SEMI-FLUID: IMPLEMENTING A HIGH SPEED CONTENT DISTRIBUTION MODEL FOR PEER-TO-PEER OVERLAY NETWORK

Salah Noori, Mousa Al Falayleh, Manal M. Nasir
College of Computer Information Technology
American University in the Emirates (AUE)
Dubai, United Arab Emirates
{salah.alnaemi, mousa.falayleh, manal.nasir}@aue.ae

ABSTRACT

This paper proposes and implements a novel content distribution model for reducing or minimizing delay in data dissemination. Currently, content distribution is based on two models: the Fluid model and the Chunk model. The Fluid model provides continuous transferring of the content from the source to multiple receivers. For high throughput, a receiving node should distribute a bit once it has received that bit. However, working with the Fluid model in a heterogeneous network needs special care because the model incorporates tightly coupled connections between adjacent nodes. This imposes fundamental performance limitations, such as dragging down all transfer rates in the system to the rate of the slowest receiving node. In the Chunk model, contents are first chopped into pieces of equal size and the subsequent distribution happens in pieces. That is, a node will not distribute a piece until it has fully received that piece. A Chunk model is a loosely coupled connections; a node will not distribute a chunk until it has fully received that chunk, making nodes wait to receive the entire chunk before they can start distributing it. The novel Semi-Fluid content distribution model proposed in this paper will distribute chunk content in different heterogeneous networks in a fluid manner, without having any backpressure caused by Fluid content distribution model, or encountering chunk transition delay caused by Chunk content distribution model.

KEYWORDS

Peer to Peer Network, Overlay Network, Content Delivery Network, Content Distribution Network.

1 INTRODUCTION

Over the past decades, users have witnessed the growth and maturity of the Internet, which has caused enormous growth in network traffic, driven by the rapid acceptance of broadband access, the increases in systems complexity, and rich content. The ever-evolving nature of the Internet brings new challenges in managing and delivering content to users. For example, popular Web services often suffer congestion and bottlenecks due to the large demands posed on their services. Coping with such unexpected demand causes significant strain on a Web server and eventually the Web servers are completely overwhelmed with the sudden increase in traffic. The Web site holding the content might become temporarily unavailable.

A content delivery network (1,2) or content distribution network (CDN), is a system of computers networked together across the Internet that cooperate transparently to deliver content to end

users, most often for the purpose of improving performance, scalability, and cost efficiency. Collaboration among distributed CDN components can occur over nodes in both homogeneous and heterogeneous environments. CDNs have evolved to overcome the inherent limitations of the Internet in terms of user perceived Quality of Service (QoS) when accessing Web content. They provide services that improve network performance by maximizing bandwidth, improving accessibility, and maintaining correctness through content replication.

The recent work (3, 4) on CDN can be largely divided into three categories: (i) infrastructure-based content distribution like, the distributed server architecture, (ii) overlay network-based distribution (5), like application layer multicast (ALM), and (iii) peer-to-peer content distribution, which includes P2P file sharing and P2P media streaming (6).

Generally, application layer multicast (ALM) depends on the Fluid model for content distribution. The Fluid model provides continuous transferring of the content from the source to multiple receivers. However, working with the Fluid model in a heterogeneous network needs special care, because the model incorporates tightly coupled connections between adjacent nodes in a distribution environment. When participating peers are very heterogeneous, particularly in terms of the amount of download bandwidth they use, this will significantly limit the performance of all peers. One solution is to use push-back flow control to rate-limit the upstream link coming from the sender (7). This backpressure or single-rate schemes have known limitations in presence of a large number of groups: a single slow receiver can drag down the data rate for the whole group. Another solution is to employ network coding, which encodes content into a linear combination of blocks (8). Under this mechanism, slow peers can recover missing blocks after they receive enough blocks (attempt to recover the original content from the encoding symbols). However, network coding also has weaknesses. A peer may need to spend a huge amount of time on decoding the data it receives. Also, coding will add extra information (XOR operation symbols) to the original data packets, which results in reception overhead.

