An Asynchronous Collaboration and Content Management Framework with High-Performance P2P-based Data Transfer Capability for Scientific Computing

Ali Kaplan

Submitted to the faculty of the University Graduate School
in partial fulfillment of the requirements
for the degree
Doctor of Philosophy
in the Department of Computer Science,
Indiana University
July 2007

Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Doctoral Committee

_____
Prof. Geoffrey C. Fox (Principal Advisor)

_____
Prof. Dennis Gannon

_____
Prof.

_____
Prof.

28, 2008

Acknowledgements

Abstract

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

“There is nothing permanent except change.” –Heraclitus of Ephesus. This is very true for nearly seventy years of history of the computing systems and computational science. When the first modern computers emerged in 1940, computer systems were so large and expensive that they had a roomful size and only a few very big companies and institutions were able to afford them. These computer systems operated independently and did not communicate with each other. In fact, there was no reason to connect them at that time.

Throughout the 1950s, computers used vacuum tubes as their electronic components. Transistor-based electronics were used to replace vacuum tube electronics in the 1960s. In the 1950s and early 1960s, one prevalent computer networking method was built on the central mainframe method in which terminals were connected to their central mainframe via long leased lines.

A groundbreaking technology, integrated circuit (IC) technology, and creation of microprocessors were introduced in the 1970s. IC technology and microprocessor

architecture resulted in manufacture of smaller, faster, more reliable and cheaper computers. Modern computers based on ICs are millions to billions of times more capable but a fraction the space and price. Another important advancement of the late 1960s and early 1970s was the development several packet switched networking solutions to address the network interoperability problems.

By the 1980s, computers became so affordable, small and simple that regular people were able to afford to buy computers for their personal purposes and used at their homes. Nowadays, computers are ubiquities. For example, embedded computers are used in machines ranging from spaceship to industrial robots, digital cameras, children's toys, and wristwatch. Following the introduction of privately run Internet Service Providers in the 1980s, the Internet became prevalent which led to invention of high speed computer networks with multitude of connected machines.

In last seventy years, the advancement in computers and computer networks not only changed their size, speed, reliability and price but also shifted their main usage paradigm from computational-intensive to data-intensive [1]. New scientific devices such as large-scale observatories and state of the art microscopes generate massive volumes of datasets. The Internet and computational Grid [2-5] make all these archives accessible to anyone anywhere, allowing the replication, creation, and recreation of more data [6]. People interested in to analyze the data sets are geographically dispersed as well.

The scientific disciplines with the above characteristic are as diverse as high-energy physics and bioinformatics. To illustrate, petabytes of data generated by the Large Hadron Collider (LHC) experiment at CERN are required to be distributed

worldwide. Another example is that The Pacific Northwest National Laboratory is building new Confocal microscopes with enhanced capabilities. High resolution video of the subject samples, which are typically protein molecules, are provided by these microscopes of which typical use of is multidisciplinary, requiring the data stream to be multicast to multiple scientists at multiple remote institutions. Within 6 months, 625 Mbps of data rates are expected per microscope. In addition to the microscopes, there is a proteomics simulation program that will be generating 5 petabytes per year, 5 years from now [7].

Consequently, bulk data transferring and management have become one of the immensely popular research fields in computational science. Particularly, for WAN type of computer networks, a lot of work and solutions have been proposed. At the beginning, some of these works had only concentrated on data transmission aspect [8]. Solutions for data catalog and management drew researchers' attention later. However, proposed systems with regard to data catalog and management were employed for only the data location discovery and they may not be functioning as collaboration framework.

This dissertation is motivated by understanding the need for a *Peer-to-Peer based High Performance Data Transfer Framework* that effectively combines "*Data management and collaboration services"* and "*SOA principles*"; evaluating the system design parameters in terms of simplicity, scalability, security, performance, and platform independency; and evaluating how these factors influence the overall infrastructure.

1.1 Motivation

The number of the Internet users greatly increased with the dramatic expansion of the Internet over the past few years. As a consequence, the high-performance networks with low-cost, powerful computational machines were proposed, which led to the creation of new distributed computing infrastructure termed Computational and Data Grid, which is the base of large-scale distributed computing systems by interconnecting geographically distributed computational resources via very high-performance networks [9]. High performance data transfer technique with collaboration and data management is required for handling widely geographically dispersed data resources; however, existing solutions have shown limitations since most of their systems and unpublished protocols are incompatible with each other and they are platform dependent solutions [8].

Firstly, existing solutions were built on the client-server architecture that makes them vulnerable to its disadvantages such as central server failure and bottleneck problem under heavy load. In addition, most of the existing solutions focus on the problems of aggressive high performance data transfer rate and do not take into consideration of collaboration and optimum use of resources seriously. Moreover, some of these techniques are data type centric solutions, which hinders them to employ systems with different data type. With the ability of handling any data type, data transfer techniques based on peer-to-peer network might be remedy to this situation by making feasible effective resources (e.g.. network, CPU, storage) usage by exploiting unutilized resources; however, very few efforts have yet been devoted to harness fully peer-to-peer network

Secondly, existing data management systems are separate and heavy systems and they are tightly-coupled to their underlying data transfer mechanisms. Administrator tasks which requires a great deal of the knowledge have been mainly been performed by people. There is a need for simple, lightweight data management frameworks with the minimum administrator tasks' requirements.

Thirdly, existing data management systems lacks collaboration feature. As the large datasets are generated and users of them dynamically distributed, sharing, discovering, and transferring of these datasets are become more problematical. To this end, we see a greater need for a collaboration framework that is very valuable and needed for geographically dispersed scientific communities to escalate scientific research output.

Finally, as it is evident that Service Oriented Architecture (SOA) [10] and its current implementation, Web services [11], will have profound impact on the next generation of distributed systems, it would be great opportunity to investigate challenges and benefits of exploiting SOA in a high performance bulk data transfer service.

1.2 Use Cases

Organizations have resources (e.g. CPU power, network bandwidth capacity, disk performance) with widely varying characteristics. These characteristics determine the requirements for data management and transferring. We give a description of two sample use cases that are explanatory of the applicability of our proposed system.

Replication: Replication is the process of copying data from one location to another so as to ensure consistency between redundant resources to improve reliability,

fault-tolerance, or accessibility. Data transfer techniques based on peer-to-peer network architecture would convert machines which download data from master-source into active sources and enable to utilize their unused resources, particularly their network bandwidth capacity.

Data sharing in systems with varying CPU power and network capacity: Not all organizations have opportunity to have the super-powerful machines and very high-performance networks, yet they still desired to share their research results with geographically distributed participants. In fact, aggregation of average personal computer CPU power yields an astonishing computational power. Seti@home [12-14] and Genome@home [14-16] are the well-know projects tapped into this power. Similarly, aggregating parallel data streams in modest high-performance network produce very high performance data transfer capability. Almost all high-performance data transfer techniques utilize parallel data streaming in order to boost their performance. However, a peer-to-peer system, Bittorrent, has outperformed GridFTP, the de facto data transfer mechanism in many scientific community, in network areas where only limited bandwidth is available [17]. Thus, our lightweight system might perform well both on very high-performance networks and networks with limited bandwidth and can be deployed on any type of platform.

1.3 Research Issues

In this thesis, we describe the architecture design and implementation of a peer-to-peer based high performance data transfer technique with data management and collaboration framework. We have thoroughly analyzed the system to determine how the system would respond and have presented benchmarks on different number of

download clients. A major goal of this thesis is to provide a simple, lightweight data transfer framework from which any size of scientific community would benefit.

We now summarize the research issues we plan to address in this dissertation:

1. How can we build a peer-to-peer data transfer mechanism which utilizes SOA for scientific community? Which one of available peer-to-peer system is best for this purpose and what type of modifications and new features are needed to be added to it?
2. How can we provide a medium that allows participants to manage, share, discover, and download their contents and integrate it with data transfer mechanism?
3. Is the data transfer mechanism scalable?
4. How is the performance of data transfer mechanism and it is acceptable?
5. What is the overhead of this system and is it reasonable?
6. Security is not a concern in peer-to-peer to networks for non-scientific community. How can we make it enough secure for scientific community?

1.4 Contributions

In this thesis, the expected contributions can be summarized as following. We will identify a novel system for Collaboration Framework for scientific community. We will also define the minimum set of requirements to build Collaboration Framework dedicated to share and cooperate on users' data with given access control rights. We will identify the additional features needed to be added into selected best practice of peer-to-peer systems.

Existing data transfer techniques focus on the problems of aggressive high performance data transfer rate and do not take into consideration of collaboration and optimum use of resources.

There are some systems used for the data location discovery and they may not be functioning as collaboration framework. We are going to look into for determining the features for collaboration framework for science environment.

Available high performance oriented data transfer techniques for scientific computing are broadly categorized FTP and HTTP based techniques. GridFTP [8] is an example of FTP based data transfer means. Due to implementation in C, it is not portable and is deployable only Unix/Linux installed machines. However, GridHTTP [18] is implemented by adding new features like authentication and authorization to HTTP. There are some other techniques; however, they use GridFTP as underlying data transfer mechanism.

Peer to peer network structure providing optimal usage of resources is another data transfer way. It is utilized as a data transfer method for non-scientific community to share mainly video, mp3, and games files. We can also investigate existing peer to peer systems and try to find which of them are more suitable and in addition which features are needed to be implemented and added to selected peer to peer system to meet the requirements of scientific data transfer.

In designing our architecture, we have identified following requirements to build a Collaborative Framework with High Performance Oriented Data Transfer Technique for Scientific Computing. First, the architecture should have a medium to let users to publish and subscribe to their contents. Second, the architecture should allow users to

create a group or friend list in which they can add other user. Third, the system should enable users to set access control rights to a specific content for a specific user or groups due to great sensitivity attribute of scientific data. In order to provide a Collaboration Framework satisfying the requirements stated above, there are several issues that we will study and research in building our architecture.

1.5 Organization of the Thesis

This thesis entails nine chapters. In the next chapter, we present briefly related work information on this area, for instance, GridFTP, RFT, and prominent peer-to-peer technologies. We will also give the overview of BitTorrent algorithm because it inspired us to implement our data transferring mechanism.

In chapter three, we give an overview of the overall architecture and explain shortly the main components of the system to provide a clear understanding of whole system.

In chapter four, we present the architecture of the first major component of the GTF, the GTF Client. The GTF Client is responsible for actual data sharing and transferring tasks. It has modular and layered structure so that it can be enhanced and maintained easily. Each of it modules and layers are discussed in detail in this chapter.

The details of Collaboration and Content Manager Module which is of paramount importance to data sharing and participant cooperation are given in chapter five. Users can publish –make their contents available to other users- their contents and subscribe –content download process- to available contents through CCM. Similar to the GTF Client structure, it has modular structure. At the beginning of the chapter, the

big picture of architecture is presented to demonstrate the general idea and principles. Its modules and layers are explored in the remainder of the chapter.

The WS-Tracker is very essential part of the GTF and at the center of the whole system. It is major role is to orchestrate communications taking places among the GTF Clients through sending and receiving meta-data. Since our WS-Tracker is designed to serve in scientific community, it needs to satisfy their conditions such as security and, access. As a result, it is very different and more advanced than BitTorrent's regular Tracker. Its architectural details and differences between them are explained in chapter six.

In chapter seven, we present the security issues and its implementation in the GTF. There are different security requirements at different communications happening between above-mentioned components. It is very difficult to deploy all the security protocols, inasmuch as it is availability of myriads of choices. Consequently, we had to choose some of them. We selected the Grid Security Infrastructure (GSI) not only the Grid community fits our targeted user profile, but also it is the largest scientific community in the world. It is a proven and common standard and its participants are exceptionally diverse and geographically dispersed. At the beginning of chapter seven, we give short and historical overview of the GSI. How the GSI and other security schemas are utilized in the GTF components are explained in the remainder of the chapter.

We introduce our prototype test results in chapter eight and provide detailed analysis for them. The tests were conducted in two types of computer networks: (1) LAN; (2) WAN. We have three different sets of scenarios. These tests and different

scenarios help us determine where the system is performing well and very useful and what the limits of the system in terms of the maximum number of data seeders and clients that can be supported without a significant loss in performance.

Finally in chapter nine, a very brief précis of thesis with the overall lessons learned from our study are provided and answers are given to the research questions identified in chapter one. At the end of this chapter, we provide concluding remarks, the contribution of this thesis and outline future research directions.

Chapter 2

Literature Survey

2.1 Introduction

A basic computer system consists of three fundamental services: data processing, data storing, and data transferring. Although all of these services are of equal importance, data transferring is the most appealing, functional and active one among three of them since it renders the data meaningful and useful by moving it from one place to another. The distance between the source and the destination of the data which will be transferred ranges from few millimicrons to a few terrameters [19]; hence, application fields of data transferring are much more diverse than that of others.

It is natural that each system has widely varying characteristics with respect to data and conveyance of data such as size, importance, and security of data and medium

of transmission; therefore, these characteristics require different methods and apparatus for the data transmission. For instance, the size of data at Level1 cache is only few KB and data loss probability is nearly impossible while transferring data from Level1 cache to central processing unit (CPU). The transmission time only takes few CPU cycles. On the other hand, transferring data from one computer to another one on wide area network may take seconds to hours or maybe days depending on the size of data, capacity of bandwidth, and physical distance between source and destination.

Another extreme example is data transmission from Voyager 1[19] to Earth. The Voyager 1 spacecraft, launched on September 5, 1977, is a robotic space probe that was sent for an expedition to Jupiter, Saturn, Uranus, Neptune and the outer solar system and beyond. It is over 16 billion kilometers from the Sun as of May 9, 2008. It returns its data through the Deep Space Network (DNS) with X band transmitter that provides downlink telemetry at 160 bit/s normally and 1.4 kbit/s for playback of high-rate plasma wave data. It is expected that Voyager 1 will continue to return valuable data until at least 2025 [19-21].

Another important example of data transfer technique is utilized in wide-area-network (WAN) environments and being part of this thesis subject as well. This type of data transfer has been gaining great importance because of the dramatic expansion of the Internet in recent years, availability of low-cost high-performance powerful computational engines, production of huge amount of data –either raw or processed– by scientific gadgets, and interconnection of geographically distributed computational resources via very high-performance networks [22]. These factors have given rise to development of a new set of technologies termed Computational Grid [22]. The

appearance of Computational Grid resulted in dramatic increase in state-of-the-art high performance distributed applications.

For high volume data transfer in WAN type of computer networks, a lot of work and solutions have been proposed. At the beginning, some of these works had only concentrated on data transmission aspect [8]. Solutions for data catalog and management drew researchers' attention later. However, those systems the results of studies in the area of data catalog and management are utilized for only the data location discovery and they may not be functioning as collaboration framework.

Although some of these existing data transfer techniques may be considered successful, most of their systems and unpublished protocols are incompatible with each other and they are dispersed solutions [8]. Moreover, they focus on the problems of aggressive high performance data transfer rate and do not take into consideration of collaboration and optimum use of resources seriously. In addition, they failed to harness fully newly emerged technology such as peer-to-peer network. As a result, there is still a need for new solutions to address above requirements.

The objective of this thesis is to leverage data transferring mechanism in distributed systems for achieving fast and economical data transmission by exploiting peer-to-peer protocol -BitTorrent algorithm- and providing collaboration framework. In this chapter, we present an overview of the various strategies relevant to our work.

2.2 Overview

Data-intensive applications have gained widespread attentions and become more prevalent with the infrastructure provided by computational grid, and in consequence there is a growing need for the efficient management and transfer of

information, in terabyte-scale or even petabyte-scale, in wide-area computing environments [23].

There are many aspects that can be used to categorize data transfer techniques; the place where they operate, the characteristics of their communication over an IP structure (e.g., point-to-point, multicast, peercasting), their design architecture (e.g., client/server, peer-to-peer), and the characteristics of data that they used for to name a few. Since the operating place is the relatively broader and the most accepted perspective, we, as well, categorized them with respect to their operating places.

Data transferring techniques can be deployed at network-level, system-level, or application-level. The data movement technologies at network-level, as the name implies, are network-based solutions and they are categorized into NAS and SAN [24]. There are certain similarities between system-level and application-level data transfer techniques; both of them involve endeavors in order to overcome the limitations of Transmission Control Protocol (TCP) [25]. However, they adopt different approaches to cope with the limitations of TCP. The system-level solutions usually include modifications to the operating systems of the machine, of the network apparatus, or of both [25]. These techniques sometimes require development of new versions of TCP such as Selective Acknowledgment TCP [26], High Speed TCP [27], and Scalable TCP [28]. In order to accomplish some of those modifications, the privilege of system's super user is required.

Unlike application-level solutions, both network and system-level solutions can yield very good performance. In exchange, they usually require substantial costly upgrades/updates of the networking structure, and considerable system-level

modifications. In spite of their better performance, they suffer from the distance limitations and high network cost.

Application-level techniques, on the other hand, attempt to overcome the limitations of high bandwidth delay product networks by using software based techniques such as exploiting parallel streams, increasing TCP window size when it possible, or employing rate-based control algorithms [25]. Application-level solutions are based on either TCP (Transmission Control Protocol) or UDP (User Datagram Protocol) [25]. They have much broader use because they do not require any system or network level modifications or upgrades to deploy them.

After our extensive survey of the data movement techniques, we must acknowledge that there has been a tremendous amount of research with regard to the development of application level data transfer protocols; thus, it is impossible to cover all types of data transfer techniques and solutions in spite of our best efforts.

2.3 System-level Data Movement Techniques

The system-level solutions usually include modifications to the operating systems of the machine, of the network apparatus, or of both [25]. Group Transport Protocol for Lambda-Grids (GTP) [29] is a good example of system-level data delivery techniques. The development team of GTP focused on achieving high performance in complex network structures in lambda-grids [29]. Multicast data delivery and shifting the rate and congestion control to end-points were their motivations to develop receiver-driven transport protocol [29]. Even though, it is a software package, operating on TCP and UDP, it is specific for Lambda Grids and requires very high speed dedicated links. A Lambda-Grid is a set of distributed resources directly connected with Dense

Wavelength Division Multiplexing (DWDM) links [29]. DWDM is a technology that multiplexes multiple optical carrier signals on a single optical fiber by using different wavelengths of laser light.

2.4 Application-level Data Movement Techniques

Having communication links cover a large geographic area subjects the wide-area networks (WAN) to very high round-trip latencies. It is because of this that they are often termed high Bandwidth Delay Product (BDP) network [25]. TCP is the most widely used protocol in the Internet and de facto data transmission protocol on any type of computer networks including WAN for reliable data movement. However, it substantially underutilizes network bandwidth over high-speed connections with long delays because TCP employs its window size as the congestion control techniques [29] in order to impose a limit on the amount of data it will send before it waits for an acknowledgement [25]. This is because traditional TCP and its variants were developed for shared networks where the bandwidth on internal links is a critical and limited resource. Hence, as stated in [29], accomplishing high performance data transfer in high BDP networks is a long-standing research challenge for point-to-point data transfer.

2.4.1 **TCP -based Data Movement Techniques**

All TCP-based data transfer techniques use TCP connections to overcome TCP's window size problems by using parallel streams. In this method, an aggregated congestion window is acquired so as to be able to fully utilize the available capacity

provided by the high BDP network. In other words, the larger congestion window size, the higher the throughput [30].

As the classic File Transfer Protocol [31] (FTP) is the most common protocol used for bulk data transfer on the Internet [32]. It is a well-understood IETF standard and widely implemented protocol, and it supports dynamic discovery of the extensions [22]. Most TCP-based Data transfer techniques, therefore, are the derivations of the classic File Transfer Protocol (FTP); for instance, GridFTP [32] and bbFTP [33]. The Secure Copy Protocol [34] (SCP) is another data transfer tool provided by Unix/Linux based operating systems. The Babar Copy Program [35] (bbcp) is an example of data movement technique implemented based on the peer-to-peer architecture.

2.4.1.1 GridFTP

GridFTP is a common data transfer and access protocol that extends the standard FTP protocol. The standard FTP protocol does not meet the key features necessary to Grid applications such as advanced security support, third-party control of data transfer and striped data transfer. In order to make GridFTP a high-performance, secure, reliable data transfer protocol [32, 36, 37], the GridFTP development team has defined new extensions to enhance the standard FTP by providing new features. The GridFTP protocol includes the following features that are new extensions to the standard FTP:

- **Grid Security Infrastructure (GSI) and Kerberos support**: Security is one of the crucial features required in Grid computing when transferring or managing files. To meet the security requirements, GridFTP implemented the GSS API extensions defined in RFC 2228 (FTP Security Extensions) [8, 32, 38] in order

to support GSI and Kerberos authentication, with user controlled setting of various levels of data integrity and/or confidentiality[36, 39].

- **Automatic negotiation of TCP buffer (window sizes)**: The performance of data transfer in wide area networks can be improved significantly by using optimal settings for TCP window sizes. However, manually setting TCP window size is not an easy operation since it requires super user privileges to perform it. Therefore, in order to support both manual setting and automatic negotiation of TCP buffer sizes for large files and for large sets of small files, GridFTP extends the standard FTP command set and data channel protocol [36].
- **Third-party control of data transfer**: Authenticated third-party control of data transfers between storage servers is requisite in order to manage large datasets for distributed communities easily. By adding GSSAPI (Generic Security Services Authentication Programming Interface) security to the existing third-party control of data transfer, a user or an application at one site can initiate, monitor and control a data transfer operation between storage servers [22, 36].
- **Parallel data transfer**: Aggregating bandwidth by using multiple TCP streams in parallel (even between the same source and destination) improves high performance data transfer in high BDP networks [36]. Through FTP command extensions and data channel extensions, GridFTP supports parallel data transfer not only from a single server but also from multiple servers as well [40].
- **Striped data transfer**: Besides using multiple TCP streams in parallel, striping or interleaving data across multiple servers may be used to provide further bandwidth improvements, as in a DPSS network disk cache or a stripped file

system [36]. Striped transfers allows portions of the data to come from different servers. In other words, striped data transfer permit having multiple network endpoints at the source, destination, or both when the same file is transferred among them.

