SEMI-FLUID: A CONTENT DISTRIBUTION MODEL FOR FASTER DISSEMINATION OF DATA

By

SALAH NOORI SALEH

Thesis submitted in fulfillment of the requirements
for the degree of
Doctor of Philosophy

March 2010

ACKNOWLEDGEMENTS

بســـــــــــم الله الرحمن الرحيـــــــــــم

" اقرأ باسم ربك الذي خلق "

I would like to take this opportunity to convey my sincere thanks and deepest gratitude to my supervisor, Prof. Dr. Sureswaran Ramadass, for all the help and valuable guidance provided to me during the preparation of this thesis. I consider myself privileged to have had the opportunity to work under his guidance.

Moreover, I would like to convey my appreciation to the MCS ver. 6 core team, all NAv6 center members, the School of Computer Sciences, the Institute of Postgraduate Studies, the university library, and the Mlabs staff for their help and support. A special thanks to my friend Ayman Helweh for his valuable help.

My sincere gratitude also goes to my wife Dr. Ban. I thank her for her support, understanding, and encouragement during every step of my study and my writing of this thesis.

The favour, above all, before all, and after all, is entirely Allah's, to Whom my never-ending thanks and praise are humbly due.

DEDICATION

I would like to dedicate this thesis to the dearest ones, my father for his patience and the encouragement he provided me with during the entire period of the study, and my late mother (May Almighty Allah rest her soul) who shared the stress in my life, encouraged me in times of dismay, cheered me up in times of distress, and renewed my hope in times of despair. I would also like to dedicate this thesis to my darling wife, lovely son, brother, and sister.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ADSL	Asynchronous Digital Subscriber Line
ALM	Application Layer Multicast
BRB	Bind Recovery Buffer
CDN	Content Distribution (Delivery) Network
CLR	Common Language Runtime
DSDFT	Data Size Dependent File Transfer
E2E	MCC End-to-End Multicast Congestion Control
FEC	Forward Error Correction
FTP	File Transfer Protocol
IL	Intermediate Language
IP	Internet Protocol
LAN	Local Area Network
LDS	Local Data Structure
MCS	Multimedia Conferencing System
MDC	Multiple Description Coding
MDNF	Most Demanding Node First
MFTS	Multipoint File Transfer System
MS-MFTS	Multi-Server Multipoint File Transfer System
NAv6	National Advanced IPv6 Center
P2P	Peer-to-Peer
PRIM P2P	Receiver-Driven Mesh-Based Streaming
QC	Quality Control
QoS	Quality of Service
RAP	Rate Adaptation Protocol
RDS	Remote Data Structure
REALM	Reliable Application Layer Multicast
ROMA	Reliable Overlay Multicast
RPF	Rarest Piece First
SFCD	Semi-Fluid Content Distribution
TCP	Transmission Control Protocol
TFRC	TCP-Friendly Rate Control
UTC	Coordinated Universal Time

WAN	Wide Area Network
XOR	Exclusive OR

SEMI-CECAIR: MODEL PENGEDARAN KANDUNGAN UNTUK PENYEBARAN DATA YANG LEBIH PANTAS

ABSTRAK

Tesis ini mencadangkan serta melaksanakan suatu model agihan kandungan bagi mengurangkan atau meminimumkan kelengahan penyaluran data sebaya. Buat masa ini, agihan kandungan dalam rangkaian tindihan-atas adalah berdasarkan dua model berikut: model Kelulan dan model Bendalir. Model Bendalir menyediakan penghantaran kandungan secara berterusan daripada sumber kepada penerima berbilang. Bagi truput (throughput) yang tinggi, suatu nod sebaya sepatutnya mengagih satu bit apabila ia menerima bit tersebut. Walau bagaimanapun, bagi model Bendalir dalam rangkaian heterogen, ia memerlukan penjagaan khusus kerana terdapatnya gandingan yang agak ketat di antara nod bersebelahan. Maka ia menyebabkan batasan prestasi asas, seperti melambatkan kadar pemindahan dalam sistem mengikut kadar pemindahan nod sebaya yang paling perlahan. Dalam model kelulan, kandungan terlebih dahulu dipenggal ke kepingan yang sama saiz dan pengagihan berlaku dalam kepingan. Maka, nod sebaya tidak akan mengagihkan sesuatu kepingan, sehinggalah ia telah menerima kepingan tersebut sepenuhnya. Model kelulan merupakan gandingan pautan longgar: sesuatu nod sebaya tidak akan mengagih sesuatu kelulan, sehinggalah ia telah menerima kelulan tersebut sepenuhnya, menyebabkan nod sebaya menunggu untuk menerima keseluruhan kelulan sebelum ia mengagihkannya semula. Keadaan ini tidak diingini kerana pindahan kandungan mungkin mengambil masa yang lama. Lebih-lebih lagi, dalam tempoh tersebut, kapasiti muat naik bagi node sebaya yang muat turun tidak digunakan sepenuhnya. Lengahan adalah kritikal bagi aplikasi interaktif masa nyata. Model agihan kandungan yang lemah menyebabkan tempoh agihan yang

