

HEAD-TAIL VIDEO STREAMING OVER PEER TO PEER SYSTEMS

Ali Sianati[†], and Maghsoud Abbaspour[†], and Maryam Norouzi[†]

[†] Faculty of Electrical and Computer Engineering, Shahid Beheshti University G.C.,
Tehran, Iran.

asianati@yahoo.com, maghsoud@sbu.ac.ir, m.norouzi@mail.sbu.ac.ir

ABSTRACT

P2P systems similar to file sharing applications are being used vastly due to unrestricted nature of these systems. Their unrestricted comes from their ability to cooperate and aggregate peer's resources and their scalability. On the other side, today technologies are traditional client-server applications. These applications can perform strongly but they are not scalable due to limitations on server resources. This limitation of the client/server technology leads designers to use alternative technologies mainly P2P. As a streaming system, P2P streaming network can be formed into two types, Tree-based and Mesh-based. In this paper a new mesh-based P2P system named Head-Tail streaming is proposed. Head-Tail simplifies packet scheduling and node failure recovery by using paired-peer sending and node failure prediction. Our system outperforms ordinary systems comparing as delay and receive time. Our system performs better than ordinary systems based on two reasons :Overlapping sequence of chunks and node replacement policy.

KEYWORDS

Peer-to-Peer Video Streaming, P2P Packet Scheduling.

1. INTRODUCTION

Although today servers are very powerful to process Internet requests, they are so vulnerable to flash crowds similar to sport events. There are several methods to increase the capacity of the servers. One of them is to add more and powerful links to the servers or add more server devices to the server farm, but the more devices, takes the more costs. Another technique is to distribute load from the original servers to the edges of the Internet similar to Caches [1] [2].

In recent years, P2P systems are used as alternative to the current client/server approaches [3]. P2P gained visibility with Napster's support for music sharing on the Web and today it is increasingly becoming an important technique in various areas, such as distributed and collaborative computing systems.

P2P systems and applications employ distributed resources to perform a function in a decentralized manner. The resources encompass computing power, data (storage and content) and network bandwidth [3]. Being distributed and cooperative makes such systems powerful means for many purposes similar to file sharing and video streaming.

File sharing and video streaming systems are different in many aspects. File sharing systems provide files to large number of user. Downloading files in these systems are not time dependent. Hence, they can be downloaded in any order and be used later. But in streaming

case, data must be generated in real-time by source. In addition, data must be received by receivers at constant rate, and chunks must arrive almost in sequence order so that they can be played with short delay [4].

Video streaming system over P2P networks can be formed in to two structures: Tree-based and Mesh-based [5]. Tree-based streaming systems are suitable for one-to-many streaming or what is so-called live streaming in which video playbacks on all users are synchronized. Tree-based systems can be used in multicast applications where IP Multicast is not applicable. Mesh-based streaming systems are mostly used for on demand video in which playbacks of same video clip on different users are not synchronized [6].

Constructing best tree, dealing with heterogeneous bandwidth, free riders, and node failure are some drawbacks of tree-based systems. Node failure is the worst drawback in these systems because in this case failed node affects all child nodes. Beside these drawbacks, the main advantage in this design is that multiple consecutive packets are pushed down the tree along the same paths, resulting unpredictable traffic flows and low control traffic [7].

Backup nodes are mostly used in these systems to recover from node failures [8] [9] [10] [11]. Some structures similar to SplitStream [12], split frames into layers and stream each one from different branch. Each node is part of all stripes and node failure causes just one stripe to be lost. Other structures similar to ZigZag [13] make use of both above methods to reinforce the tree. Combining tree-based methods with mesh-based method similar to [14] can empower tree-based systems. In such a system instead of one streaming source, several sources try to stream the video, or child nodes can choose other parents in the case of low quality.

Mesh-based systems mainly suffer from scheduling. Packet scheduling must be accurate enough to prevent packet duplicate and late arrival of packets. In comparison with tree-based systems, construction and maintaining mesh-based systems are simpler and offer good resilience to node failures. In this paper a novel mesh-based streaming system called Head-Tail is presented. This system uses an efficient and yet simple methods to schedule packets between senders. Hence recovering from node failure becomes more efficient.

The rest of this paper is organized as follows. In Section 3, we introduce the background as well as related work. Section 4 describes the Head-Tail streaming system while section 5 provides key issues in designing this system. Simulation results are presented in Section 6. Finally, Section 7 concludes the paper.

2. RELATED WORKS

As mentioned in section 2, four factors reduce the performance of tree-based systems: constructing best tree, dealing with heterogeneous bandwidth, free riders, and node failure. Nodes can leave or arrive at any time, therefore constructing the best tree is challenging. Furthermore the maximum received quality is constrained by nodes with lower bandwidths.