Peer-to-Peer (P2P) always depends on the Chunk model, where all connections among the peers are completely loosely coupled. This definitely fits with the heterogeneity of the internet. In order to maximize the participation of each of the peers in the network, large content is typically divided into many small pieces (or "chunks") that are directly exchanged between the peers (9, 10). Chunk model systems have a key difference with Fluid model systems: the content is organized into chunks whose size is significantly greater than IP packets. A peer will not distribute a chunk until it has fully received that chunk, making peers wait to receive the entire chunk before they can start serving it. This becomes untenable because content transfer may take a long time, during this time the upload capacity of downloading peers is wasted. The Chunk model imposes a fundamental performance constraint; where peers' uploading bandwidth is not fully utilized. This constraint adds a significant delay for peer-to-peer file sharing which increases the final distribution time for the file. Subsequently, peer-to-peer streaming

applications suffer from low-quality video, periodic hiccups, and high delay (stream diffusion metric).

This paper proposes and implements the novel Semi-Fluid content distribution model that will distribute chunk content in different heterogeneous networks in a fluid manner, without having any backpressure caused by Fluid content distribution model, or encountering chunk transition delay caused by Chunk content distribution model.

This paper is organized as follows. Section 2 introduces the proposed content distribution model. Section 3 explains the experimental setup of implementing the proposed model while Section 4 focuses on the performance of the implemented model. Finally, Section 5 gives some concluding remarks.

2 THE PROPOSED SFCD MODEL

The proposed content distribution model is a combination of Chunk and Fluid models, to enable multiple-rate reception, with individual rates that match the end-to-end available bandwidth along the path. The most important aspect in designing SFCD model is to keep the support for simultaneous parallel downloads from multiple sources rather than a single source. From Figure 1, we can see that inside each peer, and for each file distribution session; there are a polarity number of SFCD models. Each model handles the distribution of one part of the data content, by combining one upstream link to multiple heterogeneous downstream links.

In a real peer-to-peer environment and for content distribution, this content is partitioned into few parts, where each part is distributed separately using different establishment for content distribution model. Usually these parts have the same size in terms of number of bytes or chunks, as content parts are first chopped into chunks of equal size and the subsequent distribution happens in

