Topology Optimization in Hybrid Tree/Mesh-based Peer-to-Peer Streaming System

Tran Thi Thu Ha[1], Jinsul Kim[1], Jaehyung Park[1] Sunghyun Yoon[2], Ho-Yong Ryu[2]

[1]School of Electronics & Computer Engineering, Chonnam National University, Republic of Korea
[2]Smart Network Research Department, Electronics and Telecommunications Research Institute, Republic of Korea
thuhabkhn@gmail.com, jsworld@jnu.ac.kr, hyeoung@jnu.ac.kr, shy72@etri.re.kr, hyryu@etri.re.kr

Abstract: Peer-to-Peer (P2P) video streaming is the fastest growing application of the Internet. One of the main challenge is to provide a high quality of service through the dynamic behavior of the network because a peer may join or leave anytime. Currently, P2P streaming network exist two types of users: streaming users - who use mobile devices with 3G/4G connection expect to watch the live video immediately and storage users - who use PC with wired Internet will download and then watch the video later. We realized that the streaming users may stop watching live video after a while if they find the video is out of their interest. Users leaving causes dynamic and affect the data delivery. On the other hand, the storage users that are downloading the video do not have the concern of interest and playback quality, until they start to watch the video. Hence, the storage users are relatively more stable than streaming users. This paper, we investigate the strategies on the topology construction and maintenance of P2P streaming systems with storage users are closer to the broadcaster than streaming users. And also we apply our idea on hybrid push-pull protocol that combines the benefits of pull and push mechanisms for live video delivery to provide better video streaming quality

Keywords: peer–to–peer, live video streaming, file sharing, chunk, tree-based, mesh-based

1. Introduction

Streaming is a method for intelligent broadcasting of data on the network, it differs from conventional multimedia services because it isn't necessary to wait for the end of downloading video and able to start playing back. Current approaches in P2P video streaming can be classified as tree-based, mesh-based or hybrid. With tree-based model uses a push method to transfer data. This model has low start-up delay. However, there are two main problems in this method: if the bandwidth of parent node is low, children nodes will be lose data and when parent node failure, other nodes can't receive data until completing the recovery of the tree. On the other hand, mesh-based model uses a pull method to request necessary data from a number of neighbor nodes. However, mesh-based model requires large buffers to support pull data from neighbors and there is an adjustment between minimum delay by sending pull request and overhead of whole system. So, both models have their own strengths and weaknesses. This paper proposes a new architecture system design for P2P live video streaming that combines the advantages of pull and push methods for broadcasting live video. This consists of two states: tree-based and mesh-based. Also, we design network topology with storage nodes are adjacent to the broadcaster, because they are more stable than streaming nodes which can leave system anytime. The remainder of this paper is organized as follows. We briefly discuss the related work in Section 2. The formulation for topology optimization in tree/mesh-based P2P streaming systems, simulation setting and numerical results are presented in Section 3. Finally, the paper is concluded in Section 4.

2. Related Work

In a tree or multi-tree, a video stream is pushed along defined routes, from parents to their children. Tree topologies use pushing method as show in Figure 1 (left side) by choosing which child node to push the selected chunk based on the tree structure. Tree-based provides low latency. In a mesh, multiple and dynamic neighbors may have video data available to send, a node has to pull data to avoid significant redundancies. Mesh topologies use pull-based scheduling as show in Figure 1 (right side). Choosing which video frame to be request based on own buffer map, choose which peer to send chunks request based on the neighbor's buffer map, resend request to partners for necessary chunks. A mesh-based system is therefore more potent, but longer delays and higher control overhead. Compared with mesh-based systems, the tree-based systems have well-organized overlay structures and typically distribute video by actively pushing data from a peer to its children peers. However, maintaining the tree overlay with node churns is a difficult task.
In hybrid pull-push scheduling is proposed to enable push in a mesh-based system to lower the system overhead.
We make a survey about the overlay of streaming video systems and comparison these systems with three metrics: throughput, low latency, scalability shown in Table 1. Overcast [1] constructs and maintains a high bandwidth distribution tree from a source to multiple nodes. So, it utilizes a tree algorithm to maximize the bandwidth from the root towards each node. SplitStream [2] proposes using multiple trees to distribute parts of the whole video stream to balance load across different nodes and provides better scalability compared to a single tree system. SplitStream doesn't provide additional means to recover from node failure.
CoolStreaming [3] is a mesh-based overlay. The video stream is segmented into chunks and is broadcasted to neighbors. Comparing to a simple tree-based overlay, observe that playback continuity is much better. Hence, some mixed strategies have also been proposed to exploit the advantages of both schemes. Prime [4] presents topology design for mesh-based live streaming. Prime is designed to work with MDC to minimize content bottlenecks and