In spite of the well performance of tree based systems in reducing network traffic, they are really hard to implement. Moreover, these systems are proper for live situation. Hence, during normal times these systems cannot be used. In such situations mesh-based system can play a great role. In tree-based systems receiver have to choose senders from a set of leaf nodes. But in mesh-based systems senders are chosen from the whole set of senders. Hence Receivers in these systems choose their senders more efficiently. They can control their session freely without any interference to other receivers. In the following some of mesh-based systems are presented.

[5] and [15] present numerical scheduling methods for mesh-based systems. In their methods, delay and bandwidth are estimated by numerical formulas and use the results as inputs to their scheduling method.

In Gridmedia [16] a push-pull method is presented. At the beginning of the streaming, the receiver is in the pull state. In pull state receiver requests all packets and monitors all senders. After some steady period, the receiver enters the push state. In this state senders cooperatively send required packets to the receiver. Using push state in this system reduces control packets sent to senders. The biggest challenge in this system is scheduling senders in the push state.

LSONet [17] tries to find the best peers by considering the physical mesh constructed between peers. Using Gossip based protocol; this system can monitor the membership of each peer. Buffering techniques utilize request processing either. Constructing and maintaining an efficient overlay network is the key challenge in LSONet.

Main purpose in MeshCast [18] is to avoid congestion. In MeshCast, articulation nodes are used as points to redirect the congested flow to another path in the network. MeshCast can perform well in situations that the congested link is not located at the neither receiver nor sender side.

In [19], during download phase, each receiver becomes a new sender to new receivers. This property is used as a mean to supply system with more senders. This is similar to BitTorrent system where each receiver is a seed to other nodes [20].

3. HEAD-TAIL

One major component of mesh-based systems is scheduling. The scheduler must be accurate enough to prevent duplicate or delayed data. Almost all of the mesh-based systems mentioned before employ a similar scheduling method. Such a scheduler schedules chunks (pieces) of layers in a sequential order. For example it assigns chunk 1 and 2 to sender 1 and chunk 3 to sender 2 and so on. These kinds of schedulers behave complex mainly when node failure occurs.

If sender 1 failed and left the network, sender 2 or sender 3 would give up packets which were being sent by them to support sender 1. Of course scheduler should precisely consider it's preferable that other senders give up the packets or to continue sending own packets. Therefore schedulers of such systems variously would make a mistake in scheduling. By losing scheduling, this mistake would be solved; however useful bandwidth would be increased too.

This task becomes more difficult when the newly added peer to the sender list cannot meet the requirements of the receiver. For example the new sender has a lower upload bandwidth compared to failed node.

Since current video codec used in the Internet are layered encoded, they are sensitive to packet loss. For example LC is a stack-based codec in which completeness of one layer is constrained to completeness of all lower layers [21] [22]. In such a condition, decisions made by scheduler are so critical because a mistake in scheduling can lead to one or more layer losses.

The proposed method called Head-Tail, schedules chunks simply while it can recover temporarily from node failures. Head-Tail is based on the fact that each bit stream has two ends. Having two ends helps scheduler to schedule chunks from two points in the data stream.

This is similar to the operation of Bubble and Quick sort. In these two sorting methods, operation is done from beginning and the end of the array which results to better performance of these sort algorithms .

Head-Tail method pairs each two suitable peers and assigns each end of the layer to one of them. Peers start to send chunks from each end until they reach to a rendezvous point. Since bandwidths of peers are heterogeneous and pairing them results to unequal total bandwidth, rendezvous point may vary from pair to pair. Pairing helps scheduler to choose the best pair based on the aggregated bandwidth. The operation of Head-Tail is simply demonstrated in Figure1.

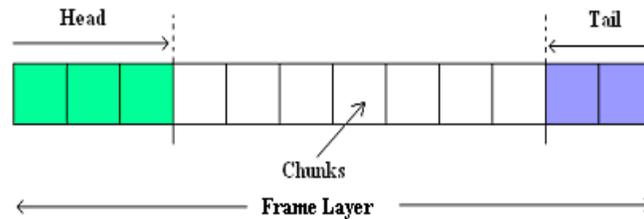


Figure 1. Simple Operation of Head-Tail

The first advantage of Head-Tail method is its simplicity. As expressed in Figure 1, after choosing and pairing peers, one peer of each pair is chosen as Head and the other one as Tail. Head is mostly the peer with more bandwidth and more stability. Head sends chunks from the beginning of the layer and the Tail from the end of the layer. This operation continues until they reach to a same chunk in the layer. In this point scheduler may start next layer or the same layer in the next frame using the pair.