panjang. Sebaliknya, model yang baik memendekkan masa pelengkapan dan penggunaan sumber seperti lebar jalur rangkaian secara cekap. Model agihan kandungan Separa Bendalir yang baru ini akan mengagihkan kandungan kelulan dalam rangkaian tindihan-atas heterogen yang berbeza dalam bentuk bendalir, tanpa mempunyai sebarang tekanan–berbalik (backpressure) yang disebabkan oleh model agihan kandungan Bendalir, atau lengahan transisi kelulan yang disebabkan oleh model agihan kandungan kelulan. Pembuktian secara matematik dan keputusan ujian pelaksanaan sebenar menunjukkan bahawa model yang dicadangkan ini menunjukan penyelesaian optimum bagi semua kes yang diuji bagi rangkaian hetrogen.

SEMI-FLUID: A CONTENT DISTRIBUTION MODEL FOR FASTER DISSEMINATION OF DATA

ABSTRACT

This thesis 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. This becomes untenable because content transfer may take a long time and during this time the upload capacity of downloading nodes is unutilized. Delay is critical for real-time and interactive applications. A poor content distribution model could result in considerably longer distribution time, while a good model could shorten the completion time and efficiently utilize resources like network bandwidth. The novel Semi-Fluid content distribution model proposed in this thesis 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, by optimizing the existing (Chunk and Fluid) content distribution models, and enabling better utilization of node's resource, such as local storage and bandwidth. Mathematical proof and real implementation test results show that our proposed Semi-Fluid content distribution model finds an optimal solution for all cases tested in heterogeneous networks.

CHAPTER ONE
INTRODUCTION

1.1 Motivation

For high-concurrency applications ranging from live streaming to reliable delivery of popular content, recent research trends proposed serving these applications using an end-system, or overlay network. Overlay network is a virtual network of nodes and logical links that is built on top of an existing network with the purpose to implement a network service that is not available in the existing network. Overlay networks offer a powerful alternative compared to traditional mechanisms for content delivery, especially in terms of flexibility, scalability, and deploy-ability. In order to derive the full benefits of the approach, some care is needed when providing methods for representing and transmitting the content in a manner that is as flexible and scalable.

Distribution in overlay networks leverages on the uploading capacity of the receiving nodes (peers) to aid in the content distribution process. Specifically, once a node has received any portion of the content, it can redistribute that portion to any of the other receiving nodes. Content distribution in overlay networks are based on two models: the Chunk model and the Fluid model.

Prior research on overlay networks mainly focuses on peer and content discovery and scheduling, overlay topology formation, fairness and incentive issues, etc, but seldom investigates the content distribution problem which is also a core component in many overlay network systems, like peer-to-peer file sharing and media streaming. A poor content distribution model could result in considerably

longer distribution time, while a good model could shorten the distribution time and efficiently utilize resources like network bandwidth.

1.2 Background

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 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 (2004) on CDN can be largely divided into three categories: (i) infrastructure-based content distribution like, the distributed server architecture, (ii) overlay network-based distribution, like application layer multicast (ALM), and (iii) peer-to-peer content distribution, which includes P2P file sharing and P2P media streaming.

1.2.1 Fluid Model

The Fluid model is applied in application layer multicast (ALM) for replacing IP multicast and providing a reliable content delivery network. The Fluid model incorporates tightly coupled connections between adjacent nodes in a distribution environment. For high throughput, a receiving node should distribute a bit once it has received that bit.

The Narada protocol (Yang-hua, Sanjay, & Hui, 2000; Yang, Sanjay, Srinivasan, & Hui, 2001) was one of the first application layer multicast protocols that demonstrated the feasibility of implementing multicast functionality at the application-layer, in which streaming content is replicated and forwarded using only the resources of peers who themselves want this data. The inherent advantage of these schemes is extreme scalability. This is because these protocols proportionately increase the amount of resources devoted to transferring data as the number of clients who want the data increases.