bandwidth bottlenecks. LiveSky [5] uses a hybrid approach, where the stream is split into different chunks which are then distributed in a sub-trees. Missing chunks are received by pull-based mesh overlay. ToMo [6] combines low overhead of tree-based system with the stability of a mesh-based system. The source node divides the video stream into several sub-trees, with each chunk being sent to multiple sub-trees. ToMo also designs the mesh-based structure for stable nodes close to the source, and the tree-based structure for dynamic nodes far away from the source.

Table 1. Comparison of P2P live video streaming systems

System	Method	Throughput	Low latency	Scalability
Overcast (2000)	Push	+	-	-
SplitStream (2003)	Push	0	+	+
CoolStreaming (2005)	Pull	0	0	+
Prime (2009)	Hybrid	0	+	+
LiveSky (2009)	Push/ Pull	+	+	0
ToMo (2010)	Push	+	+	-

3. Topology Optimization and Implementation

3.1 Topology Optimization

We design the storage nodes are nearby to the broadcaster. We call the storage users are storage nodes and streaming users are streaming nodes for design new network topology [7]. Figure 2 shows the organization of two types of nodes (storage node and streaming node) in the overlay. For this case, we can split overlay tree into three sub-trees. User are organized in separate sub-trees, except the broadcaster. That means a node belongs to only one sub-tree. Each node also has links to other nodes of other sub-trees. Each node maintains two kinds of connections: connection belonging to a sub-tree (push link) and connections of nodes in different sub-trees (pull link-see in Figure 4).

Broadcaster sends live video to each sub-tree using push mechanism. Live video is divided into many chunks. Each node will receive from its parent at least one chunk in the pushing phase. Then, node pulls other chunks from other nodes in same level to get completed video and improve the quality of service [8]. Consider that a local network consisting of 7 users: Broadcaster captures live video and then broadcasting this video on P2P network. We assume that this video is split into three chunks c_1, c_2, and c_3. Broadcaster pushes three chunks to users 1, 2, and 3 with respectively. After finishing receiving each chunk, then users 1, 2, or 3 send requests to each other for remain chunks of video using pull requests. For example, user 1 receives chunk c1 from broadcaster and then it sends pull request to user 2 and 3 to get chunk c_2 and c_3 of video. Now, user 1 is watching a chunk c_1 which receives from broadcaster and also can broadcast chunk c_1 for users 2, 3 if they request. At the same time, user 2 is also getting chunk c_2, c_3 from users 2, 3 and continues watching whole live video. We can extend this scenarios with many users, building tree overlay with a number of chunks. In each sub-tree, push method is used like the case broadcaster pushes data to users 1, 2 and 3. Besides that, in each level, pull method is used like the case user 1 pull data from user 2 and 3. The processes shown in Figure 3, once a peer joins, the application creates the overlay and starts streaming. If the node leaves, the application is stopped. When a node joins in P2P network and connects to the tracker. The tracker replies with information on the chosen sub-stream and the initial neighborhood for the peer.

Based on the information, the peer selects the level depend on the configuration of the peer: streaming user (using mobile phone with 3G/4G connection) or storage user (using PC with wired connection). After selecting the level and sub-stream, the peer starts to download matching chunks into the buffer. If a chunk is not available in the buffer at the time it is supposed to be played, the user wait until the chunk has been successfully downloaded. This process continues until the peer leaves. This helps in preventing undesired effects, such as dynamic behavior of the network, other peers can pull missing data from another peers in same level instead of pulling data from peer which dropt out.

As shown in Figure 4, at the pushing phase, broadcaster pushes three chunks to nodes at level 1. These nodes continue to push these parts to other child nodes in their sub-trees. As the result, through the pushing phase, nodes have minimum data required to display. At the pulling phase, a node will send pull request to other node with same level to receive remain parts of video. The user requires pulling requests of other users which belong to other sub-trees. If this user didn’t respond pulling requests of other users, it will inform other users about its rejection. These users must look up other users that can send data to them. If it doesn’t send data to any user in a sub-tree, it may not able to pull data from any user in other sub-tree. So, streaming data is distributed to every node in the overlay network by both pushing and pulling methods

3.2 Simulation Setting and Implementation

The tree length can be too high if there are a large number of nodes. We expect a topology, in which both the length and the number of nodes in each level grow linearly.