Dealing smoothly with node failures is another advantage of this method. One node failure (in the best case) affects flow of one end. Scheduler without any effort may continue the process of receiving chunks from the other end until a substitution for the failed node is found. Or in the case of low bandwidth of the other peer scheduler may choose another peer for the pair. In the worst case scheduler may drop the layer.

4. HEAD-TAIL STREAMING SYSTEM ARCHITECTURE

General schema, components of Head-Tail system and their relationships have been depicted in Figure 2. There are some points in architecture of the system which should be attended: (a) Video codec, (b) methods of Accessing to video information, (c) Streaming architecture and protocol, (d) Node replacement policies, (e) Scheduling

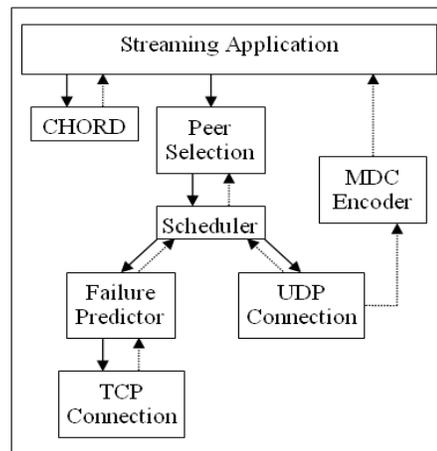


Figure 2. Head-Tail system architecture

4.1 Video Coded

All of video formats which are used in the internet are layer base not to interrupt sending packet if the bandwidth of sender and receiver are not enough. In such a codec, layers would be sent according to the bandwidth between two nodes. MDC codec acts better than LC in networks with high failing rate. In MDC, contrary to LC, if any problem in receiving one layer occurred, it would not affect higher layers. Moreover this video format needs high bandwidth.

Due to high rate of node failing in P2P networks, MDC decreases the video quality. Because a problem in one layer does not affect on any other layers, MDC is used for Head-Tail system. It is supposed that in the architecture in every frame, size of all layers is equal.

4.2 Methods of Accessing to video information

CHORD is used as a method of maintaining and searching information in Head-Tail system. Three features that made us to use Chord instead of many other peer-to-peer lookup protocols were its simplicity, provable correctness, and provable performance even in the face of concurrent node arrivals and departures. Constructing its overlay network is easy and its defects are removable. On the other hand it can be used to obtain node stability and find Free Riders because CHORD has periodical messages to find failings [23].

The CHORD maintains the pieces of information of nodes, but in the proposed system estimated available bandwidth of nodes also is considered. Therefore available bandwidth of each node was added to hash table of Chord network. This change decreases the number of exchanged messages between nodes in order to find suitable nodes.

Similar to other video distributing systems, hardware resources cannot be used for special purposes in our system and current algorithms of internet should be observed. But one of these resources, i.e. the allocation of output bandwidth of nodes, can be controlled by our system. Output bandwidth can be allocated to each node; on the other hand bandwidth is saved for a specific receiver. Updated available bandwidth is stored in CHORD network. Using stored data in CHORD network, each node can estimate its bandwidth via suggested methods in [24] and [25].

4.3 Streaming architecture

After joining the network, receiver may request peers which have a certain file from the network. On receiving peers list, Head-Tail chooses a set of peers based on its download bandwidth and the aggregated bandwidth of senders. In our implemented method Head-Tail chooses peers based on the best fit and keeps other nodes as backup. Best nodes are chosen based on their available bandwidth, packet loss and availability. Hence, nodes that meet best criteria are chosen to be the main senders and the others are kept in a list as backup nodes in the case of node failure.

4.4 Streaming protocol

Each network application has its own protocol to communicate with other nodes in the network. In Head-Tail TCP/IP has been used as underlying network protocol and build our own protocol above this layer.

Head-Tail uses out-of-bound signaling to control the entire streaming session. One connection is used for sending control messages while the other one is to transfer video data. Control messages are sent by TCP and data packets are sent using UDP. TCP helps to send reliable messages while UDP helps to have more control on the flow of video data. The control messages are small enough not to delay or congest and to be hold in one TCP message.

UDP suffers from unreliable transfer of data. Hence we added a simple ARQ method to our protocol in order to requesting lost packets from senders. In our method scheduler estimates RTT of each connection based on the history of that connection and requests probable lost packets from the sender. RTT is calculated based on Jacobson/Karels algorithm [26].

Control protocol's connections are two-way handshake. At first, receiver's willing to make a connection will be announced to sender by a control message, and then sender will send an ACK message. After receiving this message by receiver, it opens UDP port.