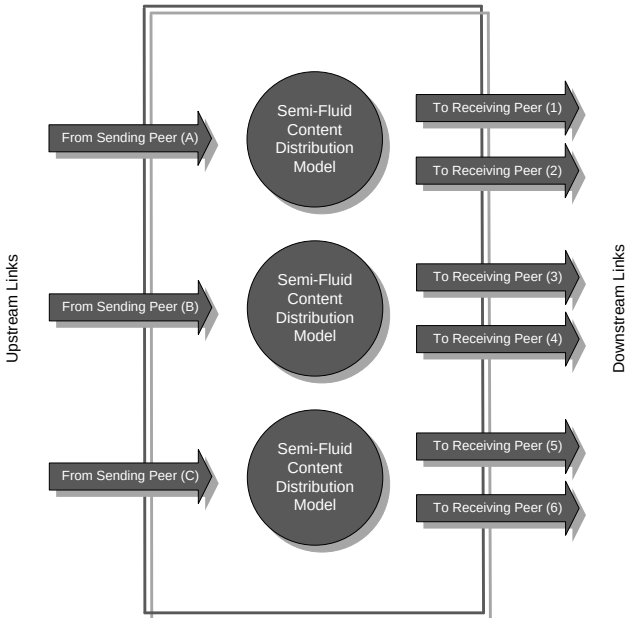

Figure 1. SFCD model with parallel download/upload

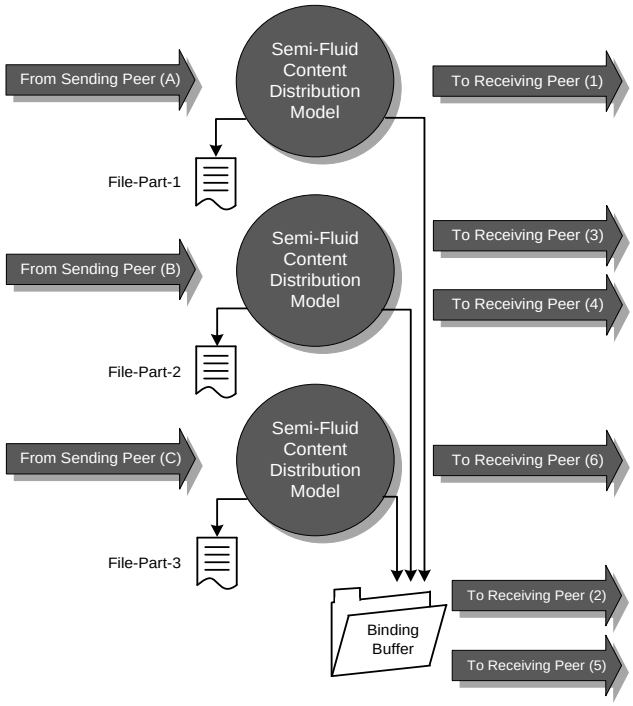

Figure 2. An Overview of the SFCD model.

chunks. Figure 2 shows how each SFCD model is responsible for receiving and distributing one part, whereby each intermediary SFCD model forwards only those received chunk's packets to fast peers that can immediately be written into the downstream link. It also builds a group of adjacent chunks in a single application buffer to be sent to all other slow receivers. The application buffer is represented by a temporary storage that takes place on hard disk, instead of system buffer, when the size of content is large.

2.1 SFCD Model Main Units and Structure

The structure of a single SFCD model is shown in Figure 3. Each model consists of: Data Receive, Flow Distribution, Traffic Monitoring, and Binding Buffer.

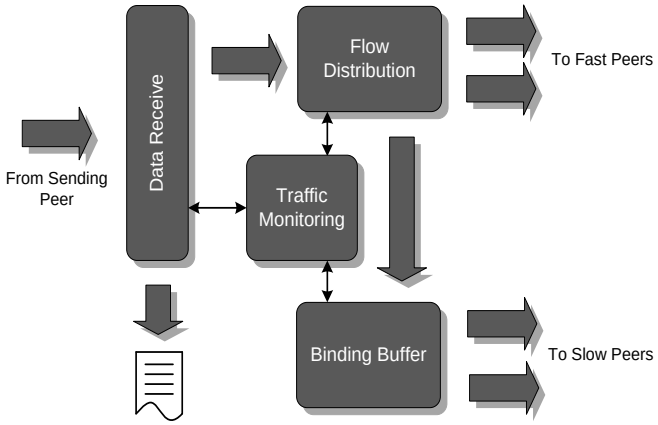

Figure 3. SFCD Model Architecture and Its Main Units.

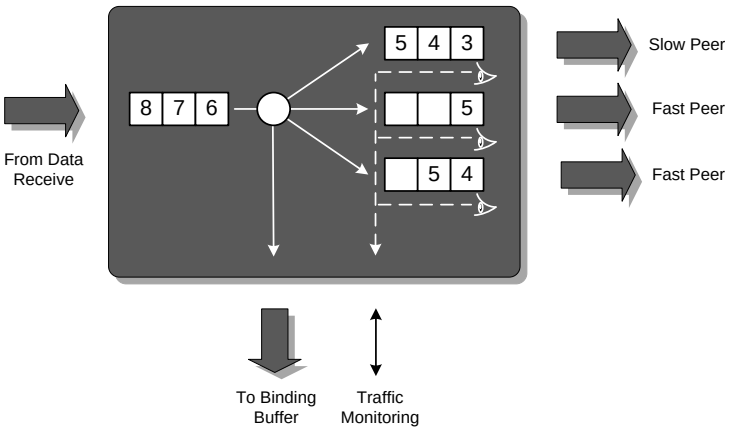

Figure 4. An Overview of Flow Distribution

The Data Receive handles all the communication with the sending peer. It also receives the chunks' packets from the incoming TCP buffer and stores those packets on the hard disk as part of the data content, and passes these chunks to the Flow Distribution. It is very clear, as we are claiming a combination between Chunk and Fluid model, to see that Flow Distribution represents the Fluid model part without any network coding or backpressure mechanism while the Binding Buffer represents the Chunk model part in our SFCD model. Finally, the Traffic Monitoring which is the core of the model, handles flow and congestion control.