- **Partial file transfer**: Transferring portions of files rather than complete files could be beneficial for some applications: for instance, high-energy physics analyses that require access to relatively small subsets of massive, object oriented physics database files [36]. GridFTP supports the capability by specifying the byte position in the file to begin the transfer [36].
- **Support for reliable and restartable data transfer**: Due to distributed nature of Grid computing, reliable transfer and fault tolerant features are of great importance for many applications that manage data. Fault recovery methods are needed to handle failures such as transient network and server outages. GridFTP exploits these features and extends them to cover the new data channel protocol [36].

Although GridFTP has many good features and impressive data transfer performance, in our opinion, it still suffers from some drawbacks that stemmed from the nature of the standard FTP and TCP. GridFTP team has been devising and adding new features or systems on top of the GridFTP to circumvent these problems. For instance, Reliable File Transfer [41-43] (RFT) was developed to provide reliability in the face of local failure. When the client loses its state, transfer process has to restart [22]. In addition to RFT, Globus-url-copy [44] and UberFTP clients are other well-know clients of GridFTP. TeraGrid Copy [45] (TGCP), designed for taking full

advantage of a 10 or 30 Gb/s network link for an individual file transfer, is a wrapper over globus-url-copy and RFT in order to provide a SCP-style GridFTP interface to users [45]. The Replica Location Service (RLS) [46, 47] is design for creating and managing multiple copies of files by providing a framework for tracking the physical locations of data has been replicated. At its simplest, RLS maps logical names to physical names, and it is intended to be used in conjunction with other components like RFT service, GridFTP, the Metadata Catalog Service, and reliable replication and workflow management services [48].

Another problem is the performance of GridFTP servers which suffers drastically if the dataset is large but consists of many small files (smaller than 100 MB), known as the “lots of small files (LOSF)”. This poor performance is because of the command/response semantics of the RFC959 FTP protocol. It is similar to acknowledgement process of TCP. If a client has multiple files to receive, it waits to initiate another request until it receives “226 Transfer Complete” acknowledgement message. To solve the LOSF problem, they use pipelining approaches by forcing the client to make next request instead of waiting for the 226 Transfer Complete” acknowledgement message [49]. According to test result in [49], pipelining improves the throughput of LOSF transfers considerably.

GridFTP has attempted to circumvent TCP problems by incorporating UDP based solutions and diving a TCP connections into a set of shorter connections into their latest version [49].

2.4.1.2 GridHTTP

GridHTTP [18] is a protocol, defined within the GridSite [50] framework that supports bulk data transfers via unencrypted HTTP connection, but first requires authentication of the clients via HTTPS. The aim of this protocol is to allow large (gigabyte) files to be transferred at optimal speeds while still maintaining some level of security. The problem of authentication over HTTP (usually achieved via usernames and passwords) is avoided by using the certificate handling capabilities of the GridSite software as well as the access control list functionality. Clients must connect to a web server over HTTPS and set the value of the "Upgrade" header in the request to "GridHTTP/1.0" in order to retrieve a file using the GridHTTP protocol [18].

2.4.1.3 bbFTP

bbFTP [33] is an open source file transfer software which implement its own transfer protocol optimizing for large files (larger than 2 GB). Similar to GridFTP, bbFTP is built upon the standard FTP protocol and uses parallel TCP streams [51]. The main strength of bbFTP is the ability to use SSH and certificate based authentication, data compression on-the-fly, and customizable time-outs [52]. Due to being more secure and attempting to optimize network bandwidth usage, it is preferable over than traditional FTP [53].

There are other FTP based solutions such as SafeTP [54, 55], which is developed at the University of California at Berkeley to provide a secure method for file transfer between Unix/Windows clients and secure FTP server [54], but their main concerns is to provide secure data transfer rather than high-performance data transfer.

2.4.1.4 The BaBar Copy Program (bbcp)

The Babar Copy Program [35], successor of Secure Fast Copy (sfcp), is another high-performance data transfer program and it was purely built upon the peer-to-peer architecture, in contrast to many other solutions. It is because of the bbcp's peer-to-peer architecture that it is well suited to environments where information flow is equal. The important features of bbcp are that carrying with a very low administrative overhead, using SSH [35] for authentication, providing an elegant and simple model. Similar to previous solutions, it exploits multiple TCP stream in parallel in order to accelerate the movement of data [35].

2.4.2 **UDP -based Data Movement Techniques**

To improve the bandwidth throughput in wide-area networks, the other alternative approach is UDP-based application-level solutions. UDP is a connection-less unreliable messaging protocol, whereas TCP is a connection-oriented reliable data streaming protocol[56]. Unreliability is a major drawback for data transfer. Therefore, UDP-based application-level solutions implement congestion control algorithm and reliability control mechanism at application layer that is fifth layer and above the transport layer, consisted of TCP and UDP, in the layered Internet architecture [56]. SABUL[57], UDT[56], FOBS [9], RBUDP [58], Tsunami [59, 60], UFTP [61], and FRTP [62] are ongoing works using rate-based UDP for high performance data transfer to overcome inefficiency of TCP [56]. Since some of these UDP-based solutions are derivations of existing ones, we only present main or important UDP-based solutions here.

2.4.2.1 Simple Available Bandwidth Utilization Library (SABUL)

SABUL is an application-level a rate-based protocol designed for data-intensive applications over high BDP networks to transport data reliably [22, 57]. Although SABUL uses UDP as data transfer channel and TCP as a control channel, it can coexist with TCP since it was not designed to replace TCP [57]. As stated by Gu and Grossman [57], it is designed for reliability, high performance, fairness and stability. In addition, since SABUL has implemented as an open source library, it can be easily deployed without requiring any significant modifications to network stacks of an operating system or to the existing network infrastructure [57]. Experimental studies, As claimed in [57], have demonstrated that SABUL can efficiently use available bandwidth in links with high BDP.

Fixed Rate Transport Protocol (FRTP) is a modified version of SABUL for end-to-end circuits [62]. SABUL was designed for packet-switched networks according to [62]; thus, it has poor performance on circuit-switched networks since congestion control service is not needed when the circuit is provisioned successfully due to fact that resource reservation and congestion is handled during the circuit setup [62]. Congestion control mechanism, on the other hand, adjusts data sending rates during the data transmission in packet-switched networks [62]. Therefore, eliminating problems of SABUL stemmed from end-to-end circuit is the main motivation behind the FRTP.

2.4.2.2 UDP-based Data Transfer Protocol (UDT)

The UDT protocol, a UDP-based protocol and designed to effectively utilize high-speed wide area optical networks, is an application-level high performance bulk data transfer protocol [56]. In order to attain high throughput data transfer with low

data loss, UDT combines rate-based, window-based and delay-based congestion control mechanisms[25, 56]. It is more TCP friendly than other rate-based schemes due to its slow start and AIMD control schemes for flow control [25].

Notwithstanding the fact that it is the successor of SABUL, it is a re-implementation from scratch with a new protocol design [56]. The main reason for redesigning it, as stated by Gu and Grossman [56], is the use of TCP as an control message channel for the simplicity of design and implementation in SABUL because TCP's own reliability and congestion control mechanism can result in unnecessary delay of control information in other protocols with their own reliability and congestion control mechanism. Therefore, UDT uses UDP protocol for both data and control packet transmission.

Similar to SABUL, it does not require any changes to network stacks of an operating system or to the existing network infrastructure and it is released as free software [22]. In addition, it permits applications to send data of any size by removing the concept of sending data block by block over UDP [56]. Moreover, it can be employed above other packet-switched network layer in such a way that it can be deployed as a transport layer protocol by using IP directly [22, 63].

2.4.2.3 Fast Object –Based data transfer System (FOBS)

FOBS [9] is an application-level, UDP-based, highly efficient large scale data transmission system designed for the high-bandwidth, high-delay network environment typical of computational Grids [64]. Similar to many other UDP-based solutions, it utilizes UDP protocol for actual data transfer and TCP protocol for control information exchange. It uses, however, two TCP channels to transfer control information between

sender and receiver; one channel for ENDOFSEGMENT/DONE/FEEDBACK/ACK packets and one for COMPLETEDPKT/WRITECOMPLETEDPKT packets [9]. Developing multiple congestion control mechanisms with the ability to dynamically switch between mechanisms to adapt to changes in the state of the end-to-end system is the uniqueness of FOBS [64].

2.4.2.4 Reliable Blast UDP (RBUDP)

The Reliable Blast UDP (RBUDP) is an aggressive bulk data transfer scheme designed for extremely high bandwidth, Quality-of-Service enabled networks, such as optically switched networks [58]. In order to fully leverage the underlying high-bandwidth network structure for pure data delivery, it not only purges slow-start and congestion control mechanisms of TCP, but also aggregates acknowledgements [58]. Similar to SABUL, hosts exchange data packets via UDP, and control packets via TCP [25].

2.4.2.5 Tsunami

Tsunami is a reliable file transfer protocol intended for faster transfer of large files over the uncongested high-bandwidth, high-delay networks [59]. As stated by Ansari [65], the architecture of Tsunami follows a classic client server model typical of conventional FTP. Tsunami, similar to FTP, uses a control channel to authenticate and negotiate the connection and a data session to transfer data [65]. Tsunami contrasts with FTP in regard to the use of UDP as data transfer channel. In order to regulate the data transfer rate, it uses the delay time between packets instead of the TCP's sliding window algorithm [60]. In addition, it implements negative acknowledgements to

notify the sender for lost packages as opposed to TCP's sending the acknowledgement of received data [59].

2.4.2.6 UFTP

UFTP, utilizing a protocol based on Starburst MFTP, is a UDP-based multicast file transfer program [61]. It is designed for efficient and reliable bulk data transfer to multiple receivers concurrently [52]. This is useful for distributing large files to a large number of receivers [61]. Mattmann et al [52] commented that although UFTP is particularly effective for data dissemination over a satellite link with two way communication or high-delay Wide Area Networks (WANs) where the reliability and congestion mechanisms of TCP cause underuse of throughput capabilities of the available network, it suffers the disadvantage of having extremely poor reliability with the fault rate a function of the total dataset volume.

2.5 Peer-to-Peer based Data Movement Techniques

Peer-to-peer (P2P) has become one of the most widely argued term in information technology [66]. Due to wide application areas of P2P systems, it could be considered as a set of protocols, an IT architecture, decentralized design model, or a business model [67]. Therefore, there is a great deal of number of different definitions of P2P.

According to Androutsellis-Theotokis and Spinellis [14], completely distributed systems composed of completely equivalent nodes in terms of functionality and tasks they perform is the most meticulous definition of "pure peer-to-peer" system. In other words, there is no discernable client or server role in a P2P architecture, although two

nodes communicate with each other [35, 67] using appropriate information and communication systems and are able to spontaneously collaborate without necessarily needing central coordination. However, there are some systems which employ the concept of super-nodes, function as mini-servers, such as Kazaa [14, 68], which are widely accepted as peer-to-peer systems. The definition of P2P given by [14] is broader enough to encompass all type of peer-to-peer systems:

> *Peer-to-peer systems are distributed systems consisting of interconnected nodes able to self organize into network topologies with the purpose of sharing resources such as content, CPU cycles, storage and bandwidth, capable of adapting to failures and accommodating transient populations of nodes while maintaining acceptable connectivity and performance, without requiring the intermediation or support of a global centralized server or authority.*

Even though, the client/server model appears more prevalent architecture than peer-to-peer model for today's Internet, the peer-to-peer architecture, at the outset, is the foundation that the original Internet was essentially built upon [69]. It has, however, changed into increasingly client/server model when the millions of clients communicating with a relatively privileged set of servers [69]. Yet there are, still a good deal number of peer-to-peer applications employed in the Internet.

In order to classify peer-to-peer systems, there are many distinguishing characteristics such as their network structure, the degree of network centralization, their purpose, their methods for distributed object location and routing, etc. We should note here that Androutsellis-Theotokis and Spinellis have made a very detailed study about peer-to-peer systems in [14] and analysis of peer-to-peer systems according to

their purpose of use is an adaptation and simplification of the one presented in their work. As to their use, they categorized peer-to-peer systems into five groups; (1) communication and collaboration, (2) distributed computation, (3) Internet service supporter, (4) database systems, and (5) content distribution.

1. **Communication and Collaboration**: Peer-to-peer systems of this category provide the infrastructure for assisting direct, generally real-time, communication and collaboration between peer computers. The foremost applications in this category are chat and instant messaging applications, such as Chat/IRC, Instant Messaging (AOL, ICQ, Yahoo, MSN, Google Talk), and Jabber [14, 70].
2. **Distributed Computation:** The main purpose of systems in this category is to exploit the available peer computer's resources, for instance, processing power (CPU cycles) [14]. Seti@home [12-14] and Genome@home [14-16] are the well-know projects in this category. Since the main purpose of the Grid Computing [2-5] is to enable the large-scale coordinated used and sharing of geographically dispersed resources, it can be, to some extent, considered a system in this category.
3. **Internet Service Support**: This category includes systems that support wide assortment of Internet services. Such applications as peer-to-peer multicast systems [14, 71, 72], Internet indirection infrastructures [14, 73], and security applications are used to leverage IP independent multicast routing and to provide protection against of denial of service or virus attacks [14, 74-76] respectively.

4. **Database Systems**: Distributed database systems are one of the most attractive research fields for peer-to-peer applications. The Local Relational Model (RLM) [14, 77] proposes translations rules and semantic dependencies between the set of all data stored in peer-to-peer network. PIER [14, 78] is a distributed query engine built on top of a peer-to-peer overlay network topology.
5. **Content Distribution:** This category includes most of the existing peer -to-peer systems. They are intended to share of digital media and other data between users [14]. Peer-to-peer systems designed for content distribution range from simple direct file-sharing applications to more complex systems. The sophisticated peer-to-peer content sharing systems provides services of publishing, organizing, indexing, searching, updating, and retrieving data with security and efficiency [14]. There are a great number of systems and infrastructures fall into this category. Examples of such systems include the Napster [79], Publius [80], Gnutella (RIP) [81], Kazaa [68], Freenet [82], MojoNation (RIP), Past [83], Chord [84], FreeHaven [85], BitTorrent[86] and JXTA[87].

2.5.1 **BitTorrent**

All of these peer-to-peer systems are widely used applications, but the last technique, BitTorrent, deserves more attention because it has been gaining in popularity and gathering momentum since its first appearance. Before explaining the rationales behind its success, we present a brief overview of what BitTorrent is.

BitTorrent is a peer-to-peer file sharing protocol like FTP in client/server paradigm. According to [88], “The key philosophy of BitTorrent is that users should upload (transmit outbound) at the same time they are downloading (receiving inbound.) In this manner, network bandwidth is utilized as efficiently as possible. BitTorrent is designed to work better as the number of people interested in a certain file increases, in contrast to other file transfer protocols”.

Despite its peer-to-peer nature, there is still a central server (called a *tracker*) which is only responsible for coordination of peers connections without having any knowledge of the content of the files being distributed. This feature enable tracker to support a large number of users with relatively limited tracker bandwidth [88].

The peer that has a complete copy of certain content and serves it is called *seed*. When the seed wants to share its file, the first process that the seed performs is to create a small static metadata file (it ends in *.torrent* and called *torrent* file) which contains information about the file that is to be shared and the location of the tracker. Then the seeder uploads this metadata file to tracker. This torrent file is vital for BitTorrent as all peers need to acquire it to start the download process. A *peer* is the opposite of the seed. In other word, it does not have the complete file; therefore, it demands pieces of the file from the seed and other peers [86, 88]. A file is split into fixed-size *pieces* that are all the same size except for the last one. The length of pieces and its corresponding SHA1 hash are described in the torrent file. However, a peer requests a *block* (a portion of data and two or more blocks made up a whole piece) from a peer.

The main reason behind wide acceptance of BitTorrent is that it defines a peer-to-peer content distribution protocol with very high data transfer speeds rather than

offering another peer-to-peer application for end-users to share their contents such as movie and mp3 files. In addition, it not only separates content distribution medium from content discovery and access services, for instance, indexing and searching, but also ensures integrity of the file content and prevents free-riding [89-91], which is a major problem from which many peer-to-peer applications suffer. It uses an embedded set of incentive mechanisms, such as SHA1 hash and tit-for-tat-ish algorithm, in order to compel the participating peers to contribute. These features promote it to suitable candidate for frameworks which provide storage service, for instance, Amazon Simple Storage Service [92].

Notwithstanding BitTorrent's great features, it has a need for modification and improvement in order to be used as a data distribution medium in scientific community. The first reason is that the tracker, in BitTorrent, [88] is a basic HTTP/HTTPS service that responds to HTTP GET requests. Since HTTP is ubiquitous protocol in the Internet, it is conceivable that using HTTP protocol may be of great advantage to it. However, it is unsuitable for an environment that is very dynamic and requires complex services (explained in next sections) to coordinate participating nodes, and communications taking place not just between the GridTorrent Framework peers, but even between the users and their GridTorrent Framework peers.

In BitTorrent, the communication happening between peers and tracker is passive communication; in other words, the tracker only delivers a list of available seeders and peers of a requested file, and collects statistics of uploading and downloading processes from the peers. File downloading process is the only required responsibility of a general BitTorrent peer and each downloading task is independent

from each other. After the initial communication, peer can continue its downloading process without the help of its tracker.

The second important reason is the dissimilarity between the characteristics of the users in scientific community and standard peer-to-peer community. In regular peer-to-peer community, there is no competition between users. In other words, there is one type of user, a passive user, and any user can access any data as long as he or she gets the torrent file. However, in the scientific community, due to expertise or research agenda and competition between institutions, only authorized users are permitted to access to pre-determined data sets with some access rights. While the passive user type in BitTorrent, the users in scientific community area very active and some of them cooperate on some files as a group. This creates diverse users' and groups' profile in scientific community.

The third reason is stemmed from the importance of data and its access. As the current design of BitTorrent, by itself, does not provide a search facility to find files by name or by other keywords; a user must find the initial torrent file by other means, such as a web search. On the other hand, searching, finding, and accessing to desired data are of paramount importance in scientific community, hence a reliable search service must be offered to scientific users. Therefore, even though there is a need to integrate BitTorrent into a content and collaboration framework with a search facility to use BitTorrent in scientific community, a mechanism involving regulation for content access with pre-defined rights and security is vital for a data sharing system in scientific community.

To summarize, BitTorrent is especially useful for large, popular files; however, it cannot directly be implemented in the grid environment because of its limitations like, data security, requirement of separate software, lack of flexibility with centralized tracking and the lack of partial usage. On the other hand, it was designed to separate file transfer mechanism from search and network maintenance components, which enables to implement each of them as an independent service from other components, and which renders it as one of the ideal data transfer mechanisms for scientific communities, for example, Grid computing, if the above missing features are provided and integrated with the tracker.

2.6 Network-level Data Transfer Techniques

The network-level solutions (i.e. NAS and SAN [24]) are low-level data transfer techniques designed to handle large data transfers. Systems such as WAN file systems and data storage systems are either built on top of network-level solutions or use them as data transfer layer. The prominent examples of SAN-based WAN file systems are GFS 5 (Sistina), Stornext (ADIC), SAN-FS (IBM), QFS (SUN), CXFS (SGI). Examples of data storage systems include the Storage Resource Broker (SRB) [93], the Distributed-Parallel Storage System (DPSS) [94], High Performance Storage System (HPSS) [95], and Hierarchical Data Format 5 (HDF5) [96]. Even though those systems are classified as data storage systems, most of them usually entail data management systems to handle the large volumes of data.

In traditional approach, in order to allow the end users and higher applications to access and modify the data stored in local storage devices, a local file system is used as mediator between them and disk subsystems. Nevertheless, this approach is not very

scalable and results in reliability and bottleneck problem. To circumvent this limitation, network-attached storage systems have been devised. In this paradigm, the system is composed of dedicated preconfigured file servers and network attached storage devices. Both **Storage Area Network (SAN)** and **Network Attached Storage (NAS)** provide network-based solutions. There are similarities between SAN and NAS, but there are differences as well.

NAS describes a complete file-level storage system designed to be attached to a traditional network such Ethernet and TCP/IP. NAS does not allow direct access to individual storage. Clients uses higher-level protocols (i.e. Network File System (NFS), Common Internet File System (CIFS), File Transfer Protocol (FTP), and Secure CoPy (SCP)) built on top of TCP/IP.

On the other hand, a SAN is a type of local area network (LAN) network to which storage devices are attached, and is designed to handle large data transfers. In contrast to NAS file-level access and TCP/IP support, SAN traditionally utilize low-level network protocols and supports high-speed block-level access to the storage devices. A SAN commonly uses Fibre Channel interconnection technology which provides better performance. In most cases, a NAS system is less expensive to purchase and less complex to operate than a SAN system.

Wide Area Network (WAN) File systems are high-performance scalable file management solutions that provide shared, fast, reliable data access from a single computer to hundreds of systems [97]. Despite the better performance of WAN file systems, they are very expensive systems since they serve the data over SAN which requires expensive FC-interfaces and SAN-switches.

Storage Resource Broker (SRB) is a middleware software system developed by the San Diego Supercomputer Center (SDSC). It depends on other lower-level systems, for instance, archives, files systems, networks, a DBMS for the metadata catalog, etc , but, like other grid technologies, it can be integrated with higher-level software [93]. The main focus of the SRB is to provide a uniform client interface to heterogeneous data collections by connecting these repositories, and to provide metadata for use in discovering and locating data within the storage system [32]. Although it uses Sreplicate and Scp with parallel I/O to improve performance, it utilizes other data transfer techniques such as GridFTP, DPSS, and SAN/ NAS[24, 98] by integration of HPSS [95].