4.5 Node replacement policy

Node leaving is based on two types: aware and unaware. In aware leaving sender informs receiver that it's leaving the network. Receiver does not let sender to leave unless the current receiving layer is completely downloaded. During transmission of the current layer scheduler has enough time to choose a new peer and replace the leaving one.

Unaware leaving refers to node failure. Node failure may occur by link failure or closing the streaming application without notification. In this case, last requested chunk is delayed or lost. Based on the estimated RTT of the connection, scheduler requests last chunk again while the second peer is still sending its assigned chunks. If the last chunk is received in the next RTT, scheduler discovers a delay. If the chunk is not received during next RTT, scheduler discovers a node failure and replaces the failed peer with another. During finding new peer and replacing failed node, scheduler makes no effort to recover the lost chunks because the second peer is sending the lost chunks from the other end of the layer. One technique to improve the performance of this remedy is to choose replacing peer and make necessary connection to it during retransmission of lost chunk.

Hence in the case of node failure, replacing peer takes part immediately. By using this remedy, initializing connection and preparing replacing peer is overlapped with the discovery of the node failure.

Failure of the replacing peer is a major issue in node replacement technique. If the replacing peer fails, the system experiences a double failure, one for the first failed node and another for the second failed node. Hence more time is lost. Each time a delay occurs, the system predicts a possible failure. Then it tries to find a suitable replacing peer. If replacing node fails then the system has enough time to find another peer due to the long failure discovery phase delay.

4.6 Scheduling

Since packet lost or retransmitting information concurrently occurs, scheduling in mesh based distributed systems is the most important part in designing. It means since information is fragmented and distributed between nodes, there is no mapping between information and nodes.

Due to special design of Head-Tail System, scheduling is easy and make the system self supporter. In this system, scheduler assigns information in packets from head and tail of layer. Therefore lose propagation is zero and repetitive receiving or transferring propagation is about zero. High performance of this system is because of none overlapping between assigned chunks to senders. The nodes always try to reach each other, not to overtake each other.

It is probable to send a chunk twice only at the point of node meeting the same chunk which happens rarely. Delay of first chunk has been sent with one node causes this problem, furthermore the second node retransmit the chunk.

One of the most important differences between scheduling in a Head-Tail system and other systems is the way of packet retransmitting. Traditional systems use a one way one node method for transmitting. In the Head-Tail System though, we have a two way transmitting system in which the nodes are moving toward each other until they meet at a meeting point (rendezvous point). Hence if delay happens in one of the nodes there's no need to change the scheduling and there's no need to consider the other nodes' packet delay.

5. SIMULATION RESULTS

We implemented our proposed streaming system in our new network simulator NS.Net. Our scenario is similar to what is presented in [27]. But in our scenario we used low bandwidth nodes mostly with bandwidths similar to dialup connections. In our scenario a receiver is placed behind a router. Senders may join the network from the LAN side or from WAN side. Local router is connected to WAN through a gateway. For the case of simplicity, WAN side nodes are connected directly to this gateway. Figure 3 illustrates our simulation setup.

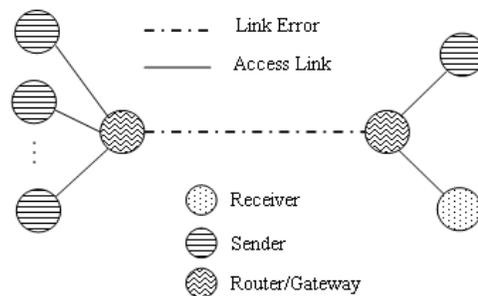


Figure 3. Simulation Setup

At first, system searches nodes containing a specific video. After finding nodes, scheduler sorts them according to their available bandwidth, then selects best of them and prepares each 2

nodes to become a sender pair. After preparation, communications in order of strength pair are established, and then the pair starts to send the video. Each receiver send a control message when it receives a packet, in this way sender will send next packet.

When a chunk is received, scheduler checks status of the other sender of pair to control if whole layer has been received or not. If pair senders have not met each other yet, scheduler assigns next chunk of layer to one of the senders. If the entire layer has been received, scheduler keeps first sender waiting until receiving the last chunk of the layer.

A simple system has been implemented to compare with Head-Tail system. In this ordinary system each sender is responsible to send one layer completely. Therefore the problem of sender meeting will not happen, so its implementation would be easy. For more coordination between this system and Head-Tail system, bandwidth of each ordinary system's node is equal to sum of pair nodes' bandwidth in Head-Tail.