2.1.1 Flow Distribution

This unit (Figure 4.) is the core of immediate content distribution; where upstream link is tightly coupled to all other fast downstream links. This unit dequeues the arriving chunk packets from the Data Receive unit in a small incoming buffer, and copies the packets to a small finite application layer buffer for each downstream link. At the same time, it sends these packets directly to

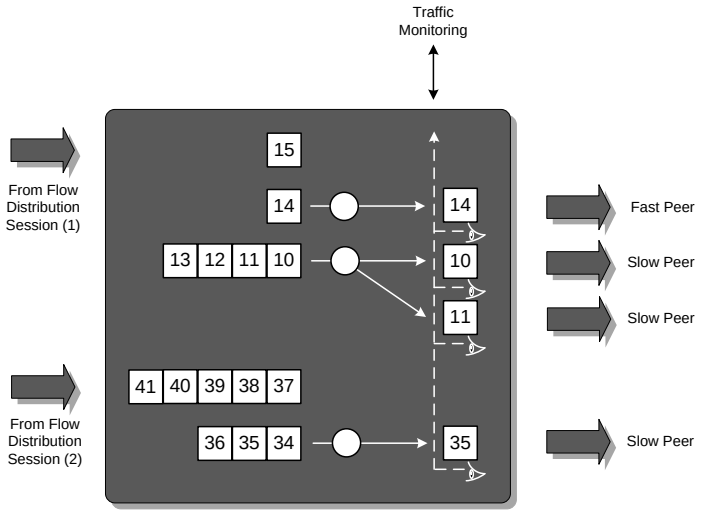

Figure 5. An Overview of Binding Buffer Unit.

the Binding Buffer unit. This unit copies data to all outgoing buffers that have available buffer space. If no space is available the unit will not receive any more packets from the Data Receive unit, and waits for the Traffic Monitoring decision to move any slow downstream link that causes the congestion to the Binding Buffer unit.

2.1.2 Binding Buffer

This unit (Figure 5) is a dynamic unlimited temporary buffer. It binds the main upstream link to all slow downstream links, in a loosely coupled way. This unit prevents any congestion that might happen to the main upstream link. It always keeps a smooth flow control for the whole model. The unit receives the chunks from the Flow Distribution and builds a group of adjacent chunks in small files to be used later by slow peers. It stops the building of the current group of chunks and starts a new group whenever there is a request from a slow downstream link to any of the chunks included within the group, as the downstream link cannot open the file for reading while the Binding Buffer is opening it for writing. When any downstream link keeps on requesting the current receiving chunk, the Binding Buffer unit will continue building a small file of a one chunk size, until a decision is made by the Traffic Monitoring to move that link to the Flow Distribution and consider this link as a fast downstream link.

The Binding Buffer unit is used and shared by all other SFCD models, which build the whole file distribution session, and is responsible for distributing one single file. The system will build other Binding Buffer units for other file distribution sessions.

2.1.3 Traffic Monitoring

This unit (Figure 6.) is the main control for the whole SFCD model. Without this unit, the model is unable to perform smoothly. This unit's main function is to collect information from all other units inside the model, collect information