The Distributed-Parallel Storage System (DPSS), originally developed as part of the DARPA funded MAGIC Testbed, is a scalable, high-performance, distributed-parallel data storage system [94]. It is defined, according to its website [94], as "a data block server, which provides high-performance data handling and architecture for building high-performance storage systems from low-cost commodity hardware components. This technology has been quite successful in providing an economical, high-performance, widely distributed, and highly scalable architecture for caching large amounts of data that can potentially be used by many different users." Similar to the SRB, the DPSS uses parallel data transfer streams or striping across multiple servers to improve performance [8], and its team is currently working on integrating the DPSS into SRB and Globus [94].

High Performance Storage System (HPSS) is highly flexible and scalable hierarchical storage management software developed to manage petabytes of data on

disk and robotic tape libraries [95]. In order to create a single virtual file system by aggregating the capacity and performance of many computers, disks, and tape drives, HPSS uses cluster, LAN and/or SAN technology [95]. This approach enables HPSS to eliminate the limitations resulting from total storage capacity, file sizes, data rates, and number of objects stored. A variety of user and file system interfaces such as VFS, FTP, PFTP, GridFTP, Samba, NFS, client API, local file mover and third party SAN (SAN3P) are supported by HPSS [95].

Hierarchical Data Format 5 (HDF5) is defined by the HDF group as "technology suite is designed to organize, store, discover, access, analyze, share, and preserve diverse, complex data in continuously evolving heterogeneous computing and storage environments" [96]. Unlike above-mentioned technologies, it focuses on the structure of data and provides interfaces that enables client to access structured data from a variety of underling storage systems [8, 32].

2.7 Discussion

All of the systems we have discussed above have advantages and disadvantages. Network-level solutions deliver enhanced storage capacity and offer much greater performance than application or system level solutions, but they are still prohibitively expensive propositions for most organizations. In addition, despite their very high technologies and low-level protocols, they are in need of integration with application-level solutions (GridFTP) in order to be compatible with existing systems.

Although system-level data transfer techniques can provide better performance than application-level solutions, they usually require modifications to the operating systems of the machine or network infrastructure, which hinders the prevailing usage of

them. Application-level solutions can provide high-performance data transfer, yet they do not utilize network bandwidth as efficiently as possible since they only use receiving inbound –not transmit outbound. Moreover, they suffer from the drawbacks of data transferring protocols (FTP) based on client/server paradigm.

Peer-to-peer solutions could propose solutions that use available systems' resources including network bandwidth as efficiently as possible; nevertheless, only few of them are suitable file distribution for scientific community and lack of key features essential to scientific community such as security support, access right management, and collaboration framework.

The work discussed in this thesis harnesses the advantages of peer-to-peer system to leverage the available resources of participating entities including network bandwidth efficiently by implementing features that are not supported by the selected peer-to-peer system (BitTorrent). Besides, as our proposed system is an application-level solution, it is so flexible and lightweight application that it can be deployed on any of existing operating systems and network infrastructures or can coexist with other data transfer techniques or data management frameworks. Furthermore, unlike other data transfer techniques, because of the architecture of our proposed model, it provides a simple and lightweight data management system that functions as collaboration framework as well.

Chapter 3

The GridTorrent Framework Architecture

3.1 The Overview of the GridTorrent Framework

The GridTorrent Framework (GTF) we have studied and implemented is a software system that provides high performance, secure data transferring and data sharing medium and collaborative environment with capability of efficient and optimum use of available resources of underlying system. Although, with the secure collaboration and content management environment, scientific community is the primary target users of the GTF, it can be deployed in and utilized by any communities that would get benefit from optimal and productive data transferring mechanism with the collaboration framework component.

The main characteristics of applications in the scientific communities as mentioned in chapter one and two have been evolved from computational behavior to informational behavior. As a result of that, unprecedented data volumes has been generated by computers, conducted experiments, and scientific instruments and gadgets ranging from tiny microscopes to enormous earth and space telescopes such as The Robert C. Byrd Green Bank Telescope (GBT) Lovell telescope at Green Bank [99], West Virginia, USA and NASA's Hubble space telescope[100].

Besides the very large size of the produced data, participators are not only the members of the same institutions, organizations, universities or laboratories, but also scattered all around the world. Consequently, delivering data to geographically dispersed organizations over the wide area networks and managing it are of paramount importance, and it is proven that data management is the one of the hardest jobs to do in the distributed computing environment [101]. As explained in chapter two, there are many studies and works to address data management and transferring. There are several reasons for myriad of researches in this field:

- Each organization decides their data management schema, data transferring protocol, security policy, and data format that are suitable to characteristics and constraints, e.g. time, money, etc., of researches that they conduct, features of available resources such as human experience, technologies, and equipments. Those variant requirements may result non-uniform data format. However, when there is a collaboration between institutions, they attempt to standardize to tools, i.e. data management schema, data transferring protocols, the data format, which are going to be used among them.

- There is none perfect data management and transferring solution for all cases. Each study tries to solve the above-mentioned varying requirements, either by focusing on the most important issue or applying different techniques considered as better approach. Sometimes, emerging technologies either in software or hardware areas imposes substantial modifications on current systems in order to enhance the performance of the existing system or upgrade it, or exploit the available technologies. For instance, after Web Service appearance, the Grid community has adopted itself to conform to Web Service standards in order to harness the power of it. Another example is that Reliable File Transfer [42, 43, 101] that is [102]devised to address the shortcomings of GridFTP [49] by utilizing Web Service [43] features. For this reason, it is very inevitable to have many and diverse systems in this area.

In spite of the availability of a great number of systems, we consider that there are still some issues which have not been settled permanently –which is not possible, and new technologies which have been exploited to the full. It would better to say optimum use of underlying system's resources by exploiting peer-to-peer technologies such as BitTorrent has not been experimented thoroughly in our opinion. Consequently, current systems are still lack of some important features from which scientific community benefited greatly. In this chapter, we describe our approach which not only provides above-mentioned missing features but also conforms to Web Service standards.

3.2 Main Components of the GridTorrent Framework

The GridTorrent entails three major components because of distributed and collaborative requirements of scientific applications;

1. The GridTorrent Framework Client (GTFC) which provides the services required to data sharing among the peers,
2. The WS-Tracker is responsible for coordination between the GTFCs,
3. The Collaboration and Content Manager (CCM) which enables users to publish, subscribe, and manage their contents.

Each of these components consists of several internal services for different purposes ranging from security to different data transfer protocols. We discuss the details of these components in the following chapters; in this section, we just give a brief overview of the architecture of the whole system and basic information about the interactions between components.

Since each component depends on another component in order to perform correctly and successfully, every part of the GTF is as important as one another. Figure 3-1 presents the overall architecture of and interactions between GridTorrent components.

A user is a real person who is interested in sharing and managing his/her contents through the CCM, initiates the GTFC so that the actual data transferring process can start. While there is no restriction imposed upon choosing the users' accesses to their GTFCs, they access the CCM through HTTP protocol similar to regular web site's access. A user is permitted to own and manage more than one GTFC if it is necessary.

The first component is the GridTorrent Client which is software that runs on users' computers and provides services to support transferring data among interested peers, imposing security constraints, and delivering fresh information to WS-Tracker in order to help its coordinator task among the peers. Owing to variety of available network configuration and diverse characteristics of data sharing process and the GTF's components, the GTFC uses several protocols. In addition to above reasons, using different data transfer protocol might sometimes improve the performance or provide better system resources usage. Therefore, the GTFC, which has modular transport layer architecture and in our prototype, uses two data transfer protocols: (1) TCP, (2) PTCP.

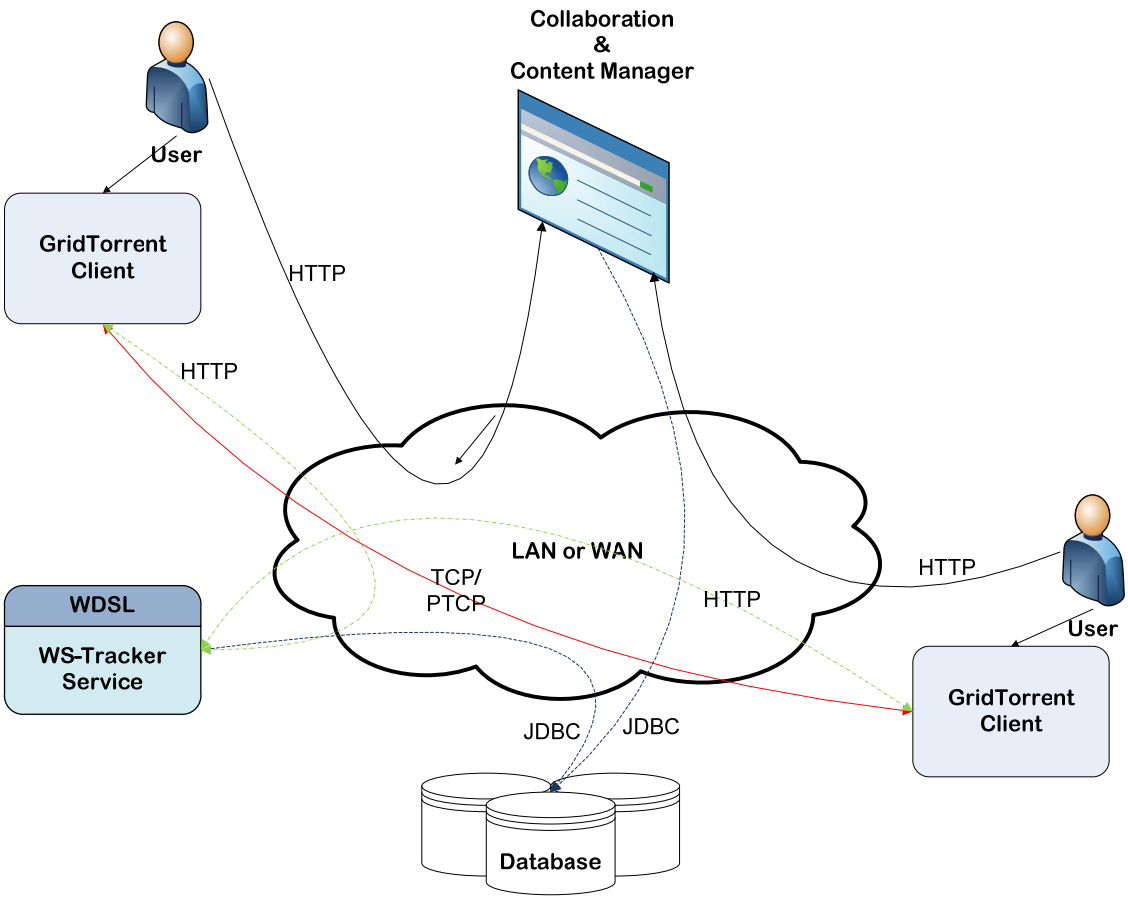

Figure 3-1 GridTorrent Framework is composed of a client, a Web Service Tracker, and a Collaboration and Content Manager. Each component communicates with another one via different protocols such as HTTP, TCP, and PTCP

As it is illustrated in Figure 3-1, whereas it communicates with other GTFCs via TCP and PTCP for data transfer, it interacts with the WS-Tracker as a traditional Web Service client and request and response SOAP messages over HTTP are exchanged with the WS-Tracker in order to update its current information. Details of the GTFC are discussed in Chapter 4.

The WS-Tracker is the second component of the GTF and a Web Service version of a regular BitTorrent tracker with added several vital features. A regular

tracker is a simple HTTP/HTTPS service [88] which responds to HTTP GET requests and it is impossible to include additional features such as access control list feature which is generally vital and required for scientific data sharing. Additionally, because of the nature of Web Service feature, new functionalities and services can be deployed easily on WS-Tracker through WSDL interface which makes the WS-Tracker very adaptable for newly emerging cases and special requirements. As a result, we developed a Web Service version of tracker in our design. We explain its design in details in Chapter 6.

Although other network protocols can be utilized for exchanging SOAP messages, the WS-Tracker uses only HTTP in prototype version. Security issues are addressed by integrating the Grid Security Infrastructure [103] and it is discussed in chapter 7.

Users' contents sharing settings are stored in to a database through the CCM. We use MySQL [47] database server for this purpose. The WS-Tracker retrieves the stored information from MySQL through JDBC connections as in shown in Figure 3-1.

The last unit is the Collaboration and Content Manager which is accessed by users via HTTP. It consists of several JSP based interfaces leveraging newly emerged AJAX [104] techniques, and is composed of two sub major components: Collaboration Manager and Content Manager. As their names denotes, Collaboration Manager provides a collaborative substratum to registered users to by permitting them to create their friend lists. This list is used to inform all the users in it when a new content is published by the owner of the list.

The Content Manager offers services which permit users to do search on available contents, publish their contents, and subscribe to contents which are allowed them to download. All required information to achieve above services is stored into MySQL database, accessed by WS-Tracker Service as well, via JDBC connections.

In Chapter 5, details of both Collaboration and Content Managers are discussed in greater detail.

After reviewing briefly each component, we can summarize interactions between components. Users are the initiators of the whole system. They register to the CCM and start their GTFCs. Additionally; they share their contents and collaborate with each other through the CCM.

The CCM stores users' information to database server using JDBC connections. The same information are retrieved and delivered to the GTFS by the WS-Tracker Service via same database access protocol –JDBC.

The GTFCs start either a downloading or uploading process corresponding to messages delivered by the WS-Tracker Service. In addition data sharing task, it uploads its current statistical information into the WS-Tracker Service to help carry out its coordinator job successfully.

3.3 Summary

In this chapter, we have presented an overview of our GridTorrent Framework architecture. The GTF architecture consists of three major components in order to meet requirements of data sharing process of scientific community. We adopt open peer-to-peer standard for data sharing task and Web Service standards for implementing the coordinator of peers, WS-Tracker Service, so that new and complex features could be

deployed easily without modifying the whole architecture. We also summarize briefly each component and their functionalities and the protocols used between them.

Chapter 4

The GridTorrent Framework Client Architecture

4.1 Introduction

In the previous chapters we explained the motivation and rationale for GridTorrent Framework Architecture which offers a framework for high performance data transferring and sharing primarily for scientific communities, and provided a high level overview and description of our design decisions and overall approach.

In this chapter, we extend more details about low level and architectural design decisions of the GridTorrent Framework Client which is responsible for real data sharing and transferring processes between the peers. In addition, we present a thorough description of key components and their implementation.

Available techniques for data transferring are classified into two categories: client/server and peer-to-peer. In the former model, client initiates data transferring process and server delivers the requested data to client. As it is shown in Figure 4-1, this model could support a heterogeneous collection of clients which span a very wide spectrum that includes desktops, PDAs and other handheld devices, appliances, and other networked resources when the client service is kept very light service. However, this model has couple of disadvantages under a certain scenario nevertheless. For example, if there is more than one client and all of them are interested in the same data, the server must provide the demanded data to all demanding clients. In other words, none of the active clients involves themselves in any part of data transferring processes except the one taking place between the server and itself. As a result, first disadvantage of client/server model might cause a severe data access bottleneck because of a considerable demand for a particular data at the server that hosts the demanded data. Second drawback is the result of the bottleneck problem and that all the available computing power of, I/O and network bandwidth resources of the clients stand idle during the data transferring process unless there are other jobs keep them busy.

In the second data transfer model, similar to client/server model, a client commences data transferring process. However, peer-to-peer model allows each peer to serve both as a client and a server at the same time by uniformly dividing all responsibilities among all participants. Therefore, a downloading peer (client) can deliver the downloaded segment of the downloading data to another peer (client) as a server. This model might address above-mentioned disadvantages caused by client/server model. Although, similar to client/server model as shown in Figure 4-2, it

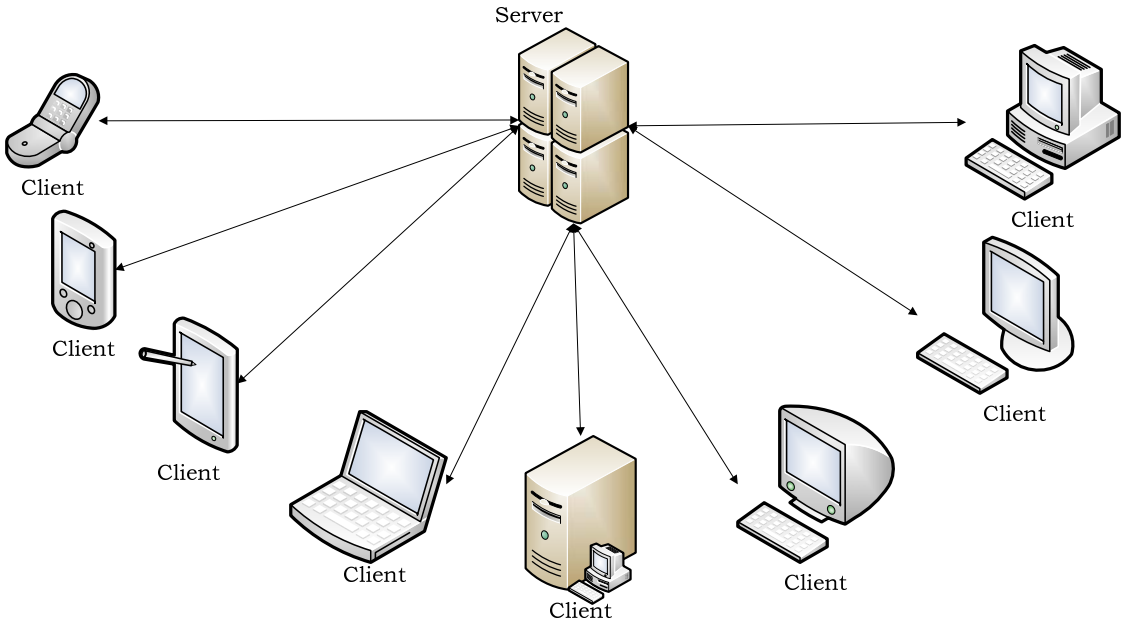

Figure 4-1 Client/Server model

is possible that to support highly diverse collection of peers with different hardware features, it is very unpractical to use very tiny gadgets with limited memory and low computing power since each peer has to function both as a client and a server simultaneously.

As a result of this, we have developed a novel data transfer layer which uses peer-to-peer data sharing algorithm of BitTorrent [86, 88] with the underlying TCP and PTCP [105] as data delivery protocols in our current prototype.

4.2 Overview of the GridTorrent Framework Client Architecture

To motivation behind our work is to provide a lightweight data transfer/management middleware that has simple, extensible, easily modifiable

architecture, as well can effortlessly be deployable in large numbers of distributed nodes and that requires the least amount of possible central control or management

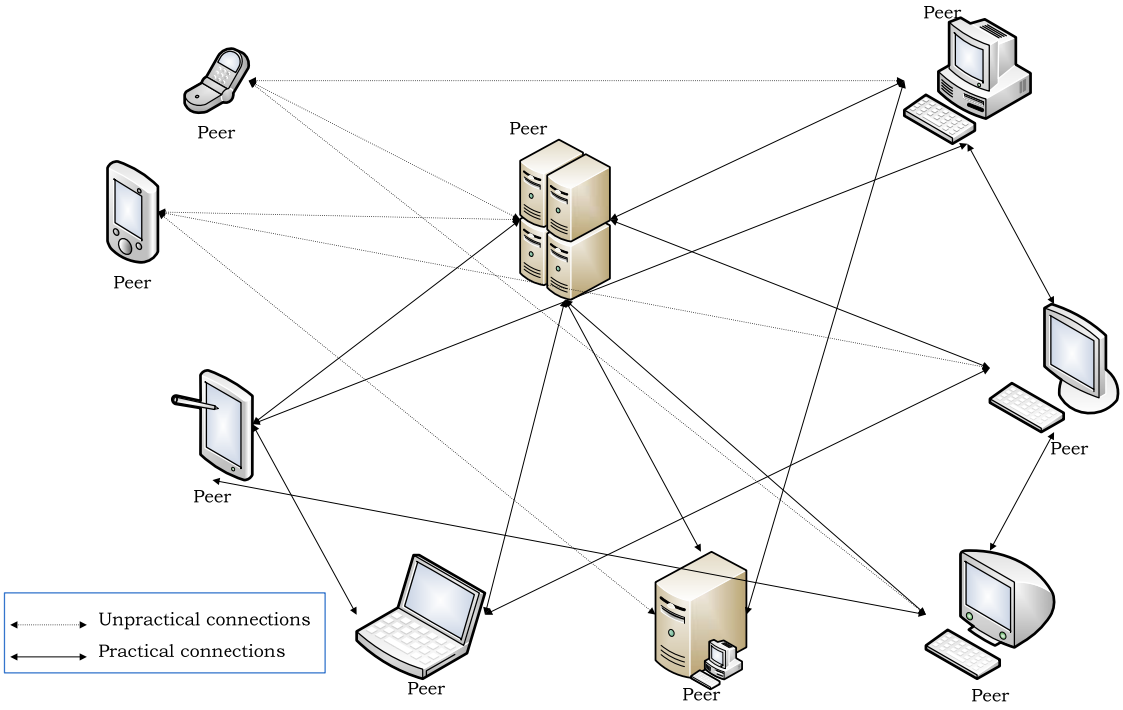

Figure 4-2 Peer-to-peer model

tasks. For this purpose, we have chosen layered architecture for the GTFC design by taking advantage of peer-to-peer infrastructure and service oriented architecture. The layered architecture of and the components of the GTFC are illustrated in Figure 4-3. These layers are: security layer, core services layer, and data sharing algorithm layer. The services provided by these layers are described in next sections of this chapter.