Whereas the Head-Tail system depends on control messages, there is no mechanism for error detection in sender. Therefore sender failing would be detected late. In order to solve this problem RTT, which is calculated with Jacobson/Karels [26], is used to predict arrival time of the packets. In this method, RTT is calculated by equations (1) to (4).

$$Difference = SampleRTT - EstimatedRTT \quad (1)$$

$$EstimatedRTT = EstimatedRTT + (d * Difference) \quad (2)$$

$$Deviation = Deviation + d * (|Difference| - Deviation) \quad (3)$$

$$TimeOut = u * EstimatedRTT + q * Deviation \quad (4)$$

Parameter d is a random decimal number between zero and one. U and q are considered constant equal to one and four respectively. Changing these parameters causes increasing or decreasing System ability of error detection.

After receiving each packet, RTT value is calculated to be used for next packet. If the next packet would not be received after this delay, the packet will be requested.

If the scheduler keeps faster sender waiting for slower sender at the rendezvous point, time of faster sender would be wasted. We changed scheduler so that faster sender does not wait for the other sender. Thus the scheduler permits faster sender to send a new layer immediately. Consequently sending information improved and layers were completed sooner.

This change in scheduler makes another problem and that's bad scheduling when the slower sender fails, but faster sender has started to send new layer. In this situation, instead of finding new peer in order to send lost chunk, Head-Tail acts like other systems and obligates faster sender to send lost chunk and stop sending new layer. Of course this helps scheduler to find replacing node and add it to the network, during receiving lost chunk.

A dynamic network has been simulated with more than 100 nodes and 10 routers in average. Routers randomly placed and nodes randomly connected to routers. Speed and error rate of link were chosen randomly. Links between nodes and their immediate routers are without error.

Figure 4 depicts Min, Max and Average hop count of different packets in 100 run. Of course Min hop count is always one because in our random network, a router always is between two nodes which are at the same network. The figure completely states that packets in different runs had different delay which shows network dynamism in case of failing or error rates.

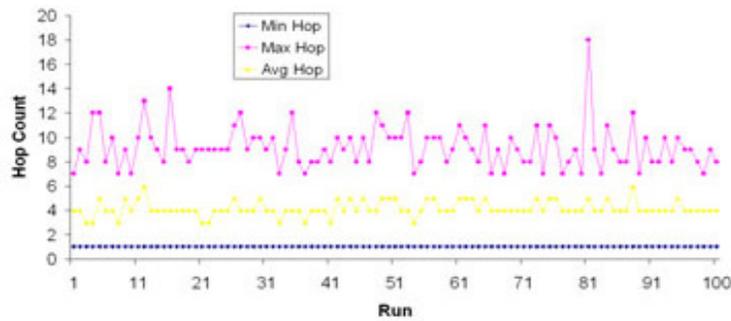


Figure 4. Packet's hop count in random network

Figure 5 depicts Max and Average packet loss rate in links between routers. We considered it between zero and %30. Different loss rates make different situations for receiver. To simulate network congestion, packets are dropped loses without any announcement to sender. Congestion percentage has been shown in Figure 6.

Figure 7 depicts Max, Min and Average delay of packet receiving on receiver side. In some runs, packet delay is so high because packet is lost or traversed a long path to the receiver. Nearly Min and Average delay overlapped because the number of packet with high delay is little.