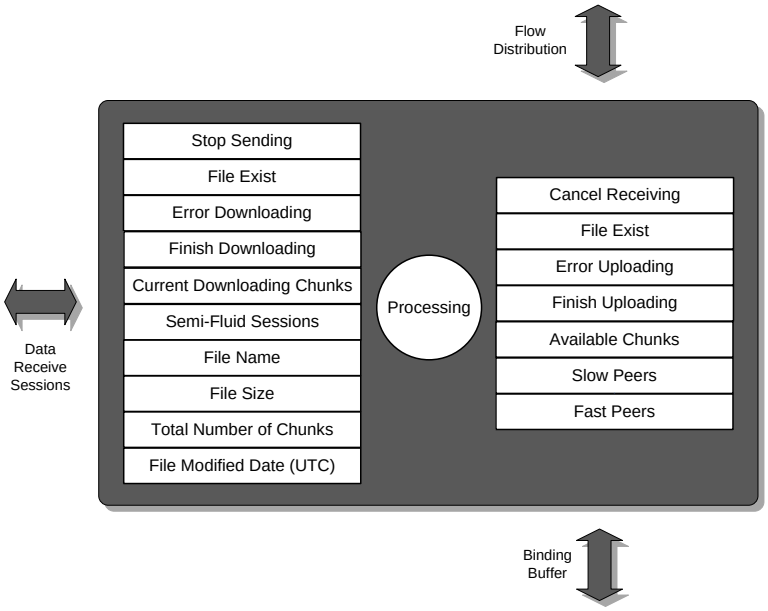

Figure 6. An Overview of Traffic Monitoring Unit.

about the whole file distribution session, and process these information for decisions related to a smooth congestion free file distribution. It provides all downstream links with detailed information about the upstream link status and vice versa.

Information that is exchanged between upstream and all downstream links via the Traffic Monitoring units include details like file information (name, size, and modified date in coordinated universal time (UTC) format), status information (like the sender stopped sending or the receiver canceled receiving), besides the main important status information (failure in connection), which is detected through a failure of the related TCP connection.
For marking the downstream links status, the unit depends on the accumulative result for all the links, like marking an error uploading, this value will remain false, unless when all the downstream links have an error in communication. In this case only, the Traffic Monitoring will raise a flag of error uploading. The same is true for canceled receiving, file exist, and finished uploading.

The Traffic Monitor's main function is moving slow peers from Flow Distribution to Binding Buffer and fast peers from Binding Buffer to Flow Distribution. There is a threshold for moving downstream links between Flow Distribution and Binding Buffer, in other words, considering them as fast peers or slow peers.

Inside Flow Distribution the Traffic Monitor keeps tracking all downstream links, and manages outgoing buffers overflow. For two consecutive chunks, if there is still insufficient buffer space, then the unit moves that downstream link to the Binding Buffer unit, and it will be considered as a slow link. The same procedure happens inside the Binding Buffer, when any downstream link becomes fast enough and tries to catch-up with the speed of the upstream by requesting the latest building chunks. Then a decision is made after requesting two consecutive chunks to move that downstream link to Flow Distribution and consider that link as a fast link.

We chose two chunks as our threshold value. This value is used inside Traffic Monitor unit, for making decisions on marking a peer, whether it is fast or slow. This value is chosen, after a serious of long testing and evaluation to the proposed SFCD model, for finding the optimal value. We found that making a decision with only one chunk makes the system too fluctuated by keeping moving links between Flow Distribution and Binding Buffer. On the other hand, choosing three chunks or above for the threshold value, will slow down the system, since the upstream link will start to add more backpressure on the sending peer, when any of the downstream buffer is full.

3 EXPERIMENTAL SETUP

We conducted our experiments on PlanetLab (11), a wide area network test-bed, using 80 nodes located in 28 sites that are geographically distributed around the world. We chose the sites so that they represent a heterogeneous environment in terms of bandwidth assigned to them, where some of the sites offer high throughput typical of corporate and university networks, while

others provide much lower throughput typical of home and dial-up users.

We experimented with all existing content distribution models including our proposed one, on two different files with sizes 50 MB and 300 MB. These experiments were useful to precisely quantify the SFCD model distribution speed and benefits, and also to compare SFCD model against Chunk, Fluid with Encoding, and Fluid with Backpressure models, in a predictable environment. All these other content distribution models were also deployed on the PlanetLab wide-area testbed.