Each layer is built on top of another layer and uses the services offered by the lower layers or adjacent modules. To provide services pertinent to security, the GTFC's layered architecture is based on Grid Interface which consists of Java CoG Kit and Java WS Security middleware as shown in the figure.

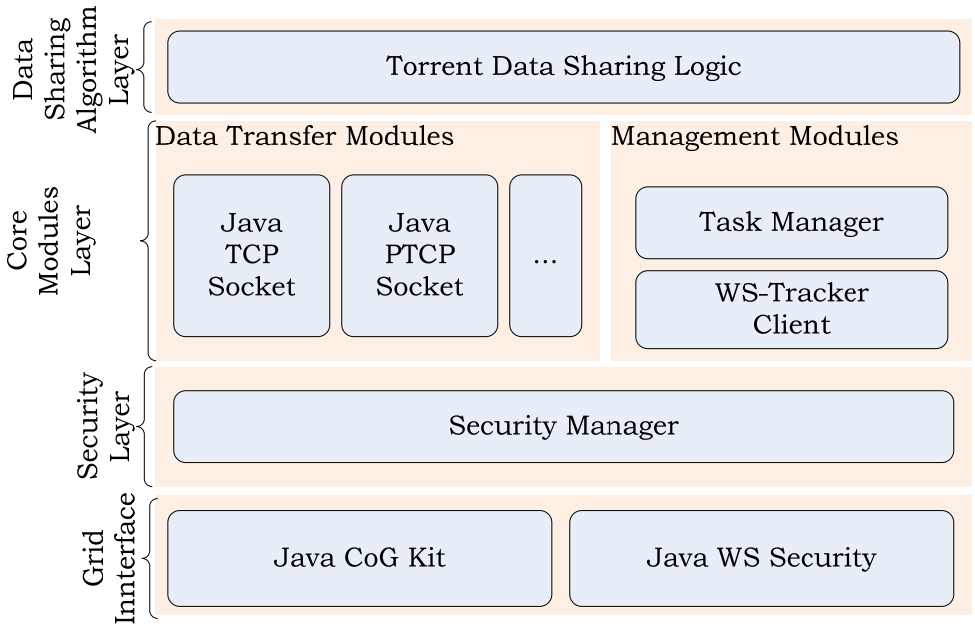

Figure 4-3 GridTorrent Framework Client Architecture

The components of the GTFC can be organized in a layered architecture as in shown in the figure. These modules are: (1) Torrent Data Sharing Logic, (2) Task Manager, (3) WS-Tracker Client, (4) Data Transfer Modules, and (5) Security Manager.

4.3 Torrent Data Sharing Logic

Data Sharing Algorithm layer is responsible for execution of and monitoring of rules and specifications which are required for fair, efficient and high performance peer-to-peer data transfer and defined by BitTorrent protocol. The services provided by Data Sharing Algorithm layer are as follows:

- Service of creating a .torrent metafile for a given file or directory structure to be desired to share or download, i.e. ***content***.
- Service of generating a bitfield map for a given content

- Service of allocating memory and disk space required for them.
- Service of handling current peer connections for data transfer.
- Service of gathering statistical information about the uploading and downloading process of each file.
- Service of reporting statistical information to Task Manager in order to deliver it to WS-Tracker service.

It provides a data listener for each of the shared contents at each node. Before starting the actual data transfer process, it checks every incoming connection's IP and port information with the ACL Registration Table (ACLRETAB) whether they have access right for the requested content. If their IP and port information is not registered to ACLRETAB, they are rejected immediately by closing their incoming socket. As shown in Figure 4-4, Data Sharing Algorithm layer interacts with Data Transfer Modules, Task Manager and Security Manager.

4.4 Core Modules Layer

This layer consists of two modules: Data Transfer Modules and Management Modules. Likewise, each of these modules comprises subcomponents as depicted in the figure.

4.4.1 **Data Transfer Modules**

Data transport is a very important task for data-intensive applications in scientific disciplines such as High Energy Physics, Astronomy, Earthquake Engineering, and Climate Modeling. Massive datasets must be transferred in the shortest possible amount of time to a community of hundreds or thousands researches

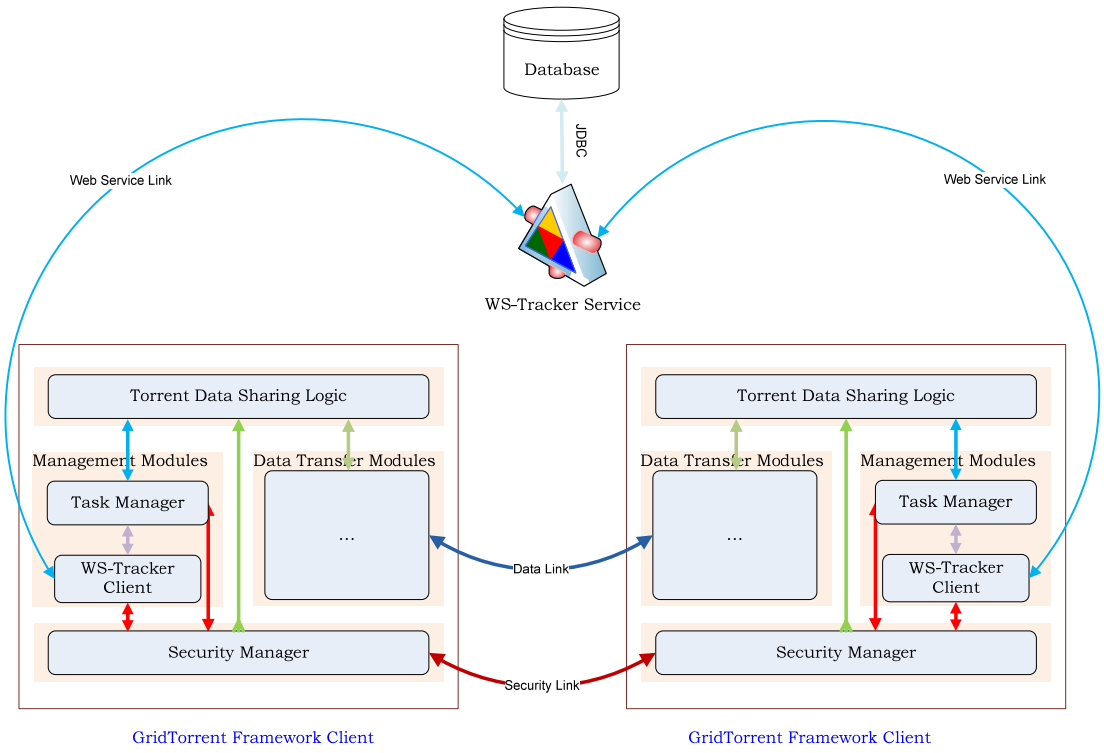

Figure 4-4 Simulation Architecture

geographically distributed so as to enable the accomplishment of satisfactory performance [23].

Although TCP is the most widely used transport protocol and is the de facto protocol of the Internet, because of its window based congestion control mechanism, it prevents [30] full-scale usage of high bandwidth-delay product. In order to overcome this problem, researches have continually worked to improve TCP and conceived several application level solutions. The latter approach emerges as a favorite solution because it supports for easy development and seamless integration with legacy systems, whereas the former one suffers from deployment difficulties [56].

Using parallel TCP is the most common technique that is used in PSockets [51] and GridFTP [40]. Using multiple TCP streams may increase the usage of network

bandwidth, but it is performance depend on many factors, such as the number of parallel streams and the buffer sizes of each flow [106].

Another approach is using rate-based UDP to overcome TCP's inefficiency. Some of ongoing works in this area are SABUL [57], FOBS [9], RBUP [58], FRTP [62], and UDT [106]. In spite of the fact that the UDP is a very simple protocol, providing reliable data streaming service to applications is an important advantage of TCP over UDP.

Data Transfer Modules is accountable for sending and receiving actual data to/from other peers. As illustrated in the figure, it only interacts with Data Sharing Algorithm layer.

Even though we are planning to use UDP based data transfer protocols in the future, in our current prototype, Data Transfer Modules consists of services which use the Internet protocol TCP/IP to transfer data between physical locations. In order to enable our GTFC transfer data on any network and utilize the network more efficiently, we both employed both a single TCP flow and parallel TCP flows. In spite of the fact that Data Transfer Modules is depicted as an independent entity in the architecture figure, because of its job description it has very close relation with Torrent Data Sharing Algorithm Layer. GTFC uses two channels: data channel and security channel. The former is used solely for the purpose of data transfer in high bandwidth. The latter is for the security purpose and encrypted.

4.4.2 **Management Modules**

The Management Modules consists of sub-modules which provide services and tools that support the tasks management, monitoring of usage and availability statistical

information such as percentage of upload and download information, and communication with the WS-Tracker.

4.4.2.1 Task Manager

Task Manager is the first module in the GTFC to be executed when the user run the GTFC. After initializing the GTFC settings, it generates a unique ID, Unique Grid Torrent ID (UGTID). UGTID is essential for each of GTCs because it is used to identify each GTC during the data sharing and communication processes. Following UGTID creation, GTC stores it into an ID file for future utilization. The user has to register other required information with this UGTID by retrieving it from the ID file into Collaboration and Content Manager (CCM).

Other important responsibility of Task Manager is to execute a task included in task list delivered by WS-Tracker service. Upon task list arrival, it parses the list and then, according to description of task, executes the appropriate services to perform a task or starts a proper module and passes it to that module to be handled.

It acts as central control unit of the GTFC and interacts all the modules except Data Transfer Modules as shown in Figure 4-4. In order to create a scheduled request needing to be passed to WS-Tracker client and to be delivered WS-Tracker service eventually, it has a time-based scheduling service as well.

4.4.2.2 WS-Tracker Client

WS-Tracker client behaves as a communication substrate between Task manager and WS-Tracker service. The relation between Task manager and WS-Tracker service is loosely coupled relation. This loose coupling feature enabled us to implement

the management part of GTF as in service-oriented architecture (SOA). SOA is defined as following on a web page dedicated to service-oriented architecture and Web services [107].

> *A service-oriented architecture is essentially a collection of services. These services communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some activity. Some means of connecting services to each other is needed.*

In other words, the service is the basic building block of SOA, and according to the same web page [107], a service is defined as "a function that is well-defined, self-contained, and does not depend on the context or state of other services."

Although, currently, the technology of Web services can be used to implement a service-oriented architecture, SOA is not a new idea and DCOM or Object Request Brokers (ORBs) might be considered as prior service-oriented architecture implementations [107].

Similar to SOA, each task of the GTF is regarded as a service so all tasks in the system are implemented as traditional web services. Thus, WS-Tracker Client communicates with WS-Tracker service as a traditional Web service client, request and response SOAP messages are transported over HTTP.

WS-Tracker service URL information is a vital piece of data since WS-Tracker Client communicates with the given WS-Tracker service during the whole data sharing process. Thus, in order to start a connection with WS-Tracker service, WS-Tracker's service URL information has to be notified GTFC by the user. This notification can be done in two ways; by updating GTFC's properties file either before, or after running it.

If WS-Tracker's address information is updated after running it, GTC will receive it after a certain time, since it checks its properties file periodically.

4.5 Security Manager

Security manager handles issues related to security, for instance exchanging certificates, encrypting and decrypting of messages. For this purpose, The GTFC's layered architecture uses third party security components: Java CoG Kit and Java WS Security middleware as shown in Figure 4-3. The former is used to provide functionalities for MyProxy [108] security credentials. Since short-term connections are the general characteristics of connections between peers, using MyProxy not only is a ideal solution by delegating short-term credentials and providing a set of flexible authentications and authorization mechanisms, but also enables us to integrate the GTFC with many other systems, for it is used in many large grid data projects such as the Enabling Grids for E-science (EGEE), FusionGrid, the Large Hadron Collider (LHC) Computing Grid, Open Science Grid, and TeraGrid, just to name a few.

The latter, as its name denotes, is used between and the Management modules and WS-Tracker service (Refer to Chapter 6) and ensures the secure conversation between them. (Refer Chapter 7 for further details about MyProxy and Java WS Security).

Security Manager communicates with other peers through security channel which is independent of data channel, dedicated to exchange of security information such as certificates and proxy certificates, and encrypted.

To authenticate incoming connections and their rights, the security manager checks provided info by them against the ACL Registration Table (ACLRETAB).

Content hash code, IP, port and UGTID are pieces of information for identity verification process and the important fields of the (ACLRETAB). They are inserted into the ACLRETAB after the parsing process of the ACL messages received in share content request message or ACL message. Following a successful authentication and authorization process, it provides a session key that needs to be delivered to data part to prove itself authenticated to incoming connection.

4.6 Task Generation and Exchange

A real user has to start the GTFC before initiating the process of task generation. After a successful registration, the user can login to Collaboration and Content Manager (CCM) to commence the sharing procedure of a content that is designated to be shared. First, a user inputs necessary data about the content into CCM. Then, he or she needs to set access control rights of the selected content. CCM stores this information and user's actions as future tasks into a database shared with CCM.

When WS-Tracker Client communicates with WS-Tracker service, it pulls out the user’s actions from the database, converts each action into a *task,* and delivers those tasks in a task list to the task manager through WS-Tracker Client components so that they will be executed by the task manager module in the user's GTFC. *Task list* is a list in XML format and contains only two types of task created by the user: share content request and download content request. These tasks with the others will be explained in details in the next section.

4.6.1 **Tasks**

A task is just simple metadata. Even though we could name it as "***message***", since it triggers actions in the GTFC, we preferred to term it as "***task"***. It is exchanged between the GTFC and WS-Tracker service in order to instruct them what to do to carry out a specific action. It can be categorized into request, response, periodic, and non-periodic. Table 4-1provides a listing of task types used within the GTF.

Table 4-1 Tasks Overview

No	Task Name	Creator	Source	Destination	Category
1	Task List Request	GTFC	GTFC	WS-Tracker	request, periodic
2	Share Content Request	User	WS-Tracker	GTFC	request, nonperiodic
3	Share Content Action	GTFC	GTFC	WS-Tracker	Response, nonperiodic
4	Download Content Request	User	WS-Tracker	GTFC	Request, nonperiodic
5	Torrent Request	GTFC	GTFC	WS-Tracker	response, periodic
6	ACL Request	GTFC	GTFC	WS-Tracker	request, periodic
7	ACL	User	WS-Tracker	GTFC	response
8	Update Status	GTFC	GTFC	WS-Tracker	periodic

There are six types of task available: task list request, share content request, share content action, download request, torrent request, ACL request, ACL and, lastly,

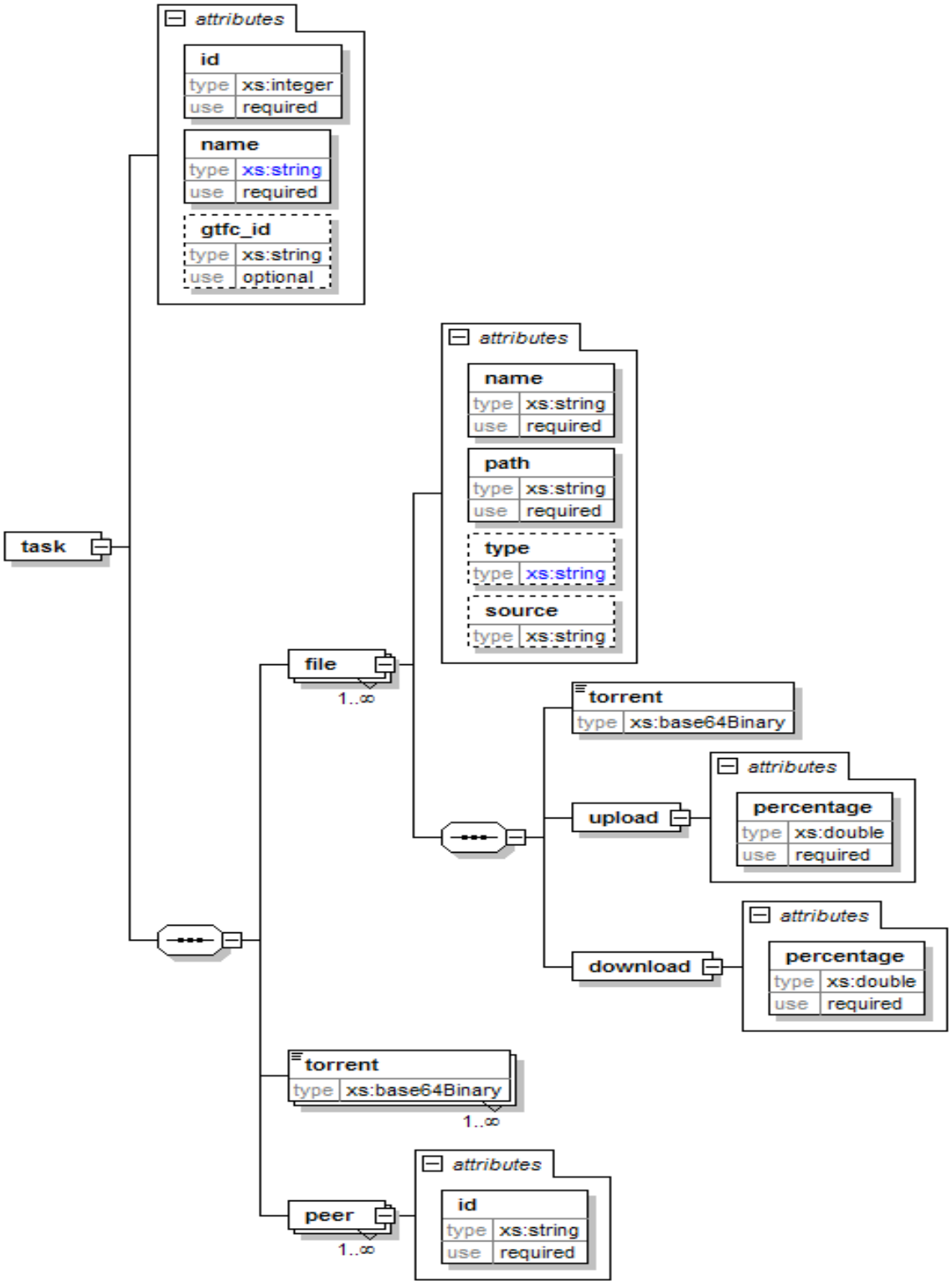

Figure 4-5 Representation of XML schema of Task

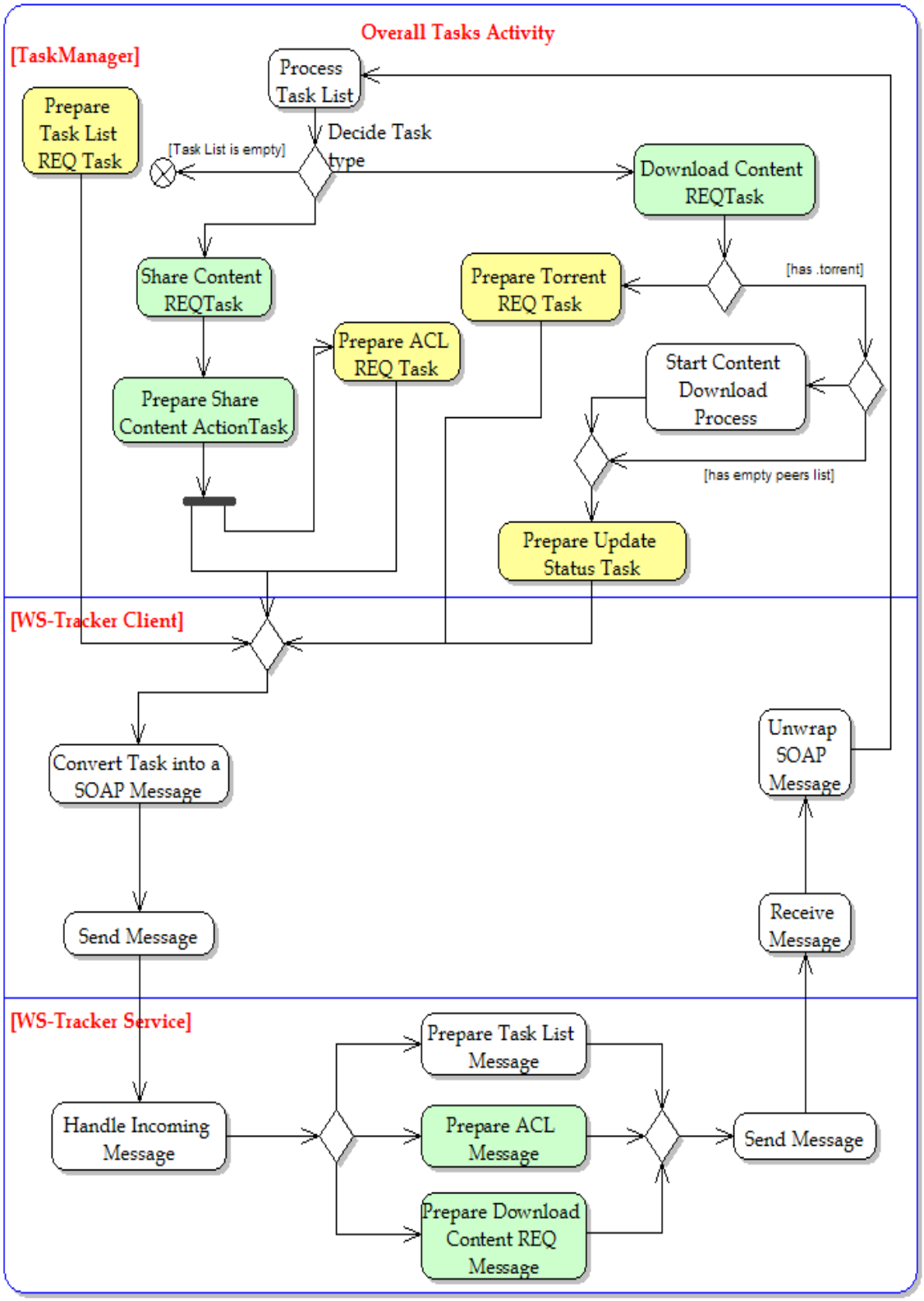

Figure 4-6 Activity diagram of GTF's tasks

update status task.

