming/DONet: A data-driven Overlay Network for Efficient Live media streaming." IEEE Computer and Communications Societies. (2005) : 2102-2111.

[8] X. Su and L. Chang. "A Measurement Study of PPStream." IEEE Communications and Networking in China. (2008)

[9] S. Tang, et al. "Topology dynamics in a P2PTV network." Proceeding of the 8th International IFIP-TC 6 Networking Conference. (2009) : 326-337.

[10] C. WU, B. Li and S. Zhao. "Exploring Large-Scale Peer-to-Peer Live Streaming Topologie." ACM Transaction on Multimedia Computing, Communications and Applications. (2008) : 1-23.

[11] Y. Huang, et al. "Challenges, Design and Analysis of a Large-scale P2P-VoD System." Proceeding of the ACM SIGCOMM Conference on Data communication. (2008) : 375-388.

[12] J. Yu and M. Li. "CBT: A Proximity-Aware Peer Clustering System in Large-Scale BitTorrent-like Peer-to-Peer Networks." Journal of Computer Communication's Special issue on Foundation of Peer-to-Peer Computing. (2008) : 591-602.

[13] Z. Chen, et al. "SOBIE: A novel super-node P2P overlay based on information exchange." Journal of Computers. (2009) : 853-861.

[14] V. Lo, et al. "Scalable supernode selection in peer-to-peer overlay networks." Proceeding of the 2nd International Workshop on Hot Topics in Peer-to-Peer Systems. (2005) : 18-25.

[15] J. Peltotalo, et al. "An RTSP-based Mobile P2P Streaming System." Journal of Digital Multimedia Broadcasting. (2010) : 1-15.

[16] H. Ketmaneechairat, P. Oothongsap and A. Mingkhwan. "Smart Buffer Management for Different Start Video Broadcasting." Proceeding of the 2nd International Conference on Interaction Sciences : Information Technology, Culture and Human. (2009) : 615-619.

[17] H. Ketmaneechairat, P. Oothongsap and A. Mingkhwan. "Buffer Size Estimation for Different Start Video Broadcasting." Electrical Engineering/Electronics Computer Telecommunications and Information Technology ECTI-CON'10. (2010) : 924-928.

[18] H. Ketmaneechairat, P. Oothongsap and A. Mingkhwan. "Communication Size and Load Performance for Different Start Video Broadcasting." Proceeding of the PGNET'10. (2010).

[19] A. Sentinelli, et al. "Will IPTV ride the peer-to-peer stream?." Proceeding of the IEEE Communications Magazine. (2007) : 86-92.

[20] B. Fallica, et al. "On the Quality of Experience of SopCast." Proceeding of the Next Generation Mobile Applications, Services and Technologies. (2008) : 501-506.

[21] X. Hei, et al. "Insights into PPLive: A measurement study of a large-scale P2P IPTV system." Proceeding of the IPTV Workshop, International World Wide Web Conference. (2006).

[22] C. Wu, B. Li and S. Zhao. "Multi-channel Live P2P Streaming: Refocusing on Servers." Proceeding of the IEEE INFOCOM'08. (2008) : 1355-1363.

[23] L. Vu, et al. "Mapping the PPLive network: Studying the impacts of media streaming on P2P overlays." Proceeding of the UIUC Technical Report. (2006).

[24] H. Liu and G. Riley. "How Efficient Peer-to-Peer Video Streaming Could Be?," Proceeding of the Consumer Communications and Networking Conference. (2009) : 1-5.

[25] The Network Simulator (NS-2). [serial online] 2006. [cited 2007 Aug 17]. Available from : URL http://www.isi.edu/nsam/ns/

[26] T. Issariyakul and E. Hossain. "Introduction to Network Simulator NS2." Journal of Springer. (2009) : 1-438.

[27] K. Eger, et al. "Efficient Simulation of Large-Scale P2P Networks: Packet-level vs. Flow-level Simulations." Proceeding of the 2nd Workshop on the Use of P2P, GRID and Agents for the Development of Content Networks. (2007).