If level 1 has 2^k nodes, then the number of nodes at level i has 2^{k-i}.

Thus, total number of nodes from level 1 to level i is:

$$N_i = \prod_{j=1}^{i} 2^{k-j+1} \quad (1)$$

Also, the total number of nodes in the network can be calculated as:

$$N = \sum_{i=1}^{k} N_i \quad (2)$$

Then, from the number of users join to P2P network, we can calculate both the number of levels and number of node in

each level for design topology structure. For the example gives in Table 2 with 100 peers.
As (1), number of peers of each layer following:

$$N_1 = \prod_{j=1}^{1} 2^{3-j+1} = 8$$

$$N_2 = \prod_{j=1}^{2} 2^{3-j+1} = 8 \times 4 = 32$$

$$N_3 = \prod_{j=1}^{3} 2^{3-j+1} = 8 \times 4 \times 2 = 64$$

As (2), the total number of nodes in the network:

$$N = \sum_{i=1}^{3} N_i = N_1 + N_2 + N_3 = 104 > 100$$

So, number of node in level 1 is 8 (k=3), number of levels is 3 (i=3).

Table 2. Parameters used in the simulation setup

Name	Parameter	Description
Simulation Length	3600s	Length of the simulated scenario
Number of Peers	100	Total number of peers
Storage Peers	20	Number of storage users
Streaming Peers	80	Number of streaming users
Chunk size	256 Kb	Size of each chunk
Video Files	2.048 Mb	Number of video files
Topology level	3	Number of levels of overlay

Number of chunks which will be pushed into each sub-stream given by equation:

$$Number\_of\_Chunks = \frac{Files \times File\_size}{Chunk\_size} \quad (3)$$

From parameter given in table 2, we create 8 chunks (2.048Mb/256Kb) as (3), and push each chunk into 8 sub-stream (because from above calculation, the number of nodes in level 1 is 8 nodes, it means existing 8 sub-trees for our topology). The application is written in Java and runs on the Java Platform for investigating peer-to-peer content distribution with implementation requirement following our topology design as shown in Figure 5. With using new topology, processing time for receive complete video is much faster than traditional transfer methods using pure mesh-based or pure tree-based.
In evaluation, we use the typical metrics: startup delay, packet loss to compare our solution with Overcast [1] (pure tree-based) and CoolStreaming [3] (pure mesh-based) overlays.

Startup delay time: For the beginning, the mesh-based solution (CoolStreaming) performs worst, because it needs a longer time to search and request for neighbor peers. The Overcast performance took only 20 seconds to startup. Even though, our solution need to aware of the coexistence of the two types of nodes and explicitly prioritizes the service to the streaming nodes but at the beginning our topology constructs as tree overlay network. So, nodes need shorter time to startup than the CoolStreaming and almost similar with Overcast startup time.

Packet loss rate: CoolStreaming with the mesh-based solution performs the best, because it is pull-based and thus is resilient to the node dynamics. On the other hand, Overcast with the pure tree-based solution performs the worst, because the tree overlay is interrupted to suffer from the node dynamics. Our solution combines pull-based and push-based, it takes advantage of both solutions, so it performs much better than the pure tree-based solution and little less than mesh-based solution. By using hybrid of overlay structure and combine design storage nodes are close broadcaster, our solution can achieve the performance of the mesh-based solution.

4. Conclusions

This paper, we recognized existence of the two types of users for video streaming - there are wired Internet users and wireless mobile users. Specifically, Internet users have better network connection, and thus they should be considered to be located close to the broadcaster. Moreover, since mobile users are usually charged for data usage, such users are not always suitable for forwarding data. These requirements call for suitable design of the architecture as shown in this paper with new topology for P2P network with more stable, minimize packet loss and provide better video streaming quality compared to the pure mesh-based, pure tree-based networks.

5. Acknowledgment

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (NRF-2013R1A1A2013740) and also supported by Chonnam National University, 2012.