As To formulate the different task types we created XML schema as illustrated in Figure 4-5. The Attributes of task are quite self-explanatory. As its name suggested, id attribute stores a unique number generated by DB. Name attribute's value can be one of predefined task names and gtfc_id is used for UGTID (Unique GridTorrent ID). The name, path and type attributes of file are used to identify the content in local file-system. Type attribute's value can only be either file or folder. File's torrent element or task's torrent element is used to transfer binary data of the .torrent metafile content from GTC to WS-Tracker. Upload and download elements of file are utilized to provide statistical information to WS-Tracker about current transferring processes. Finally, WS-Tracker returns a list of peers that have access rights to given content in peer element of task.

Figure 4-6 shows all interactions between the GTF tasks. The tasks displayed as in yellow boxes are periodical tasks that are executed repeatedly over a period of time. Non-periodic tasks just carried out as reactions to certain tasks are displayed with the green boxes as depicted in the figure.

4.6.1.1 Task List Request Task

The Task List Request Task falls into the category of request and periodical task. It is the first task generated by the task manager to initiate the communication between WS-Tracker client and WS-Tracker service. An example of Task List Request task is shown in Table 4-2. It flows from the GTFC to WS-Tracker service with the information of task id, task name, and UGTID. A list of task, which may contain some tasks or be empty, will be returned to task manager as a response to it.

Table 4-2 Presentation of Task List Request Task in XML format

```
<task id="56" name="TaskListREQ">
</task>
```

4.6.1.2 Share Content Request Task

The Share Content Request Task is categorized as a request task and generated by the user when he/she publishes the content's metadata, file name, path, size, etc., for example, into CCM. It is exchanged between WS-Tracker service and the WS-Tracker client of source GTFC, that is, it has the whole content whose owner desires to share it, and emanates from WS-Tracker service to WS-Tracker client. When a WS-Tracker client contacts with the WS-Tracker service, it pulls available task lists from database and delivers it to designated client.

Table 4-3 Illustration of Share Content Request Task in the XML message

```
<task id="59" name="ShareContentREQ">
        <file name="test1.data" path="c:\test-results" type="file">
        </file>
</task>
```

Apart from task id and name information, the file name, path and type are the parameters transferred with this task as shown in Table 4-3.

Upon its arrival, after the message and task handling processes, Task Manager starts a ConnectionListener object. Then, the ConnectionListener object locates the content, desired to be shared, by using information delivered via ShareContentREQ task and creates a .torrent metafile of the content and passes it to Task Manager in order

to enable it to generate Share Content Response task (ShareContentRES Task) as illustrated in Figure 4-7.

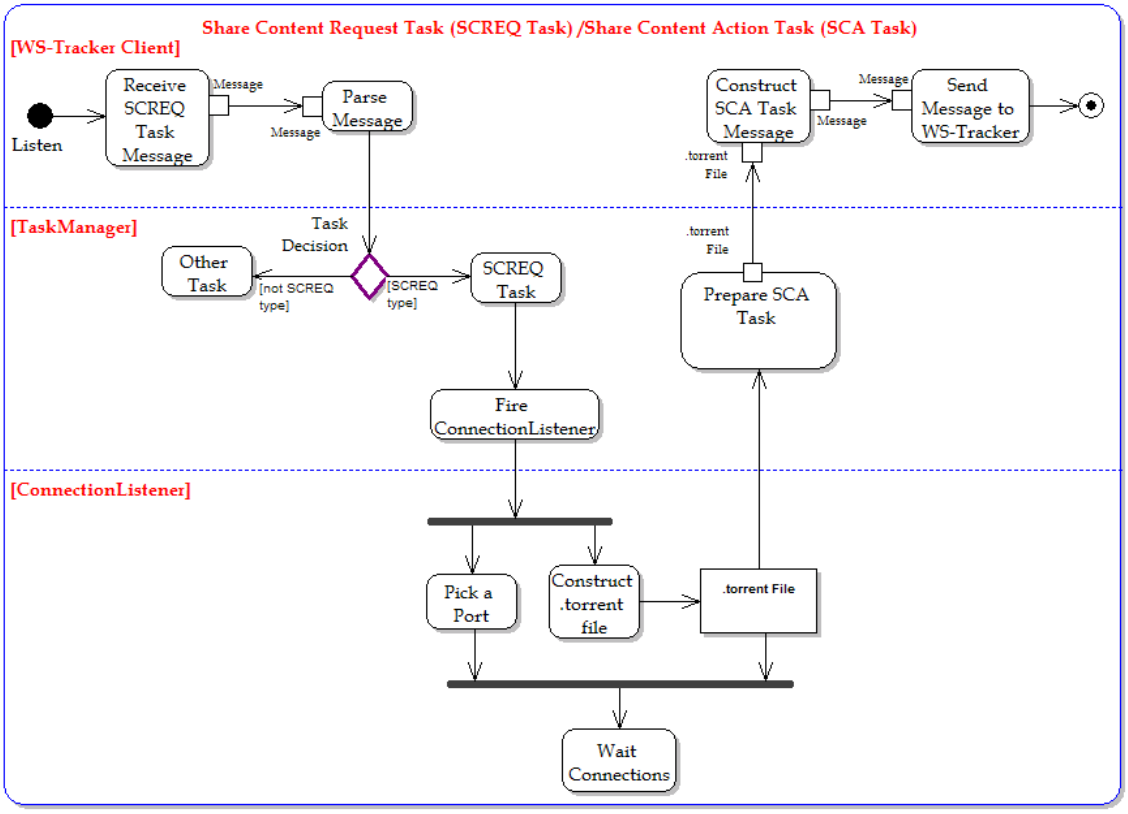

Figure 4-7 Processes workflow of share content request task and share content action task

4.6.1.3 Share Content Response Task

The Share Content Response Task is GTC's reaction to the ShareContentREQ task when it is delivered to GTC by means of its WS-Tracker Client. GTC creates a .torrent file, i.e. a metafile, for the requested the content. Then it passes this metafile to WS-Tracker via its WS-Tracker Client. In other words, the Share Content Response Task flows into WS-Tracker from GTC. WS-Tracker stores incoming metafile into both memory and database. While the former is used for the performance purpose, the

latter is employed for the persistency purpose. This task including the .torrent metafile, then, is passed to WS-Tracker client. Finally, it constructs a SOAP message of the Share Content Response Task and sends it to WS-Tracker service over HTTP.

Table 4-4 Representation of Share Content Response Task in the form of XML message

```
<task id="63" name="ShareContentRES">
        <file name="test1.data" path="c:\test-results" type="file">
                <torrent>TORRENT BINARY DATA IS HERE</torrent>
        </file>
</task>
```

4.6.1.4 Download Content Request Task

Similar to the Share Content Request Task, the Download Request Task is a request type of task and generated by the user when the user picks any content which is available to the user and wants to download it. CCM stores this process into database as a Download Request Task. The same as Share Content Request Task does, it originates from WS-Tracker to WS-Tracker Client, and will be pulled from DB by WS-Tracker and delivered to the client when it communicates with WS-Tracker.

Table 4-5 An example of Download Content Request Task in XML format

```
<task id="67" name="DownloadContentREQ">
        <file name="test1.data" path="c:\test-results" type="file" source="GID">
        </file>
</task>
```

4.6.1.5 Download Content Response Task

The Download Content Response Task message generated by GTC is sent to WS-Tracker in reply to the Download Request Task message. It demands WS-Tracker to send .torrent metafile of the requested content. Upon receiving the Download Content Response Task, WS-Tracker checks whether the requested .torrent metafile is

Table 4-6 Illustration of Download Content Response Task in the XML message

```
<task id="70" name="DownloadContentRES">
        <file name="test1.data" path="c:\test-results" type="file" source="GID">
        </file>
</task>
```

available. It is important to emphasis that requested .torrent metafile may not be ready for delivery when WS-Tracker receives Download Content Response Task message inasmuch as GTF message exchange mechanism is based on loose coupling design. In other words, the source of the content may send the .torrent metafile of the shared content after downloader's request. Consequently, WS-Tracker has two options to reply incoming Download Content Response Task message.

- If .torrent metafile is accessible, then it delivers it to demanding client in the following message format.

Table 4-7 Presentation of Torrent Data Task in XML format

```
<task id="71" name="torrentDATA">
        <torrent>
            TORRENT BINARY DATA IS HERE
        </torrent>
</task>
```

When the WS-Client receives the .torrent metafile, it parses the metafile to extract the encoded meta-data of the shared content. Next, it starts to actual data download process by asking pieces from the sources and leeches.

- If it is not ready yet, then it sends “torrent Not Available” message to the client.

Table 4-8 Representation of Torrent No Data Task in the form of XML message

```
<task id="71" name="torrentNODATA">
</task>
```

WS-Tracker Client will ask the same torrent file after some specified time, like 30 minutes. Until it obtains the metafile, it will ask for it periodically.

4.6.1.6 Access Control List Request Task

This message originates from WS-Tracker Client to WS-Tracker in order to update access control list (ACL) of a shared content. Similar to Download Content Request Task message, it is a periodical message. It demands to WS-Tracker to deliver given shared content’s ACL. The processes workflow of Access Control List Task is fully illustrated in Figure 4-8.

Table 4-9 Illustration of Access Control List Request Task in the XML message

```
<task id="83" name="ACLREQ">
        <file name="test1.data" path="c:\test-results" type="file" source="GID">
        </file>
</task>
```

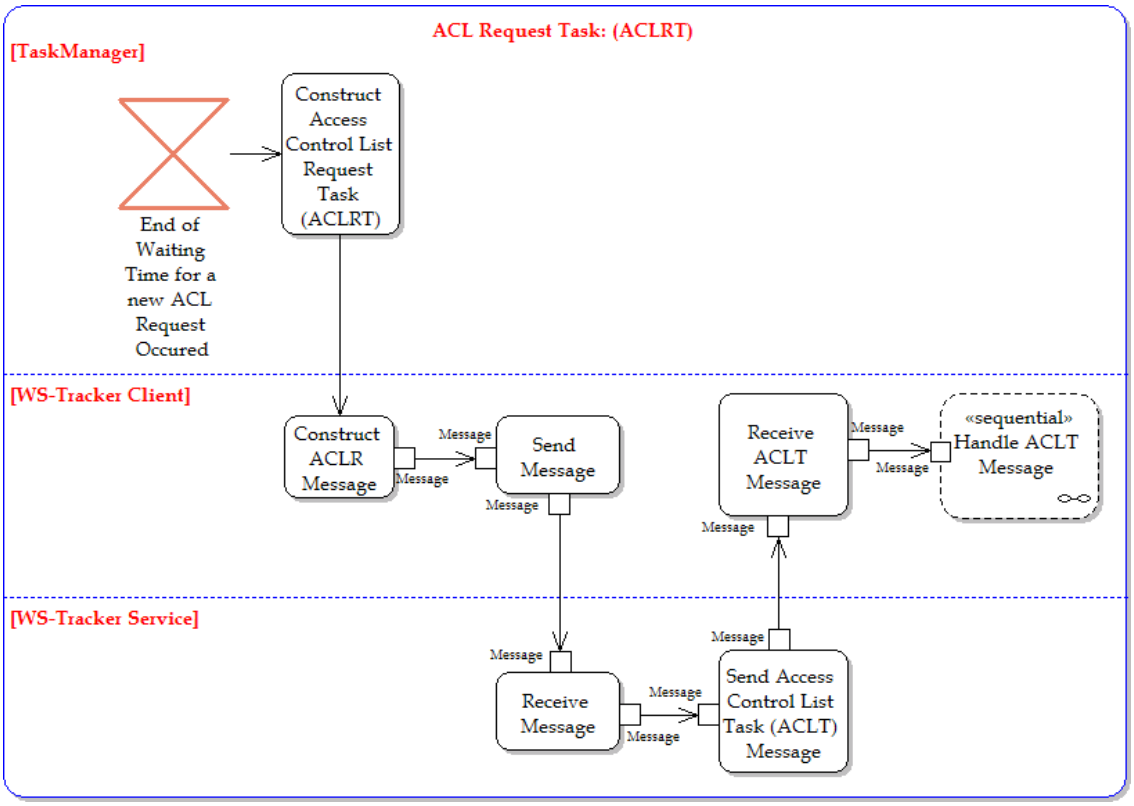

Figure 4-8 Access Control List (ACL) request task's processes workflow

4.6.1.7 Access Control List Response Task

The Access Control List Task message is a reply message to the Access Control List Request Task, and it emanates from WS-Tracker to WS-Tracker Client. It conveys IDs of peers which have a right to download stated content or an empty list. The owner of the content later approves or rejects incoming download connections associated with respect to a given content by checking their provided GIDs against the GIDs represented in this ACL response list.

Table 4-10 An example of Access Control List Response Task in XML format

```
<task id="89" name="ACLRES">
        <peer id="GID1">
        <peer id="GID2">
        <peer id="GID3">
        ……..
        <peer id="GIDN">

</task>
```

4.6.1.8 Update Status Task: (UPT)

This message is used to collect Clients' current statistical information, such as upload, download percentage, etc. Even though, in current prototype, we only provide upload and download information, new fields can effortlessly be added. It is a periodical message and sent from WS-Tracker Client to WS-Tracker. The message flow between various components and process workflow for the update status task is illustrated in Figure 4-9.

Table 4-11 Presentation of Update Status Task in XML format

```
<task id="" name="UpdateStatusCMD">
        <file name="test1.data" path="c:\test-results" type="file" source="GID">
                <upload percentage="0.25"/>
                <download percentage="0.47"/>
        </file>
</task>
```

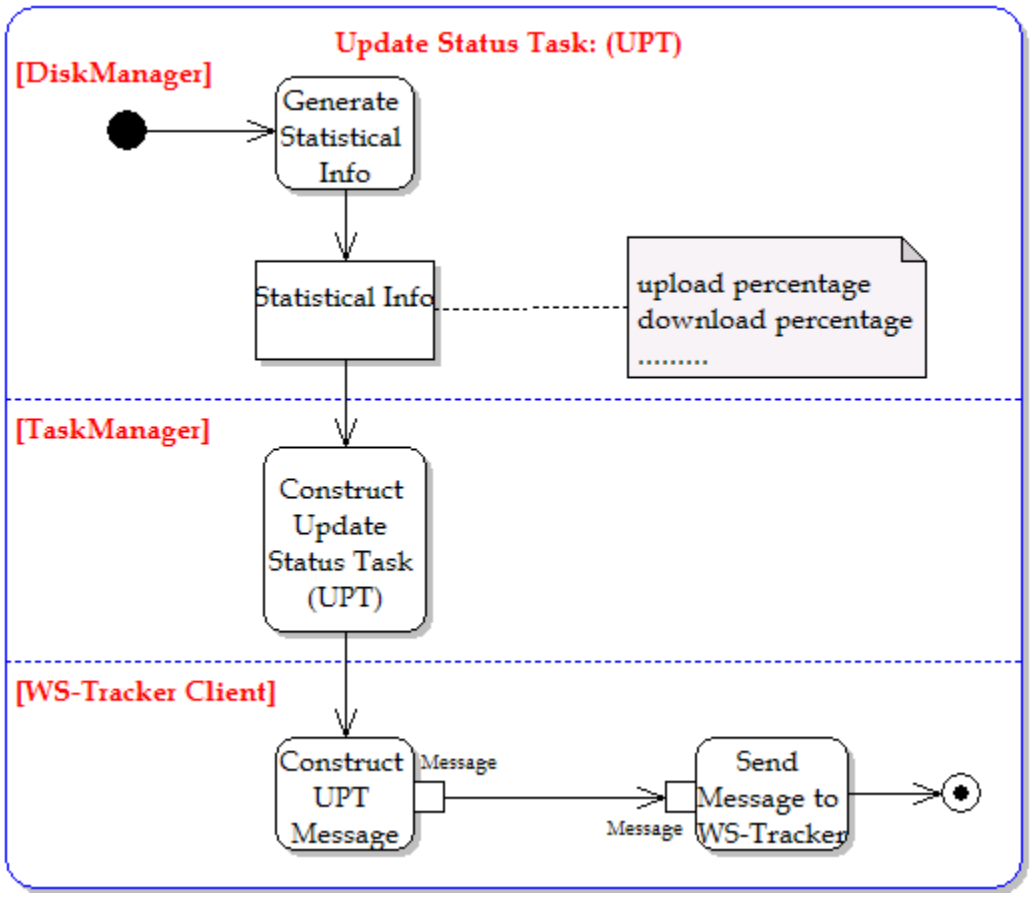

Figure 4-9 Update status task (UST) is one of the important and periodic task that is used to inform WS-Tracker service about current status of the GTF clients

4.7 Summary

In this chapter, we explained the architecture of the GridTorrent framework and motivations and design decisions behind it. Then, we presented a very detailed description of its components: data sharing algorithm layer, data transfer module, core modules responsible for management and task handling and processing, security manager, and finally all tasks. In the next chapter, we are going to explain Collaboration and Content Management which enables the GTF to provide a collaborative environment.

Chapter 5

Asynchronous Collaboration and Content Management Architecture

5.1 Introduction

Bhartrihari, one of the ancient Indian philosophers (c. 450-510 CE?), says "Knowledge grows when shared." This is very true for science as well, when we consider our contemporary knowledge and its output, i.e. technology, as the accumulation of previous collaborative studies and works since the first appearance of ancient thinkers.

In other words, the essence of science is analyzing, suggesting, and sharing ideas, data; to put it simply, exchange and communication. Christopher Surridge, editor of the Web-based journal, Public Library of Science On-Line Edition [109] (PLoS

ONE), enunciates it clearly as saying “Science happens not just because of people doing experiments, but because they're discussing those experiments,” according to [110]. The collaborative projects can be as diverse as that it may be a work of two scientists or of many universities and intuitions scattered around the world. Therefore, there is a growing need for a framework which enables community groups, academics, and scientists to develop collaborative research projects between them.

5.2 The concept of Asynchronous Collaboration and Content Management

5.2.1 **Definition of term “Asynchronous Collaboration”**

Even though *collaboration*, *content*, and *content* management are the buzz-words of the Web 2.0, they have broader definitions in computer environment; therefore, we would like to clarify the use of terms *content*, *content management* and *asynchronous collaboration* as used in this dissertation before we discuss the aspects of asynchronous collaboration and content management. In addition, when we refer to *asynchronous collaboration* feature of our prototype we use *collaboration* and *asynchronous collaboration* interchangeably in the rest of this thesis since the asynchronous collaboration is the only type of the collaboration employed in this thesis.

As collaboration has wider and prevalent usage in today’s computerized world, it is perfectly correct to regard email, video teleconferencing, Internet chat or the World Wide Web as collaboration. Therefore, collaboration can be defined “as the integration of many technologies in to a single environment to facilitate information sharing and information management” according to [111].

The accesses of resources in collaborative systems are other aspect of collaboration and they can be synchronous or asynchronous. In synchronous collaborative systems –real-time collaboration, users work with others at the same moment taking turns communicating ideas and controlling resources. On the other hand, users does not required to be present to participate in asynchronous collaborative systems which allow users to collaborate with other people at their convenience [111].

5.2.2 **Definition of term "Content" and "Content Management"**

The term of *content* is generally used to refer to various kinds of digital media and electronic text such as computer files, image, audio and video files, electronic documents, and Web contents.

Content management refers to create, edit, manage, search and publish digital content. Although the content management term has similar meaning in our prototype, we used the content term with a broader definition in order to include any type of electronic data which is desired to be shared with other users by its owner and can be distributable on computer networks without concerning and processing according to what is stored in it.

5.2.3 **Requirements**

A framework designed for collaborative research projects, in our opinion, should satisfy the following five important requirements.

- It should enable participants to share and exchange their ideas –a collaborative substrate.

- It should provide a basic content management service that allows users to publish and manage their content and search, find and access any data available in the system.
- It should be platform and scientific discipline independent, lightweight and simple as possible it can be.
- It should be customizable and extensible so that organizations with different needs could modify or add new features to it seamlessly.
- It should allow users to transfer their shared data from one location to another one in a high-performance, reliable and efficient way.

5.3 Related Work

There are great deals of commercial and academic products available for collaboration and content management. They are fall into three main categories: (1) systems aimed to deal with great variety of content on a website, (2) systems designed for handling distributed documents produced by office productivity programs (text processing, spreadsheet, etc.) in addition to web content and (3) systems dedicated to courseware management. Bittorrent, SharePoint [112] from Microsoft, Drupal [113] and Sakai [114] from Indiana University are the prominent examples of their area.

Bittorrent provides high performance data transfer techniques but it lacks of content management feature and does not meet requirements for scientific contents such as security, and reliable and comprehensive search feature (Refer Section 2.5.1).