Yoid (Francis, 2000), along with Narada, is one of the first application layer multicast protocols. Since Yoid directly creates the data delivery tree, it has a direct

control over various aspects of the tree structure. This is in contrast to the mesh-first approach (Narada) which has an indirect control over the tree structure.

Initially, ALM with Fluid model proposed the use of a Tree topology to distribute streams among peers, where all peers are arranged into a tree rooted at the source. The content is streamed down from the source to every peer along the tree edges in a push-based manner. Though the tree approach is simple and achieves low delay, the failure of a node can seriously affect the streaming quality of all its descendants due to tree re-construction. Furthermore, the streaming rate cannot be guaranteed as it is limited by the least uplink bandwidth of a node in the tree. Therefore, trees cannot accommodate network dynamics and asymmetric bandwidth well. Also the leaf nodes in the tree do not participate in the distribution process.

Single tree performance can be significantly improved by using a parallel tree PTree (multi-tree) topology, which organizes the peers in k different trees such that each peer is an interior peer in at most one tree and a leaf peer in the remaining $(k-1)$ trees. The content is then striped into k stripes, where each stripe is distributed on a different tree. This approach has three important limitations: (i) in the presence of churn (where peers may open and close connections or leave and rejoin the infrastructure at arbitrary times), maintaining multiple tree-shaped overlays with desired properties could be very challenging. (ii) The rate of content delivery to each peer through individual trees is limited by the minimum throughput among the upstream connections which could be even smaller than the bandwidth of a single sub-stream. (iii) Peers cannot share the content more than one time.

1.2.2 Chunk Model

In some distribution systems a file is broken down into many chunks that can be downloaded independently. These chunks are then redistributed again by the receiving nodes as soon as the chunk is completely received and verified. Breaking a file into smaller units has several advantages for performance and robustness.

Chunk technology was first introduced by BitTorrent clients (BitTorrent, 2004; Cohen, 2003), though other variations have also been proposed (Sherwood, Braud, & Bhattacharjee, 2004). In BitTorrent, a file is split into chunks, typically of the order of a thousand chunks per file. To download a complete file, a user downloads different chunks of the desired file from other users. The chunks are not downloaded sequentially, but are based on the rarity of the chunk at that time. When all the chunks have been downloaded, the chunks are reassembled, and the user has their file. This method of splitting a file into many pieces greatly facilitates the sharing of large files.

Nowadays most peer-to-peer file-sharing applications depending on Chunk model use a practice which is called swarming (Stutzbach, Zappala, & Rejaie, 2005). Swarming is a new type of data transfer which leverages the cooperative nature of peer-to-peer networking to serve large numbers of users without placing a heavy burden on a centralized web server. With swarming, any user that has downloaded some piece of content from a server can then itself act as a server to other peers for that content. Because no single peer has the entire content, nor a high amount of bandwidth, peers download content from each other in parallel (Byers, Luby, & Mitzenmacher, 1999; Rodriguez & Biersack, 2002; Rodriguez, Kirpal, & Biersack,

2000), constructing the larger file from the pieces they collect. This approach frees the web server from having to deliver the entire file to all users. Instead, it gives a piece of the file to some users and then relies on those users to exchange the data among themselves.

The Chunk model meets certain requirements: (i) loosely coupled connections that accommodate asymmetric bandwidths. (ii) Fully supporting parallel download for chunks from different peers (swarming). (iii) Peers can easily make the decision to pull or push chunks; which helps a lot in content and peer scheduling. (iv) Finally, the Chunk model is the only model up till now that is used by the Mesh topology which is the most robust topology to churn.

The success of the Chunk model, and especially file swarming mechanisms, has motivated a new approach for scalable streaming of live content that is the mesh-based Peer-to-Peer streaming. In this approach, peers form a randomly connected mesh and incorporate swarming content delivery to stream live content. Peer-to-Peer streaming is classified into: video-on-demand (Annapureddy, Gkantsidis, Rodriguez, & Massoulie, 2006; Do, Hua, & Tantaoui, 2008), live media broadcasting (Bocca, 2008; Maria Elisa et al., 2009; PPLive, 2007; SpotCast, 2007), and video conferencing (Akkus, Civanlar, & Ozkasap, 2006; Hossain, Yi, & Yuan, 2009; Ponec, Sengupta, Chen, Li, & Chou, 2008).