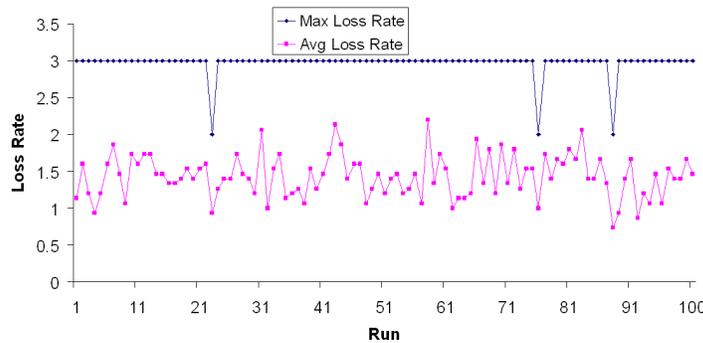


Figure 5. Packet loss rate in links between routers

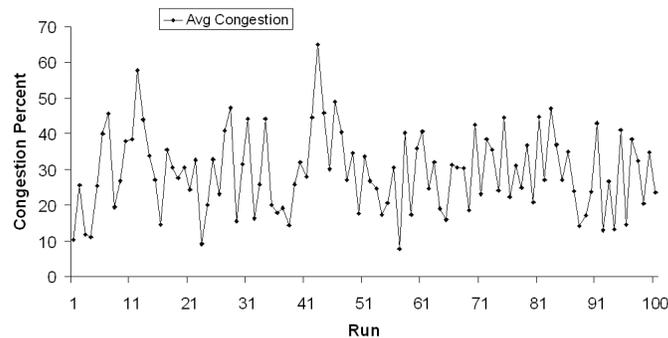


Figure 6. Packet loss rate in network

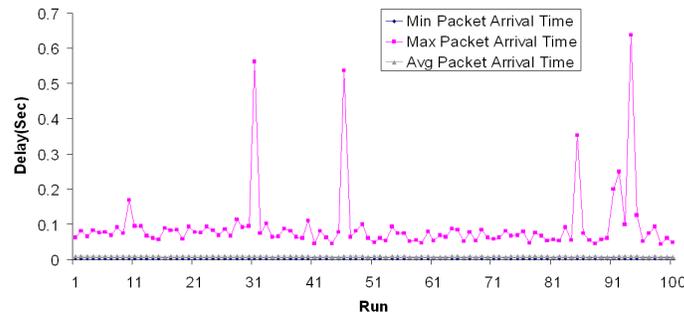


Figure 7. Packet delay on receiver side

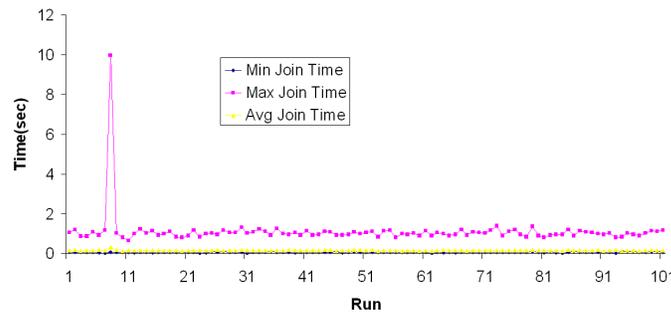


Figure 8. Join time to CHORD network

After constructing the network, CHORD overlay network must be constructed. Nodes randomly enter to this network. By entering each node to the network, CHORD's structure is updated. Figure 8 depicts Max, Min and Average join time for connecting to CHORD network. As it is obvious in the figure, in one run the time was about 10 seconds due to CHORD's network destruction and replacement requesting. Although it is possible to happen, but it hardly ever occurs.

Total node cooperation in distributed systems is shown in Figure 9. Node cooperation means the percent of video files are shared by nodes. In our simulation, nodes containing a special video files are chosen to distribute.

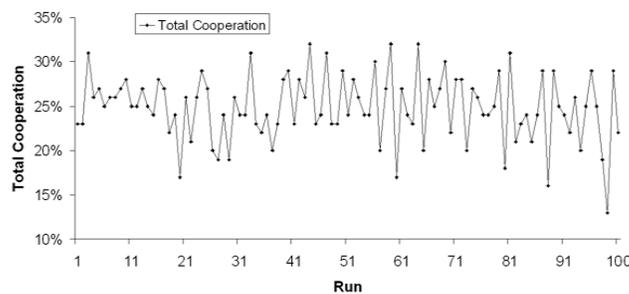


Figure 9. Node cooperation in distributed system

Figure 10 depicts propagation delay between requesting and receiving frame in Head-Tail and ordinary systems. At the simulation it was tried to have equal efficiency in both systems when there is no error or failing. Nevertheless in stable situations Head-Tail acts also better than

ordinary one. At the simulation, nodes randomly leave the network and join it again. Leaving the network without teardown message does not make any time waste therefore there is no effect in the Figure 9.

There are two sharp slopes in Figure 9. At the first one, one of the sender pair has been failed and at the second one, both of senders at one pair have been failed (double failure). These events are happened in a random time and it is not obvious when these failings exactly happened but average frame delays were depicted as 2 events in Figure 10.

At the first slope, when one sender fails consequently delay increases and performance decreases very much, nevertheless self supporter structure of Head-Tail helps to have less delay in comparison with ordinary system. Because the other sender in the pair continues sending frames and does not change the scheduling. This does not happen in other ordinary systems and node failings are not predictable.

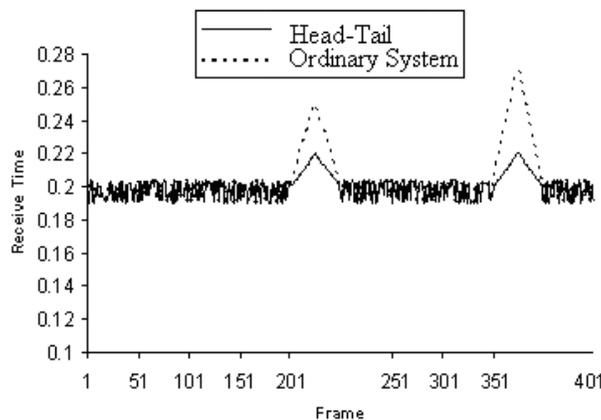


Figure10. Receive time (propagation delay between requesting and receiving frame)

At the second slope which relates to double failure, delay in Head-Tail system is very less than ordinary systems because it has predicted it. In most runs delay at single failing and double failure are the same, but when the prediction is not correct, delay increases. In spite of this, generally delay in Head-Tail is less than ordinary systems.