SharePoint was designed as the single portal and developed by Microsoft as an enterprise-level application solution for organizations seeking to deploy for their internet, intranet, and extranets with a consistent user experience [112]. It includes

browser-based collaboration and document management platform. Microsoft's SharePoint consists of two products:

1. Windows SharePoint Service 3.0 (WSS)
2. Microsoft Office SharePoint Server 2007 (MOSS)

While WSS provides platform services such as collaboration, storage, security, management, deployment, site model and APIs for extensibility, MOSS provides five server applications content management, portal, search, business intelligence, and business process management with shared services such as single sign-on (SOS), usage reporting, user profile store, business data catalog services [115]. Although SharePoint offers wide range of collaboration and document management functionalities, it is not suitable for scientific community for several reasons. First of all, as SharePoint only runs on Windows based OS using Microsoft SQL server, it is platform dependent and integrated tightly with other Microsoft products and technologies, for example, ASP.NET, ISS and SQL Server. Secondly, due to its license issue and being a commercial product, it is an expensive solution and adding new capabilities entails daunting challenges. In addition, SharePoint is built on ASP.NET; thus, customizations are done via .NET framework. The third reason is that it is difficult to utilized alternative data transfer mechanisms apart from HTTP and FTP. The final reason is that it is designed to serve small size files by storing and locating files in a central site and controlling them via central administration services, which makes SharePoint not suitable for scientific community since they are entirely independent organizations.

Drupal is an open-source content management system implement in PHP.; it allows an individual or a community of users to easily publish, manage and organize a

great variety of content on a website. The uses of Drupal ranges from community portal sites, news publishing, intranet/corporal web sites, to social networking sites and art, music and multimedia sites [116]. According to [113] the Drupal core includes the following services but not limited to:

1. Basic content management
2. User management
3. Session management
4. Localization
5. Templating
6. Syndication
7. Logging

In addition to core services, it has additional modules which enable Drupal to offer more functionality such as E-commerce, adsense, forums and workgroups. For scientific content and collaboration management, Drupal has several important advantages over Microsoft's SharePoint due to being open-source and platform independent. However, it is impractical for scientific community because not only some of its important features (e.g. e-commerce and adsense) are barren, but also most of its functionalities were not designed to manage distributed large files belongs to different organizations scattered across the world. In fact, the Drupal's primer focus is on managing small files related to websites and they reside on servers owned by a single organization. Similar to Microsoft's SharePoint, Drupal does not provide any alternative data transfer technique as requested files are transferred directly by the web server over HTTP as a static file and Drupal is not involved at all [113].

The Sakai Collaboration and Learning Environment (CLS) is a free and open-source Courseware Management System. The Sakai project began January 2004 when four universities –Indiana University, Stanford University, the Massachusetts Institute of Technology, and the University of Michigan- decided to replace their learning systems by a common courseware management system developed together [114]. As it is seen from some tools of Sakai listed in Table 5-1, the primary motivation behind the Sakai project is to accomplish a framework with rich functionality that will support pedagogies in all disciplines [114] and it does not provide a dedicated tools that manages and transfer geographically distributed large files stored on divergent systems. Currently, the Sakai CLE is used at over 160 educational institutions, in productions settings ranging from 200 to 200,000 users [117].

Even though above-mentioned products have great features, as their services are sophisticated and comprehensive and they were designed for being a management framework instead of being a content management framework, they are heavyweight process and require considerable system resources such as computational power and memory; therefore, they alone are not suitable for scientific content management. Furthermore, they provide services to manage only web content or they lack of high performance data transfer capability by using simple HTTP or FTP as data transfer mechanism. However, due to their rich functionality and modular structure or sophisticated web content management systems such as portals, lightweight frameworks designed to manages distributed files, for instance The GridTorrent Asynchronous Collaboration and Content Management framework, can be integrated with them seamlessly.

In addition to enterprise level solutions, a great deal of effort has been expended in studying this subject and many systems have been developed in order to satisfy the growing need to provide collaborative environment in scientific community. In Grid community, Replica Location Service (RLS) [48] can be considered as collaborative tools in an aspect of finding where existing files are located in the Grid by providing a framework for keeping track of one or more replicas in a Grid environment. GridFTP is

Table 5-1 Partial List of Sakai 2.5 Tools

A set of generic collaboration tools of the core of Sakai.		**The core tools can be augmented with tools designed for a particular application of Sakai.**	
		Teaching Tools	***Portfolio Tools***
Announcements	Preferences	Assignments	Forms
Drop Box	Presentation	Gradebook	Evaluations
Email Archive	Profile / Roster	Module Editor	Glossary
Resources	Repository	QTI Authoring	Matrices
Chat Room	Search	QTI Assessment	Layouts
Forums	Schedule	Section Management	Templates
Threaded Discussion	Search	Syllabus	Reports
	Web Content		Wizards
Message Center	WebDAV		
Message of the Day	Wiki		
	Site Setup		

another example of collaboration tool of Grid community used for high-performance data transfer.

In addition to previous studies and works, emerging technologies offers new tools suitable for scientific collaboration. For example, the technologies of Web 2.0 provide a new way of sharing and interacting to the end users by presenting user-oriented social networks, wikis, blogs, and information-tagging devices. Moreover, being not only more collegial than the traditional variety, but also considerably more productive is another attractiveness of the Web 2.0 based collaboration. Although, offered facilities are very important and it promotes the productivity, they address only the first requirement and are used a small but growing number of researchers, yet their efforts are still too scattered [110].

The OpenWetWare [118] project at MIT is another example of collaboration project which harnessed Web 2.0. It is designed to collaborate on synthetic biology. OpenWetWare, a collaborative Web site, is based on wiki that can be edited by anyone who has access to it [110].

A great deal of studies and efforts in this area are solutions either that deal with just some aspects of the collaboration issues instead of all aspects of them or, likewise OpenWetWare, clearly fall into the category of discipline specific studies which is almost impossible to use for another scientific discipline because of imposing data type germane to particular discipline . Even though some of them are very successful at their targeted field, they still suffer either from above-mentioned shortcomings or from not

having been addressed some very important requites such as privacy and security issues. For example, the privacy issues are hotly debates in scientific communities whose members use the technologies of Web 2.0 [110].

As a result, bearing in mind the deficiencies of previous works, we developed a system that is highly platform and scientific discipline independent and capable of high-performance data transfer technique and provides services in order to enable users to share, search, and find their contents in the system.

In the previous chapters we explained the motivation and rationale for the GridTorrent Framework Client Architecture which is responsible for high performance data sending and receiving with acceptable level of security.

In this chapter, we extend more details about low level and architectural design decisions of the Collaboration and Content Management (CCM) component of the GridTorrent framework which offers a collaborative environment. Users can share, search, access, and manage their contents by the tools provided by the CCM. In addition to explanation of high level overview of the CCM, we present a thorough description of key components and their implementation.

5.4 Access Control Schemes

It is inevitable that distributed computer systems require at least one mechanism to restrict system access to authorized users. As collaborative systems permit their users to access other users' resources, access control scheme is a vital requirement for collaborative systems as well. In this section we examine existing access control strategies for collaborative system.

The simple access control mechanism is Access Control Matrix (ACM). In this scheme, current allowed accesses of the subjects, which are active protective entities such as users and processes, are controlled using a matrix which defines the access rights of each subject associated with each object in a system [119, 120]. Access Control Lists is the one of the implementation of access matrix model. Objects that associate with subjects and rights lists of a set of pairs –subjects and rights- are stored. Capability Lists is the another mechanism used to implement access matrix model. In this implementation, subjects that associate objects and rights with lists of a set of pairs -objects and rights- are stored.

In Role Based Access Control [119, 121-123](RBAC) model, access decisions are based on the roles predefined in organizations. Access rights are grouped by role name and subjects take on assigned roles. The access of system resources is authorized according to subjects' roles not their individual identity. Privilege and Role Management Infrastructure Standard [124] (PERMIS) , an implementation of RBAC model, is an authorization infrastructure based on the X.509 Attribute Certificate. Whereas RBAC scheme is very effective for collaboration systems because of simple administration and scalability issues, ACM scheme is efficient in flexibility and fine-grained control issues.

As settings of ACM and RBAC based systems are relatively static, they are suitable for asynchronous collaborative environments which require no or little real-time interactions or access control settings. On the other hand, those systems lack to coordinate concurrent activities and to manage synchronous resources in synchronous collaborative systems. Floor control [125, 126] scheme developed "to manage joint or

exclusive access to shared resources" [127] and to maintain shared state consistency in synchronous collaborative environments such as conference and media session setup, conference policy manipulation, and media control.

As users of the Collaboration and Content Management framework are not required to collaborate with other users at the same moment, the CCM display more the characteristics of asynchronous collaborative systems than that of synchronous collaborative systems; thus, we used ACM and RBAC based access control models in our prototype.

5.5 The Collaboration and Content Management

The CCM is a software application that provides a system capable of managing users, providing services to the users to govern their contents and access rights of their contents, building their collaborative team, and administering their teams. It has been developed mainly using Java, JavaServer Pages and Java Beans technologies.

The human user is the sole actor of the CCM. He uses the services (Refer Section 5.2.2 for full list of services and detailed descriptions of them) offered by the system either to build a cyber collaborative environment for his work or to benefit from collaborative environment established beforehand. All actions of the users are recorded into storage server. We used MySQL [47] database server for the data storing purpose.

MySQL is a high performance database library with Java bindings as well as for many other languages. It has a simple architecture and provides simple data access and management. Subsequently, some of these recorded actions are converted into a task format (Refer Section 4.6.1) by the WS-Tracker service later. The components of the

GridTorrent Framework and communications taking place among them are shown in Figure 5-1.

The CCM consists of two main modules: Collaboration Management Module and Content Management Module. Figure 5-2 illustrates the subcomponents of them. TheCCM interacts with the user and the database. In the following section, we clearly explicate objects and services that are used and offered by the system. Then, we explain the Collaboration and Content Management modules in details.

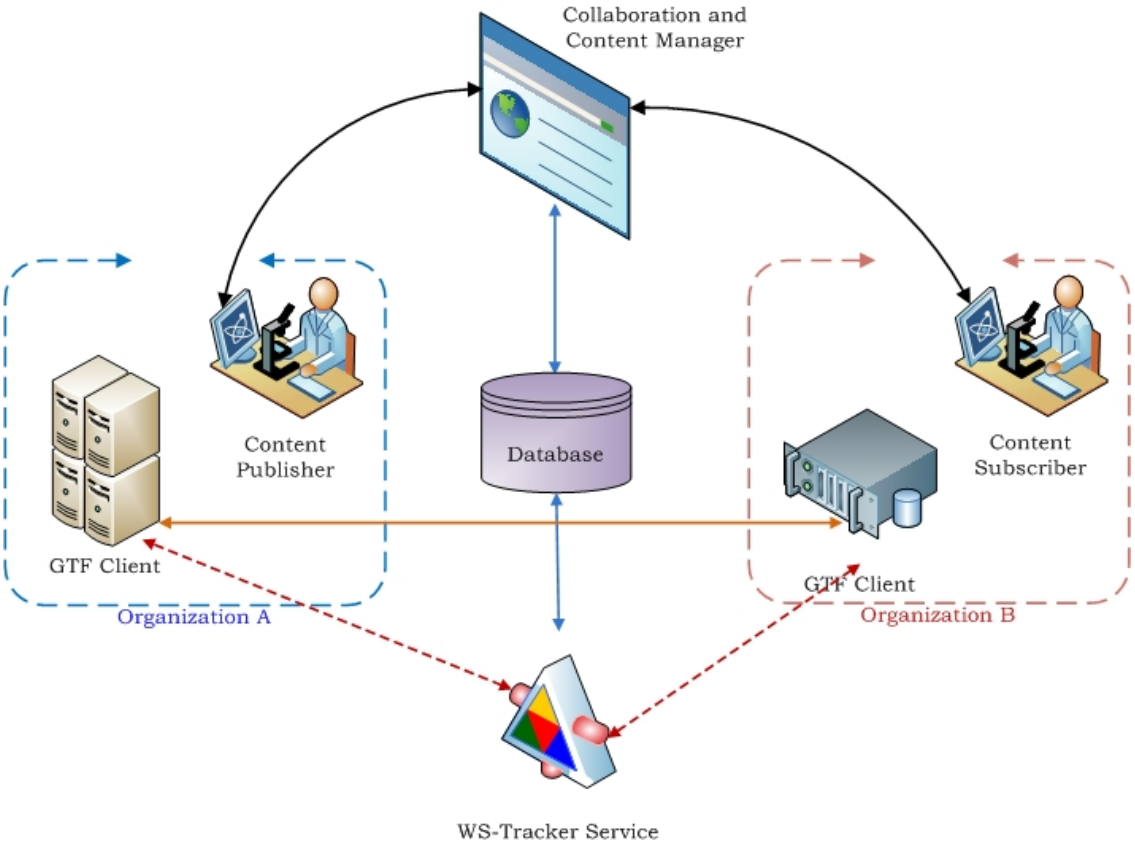

Figure 5-1 The interaction of Collaboration and Content Manager with other entities of the GridTorrent Framework

5.5.1 **Objects of the System**

We refer to any entity that is either a subject of or an object of a service provided by the CCM. Table 5-2 lists the major objects, their brief descriptions, and the services interacted with them.

The term '*a user*' is frequently used in many systems, but not in our system, for an entity that can be either a machine or a human who uses the system. When we refer to a user, this term includes only a human user who interacts with the CCM and can be either a collaborative team administrator/member, or content publisher/subscriber, or all of them.

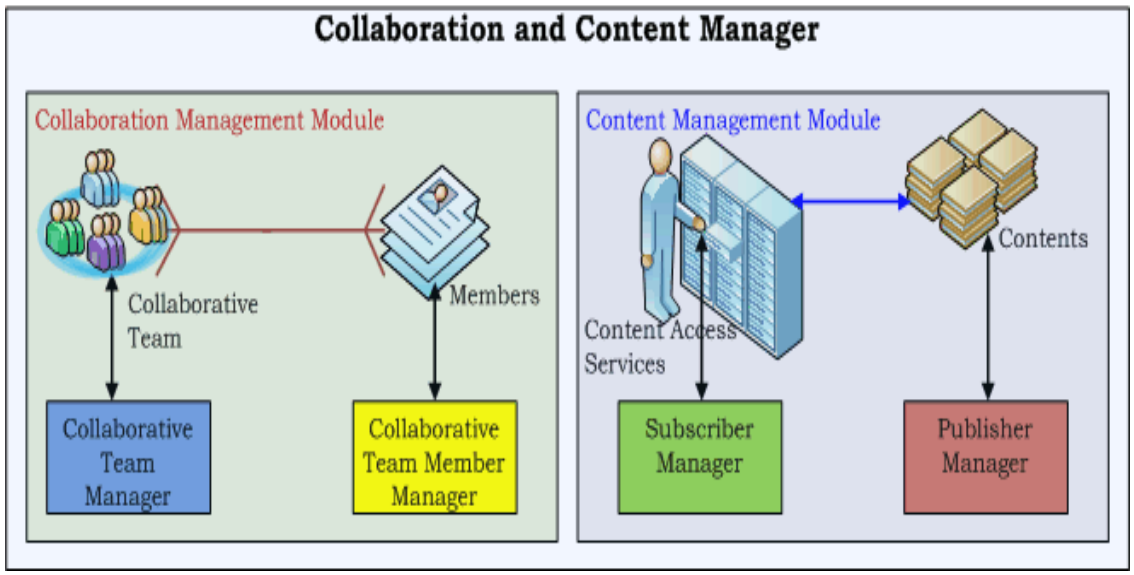

Figure 5-2 Anatomy of the Collaboration and Content Management module

A Collaborative Team is a group of users who are usually geographically dispersed and share common interest for particular contents germane to their work or project. It is composed of collaborative team members.

Table 5-2 Overview of objects used in the Collaboration and Content Manager

Object Name	Description	Services
User	a human who interacts with the CCM	registration, forming a team, disbanding a team, joining a team request, leaving a team request, searching/browsing contents, publishing/downloading contents
Collaborative Team	a group of users usually geographically dispersed and share common interest for particular subject	informing team members about newly published contents
Collaborative Team Administrator	a user initially builds a collaborative team and manages it	approving/rejecting requests, managing team
Collaborative Team Member	a user works on a project relevant to or depends on other team members projects	searching, publishing contents, downloading contents
Request	a demand for joining a particular team	-
Content	meta-info of any computerized data that can be a single file or a directory structure desired to be shared among users	-
Publisher	a user who owns contents and shares them	publishing contents
Subscriber	a user interested in other users' contents	downloading a contents

Collaborative Team Member is a user who works on a project that is relevant to or depends on other team members projects. Therefore, a team member either publishes contents for the utilization of other team members or subscribes to contents published by the other team members in order to use them in his work. For example, in replication use case, both replica master and replica slaves are both team members; however, the

former is a collaborative team administrator and a publisher, yet the latter is just a collaborative team member and a subscriber.

A Collaborative Team Administrator is a user who initially builds a collaborative team and manages it. The management process involves approving or rejecting requests of group enrollment. A collaborative team administrator is a collaborative team member as well.

A Request is a demand for being a member of a particular team. It is submitted by a user who desires to join a team related to his subject and approved or rejected by the administrator of that team.

Content is a meta-info of any computerized data that can be a single file or a directory structure which is desired to be shared among users interested in it. It can be shared among all users without any restriction, the members of a particular collaborative team, or preselected users. To keep track of data and locations, with the file name, a file info hash created by a GTF client in order to distinguish files is mapped to the physical location where that file is stored. It needs to be clarified that the content object in the CCM is a mere meta-info and does not contain any actual data at all.

A Publisher is a user who owns contents and shares his contents with others. *A Subscriber* is a user who is interested in the contents of other users' and highly likely willing to download their contents.

5.5.2 **Services of the System**

Services are the actions that may change the status of objects of the CCM directly or indirectly. Services are initiated by either a user or another service. The

major services, their brief descriptions, and the doers and subjects of those services are listed in Table 5-3.

Table 5-3 Summary of services used in the Collaboration and Content Manager

Service Name	Subject	Object	Description
Registration	User	User	Permitting a user to enroll the CCM
Forming a team	Collaborative Team Administrator	Collaborative Team	Permitting a user to build a collaborative team
Disbanding a team	Collaborative Team Administrator	Collaborative Team	Permitting a user to remove a collaborative team
Joining a team request	User	Collaborative Team	Enabling a user to make a joining request for desired a team
Leaving a team request	Collaborative Team Member	Collaborative Team	Enabling a user to make a leaving request for desired a team
Approving a request	Collaborative Team Administrator	User	Allowing a team administrator to approve joining a team request
Rejecting a request	Collaborative Team Administrator	User	Allowing a team administrator to refuse joining a team request
Team management	Collaborative Team Administrator	Collaborative Team	Allowing a team administrator to assign access control rights to team members
Publishing contents	User / Collaborative Team Member	Content / Collaborative Team	Enabling a user to share his contents
Downloading contents	User / Collaborative Team Member	Content / Collaborative Team	Enabling a user to download contents
Searching contents	User / Collaborative Team Member	Content / Collaborative Team	Enabling a user to find contents
Browsing contents	User / Collaborative Team Member	Content / Collaborative Team	Enabling a user to find contents

Registration service permits a user to enroll the CCM to access the features provided by the CCM. It is a service used only once at the outset. First and last name of the user, username, password, institution, and telephone number are the required information at the registration stage. It is the responsibility of the user to select his username and password. However, when he selects his username, he needs to comply with the rule that the selected username has to be unique in the CCM, since they are used for identifying users.

Forming a team is a service used by a user, called as a collaborative team administrator, to build a collaborative team. The name of a team must be unique, similar to username.

Disbanding a team service is the opposite of the forming a team service. It enables the team administrator to delete the team from the system. When a user desires to join or leave a particular team, he uses *joining* and *leaving a team requests* services. The administrator of the team, which the user wants to sing for or leave, either grants or refuses these requests by using *approving* or *rejecting a request* services. In addition to approving and rejecting a request services, the team administrator uses *team management* services to manage the team members and to assign access rights for them.

Publishing contents service is utilized by any users who desire to share their content. *Browsing* and *searching contents* services are used to find a specific content. Finally, *downloading contents* services are employed to start actual data transferring process between the GTF clients.

5.5.3 **Collaboration Management Module**

Collaboration Management Module offers services that enable the end users to establish their collaborative environment by forming their teams and managing their teams and members of their teams. Collaboration Management module is composed of Collaborative Team Manager (CTM) and Collaborative Team Member Manger (CTMM) subcomponents as illustrated in Figure 5-2.

The CTMM offers services (Refer Section 5.2.2) to the users to build or remove a collaborative team, and provides services to a collaborative team administrator to govern its members. The CTMM component allows the users to form their collaborative team member list without forming a group. An owner of contents can choose individual users by name either from a list of all known users or a self-maintained collaborative team member list. This unit is quite helpful for a small collaborative team which consists of very few people working on a small size project.

After building a collaborative team, the process of a collaborative team membership is started either by the collaborative team owner or by the user who makes a request for a desired group. In both cases, team membership is activated after acceptance of both sides. During the joining a team process, the collaborative team administrator can assign a new member either an admin role or a user role. Figure 5-3 displays the possible roles and their hierarchy, and rights available for a new team member. After deciding the role of the new member, the permissions to perform publish or browse contents operation is assigned to user role.

Following the team membership approval, the new member can publish or browse contents. If the right of publishing contents is granted, the member can publish

his contents for the usage of whole team members. When a new content is published into the team, the system will generate a Download Request Task (DRT) (Refer Section 4.6.1.4) on behalf of a team member if that user authorizes auto-download option. Auto-download option is a particularly practical feature in the case of certain collaborative projects, in which automated data replication on different physical locations is required.