All of our experiments were run on an uncompleted/asymmetric tree topology, where each peer of overlay participants is directly connected to three or two peers (outdegree 2 and 3), except leaf peers. This is more representative of actual Internet topologies; it allows us maximum flexibility to affect the bandwidth and have different rates between any two peers. For illustration, Figure 7 shows a snapshot of a tree in our experiment. The total number of nodes is 80, but height is 12, recalls that a full and balanced 3 outdegree tree of 80 nodes has a height of 4 only, and for 2 outdegree has a height of 6 only.

The connections among peers were chosen randomly, but we were trying to keep fast peers at the root of the tree, else if we use slow peers at the root, those slow peers will drag down the whole tree to the speed of those slow peers. We partially follow the idea of emulating fat-trees in overlays (12). Fat-Trees are like real trees in that their branches become thicker the closer one gets to the root, thus overcoming the "root bottleneck" of regular trees. We tried to keep the number of tree levels high enough. This will give us clear results especially for the Chunk model, where the chunk size is 256 KB. We configured Fluid with Encoding model to use a 256 KB block size also, which is the same as chunk size in Chunk model. Fluid with Encoding was not performing the file encoding/decoding; instead we mark the downloads as successful when a required number of distinct file blocks is successfully received, including the fixed 3% overhead that an actual encoding scheme would incur.

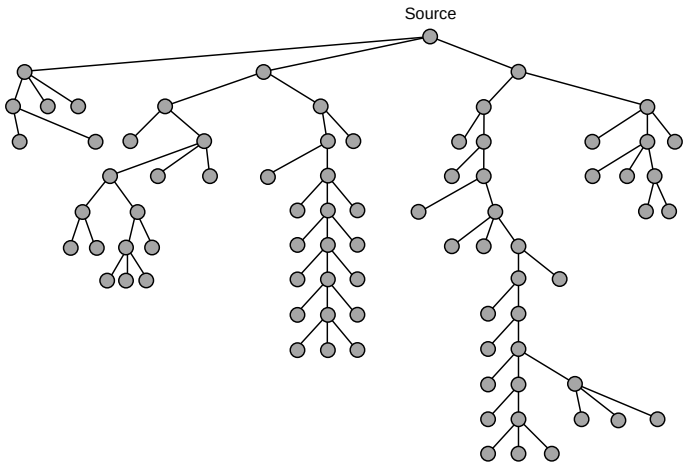

Figure 7. Snapshot for PlanetLab testing.

Finally, for most of the testing scenarios, we conducted identical experiments in four subsequent steps starting with: Semi-Fluid, Chunk, Fluid with Encoding, and Fluid with Backpressure. We conducted these experiments three times, with the same sequence, and then the average results were obtained.

4 OVERALL PERFORMANE FOR SFCD PLANETLAB TESTING

In our experiments over PlanetLab, it has been observed that some peers may slow down significantly due to congestion. If all the peers share the same queue, the uploading to the slowest peer will block the uploading to the remaining peers, and this is very clear with Fluid with Backpressure model, which is very far compared to other content distribution models, and it is worse among them, where the source's upload bandwidth will be wasted.

For these live Internet runs, we found that SFCD model can deliver the optimal performance, and added a significant improvement to the downloading speed. Figure 8 shows the testing for a relatively small size file (50 MB). It is very clear to see that peers in the beginning are having the same download time as SFCD model, especially when compared to Chunk model. Fluid with Encoding model added some extra overhead, for this reason the download time is higher for the first peer.

For all the models and for the range of peers between 30 and 40, we can see an almost constant download time. This usually happens when all the peers are in the same network, where the packet transition delay is very small. Finally, Fluid with Encoding model performance is better than Chunk model, with Chunk model peers suffering from the large chunk delay when moving from one peer to another (overlay hop-count), especially with this topology. This results in a high levels tree, with a high delay.