References

[1] John Jannotti, David K. Gifford, Kirk L. Johnson, M. Frans Kaashoek, and James W. O'Toole, "Overcast: Reliable Multicasting with an Overlay Network," In Proceedings of the ACM Symposium on Operating System Design and Implementation, vol.4, pp.14–29, 2000.
[2] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, and Atul Singh, "SplitStream: High-Bandwidth Multicast in Cooperative Environments," In Proceedings of the ACM Symposium on Operating Systems Principles, pp.298–313, 2003.
[3] Xinyan Zhang, Jiangchuan Liu, Bo Li, and Tak-Shing Peter Yum, "CoolStreaming/DONet: a Data-Driven Overlay Network for Peer-to-Peer Live Media Streaming," In Proceedings of the Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), vol.3, pp.2102–2111, 2005.
[4] Nazanin Magharei and Reza Rejaie, "PRIME: Peer-to-Peer Receiver-driven Mesh-based Streaming," IEEE/ACM Transactions on Networking, vol.17, no.4, pp.1052–1065, 2009.
[5] Hao Yin, Xuening Liu, Tongyu Zhan, Vyas Sekar, Feng Qiu, Chuang Lin, Hui Zhang, and Bo Li, "Design and Deployment of a Hybrid CDN-P2P System for Live Video Streaming: Experiences with LiveSky," In Proceedings of the ACM International Conference on Multimedia, pp. 25–34, 2009

[6] Suphakit Awiphan, Zhou Su, and Jiro Katto, “ToMo: A Two-Layer Mesh/Tree Structure for Live Streaming in P2P Overlay Network,” In Proceedings of the IEEE Consumer Communications and Networking Conference, pages 1–5, 2010.

[7] Tran Thi Thu Ha, Yonggwan Won, and Jinsul Kim, “Topology and Architecture Design for Peer to Peer Video Live Streaming System on Mobile Broadcasting Social Media,” International Conference on Information Science and Applications, pp. 1-4, 2014

[8] M. Sanna, and E. Izquierdo, “Live Scalable Video Streaming on Peer-to-Peer Overlays with Network Coding,” IEEE Latin America Transactions, vol. 11, no.3, pp. 962–968, 2013.

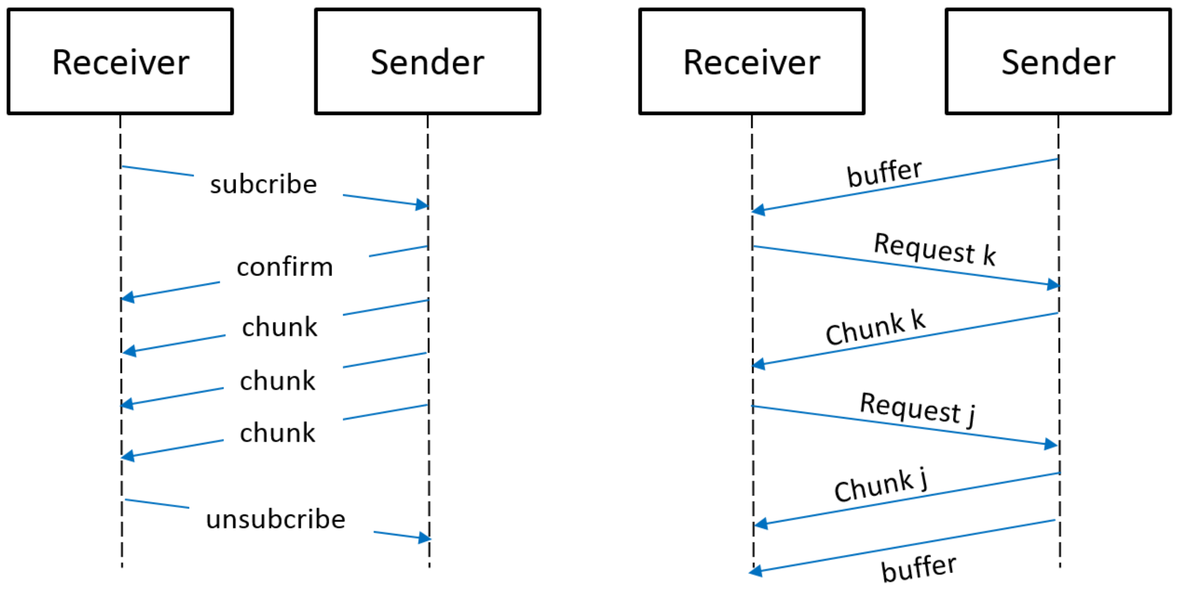

Figure 1. Message sequence charts for push (left) and pull methods (right)