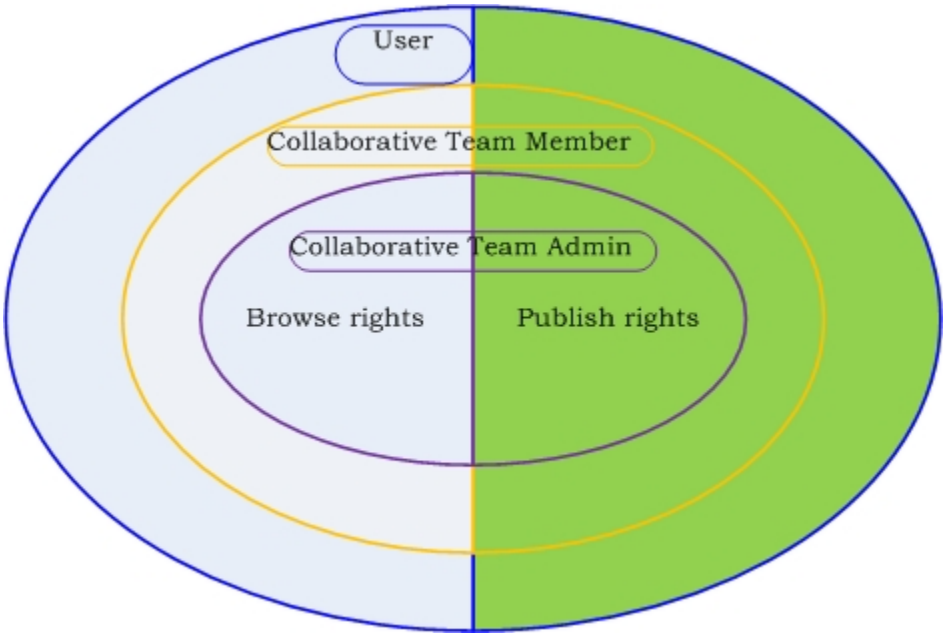

Figure 5-3 Possible roles and rights in the Collaboration and Content Management Module

Team members with browsing contents right can only download contents published to team. This feature is again very beneficial for read-only replication models such as the master-slave type. In this models slave replicas should always be identical to the master replica.

Contrary to the process of a team membership application, the resignation of a team membership is a one-sided activity because it does not require endorsements of both sides. Either the group owner or the user can revoke it.

5.5.4 **Access Control Mechanism of CCM**

To prevent unauthorized content access, we used the simplified and modified version of traditional role-based access control system. In our system, three roles are available for a user: ordinary user, collaborative team administrator, and collaborative team member. Collaborative administrator grants admin or user role for a new member with publish or browse access right as illustrated in Figure 5-3.

Table 5-4 Access levels offered by the Collaboration and Content Manager

Name	Description
Public level access	Contents with the public level access are visible to every user. They can be downloaded by any user without any restriction. All users have public access level rights.
User level access	Contents with the user level access are available for only users selected by the owner of contents. Content owners grant this right to user whoever they selected.
Collaborative team level access	Contents with the collaborative team level access are visible to every member of the team. Team members can download them without any restriction. Only Collaborative Team Administrators approve team level access to users.

As listed in Table 5-4, three types of access level are supported; public level, collaborative team level, and user level access. While public access level permits users to make their contents available for all users, collaborative team access level permits

users to share their contents with only members of that team. In user level access, content is offered to usages of users who are selected by the content publisher. The Publisher Manager takes the public level access as the default access level, unless the owner of the content sets something different.

In order to avert unauthorized content access, user and the content must be on the same access level. One user can have more than one access level at the same time. Public level access is given to every user after a successful registration process. For example, any user can browse, search, and download any contents with the public level access. When a user is included in the collaborative team member list of another user, or joins a collaborative team, he is granted for user and collaborative team level access respectively.

5.5.5 **Content Management Module**

The Content Manager module allows users to share their files with selected access control rights by providing services of publishing, downloading, browsing, and searching for contents. These services are explained in Section 5.2.2. As it is shown in Figure 5-2, the Content Manager module is comprised of Publisher and Subscriber Manager subcomponents.

Publisher Manager Module empowers user to distribute their collaborative contents among other user. Every content must be published with an access level right; thus, the procedure of assigning an access level for every content file is an imperative operation.

Subscriber Manager Module offers content access services; for instance, browsing contents and searching contents services. These services enable users to acquire to a particular content easily according to access level of contents.

5.6 Summary

In this chapter, we explained the architecture of the Collaboration and Content Manager component which provides a collaborative framework where users can create their collaborative team, share their contents, or download contents offered by other team members or other users by using the services provided by the CCM. Then, we provided very detailed description of its components: Collaboration Manager and Content Manager. We also discussed our access control system which restricts content access to authorized users. In the next chapter, we are going to describe the third major component of the GTF; WS-Tracker Service which assists in the communication taking place between the GTF clients.

Chapter 6

WS-Tracker Architecture

6.1 Introduction

In this chapter, we describe the architecture of the WS-Tracker Service and the motivations and goals behind it and argue its benefits. Moreover, we provide more details about low level and architectural design decisions, as well as a thorough description of key components and their implementation.

As it was explained in previous chapters, the components of the GridTorrent Framework form a simple distrusted system. Each of its components is independent from each other, performs dissimilar purposes and being, hence they operate on a different physically located machines. However, there is still a need for a component that assists in the communication taking place between the GTF peers, and conveys the

information that is generated by the users through the Collaboration and Content Manager to their GTF peers.

In order to satisfy the need for a coordinator component in our system, we introduced WS-Tracker Service, a modified version of BitTorrent tracker with many added features. Notwithstanding some basic similarities, there are quite differences between them in regard to the functionalities of WS-Tracker Service.

The tracker, in BitTorrent [86, 88], is a basic HTTP/HTTPS service that responds to HTTP GET requests. The main advantage of it is to use HTTP protocol, since HTTP is ubiquitous protocol. However, it is not suitable for an environment that is very dynamic and requires complex services (explained in next sections) to coordinate participating nodes, and communications taking place not just between the GridTorrent Framework peers, but even between the users and their GridTorrent Framework peers.

In BitTorrent, the communication happening between peers and tracker is passive communication; in other words, the tracker only delivers a list of available seeders and peers of a requested file, and collects statistics of uploading and downloading processes from the peers.

File downloading process is the only required responsibility of a general BitTorrent peer (Refer Chapter 4 for further details about comparison of GridTorrent Framework peer and BitTorrent peer) and each downloading task is independent from each other. After the initial communication, peer can continue its downloading process without the help of its tracker. However, in GridTorrent Framework, a tracker plays a maestro role between the Collaboration and Content Manager and the clients of the

GridTorrent Framework, and among the clients of the GridTorrent Framework. Therefore, a GridTorrent peer needs a tracker not only to download a file but also to receive its future tasks assigned by its owner.

6.2 Web Service

We overcame the shortcoming of a BitTorrent's tracker, which is a simple web application, with the help of *Web Service* technology that enables us to extend or add complex services for GTF tracker. The World Wide Web Consortium (W3C) defines a Web Service as following [128]:

> *A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.*

In other words, it standardizes the interface, operations provided by Web-based applications, discovery of the operations, and the message format exchanged for delivering and receiving service. This standardization is being accomplished by using Extensible Markup Language (XML), defining Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Universal Description, Discovery and Integration (UDDI), and using mainly Hypertext Transfer Protocol (HTTP), although supporting a variety of Internet protocols, such as Simple Mail Transfer Protocol (SMTP) and Multipurpose Internet Mail Extension (MIME). While XML is a general-purpose specification developed by the W3C and, in Web Service

context, used to tag the data, SOAP is an XML-based messaging protocol and universally used to transfer data. WSDL is an XML-formatted language used to describe the services available and data types of exchanging messages. UDDI, sponsored by OASIS, is a Web-based distributed directory and used for Web Services registering themselves and discover each other [129, 130].

Because of the standardization, Web Services have many advantages but two of them are the most significant of them and deserve to be mentioned: being (1) loosely coupled and (2) platform independent.

Loose coupling hides the implementation logic from the callers and puts an end to the overhead through making requirements and few assumptions about each end of the communication by sticking to provided service contract, i.e. programmatic interface, instead of the underlying implementation details. Because it is implementation of Service-oriented architecture (SOA) [131], it possible to build more complex services using collection of simple services.

Using open standards that includes XML, SOAP, WDSL, UDDI, and HTTP and easier to implement allow any two Web-based applications communicate with each other without regard to their running hardware platforms and operating systems or programming languages. It is also major difference between Web Services and its competitors of object-model-specific protocols such as the industry standard Common Object Request Broker Architecture [132] (CORBA) or Microsoft's Distributed Component Object Model [133] (DCOM) or Remote Method Invocation [134] (RMI) or Internet Inter-Orb Protocol [135], and gives Web Services a better chance of being

widely implemented. For example, an application written in Java and running on Microsoft Windows OS can talk with one written in C++ and running on UNIX.

Some of Web Services specifications have been developed or some of them are still under development to extend Web Services capabilities in order to meet the newly emerged requirements. These specifications have come from the W3C or OASIS or a coalition of vendors such as Sun Microsystems, IBM, Microsoft, BEA Systems, Oracle, and Tibco. Though there are many specifications associated with web services, we only provide a list of important ones [136] with short description here:

- **WS-Security** defines how to use XML encryption and Signature in SOAP message in order to provide end-to-end security. It is released by OASIS.
- **WS-Reliability** is specified by OASIS to address reliable messaging requirements that are critical to some Web Services applications. .
- **WS-Reliable Messaging** is, similar to WS-Reliability, addressing the reliability issues in Web Service Context and subsequent of WS-Reliability. It is produced by OASIS.
- **WS-Addressing** is developed by W3C and a specification of transport neutral mechanism to enable Web Services to communicate just using addressing information.
- **WS-Resource Framework** provides a set of operations to implement stateful Web Services. It is developed by OASIS with the major contribution of Globus Alliance and IBM.
- **WS-Distributed Management** is a standard for managing and monitoring the status of other services. It is approved as an OASIS standard.

6.3 WS-Tracker Service

WS-Tracker service is at the hub of the GridTorrent Framework's communications taking place between the CCM and the peers of GTF, and between the peers. In order to provide services promised, it offers a set of operations implemented based on Web Service technology using Java and Axis 1.4.

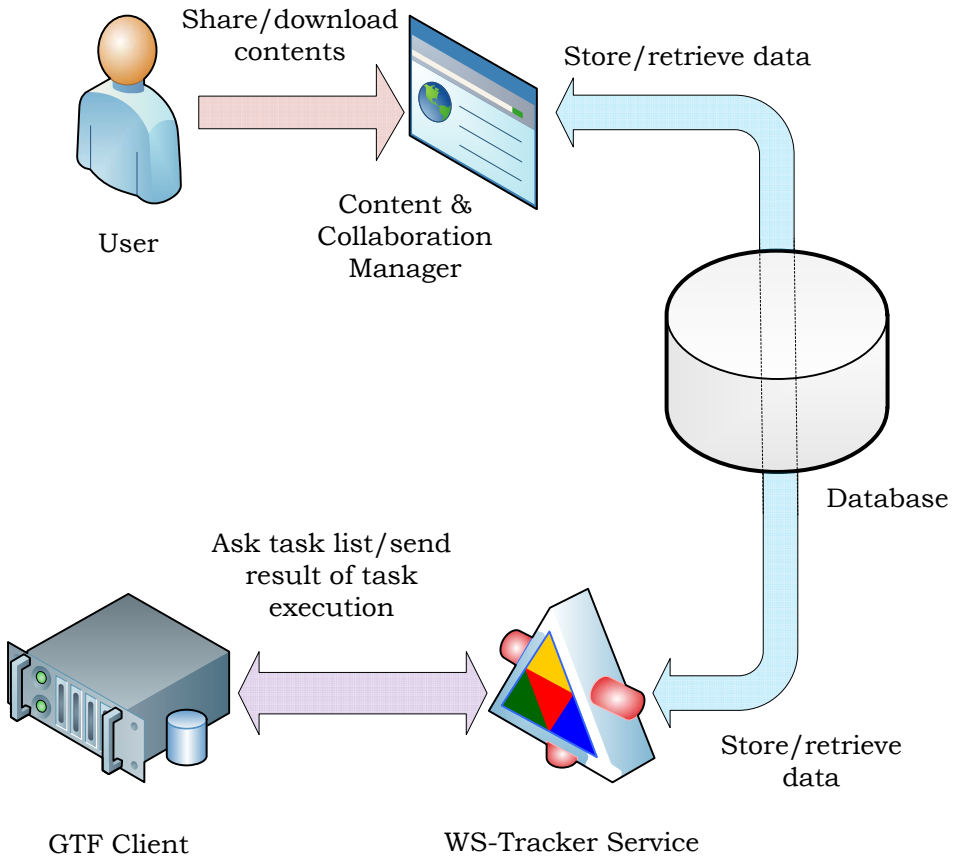

Figure 6-1 The flow of information between a user and GTF Peer through Content and Collaboration Manager and WS-Tracker service

As it is illustrated in Figure 6-1, the user commences the flow of information between a user and his GTF Peer. The user inputs information regarding to shared or downloaded content through the CCM using operations offered by it. For example, a

user makes one of his/her content available for public usage by using publishing contents' service of the CCM. Then, the CCM saves this transaction into a database server accessible both the CCM and WS-Tracker Service. The CCM also retrieves information from the database server to display the status and statistical information of the user's tasks.

WS-Tracker Service obtains information from the database server to deliver task messages (Refer Section 4.6 for type of task messages and handling them) created by the users to corresponding peers. In order to increase performance of the task delivery, we use task cache. To maintain the task cache highly updated, in addition to peer request based database access, we provide a periodical database access taking places in a certain interval of time.

When a GTF peer requests its task lists, WS-Tracker Service first checks the task cache to see whether any task is available for it. If there is no task in the task cache for requesting peer, then it sends a query to database about available tasks. According to the database server response, it updates its task cache and also returns a task list to the corresponding peers.

After receiving task list, a GTF peer begins to process each task delivered in the list. Some of these tasks require additional interaction with either WS-Tracker Service or the other peers. When a task is successfully received or completed, an acknowledgement message is generated and sent back to WS-Tracker Service in order to notify to it.

WS-Tracker Service provides a substrate where the clients exchange information about each other. The interaction between a GTF peer and another peer is

initiated after receiving content downloading task message from WS-Tracker Service. If both peer is successfully authenticate themselves and have authorization to requested content, real data transferring process has been commenced. Figure 6-2 shows overview of this information flow between peers.

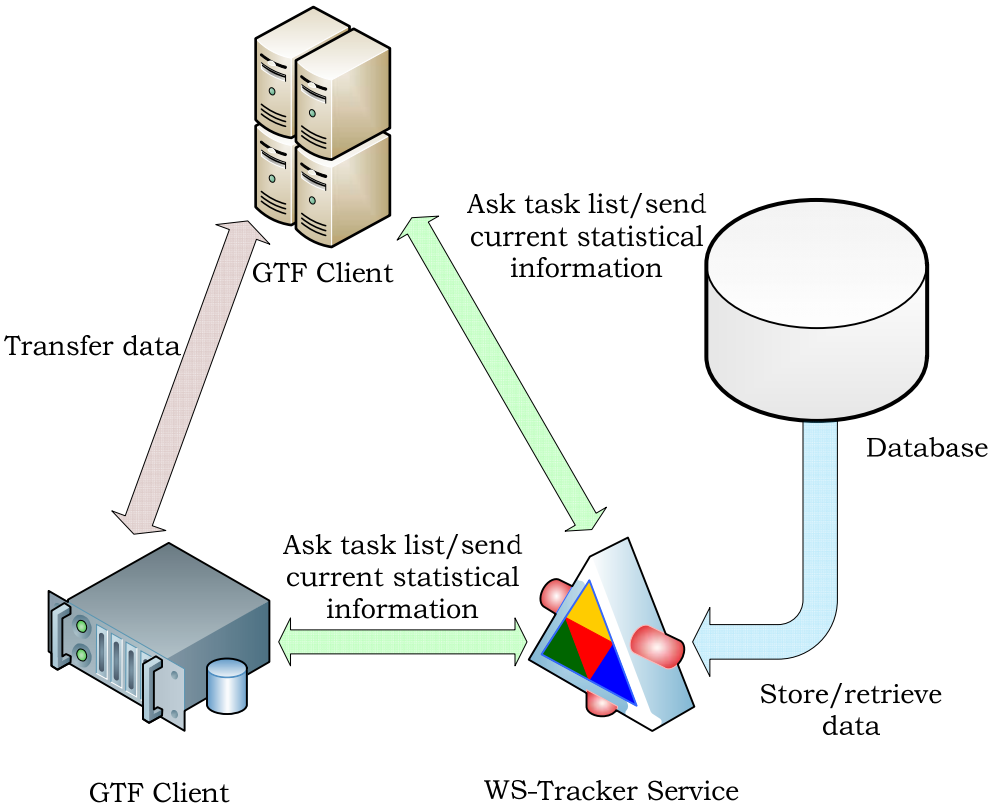

Figure 6-2 Message flow between GTF peers via WS-Tracker Service

6.3.1 **Multiple Trackers**

The WS-Trackers are passive components utilized to exchange task messages between users and their GTCs. In this sense, they are just mediums that do not store any important data not recorded in DB. In addition, all GTCs keep their internal state data in order to maintain their consistency. Moreover, the GTCs preserve their existing connections and they do not need to interact with WS-Trackers except for statistical

information update which is not essential. Therefore, in summary, the only vital issue that WS-Trackers are responsible for is to provide a steady connection between users and their GridTorrent Clients. To achieve this goal, each GTC is provided with multiple addresses of WS-Trackers services. If the WS-Tracker service fails for any reason, GTCs connected with it receive timeout exception when they send their statistical information about their current upload and download status to it. In this case, a GTC select another WS-Tracker service from provided tracker service list and upload internal state to the new one if the current one fails.

6.3.2 **Fault Tolerance**

The failure of WS-Tracker can be categorized into single instance failure and multiple instances failure. By single instance failure, we meant that there is only one WS-Tracker Service functioning in the whole system and it fails. Unlike single instance failure, there are more than two instances serving to peers in the system and at least one of them successfully functions when the others fail. In the latter failure case, when the all trackers fail, it becomes similar to single instance failure case. In both cases, we presumed that the shared database is replicated in order to eliminate the issues that arise when it fails for any reason. In the same way, all the transactions that have taken place between the users and the CCM are recovered from the database server easily since they are recorded to database server immediately.

The recovery process of multiple instances failure is relatively simple, since there is still at least one WS-Tracker Service serving to the GTF peers in the event of failure of all other trackers. An inconsistency between the running WS-Tracker Service runtime task list and active GTF peers runtime task list can be regarded as one major

possible problem. However, this can be resolved straightforwardly since the state information on WS-Tracker Service runtime memory is regularly committed to database server. In addition to that, each peer sends an acknowledgement message back to WS-Tracker Service after a successful task receiving or completion process. A GTF peer provides its failure recovery mechanism by regularly saving its runtime task list into a file and using .torrent metafile to resolve any inconsistencies taking place when a downloading process is interrupted.

When a new task list arrives and there is a discrepancy between the peer's list and the received one, it just ignores the incoming one and sends its last acknowledgements to WS-Tracker Service. Thus, available WS-Tracker Service can detect failure effortlessly in a relative short amount of time and update its runtime task cache either with the help of this acknowledgement mechanism or with the help of periodic database access.

For the case of single instance failure, although transactions of the users are stored into database server, the GTF peers cannot obtain their latest task list from the WS-Tracker Service since it is down. Our solution to this case is to start a new WS-Tracker Service and to provide its service address to the GTF peers with the help of their owners.

The recovery process is similar to multiple instance failure case as the new tracker pulls data from the database server and creates its runtime task cache. If any inconsistencies emerge when the new tracker delivers task lists to the peers, they are resolved in a similar way explained in the recovery process of multiple instances failure case.

6.3.3 **Security**

Apart from requesting unique GridTorrent Framework ID of peers in order to deliver correct task list to inquiring peer, there is no security mechanism implemented in WS-Tracker Service. The main motivations behind our decision in favor of insecure tracker are:

- The performance issues
- the metadata characteristics of data delivered by WS-Tracker Service

The performance issues arise as the direct result of encryption and decryption of a large of number of short messages in a short period of time. Apart from the UGTID of peers, IP addresses of peers and, and .torrent metadata file, data from WS-Tracker service contain no actual content data. As a result of this, even if they are captured by an unauthorized user, they would be a piece of useless knowledge for him/her. However, due to its distributed system characteristic, it is still susceptible to several security threats such as but not limited to:

- Distributed Denial of service attacks (DDoS)
- Unauthorized users accessing resource
- Man in the middle attack that impersonates an entity
- Malicious users modifying messages (such as when a messages passes over insecure intermediaries)

There is no quick and easy way to secure against a DDoS. A simple and best solution is to secure computers from being hijacked [128].

In order to receive their task lists from WS-Tracker Service, the peers must provide their UGTID generated randomly to WS-Tracker Service. Requiring their UGTIDs as an identity verification tool renders peers' task lists acquisition very difficult for an unauthorized users. Even if he/she takes possession of the task list, that list is still worthless since it does not contain any actual data or security information.

Furthermore, X.509 proxy certificates are used as proof of identity before starting actual data transfer process. Thus, asking proxy certificate at each peer eliminates any attacks result of false identity. Moreover, peers verify the hash code of every chunk of downloaded data with the one provided by its .torrent before writing it to file system. However, that torrent metafile can be used to validation process only if it is authentic file. In other words, it is obtained from the genuine WS-Tracker Service.

6.4 Summary

In this chapter, we explained the architecture of the WS-Tracker Service that is a Web Service and plays a maestro role between the Collaboration and Content Manager and the clients of the GridTorrent Framework, and among the peers of the GridTorrent Framework. Then, we provided our motivations and design decisions behind it. We also described the information flow between the users and their corresponding GTF peers, and between GTF peers. Fault tolerance and recovery, and security are the main issues we discussed for the rest of this chapter. In the next chapter, we are going to describe security issues in the GridTorrent Framework and how we addressed them.