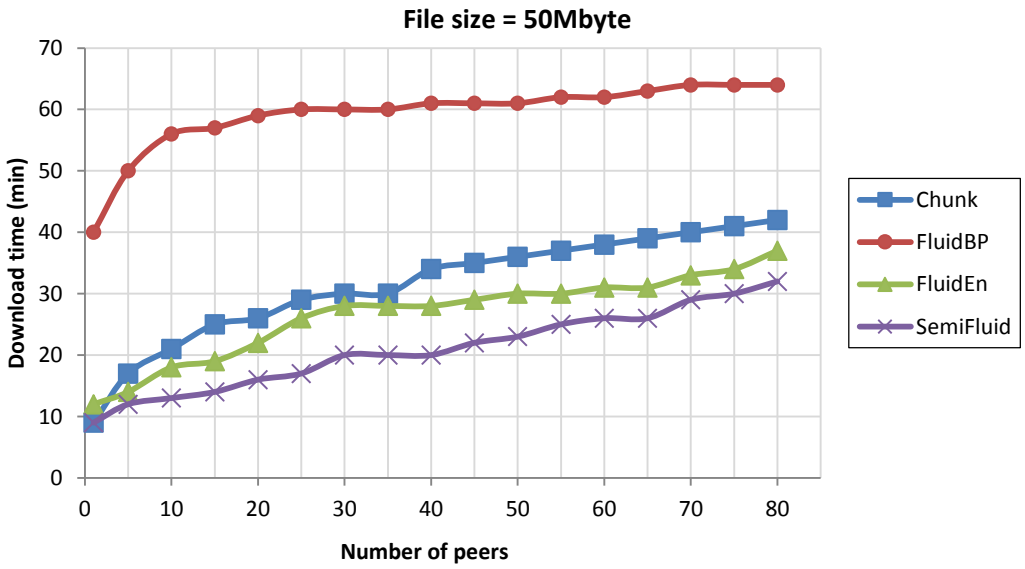

Figure 8. PlanetLab evaluation for different content distribution model: Total download time and file size = 50MB.

Figure 9 shows our result for PlanetLab testing when having a large size file (300 MB), all the models' performance is the same when testing with small size file, except that in this case and especially after reaching 48 peers, the performance of Fluid with Encoding model decreases rapidly. This is because of the extra encoding overhead added to the file. It increases with the increasing file size. But for Chunk model, the delay encountered because of chunk transition from one level to another is the same whether we are using small or large file size. The only remaining delay is the downloading of the whole file, which depends on the speed of the slowest node along the path from the sender to the receiver.

A very important observation is that the shapes of the performance for Semi-Fluid and Fluid with Encoding models are almost identical, because the concept is the same, except that Fluid with Encoding model is higher than SFCD model, from the starting to the end, because of the extra data added by the encoding overhead. The results show that our proposed Semi-Fluid content distribution model offers robust and improved downloading speed performance.

5 CONCLUSIONS

Within this paper we have presented a novel content distribution model for reducing or minimizing delay in peer-to-peer data dissemination. We show the ability of the distribution model in supporting heterogeneous networks by implementing and integrating it in a large scale distributed network of servers.

The use of the Traffic Monitoring unit in our architecture provides us with the ability to combine chunk and fluid content distribution models, and distribute chunk content in a fluid manner. The degree of freedom that the