Our system performs better than ordinary systems based on two reasons :Overlapping sequence of chunks and node replacement policy .

Overlapping sequence of chunks is useful when one peer has finished sending its assigned chunks while the pairing peer has not. In this situation first peer can start sending chunks of the same layer in the next frame. By using this method a little buffering is needed for the system .

Discovering failed node and preparing replacing node can be done simultaneously . Almost all of the today streaming systems wait until the failure is discovered. Then they try to find the best replacement for the failed node. But our system predicts the failure. Every time a delay occurs, our system tries to find the best replacement peer and initiates necessary connections. During this phase, system may request the delayed chunk again. One major advantage of this prediction is that finding and initiating connections is overlapped with the discovery phase. In some cases replacing node may be left the network and is not online anymore . Hence system faces double failure. Since our system finds replacing node during the discovering phase, in

case of node failure, system can find another node without losing extra time. Because discovering phase takes lots of time.

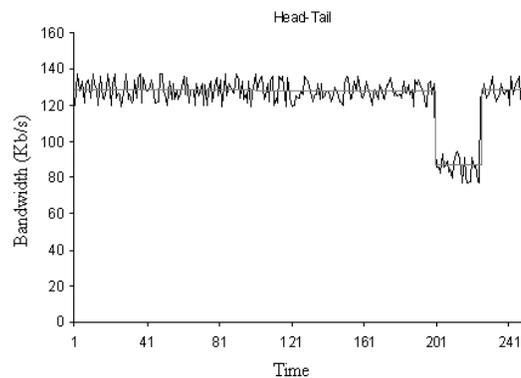


Figure 11. Bandwidth fluctuation in one run

Figure 11 depicts the bandwidth fluctuation of our system during a normal run. When there is no node to replace the failing node, bandwidth decreases and scheduler has to cancel current layer and put the other node as backup node. In this situation, bandwidth of receiving and consequently quality of video will greatly decrease. This situation is unpredictable in all systems.

In our simulated runs, nodes join and leave randomly. We forced system to have a double failure in 201second. In this point no suitable peer is found for failed node, hence system drops one description. After some time system finds new peers. Then it selects two appropriate peers and adds another description to the streaming session.

We implemented congestion in our system by using link error. Losing packets due to congestion is similar to losing them in link error but no feedback is supported in link errors.

6. CONCLUSION

P2P streaming is a powerful tool to remove load on the streaming servers. Lots of issues must be considered during designing of such systems, because such systems have restricted and weak resources. Head-Tail streaming system tries to overlap times that are wasted in ordinary systems, hence improving the overall performance of the system. Predicting can help streaming system not to face double failure during failure discovery phase.

Our system in the phase of the design suffers from two issues: real congestion and current playout time. Congestion is simulated by using error links in our NS.Net simulator. Current playout time may help our system to abandon delayed frames and chunks, hence wasting less bandwidth.

References

- [1] J. Wang, "A survey of web caching schemes for the Internet," *ACM Computer Communication Review*, vol. 29, no. 5, 1999.
- [2] A. Bestavros, and C. Cunha, "Server initiated document dissemination for the WWW," *IEEE Data Engineering Bulletin*, vol. 19, pp. 3-11, 1996.