1.3 Problem Statement

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. 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. 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. 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).

1.4 Research Objective

The main objective of this thesis is to develop a new content distribution model that distribute chunk content in different heterogeneous overlay networks in a fluid manner, to overcome the backpressure created by Fluid content distribution model, and to overcome the chunk transition delay caused by Chunk content distribution model.

1.5 Thesis Contribution

We consider the problem of architecting a reliable content delivery system across an overlay network using TCP connections as the transport primitive. The primary set of target applications are applications requiring reliability and high bandwidth, such as delivery of large files or video streams. The proposed Semi-Fluid content distribution model enables multiple-rate reception, with individual rates that match the end-to-end available bandwidth along the path, while using unlimited buffers at application-level relays, and the standard TCP protocol. The key to our

method is to make a departure from the straightforward approach in which each intermediate peer forwards all received packets to the downstream peers to achieve reliability, or using chunks only with store-and-forward approach. We apply an intermediate approach, whereby each intermediary peer forwards only those received chunk's packets to downstream peers that can immediately be written into the downstream TCP socket. It also builds a group of adjacent chunks in a single application buffer to be sent to all other slow receivers.

The main contribution of this thesis is to propose and design a new Semi-Fluid content distribution model that supports:

- Heterogeneous networks.
- Different overlay topologies.
- Swarming technique (parallel download).

The proposed Semi-Fluid content distribution model features distributed congestion control that achieves optimal bandwidth utilization. It is a combination of Chunk and Fluid content distribution models for faster dissemination of data in peer-to-peer networks.

Mathematical proof and real implementation test results show that the proposed Semi-Fluid content distribution model provides an optimal solution for the cases tested in heterogeneous networks. Therefore, we believe our model is a promising solution to be employed as the core distribution model in different overlay topologies, shortening the total download time experienced by users.

1.6 Thesis Organization

This thesis is organized into six chapters. The content is arranged such that each chapter provides a basic idea to further proceed to the next chapter. Firstly, this chapter (**Chapter 1**) introduces the background principles of content distribution models along with our research objectives and contributions.

In **Chapter 2**, we review literature and fundamental concepts related to our work and issues surrounding it. We discuss other overlay network protocols and methods, related work for improving both Chunk and Fluid content distribution models. We provide motivation for our work by describing some candidate architectures and the limitations of those proposed solutions.

Chapter 3 presents the methodology of how the proposed SFCD model was designed. The chapter also covers the evaluation of different content distribution models. Lastly, the methodology and design for integrating the proposed SFCD model in a large-scale distributed network of servers is also introduced.

Chapter 4 covers the architecture and implementation of the Multi-Server Multipoint File Transfer System, and the implementation of the proposed SFCD model, and the way it integrates and interacts with the Multi-Server Multipoint File Transfer System.

Chapter 5 covers the discussion on the experimental setup issues for evaluating our proposed SFCD model using the real implementation.

Chapter 6 covers an in-depth analysis and discussion of our proposed SFCD model mathematically, and its performance through detailed experiments and in real Internet environments and PlanetLab testbed.

Finally, **Chapter 7** covers the conclusions of the thesis, as well as recommendations for further research.

CHAPTER TWO
LITERATURE REVIEW

2.1 Introduction

The related work that we discuss in this chapter concerns prior research on overlay networks, their structure and requirements. In particular, we discuss other supporting technologies to content distribution models. We primarily focus on the work that has been done to support the Chunk and Fluid content distribution models, to improve their performance. These enhancements can be categorized into six groups. We discuss each of these groups individually, and present the advantages and disadvantage of using them. Also, a few examples are given for each group. Using different overlay topologies (peer organization strategy) is the only enhancement that we did not discuss in details within this chapter. Our aim is to study only the enhancements that directly affect the content distribution models behavior, and make them suitable for heterogeneous network.

2.2 Complete File Store-and-Forward

Traditional client/server file distribution systems depend on the store-and-forward file distribution mechanism, where the file needs to be completely uploaded first from the sender to the server; then the receivers will start downloading the file from the server.

Although store-and-forward doesn't depend exactly on Chunk or Fluid models for its distribution mechanism, if we look at it, we can consider it either Fluid or Chunk model. It is a Fluid model where the file is sent completely and

continuously in a flow stream. On the other side, it can be a Chunk model if we suggest using one chunk, which is the whole file size.