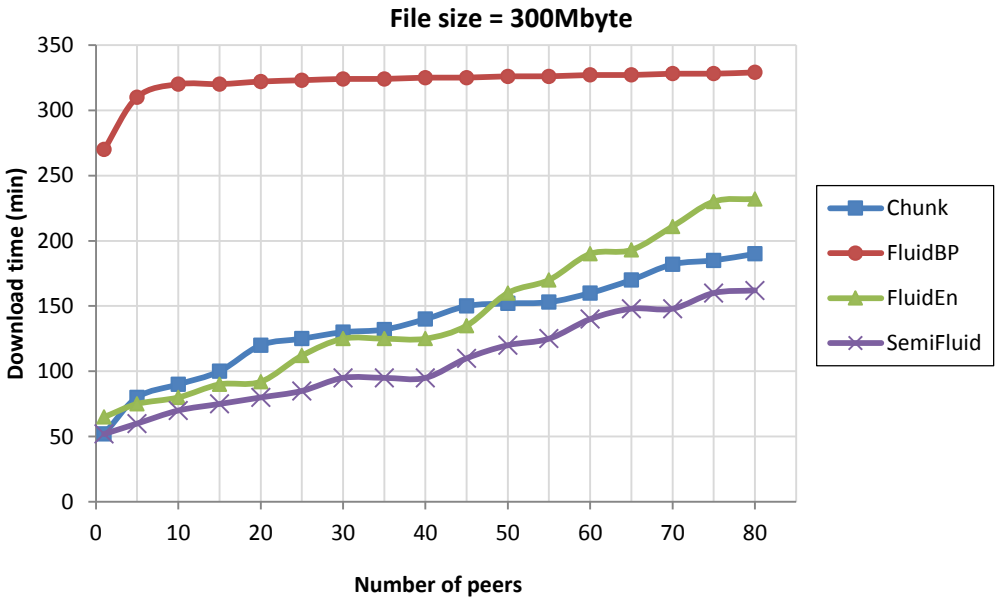

Figure 9. PlanetLab evaluation for different content distribution model:

Total download time and file size = 300MB.

use of the Binding Buffer unit in our architecture enables us to loosen the tight coupling of TCP connections that is needed in a fluid content distribution model in order to provide reliability, without any performance limitations. Also, the use of Flow Distribution unit in our architecture provides us with the ability to tighten the loose coupling of TCP connections that exists in chunk content distribution models, as a result of distribution chunks in store and forward manner.

6 REFERENCES

1. Leighton, F. Thomson, and Daniel M. Lewin: Content delivery network using edge-of-network servers for providing content delivery to a set of participating content providers. U.S. Patent No. 6,553,413 (2003).
2. Gagliardi, Joshua D., Timothy S. Munger, & Donald W. Ploesser: Content Delivery Network. U.S. Patent No. 20,120,254,421 (2012).
3. Byers, J. W., Considine, J., Mitzenmacher, M., & Rost, S: Informed content delivery across adaptive overlay networks. IEEE/ACM transactions on networking, 12(5), pp. 767-780 (2004).
4. Carra, D., Cigno, R. L., & Biersack, E. W: Introducing heterogeneity in performance analysis of p2p networks for file distribution. Univ. of Trento, Tech. Rep. DIT-04-113 (2010).
5. Trung, Ha Quoc: A new converge cast algorithm: Application of procedural distributed recursive wave model and feedback function. International Journal of Digital Information and Wireless Communications (IJDIWC), 2(1), pp. 82-85 (2012).
6. Haasn, Mahamat Issa: Semantic Technology and Super-peer Architecture for Internet Based Distributed System Resource Discovery. International Journal of New Computer Architectures and their Applications (IJNCAA), 1(4), pp. 848-865 (2012).
7. Kwon, Gu, and John W. Byers: ROMA: Reliable overlay multicast with loosely coupled TCP connections. In: Proc. of the twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies. Vol. 1. IEEE (2004).
8. Gkantsidis, Christos, and Pablo Rodriguez Rodriguez: Network coding for large scale content distribution. In: Proc. of the twenty-fourth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE. Vol. 4., pp. 2235-2245, IEEE (2005).
9. Gaeta, R., Gribaudo, M., Manini, D., & Sereno, M.: Analysis of resource transfers in peer-to-peer file sharing applications using fluid models. Performance Evaluation, pp. 149-174 (2006).
10. Ge, Z., Figueiredo, D. R., Jaiswal, S., Kurose, J., & Towsley, D.: Modeling peer-peer file sharing systems. In: Proc. of the twenty-second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies, Vol. 3, pp. 2188-2198, IEEE (2003).
11. PlanetLab Consortium. Planetlab: An open platform for developing, deploying, and accessing planetary-scale services. http://www.planet-lab.org/
12. Birrer, S., Lu, D., Bustamante, F., Qiao, Y., & Dinda, P.: FatNemo: Building a resilient multi-source multicast fat-tree. Web Content Caching and Distribution, pp. 182-196 (2004).