- [3] D.S. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu, "Peer-to-Peer computing," *HP Laboratories*, 2003.
- [4] R.J. Lobb, A. Paula, C. Silva, E. Leonardi, M. Mellia, and M. Meo, "Adaptive Overlay Topology for Mesh-Based P2P-TV Systems," in *Proc. of the 18th international workshop on Network and operating systems support for digital audio and video*, New York, 2009.
- [5] N. Magharei, and R. Rejaie, "Understanding mesh based Peer-to-Peer streaming," in *NOSSDAV*, 2006.
- [6] Y. Liu, Y. Guo, and C. Liang, "A survey on peer-to-peer video streaming systems," *Peer-to-Peer Networking and Applications*, vol. 1, no. 1, pp. 18-28, 2008.
- [7] F. Picconi, and L. Massoulie, "Is There a Future for Mesh-Based live Video Streaming?," in *Proc. of the 8th International Conference on P2P Computing*, Germany, pp. 289-298, 2008.
- [8] T.T. Do, K.A. Hua, and M.A. Tantaoui, "P2VOD: Providing fault tolerant Video on Demand Streaming in P2P Environment," In *proc. of the IEEE International Conference on Communication (ICC)*, pp. 1467-1472, 2004.
- [9] G. An, D. Gui-guang, D. Qiong-hai, and L. Chuangl, "BulkTree: an overlay network architecture for live media streaming," *Journal of Zhejiang University SCIENCE A*, vol. 7, pp. 125-130, 2006.
- [10] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller, "Construction of an efficient overlay multicast infrastructure for real time application," In *Proc. of IEEE INFOCOM*, pp. 1521-1531, 2003.
- [11] S. Banerjee, S. Lee, R. Braud, B. Bhattacharjee, and A. Srinivasan, "SRMS: Scalable Resilient Media Streaming," In *Proc. of NOSSDAV*, pp. 4-9, 2004.
- [12] M. Castro, P. Druschel, A. M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: high bandwidth content distribution in a cooperative environment" In *Proc. of IPTPS*, 2003.
- [13] D.A. Tran, K.A. Hua, and T. Do, "Zigzag: an efficient P2P scheme for media streaming," In *Proc. of IEEE Infocom*, 2003.
- [14] S. Birrer, D. Lu, F. E. Bustamante, Y. Qiao, and P. Dinda, "FatNemo: building a resilient multi-Source multicast fat-tree".
- [15] T.P. Nguyen, and A. Zakhor, "Distributed video streaming over Internet," *SPIE conference on multimedia computing and networking*, San Jose, California, pp. 189-195, January 2002.
- [16] L. Zhaot, J.G. Luo, M. Zhang, W.J. Fu, J. Luo, Y.F. Zhang, and S.Q. Yang, "Gridmedia: a practical P2P based live video streaming system".
- [17] H. Guo, K.T. Lo, and C.T. Cheng, "LSONet: overlay networks construction for multi layered live media streaming," *8th IEEE International Symposium on Multimedia (ISM)*, 2006.
- [18] Y. Ma, and R.S. Aygün, "The methodology of mesh-cast streaming in P2P networks," *7th IEEE International Symposium on Multimedia (ISM)*, 2005.
- [19] J.B. Kwon, and H.Y. Yeom, "Distributed multimedia streaming over Peer-to-Peer networks," *9th International Conference on Parallel and Distributed Computing*, 2003.
- [20] B. Cohen, "Incentives build robustness in bitTorrent," 2003.
- [21] M. Ghanbari, "Two-layer coding of video signals for VBR networks," *IEEE Journal on Selected Areas in Communications*, vol. 7, pp. 771-781, June 1989.
- [22] V. K. Goyal, "Multiple description coding: compression meets the network," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 74-94, September 2001.

- [23] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, "Chord: a scalable Peer-to-Peer lookup service for Internet applications," *In proc. Of SIGCOMM*, California, USA, August 2001.
- [24] T. Anjali, and C. Scoglio, "TEMB: Tool for End-to-End Measurement of available Bandwidth," *In Proc. of IEEE ELMAR*, 2003.
- [25] R. Prasad, and C. Dovrolis, "Bandwidth estimation: metrics, measurement techniques, and tools," *IEEE Network*, vol. 17, pp. 27- 35, 2003.
- [26] V. Jacobson, and M.J. Karels, "Congestion avoidance and control," *ACM SIGCOMM Computer Communication Review*, vol.18, no. 4, 1988.
- [27] R. Rejaie, and A. Ortega, "PALS: Peer-to-Peer Adaptive Layered Streaming," *In Proc. Of NOSSDAV*, 2003.

Authors

Ali Sianati received his BSEE degree from Shahid Bahonar University of Kerman, Iran in 2005. He is currently under graduation MSc student of Shahid Beheshti university of Tehran, Iran. He has developed several academic applications similar a new network simulator named NS.Net. He is a member of IEEE. His research interests include networking, P2P networks and c# developing.



Maghsoud Abbaspour received his B.S, M.S and Ph.D degree from University of Tehran in 1992, 1996 and 2003 respectively. He joined Computer Engineering department, Shahid Behesht University in 2005. He is interested in wireless sensor networks, multimedia over wireless sensor networks, adhoc networks, and multimedia over peer to peer systems.



Maryam Norouzi received the B.S. degree in computer engineering from Azad University (Central Branch), Iran, in 2007 and she completed her master in computer architecture. She received her M.S degree in Shahid Beheshti University, Iran, in 2011. She had studied on Image compression on wireless camera sensor networks and different routing protocols especially real-time and reliable or secure routing protocols on wireless sensor networks. Different traffic modeling especially multimedia streaming modeling and Peer-to-Peer communications are another research subjects studying too.