Combining complete file store-and-forward with peer-to-peer networks was first motivated by old random Gossip models (Frieze & Grimmett, 1985; Pittel, 1987), as part of the work in distributing file's data in unstructured networks. Initial data dissemination in unstructured networks approaches (Alan et al., 1988; Karp, Schindelhauer, Shenker, & Vocking, 2000) advocated uploading the whole file at one go. This involves users in random gossip model receiving the complete file and then uploading it to other users chosen at random.

Complete file store-and-forward could also be found in distributed network file systems. Project (Siegel, Birman, & Marzullo, 1990) is a distributed file system that focuses on file semantics in relation to efficiency, scalability, and reliability. The system uses servers that are interchangeable and collectively provides the illusion of a large, single server to its clients. Replicas of files are stored on a subset of file servers which are then forwarded to users.

Depending on Complete file store-and-forward mechanism in multi-servers is considered a slow process and doesn't fully utilize the server's resources, especially when the number of distribution servers increases.

However, for large files, making users wait to receive the entire file before they can start serving it becomes untenable for two reasons: (i) file transfer may take a long time, and during this time the upload capacity of downloading users is wasted,

and (ii) users who have received the file may depart before uploading a complete copy, resulting in the complete file being lost to others.

2.2.1 Multipoint File Transfer System (MFTS)

The existing Multipoint File Transfer System (MFTS) is an ideal platform for offering synchronous and asynchronous file distribution, mainly for sharing small and medium file sizes, which is suitable for real-time collaborative environments (Noori, Sureswaran, Budiarto, & Rao, 2004; Noori, Sureswaran, & Rao, 2004c).

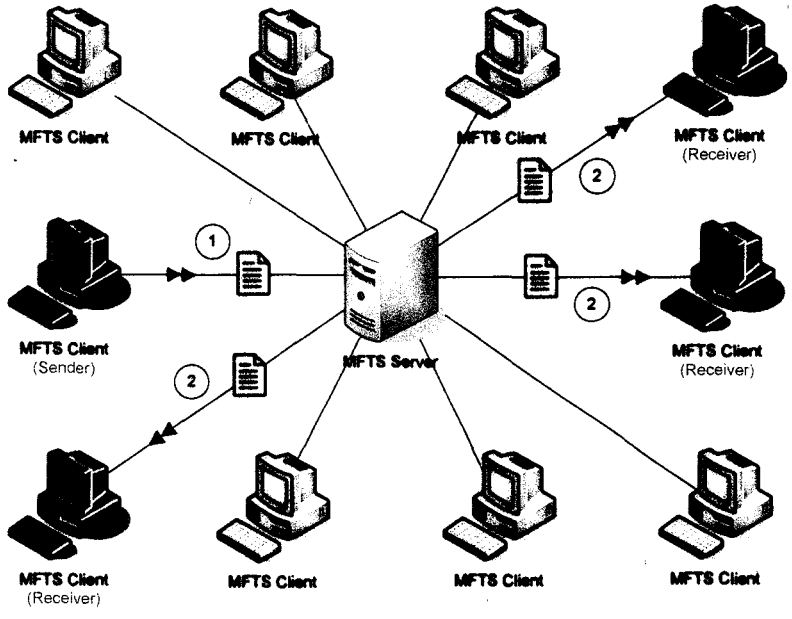

Figure 2.1: Multipoint File Transfer System (MFTS).

MFTS (Figure 2.1) is implemented on a reliable Data Size Dependent File Transfer (DSDFT) protocol (Noori, Sureswaran, & Rao, 2004a). This protocol depends on variably-sized data packets, where packet sequence numbers are derived based on the variation of the data size. This method allows for a more efficient use of the available bandwidth, by removing the sequence number portion that is used

within protocols like FTP. The MFTS design and architecture resolved certain problems inherent in file distribution systems, such as: (i) providing a connection-secure data channel, (ii) enabling clients who join the file distribution group late to download all or selected files, both previously and currently distributed (commonly known as the join-late problem), and (iii) keeping all clients up-to-date with the original file being transferred. This is achieved by means of characterizing each transferred file with its original modified date in a universal time format.