APPENDIX

The probability of peer selection to download is proof as follows:

$$Pr\{\text{Selecting K peers}\} = \binom{L}{K} P^K (1-P)^{L-K}$$

P = *Probability of the neighbor having at least one interested chunk*

$$= \sum_{i=1}^{l_{xi}} P(\text{having } i \text{ chunks})$$

$$= \sum_{i=1}^{l_{xi}} \frac{i}{N}$$

$$= \frac{1}{N} + \frac{2}{N} + \frac{3}{N} + \ldots + \frac{l_{xi}}{N}$$

$$= \frac{l_{xi}(l_{xi}+1)}{2N}$$

$$Pr = \binom{L}{K} \left(\frac{l_{xi}(l_{xi}+1)}{2N}\right)^K \left(1 - \frac{l_{xi}(l_{xi}+1)}{2N}\right)^{L-K}$$

The probability of download chunks is proof as follows:

Condition 1 : when $N-l_{xi} \geq l_{xi}^{ext}$

$$\Pr\left(X_j - l_{xi}^{ext} + 1 \leq X_i \leq X_j - 1\right)$$

$$= \sum_{X_j=0}^{l_{xi}^{ext}-1} Pr\left(X_j\right) Pr\left(0 \leq X_i \leq X_j - 1\right)$$

$$+ \sum_{X_j=l_{xi}^{ext}}^{N-l_{xi}} Pr\left(X_j\right) Pr\left(X_j - l_{xi}^{ext} + 1 \leq X_i \leq X_j - 1\right)$$

$$= \sum_{X_j=1}^{l_{xi}^{ext}-1} \frac{1}{N-l_{xi}+1} \times \frac{X_j - 1}{N-l_{xi}+1}$$

$$+ \sum_{X_j=l_{xi}^{ext}}^{N-l_{xi}} \frac{1}{N-l_{xi}+1} \times \frac{X_j - 1 - (X_j - l_{xi}^{ext} + 1)}{N-l_{xi}+1}$$

$$= \frac{1}{(N-l_{xi}+1)^2} \times \left[\sum_{X_j=1}^{l_{xi}^{ext}-1} X_j - \sum_{X_j=1}^{l_{xi}^{ext}-1} 1\right]$$

$$+ \frac{1}{(N-l_{xi}+1)^2} \sum_{X_j=l_{xi}^{ext}}^{N-l_{xi}} (l_{xi}^{ext} - 2)$$

$$= \frac{1}{(N-l_{xi}+1)^2} \times \left[\frac{(l_{xi}^{ext}-1)(l_{xi}^{ext}-1+1)}{2} - \frac{2(l_{xi}^{ext}-1-1+1)}{2}\right]$$

$$+ \frac{l_{xi}^{ext}-2}{(N-l_{xi}+1)^2} \times \sum_{X_j=l_{xi}^{ext}}^{N-l_{xi}} 1$$

$$= \frac{1}{(N-l_{xi}+1)^2} \times \left[\frac{(l_{xi}^{ext}-1)(l_{xi}^{ext}-2)}{2}\right]$$

$$+ \frac{l_{xi}^{ext}-2}{(N-l_{xi}+1)^2} \times \left[\frac{2(N-l_{xi}-l_{xi}^{ext}+1)}{2}\right]$$

$$= \left[\frac{(l_{xi}^{ext}-1)(l_i^{ext}-2)}{2(N-l_{xi}+1)^2}\right] + \frac{(l_{xi}^{ext}-2) \times (2N - 2l_{xi} - 2l_{xi}^{ext} + 2)}{2(N-l_{xi}+1)^2}$$

$$= \frac{(l_{xi}^{ext}-2) \times (l_{xi}^{ext} - 1 + 2N - 2l_{xi} - 2l_{xi}^{ext} + 2)}{2(N-l_{xi}+1)^2}$$

$$= \frac{(l_{xi}^{ext}-2)\,(2\,(N - l_{xi} + 1) - l_{xi}^{ext} - 1)}{2(N-l_{xi}+1)^2}$$

Condition 2 : when $N-l_{xi} < l_{xi}^{ext}$

$$\Pr\left(X_j - l_{xi}^{ext} + 1 \leq X_i \leq X_j - 1\right)$$

$$= \sum_{X_j=0}^{N-l_{xi}} Pr\left(X_j\right) Pr\left(0 \leq X_i \leq X_j - 1\right)$$

$$= \sum_{X_j=1}^{N-l_{xi}} \frac{1}{N-l_{xi}+1} \times \frac{X_j}{N-l_{xi}+1}$$

$$= \frac{1}{(N-l_{xi}+1)^2} \sum_{X_j=1}^{N-l_{xi}} X_j$$

$$= \frac{1}{(N-l_{xi}+1)^2} \times \frac{(N-l_{xi})(N-l_{xi}+1)}{2}$$

$$= \frac{N-l_{xi}}{2(N-l_{xi}+1)}$$