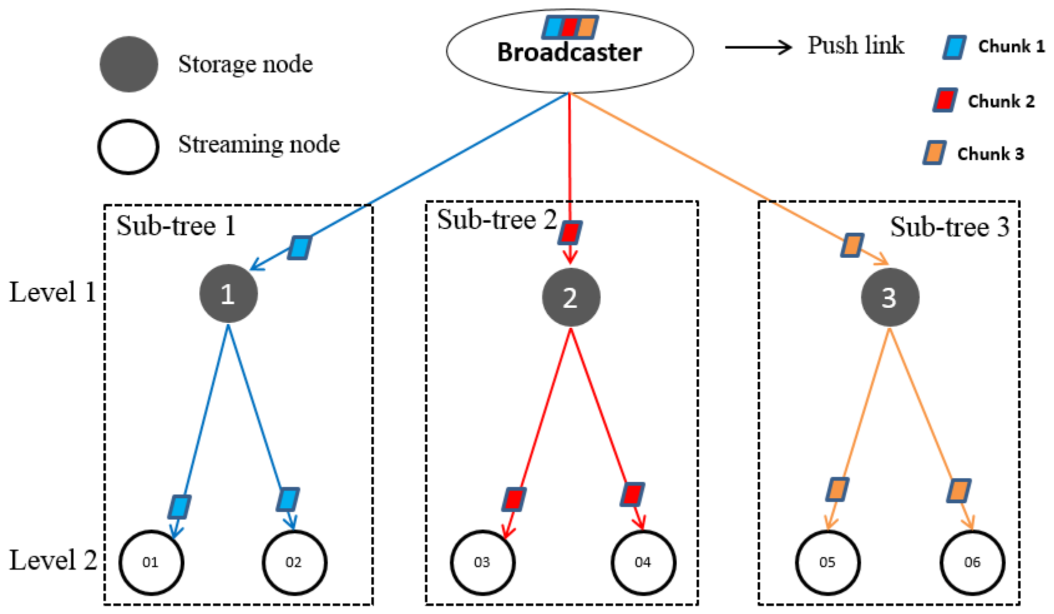

Figure 2. Overlay construction and pushing data to sub-trees

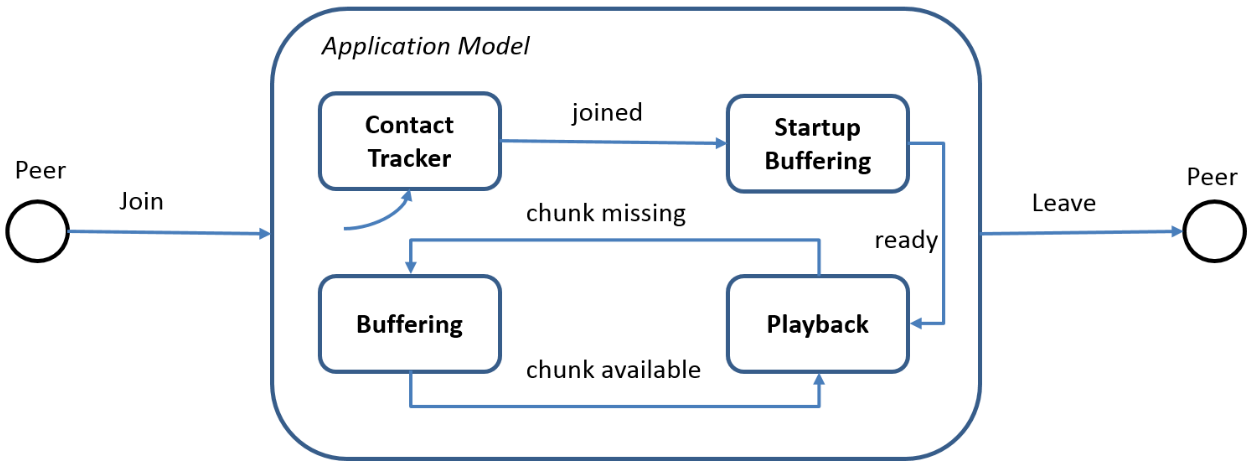

Figure 3. The life circle of peer in the network