The existing platform of MFTS is based on client-server (single server) architecture. Because of this, it suffers from a scalability bottleneck problem. As the outgoing bandwidth of the server is shared among all concurrent clients, the more the clients, the less bandwidth each client can have. Hence, the performance of this approach deteriorates rapidly as the number of simultaneous clients increases.

2.2.2 Large File Distribution within CDNs (FastReplica)

FastReplica (Cherkasova & Lee, 2003) addresses the problem of reliable and efficient file distribution in content distribution networks (CDNs) (e.g. Akamai (Akamai, 1998; Chao, 2006)), which is an infrastructure based network, that employs a dedicated set of machines to reliably and efficiently distribute content to clients on behalf of the server.

FastReplica focuses on distributing large size files such as software packages or stored streaming media files (also called on-demand streaming media), by considering a geographically distributed network of servers and a content distribution across it. The system is based on a large-scale distributed network of servers located

closer to the edges of the Internet for efficient delivery of digital content including various forms of multimedia content. The main goal of the CDN's architecture is to minimize the network impact in the critical path of content delivery as well as to overcome a server overload problem that is a serious threat for busy sites serving popular content.

FastReplica proposed a novel algorithm, for efficient and reliable replication of large files. In order to replicate a large file among n nodes (n is in the range of 10-30 nodes), the original file is partitioned into n subfiles of equal size and each subfile is transferred to a different node in the group. After that, each node propagates its subfile to the remaining nodes in the group, as shown in Figure 2.2. Thus, instead of the typical replication of an entire file to n nodes by using n Internet paths connecting the original node to the replication group, FastReplica exploits $n \times n$ diverse Internet paths within the replication group where each path is used for transferring $1/n$-th of the file.

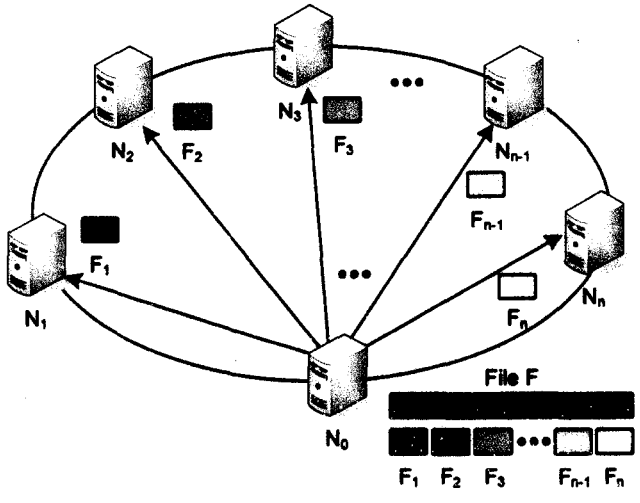

Figure 2.2: FastReplica in the small: distribution step.

In the basic algorithm for FastReplica, nodes are organized into groups of fixed size (n), with full group membership information at each node. To distribute the file, a node splits it into n equal-sized portions, sends the portions to other group members, and instructs them to download the missing pieces in parallel from other group members. Since only a fixed portion of the file is transmitted along each of the overlay links, the impact of congestion is smaller than in the case of tree distribution. However, since it treats all paths equally, FastReplica does not take full advantage of high-bandwidth overlay links in the system. Since it requires file store-and-forward logic at each level of the hierarchy necessary for scaling the system, it may not be applicable to high-bandwidth streaming. Also the latency observed by the end-users is still high as the client will not be able to receive the file until the server completely receives the whole file.

2.3 Chunk Model with Scheduling

In practice, a shared file or video stream is divided into multiple chunks. To increase the availability of these chunks, a good data distribution scheduling algorithm is needed (Chan, Li, & King-Shan, 2005; Dongni, Li, & Chan, 2008; Guo, Liang, & Liu, 2009; Jochen Mundinger, Weber, & Weiss, 2008). This is the core of any file sharing or P2P video streaming system, since without actual data transmission and distribution, no data sharing is possible. A scheduler tells each peer which chunk should be sent and to whom. A poor data distribution scheduler could result in considerably longer download time, while a good scheduler could shorten the completion time and efficiently utilize all resources, like network bandwidth.

Experiments, such as those in (Rodriguez & Biersack, 2002), have shown that using parallel downloading scheme in P2P file sharing systems, in which an end user opens multiple connections with multiple file sources to download different portions of the file from different sources and then reassembles the file locally, could result in higher aggregate download rate and thus shorter download time. The multiple connections within the parallel download scheme need a good scheduler algorithm, to get significant performance improvements in collaborative file sharing.

Before proceeding to the scheduling examples, we have to distinguish two models for determining the transmission: *Pull-based* and *Push-based* (Meng, Jian-Guang, Li, & Shi-Qiang, 2005; Sanghavi, Hajek, & Massoulie, 2007). In both models, peers have to exchange their file or video stream piece possession information periodically. They differ in how to make the decisions of which piece to send and to whom.

Most existing P2P applications use the *Pull-based* model, in which the receiver determines which file pieces he needs from others and subsequently sends request messages to the nodes he chooses. The file source who receives these request messages could choose to accept or reject the requests based on some policies, such as his available bandwidth and the requestors' contributions. While in *Push-based*, a peer determines which piece it should transmit and which peer it should send the piece to. This approach of scheduling was first adopted by (Ma & King-Shan, 2008).

In a large distributed cooperative system, finding an optimal chunk propagation scheme that minimizes the client download time is very difficult. This is

especially the case in practical systems that cannot rely on a central scheduler and, instead, allow nodes to make local decisions. The scheduling problem becomes increasingly difficult as the number of nodes increases. When nodes are at different stages in their downloads, or when incentive mechanisms are introduced to prevent leeching clients.

The *Pull-based* model is commonly used in existing applications, such as BitTorrent, in which the receiver determines which file pieces he needs from others and subsequently sends request messages to the nodes he chooses. The file source that receives these request messages could choose to accept or reject the requests based on some policies, such as his available bandwidth and the requestors' contributions. One disadvantage of this model is that there will be many short-length but frequent request messages flowing through the network, taking up network bandwidth and processing time. In addition, it may happen that all peers decide to request the same file piece from the same source, thus wasting queuing time at the source node (or even getting rejected by the source node).

Finally, there are some drawbacks or complexity with some scheduling algorithms, to achieve a valid scheduler: (i) At least one file chunk must be distributed among the peers in each cycle, if the bandwidth available for uploading or downloading is not completely utilized, leading to increased number of cycles to complete the file sharing. (ii) Weights assigned to the nodes for deciding the maximum flow should not remain constant throughout the cycle. When choosing recipients, those peers who have been assigned to receive something may still be

further assigned to receive more as their static weights remain as the highest, resulting in unfair resource allocation.

2.3.1 Rarest Piece First (RPF)

The RPF algorithm concentrates on the piece selection strategy. It identifies the rarest file pieces and tries to increase their availability in the network by transmitting them first. By doing so, the source peers for the rarest file pieces are increased and will help the peers to continue file sharing even if one of the source peers fails. Hence, RPF inherently distributes all pieces from the original source to different peers across the network as quickly as possible, such that the distribution can continue even if the original source leaves.

The Rarest Piece First algorithm is borrowed from the Rarest Element First algorithm employed in BitTorrent. In RPF, those file pieces that most peers do not have (rarest) are distributed first.

BitTorrent (BitTorrent, 2004; Cohen, 2003) is one of the most popular P2P file sharing applications with thousands of simultaneous users. A shared file is chopped into multiple small pieces (each about 256KB or 512KB). Some tracker servers are used to periodically announce the list of connected peers who participate in the same sharing session. Each peer then uses this peer list to contact other peers and report to them which pieces it currently possesses. It also requests those missing pieces it does not have from those peers who have them. A peer can maximize its downloading speed by requesting different pieces from different peers at the same time.

A poor scheduling algorithm may lead to every peer getting nearly the same set of pieces and consequently decreases the number of file piece sources which a peer can simultaneously download from. BitTorrent employs the Rarest Element First algorithm, in which those pieces that most peers do not have are downloaded first. This algorithm is good at increasing the availability of different file pieces in the network and can distribute all pieces from the original source to different peers across the network as quickly as possible.

RPF aims at increasing the availability of different file pieces in the network, such that peers may still have some pieces that other peers want. In case the file is published by a single source who may just seed (remain available to contribute) the file for a short period of time, RPF also tries to distribute all pieces from the original source to different peers across the network as quickly as possible, so that the distribution can continue even if the original source leaves.

2.3.2 Most Demanding Node First (MDNF)

The MDNF algorithm concentrates on the peer selection strategy. It identifies the peers with most missing pieces and tries to fulfill its demands first. This strategy brings down the overall download time of file pieces since the peers with highest demands are satisfied first. When the peer with highest demands is satisfied, the available copy of those pieces is increased by one (Jonathan, Victor, & King-Shan, 2007).