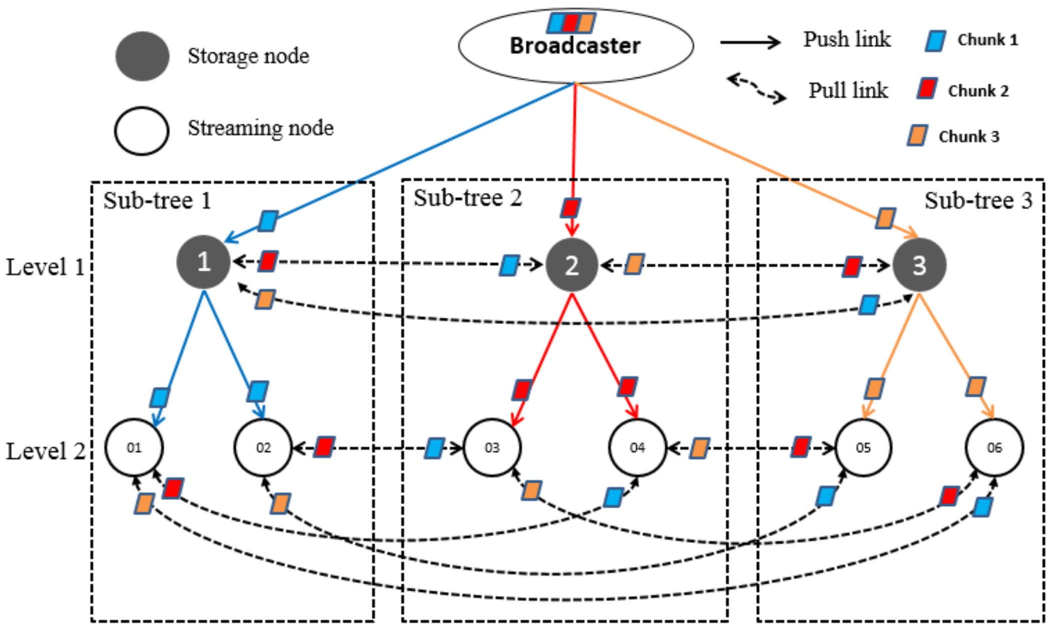

Figure 4. Peer pull missing data from node in other subtrees

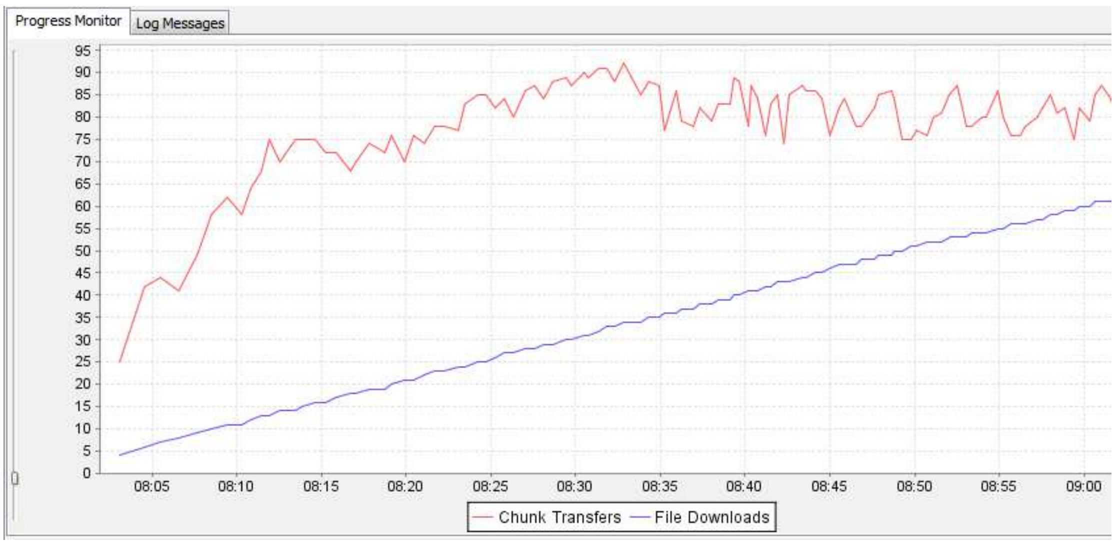

Figure 5. Comparison of processing time for transferring whole file and chunk