To fully understand the modeling of RPF and MDNF, we can follow below example for the aim of reducing the average distribution time by depending on both RPF and MDNF (Ma & King-Shan, 2008).

In the example, there are N peers forming an overlay network, distributing a file F that is divided into M pieces of equal size. F can be regarded as a set of pieces, i.e. $F = \{f_1, f_2, \dots, f_M\}$, and each peer has only a subset of F. They build an $M \times N$ matrix to store the piece possession information. For $P_{ij}(1 \leq i \leq N,\ 1 \leq j \leq M)$ in the matrix, before the file distribution starts it can take on either one of the two values: $P_{ij} = 1$ if peer i has piece j; $P_{ij} = 0$ if peer i does not have piece j. An example possession matrix is shown below:

$$P_0 = \begin{pmatrix} 0\,0\,0\,0\,1\,0 \\ 1\,1\,0\,0\,0\,1 \\ 0\,1\,1\,1\,1\,1 \\ 0\,0\,0\,1\,1\,0 \end{pmatrix}$$

Each peer maintains its own possession matrix. Each peer acquires the initial possession matrix and updates from other peers. Usually other peers broadcast their possession information so that each peer can build the initial possession matrix on its own.

Based on the possession matrix, piece rarity and peer demand are developed. The piece rarity indicates how rare the piece is and is quantified as how many peers need this piece. In other words, the rarity of piece f_k is the number of zeros across column k in the possession matrix. On the other hand, the demand of peer i refers to how many pieces it needs, which is quantified as the number of zeros across row i.

Consider the example possession matrix P_0. The rarity of piece 1 is 3 and the demand of peer 3 is 1.

In RPF protocol, once peer i makes a scheduling decision, it proceeds to contact its prospective piece recipient. If the recipient accepts the offer, piece transmission starts. Otherwise, peer i drops this scheduling decision and seeks other scheduling alternatives. Assume peers 1 and 2 in P_0 agree on the delivery of piece 5. After the transmission starts, p_{25} is updated to $1/2$ to mark the ongoing transmission. p_{25} is changed to 1 when the transmission is completed.

2.3.3 Maximum Flow (MaxFlow)

The MaxFlow algorithm concentrates on two factors that affect file distribution in P2P network. This algorithm takes into consideration the rarity of the file pieces and also the demands of the peers, and assigns weights to the node. A bipartite graph is formed with all the peers in two sets. One set being the file piece senders (L), and the other set being the file piece receivers (R). Edges are then formed between the two sets based on which file pieces can be sent from (L) to (R). Then the problem instance is transformed into a flow graph problem. Once the flow graph has been constructed, the maximum possible flow is calculated, which decides the number of transmissions that are to take place in that cycle. Hence based on the maximum matching the schedule for the cycle is formed (Jonathan et al., 2007).

The complexity of this algorithm is $O(N^2M \times \min\{\sum_{i=1}^{N} pi, \sum_{i}^{N} qi\})$ where N is the total number of peers, M is the total number of file pieces, pi and qi are the upload and download capacities of the peers. Since the MaxFlow algorithm transmits

the maximum possible files in each cycle, the number of cycles taken to download the files pieces by all peers is the least. Hence this algorithm gives the best possible performance. The problem instance has been transformed to the well-known maximum bipartite matching problem in order to find as many sender and receiver pairs as possible in each cycle. Weights are added to the nodes to achieve better matching.

2.3.4 Gossiping with Multiple Messages

The underlying motivation of random gossip protocol (Sanghavi et al., 2007), is the design and analysis of piece selection protocols for peer-to-peer networks, which disseminate files by dividing them into pieces. They first investigate one-sided protocols, where piece selection is based on the states of either the transmitter or the receiver. They found that any such protocol relying only on pushes, or alternatively only on pulls, will be inefficient in disseminating all pieces to all users.

Gossiping with multiple messages proposed a hybrid protocol (INTERLEAVE) to investigate both *one-sided* and *two-sided* piece selection protocols. For either one of the two protocols, the user needs to make a piece selection. This piece selection is said to be *one-sided* if it is based only on the user's own current state, and not that of the target. The piece selection is said to be *two-sided* if it is based on the current states of both the user and the target. In either case the selection is independent of system history or the states of other users. Different ways of making this choice correspond to different protocols.