Chapter 7

Security Modules and Issues of GridTorrent Framework

7.1 Introduction

The data shared between standard peer-to-peer community and scientific community displays different characteristics. The first reason is the nature of data. Whereas the data shared among standard peer-to-peer community is generally obtained from other people's works, like music or movie files, every scientific data is generated in scientific community. In the former, users usually do no attach importance to data integrity or authenticity. For instance, it is not very important that same movie file has different versions as long as it is playable, because people use it for entertainment and

keep it private and do not built anything new on top it. On the other hand, every bit of information is important in the latter case. For example, all information integrity and authenticity in a numerical test result of a scientific experiment is of great importance since new theories, experiment, or technologies are established on it. In addition, despite the fact that the standard peer-to-peer community's data can be shared either modified or unmodified without getting the permission of the content owner –in fact there is no rule about this, the scientific data is more sensitive and requires more complicated sharing rules.

The second reason is base on the characteristics of users. In standard peer-to-peer community, there is no competition between users. In other words, there is one type of user, a passive user, and any user can access any data as long as he or she gets the torrent file. However, in the scientific community, due to expertise or research agenda and competition between institutions, only authorized users are permitted to access to pre-determined data sets with some access rights. While the passive user type in BitTorrent, the users in scientific community area are usually very active and some of them cooperate on some files as a group. This creates diverse users' and groups' profile in scientific community.

The third reason is stemmed from the importance of data and its access. As the current design of BitTorrent itself does not provide a search facility to find files by name or by other keywords; a user must find the initial torrent file by other means, such as a web search. On the other hand, searching, finding, and accessing to desired data are of paramount importance in scientific community, hence a reliable search service must be offered to scientific users. Therefore, even though there is a need for

integration of BitTorrent and content and collaboration framework with a search facility to use BitTorrent in scientific community. As a result, a mechanism involves regulation for content access with pre-defined rights and security is vital for a data sharing system in scientific community.

To build a security infrastructure for GridTorrent Framework, we adopt the most common scientific community standards –the Grid Security Infrastructure (GSI) [103], to allow our data collaborative and sharing framework to be available to a larger scientific community, and our services to be compatible with others.

In this section, first, we present an overview of the GSI. Then, we explain the GTFSI and how we utilized the GSI in the GTF. We also elucidate our approach to build the GridTorrent framework security infrastructure (GTFSI) and give detailed descriptions of the services developed as part of this infrastructure.

7.2 The Grid Security Infrastructure (GSI)

7.2.1 **History of the GSI**

Security has been one of the widely studied fields in Grid community. Hence, the GSI has a rich history to meet new emerging requirements and cooperate with the arising technologies. While mainly message protection and authentication are supported in version 1 in 1998, X.509 proxy certificates and Community Authorization Service (CAS) [137] are introduced in version 2 in 2002. The CAS from Globus team functions as a trusted third party server which is responsible for enforcement of access policies within distributed virtual communities which govern access to a community's resources The CAS serves maintain information about Certificate Authorities, users, servers and

resources with access right lists granted to access resources to each users. When a user issues a request to the CAS server in user's community, the CAS server provides a proxy credential with his or her capabilities to the user. Upon receiving the proxy credential, the user access the resources using his or her CAS credential.

The Grid technology converged with the emerging Web service technology in release 3, and that convergence gave birth to Open Grid Services Architecture (OGSA) [3]. Finally, in the latest version –the 4 release, Web Services security specifications are implemented. Extensible Access Control Markup Language (XACML) and Security Assertion Markup Language (SAML) are the two important authorization related standards [138] supported by current version of GSI [103, 139]. Of course, there are several as well authorization systems supported by Grid Computing, such as Kerberos [140], Akenti [141], PERMIS [124], Shibboleth [142] (GridShib [143, 144]), VOMS [145].

The Shibboleth System developed by Internet2 outlines a proposed architecture to address web single sign-on across or within organization boundaries. The Shibboleth software implements SAML a product of the OASIS Security Services Technical Committee. GridShib is the project which integrates the Shibboleth SAML-based framework and Globus Toolkit's PKI-based security infrastructure to provide attribute-based authorization for distributed scientific communities. With the aid of GridShib, a Globus Toolkit Service Provider can securely request user attributes from a Shibboleth Identity Provider. To enable Grid Computing to support both Web Service security standards and other authorization systems, Multipolicy Authorization framework has been introduced recently [139].

7.2.2 **Overview of the Grid Security Infrastructure**

As Grid computing is concerned with the use of dynamic, diverse resources in distributed "virtual organizations" [145], identifying the users or services (authentication), providing secure communications, and who is permitted to perform what actions (authorization) are the primary issues and challenges in the GSI according to [146]. The GSI development team summarized the above concerns as three prime motivations behind the GSI:

1. The necessity of secure communication between the elements of Grid community.
2. The necessity of supporting security across organizational boundaries to eliminate a centrally-managed security system.
3. The necessity of supporting "single sign-on" for the Grid's users.

To realize the above requirements, Public Key cryptography and the certificate, namely standard X.509, are used as central concept in GSI authentication credentials [137]. There are two type of certificates: (1) end entity certificates (EEC); (2) proxy certificates. Whereas the former is used to identify persistent entities such as users and servers, the latter is used to support the temporary delegation of privileges to other entities [146]. Briefly, a GSI user or server has to obtain a public-private key pair and an X.509 certificate from a trusted entity called a Certificate Authority (CA). As a result, every service, server and user on the Grid is recognized by way of a certificate.

Inasmuch as the GSI merged with Web Services security standards in the 4 release above-mentioned in section 7.1.1.1 and has legacy systems, it supports both WS and pre-WS authentication and authorization capabilities [146].

7.2.3 **GT4 WS Security**

GSI offers two levels of security: (1) message- level security; (2) transport-level security. In addition to that, as depicted in Figure 7-1, GSI supports two message-level protection protocols to realize the different purposes: (a) WS-Security-compliant message-level security with X.509 credentials; (b) with usernames/passwords.

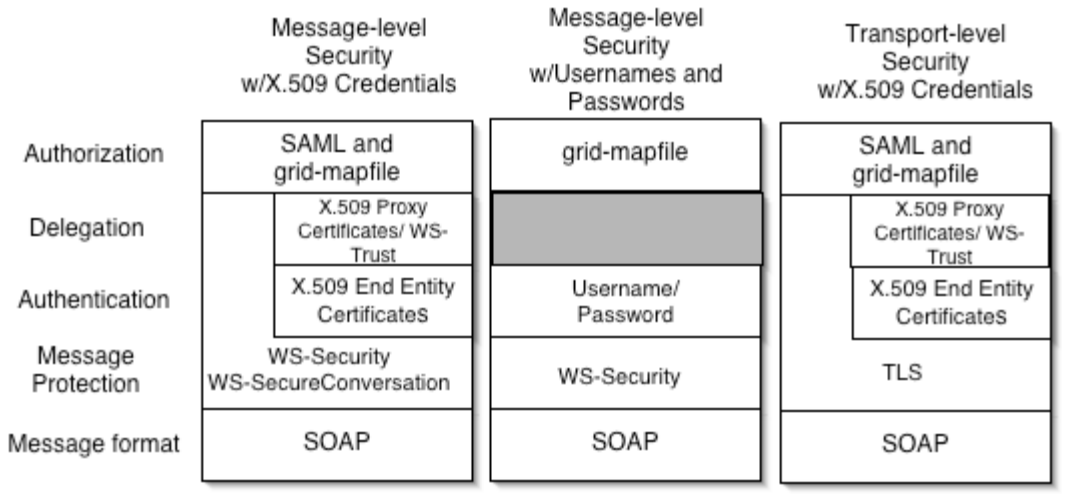

Figure 7-1 GT4 security protocols from [103]

Message-level security provides both protocols with the purpose of complying with the WS-Security, the WS-SecureConversation specification, and the WS-Interoperability Basic Security Profile. However, there are differences between these protocols. Whereas GSI SecureConverversation delivers the full security constraints, GSI Secure Message supports WS-I Base Security Profile, yet it is insecure. Even though, the latter has relatively better performance than the former does, there is a poor performance issue when their performances are compared to that of transport-level schema. Partly an implementation issue and partly a specification issues are the leading

causes of this poor performance of message-level security [103]. As a result, GT4 uses transport-level security by default due to its best performance.

The last schema, transport-level security, has one security schema providing authentication via TLS with support for X.509 proxy certificates [103]. The differences between these there protocols are highlighted in Table 7-1.

Table 7-1 Comparison of transport-level and message-level security from [102]

	GSI SECURE CONVERSATION	**GSI SECURE MESSAGE**	**GSI TRANSPORT**
Technology	WS-SecureConversation	WS-Security	TLS
Privacy (Encrypted)	YES	YES	YES
Integrity (Signed)	YES	YES	YES
Anonymous authentication	YES	NO	YES
Delegation	YES	NO	NO
Performance	GOOD (if sending many messages)	GOOD (if sending few messages)	BEST

7.2.4 **GT4 Pre-WS Security**

The pre-Web Service components use the same authentication mechanisms as the Web Services components described here, but implement application-specific protocols and message protection schemes that are beyond the scope of this document [103].

7.3 The GridTorrent Framework Security Infrastructure

Taking into account the sensitivity of data generated at scientific communities, security issues are of great importance in GTF. We need to address several security issues, some of which are pertinent to content, and some of which are pertinent to peers. Firstly, authenticity of shared contents and verification of their authenticity need to be resolved because the genuineness of data is unessential in typical peer-to-peer file sharing systems. Secondly, protection of the content has to be guaranteed. Thirdly, identification of the participated peers has to be resolved. Finally, authorization of the participated GTF peers for a specific shared content must be verified in order to avert accesses of unauthorized peers as almost all content in peer-to-peer file sharing networks violates copyright; thus, users always prefer to remain anonymous when they access to any content in the peer-to-peer network.

We addressed these security issues in three places of GTF: (1) at Collaboration and Content Manager (CCM); (2) between WS-Tracker and the GTF Peer; (3) among GTF Peers. All possible interactions requiring security measures are displayed in **Error! Reference source not found.**. In order to fully clarify what type of the security services are needed at which interactions, we numbered all interactions that are required to fulfill a complete cycle of a content publishing and downloading from one to nine. We follow the time sequence of processes when we numbered them and used apostrophe to show the concurrent interactions. For example, a process that is numbered as 3 is taken place

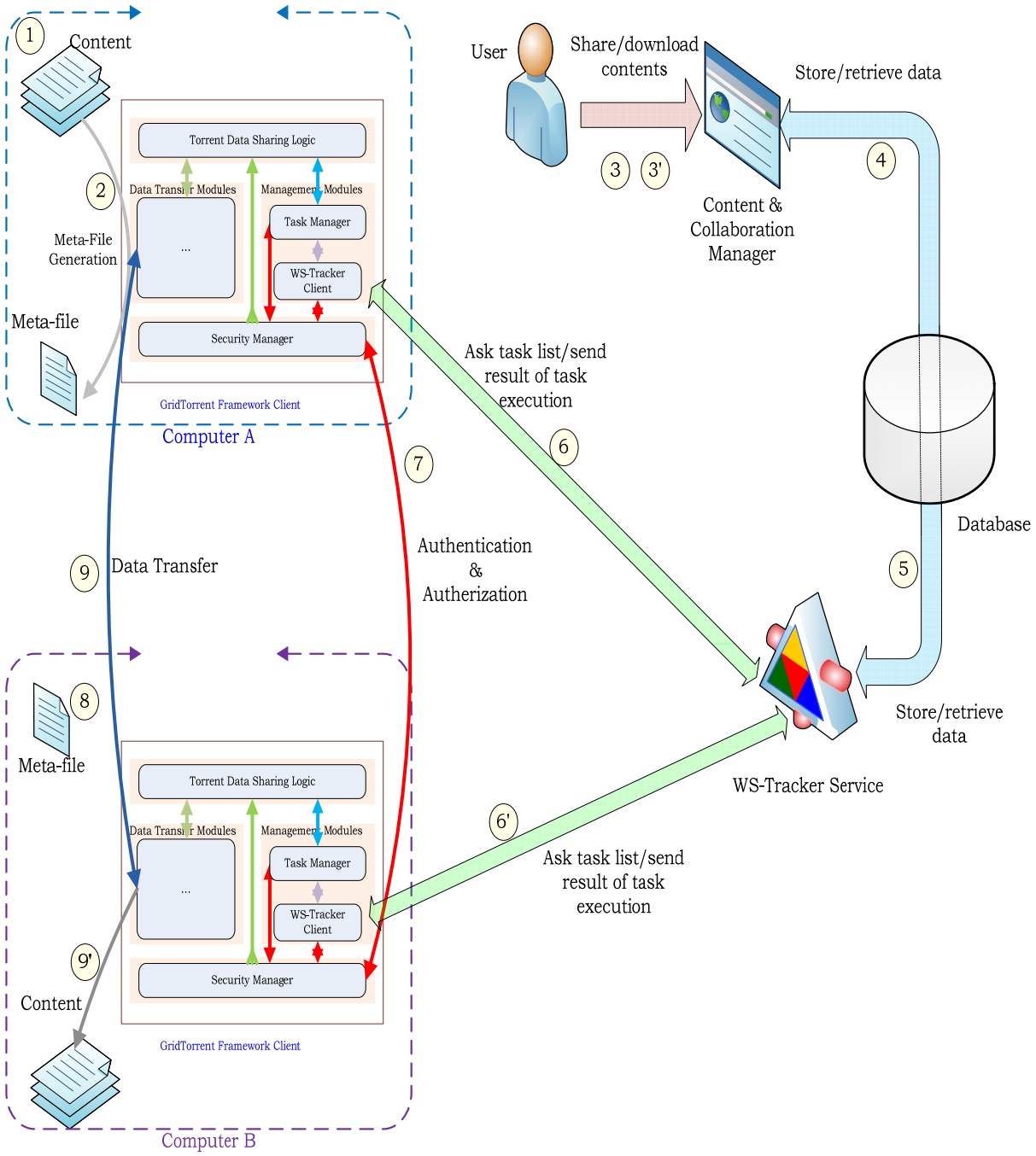

Figure 7-2 All possible interaction among components of GTF require security services.

before all processes labeled greater than 3 and after all processes labeled smaller than 3. The processes tagged as 3 and 3’ indicate that they are independent of each other and may occur concurrently. In the next sections, we explain them one by one.

Both a human user and a computer having been considered as an active system user, we used the rules of etiquette or social protocols and information security so as to perform security requirements.

A human user has direct involvement in the first stage and indirect involvement in the second stage. The possible security problems at this stage that may emerge are spurious content announcement or false meta-file generation. Content generation takes place at stage one and is performed by a human user. It is assumed that every user interacts with each other or GTF according to the rules of academic and social etiquette as GTF is design mainly for scientific communities with the collaboration features. In other words, when the content is published by a user, it is presupposed that either it is consistent with its announcement, or other users have some mechanism to verify its authenticity. Although the GTF Client software is responsible for generating meta-file (.torrent file) of a shared content at stage two, its correctness depends on correctness of content. In addition, it is possible to create false meta-file since the owner has complete access to it to do that. As a result, the rules of etiquette were used to ensure security at stage one and two.

7.3.1 **Security at Collaboration and Content Manager (CCM)**

After a successful registration process, users may publish or download contents at step 3 and 3’ by interacting with CCM. Possible security threats at these steps may be false identity and unauthorized access of information (Refer Section 5.5). Similar to

in stage one and two, the rules of academic and social etiquette were used in step 3 and 3' to address those security threats. Or to put it another way, identity information declared by users are assumed truthful information since users are being members of respected institutions. Moreover, there should be some other communication mechanisms, such as email and telephone number, to verify users' proclaimed information about their identities in order to eliminate possible false identity cases.

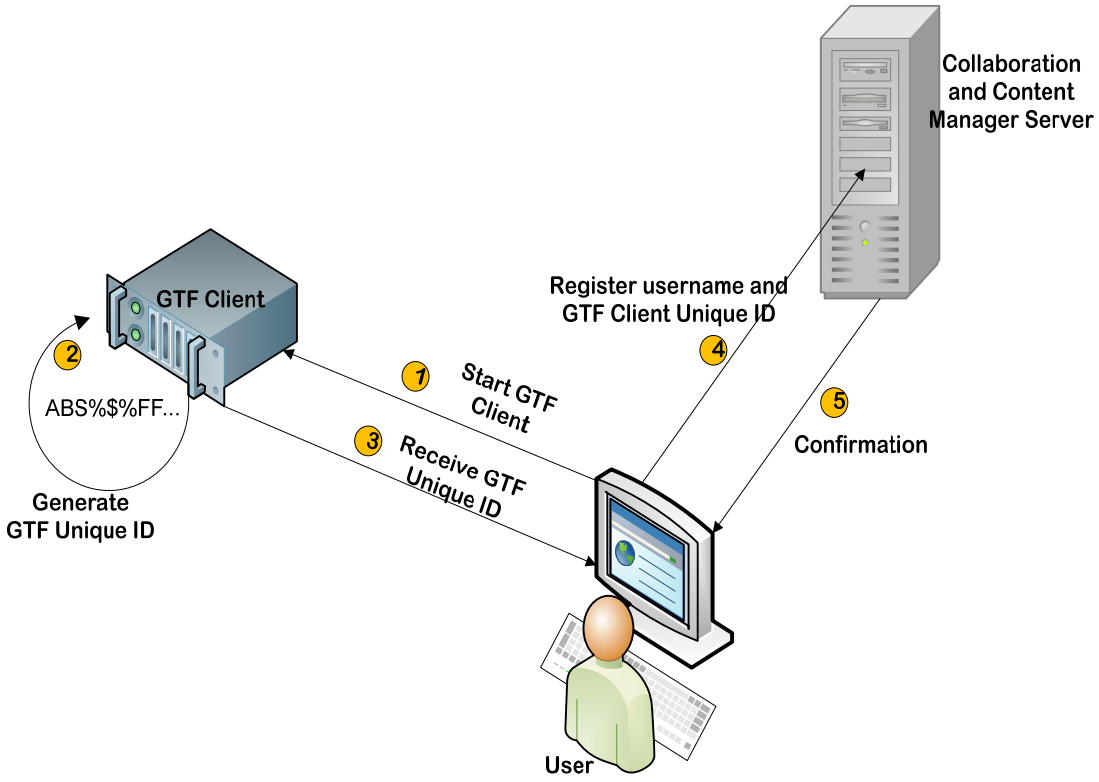

Figure 7-3 Establishing security credentials at Collaboration and Content Manager Server.

Besides access control mechanism in CCM (Section 5.5.4), the information provided by CCM has restricted access in order to prevent unauthorized access; that is

to say, it can only be viewed by authorized people on account of the implemented secure login mechanism.

In CCM, similar to many computer operating systems, a user authenticates himself by entering a user login id and a secret password known solely him and the system. In addition to them, to relate the user and his GTF Client machine and to identify each GTF Client during the data sharing and communication processes, a unique ID is essential for each of GTF Clients and required. Hence, we provide a mechanism in GTF to generate a unique Grid Torrent ID (UGTID). Following UGTID creation, the GTF Client stores it into an ID file for future utilization. By retrieving it from the ID file, the user registers this UGTID with the above-required information into CCM as in presented in Figure 7-3.

We keep the security credentials of the user in database and they are never displayed to public or system coaches in any way. The users are responsible to remember the passwords. On the other hand, CCM administrators may assign new passwords, or login ids, to users. After that, user himself can change the password but not the login id. Login id is the unique set of the characters identifying user in the system. This approach is very compatible with UNIX user authentication.

Another requirement is providing secure communication medium between the user and CCM. Instead of implementing an encrypted data transfer protocol as our own, we used a completely proven TLS (ref) technology to meet that requirement.

The interactions between users and CCM are recorded into a database server through the communication at step 4 taking place between CCM and database server

via JDBC connections. It is protected with username and password level access authorization.

7.3.2 **Security at WS-Tracker Service**

WS-Tracker service is at the hub of the GridTorrent Framework's communications taking place between the CCM and the peers of GTF, and between the peers. As a consequence, several security issues for communications taking place at step 5, 6 and 6' in the figure must be taken into consideration. Authentication, authorization and providing secure communications between the entities are the issues that we must resolve.

A username/password level access authorization is used between WS-Tracker Service and database server at step 5. This type of access is secure enough at our prototype mode. However, adding secure communication layer is an unchallenging task since many newer versions of database servers supports encrypted (secure) connections between database clients and their servers using SSL protocol.

The communications taking place at steps 6 and 6' are between GTF clients and WS-Tracker service. Authentication and authorization are the main security concerns for those steps. Requesting unique GridTorrent Framework ID of peer is the security mechanism that is deployed for the purpose of both identifying inquiring peer (authentication) and delivering correct task list (authorization) to it. At our prototype, no secure communication is provided between GTF clients and WS-Tracker service due to performance issues and the characteristics of the meta-data delivered by WS-Tracker service (Refer 6.3.3).

7.3.3 **Security between GTF Clients**

A GTF client interacts with other GTF clients and WS-Tracker service. Similar to security issues at WS-Tracker Service, identifying GTF clients (authentication), deciding who is permitted to download/upload which content with what actions (authorization), and providing secure communications between the entities are the primary issues and challenges that we must face.

Handling the security issues between a GTF client and the WS-Tracker service was explained in previous section.

To address the above-mentioned security issues, we used the GSI WS-Security schema between the GTF Clients. Accordingly, a user needs to obtain both public-private key pair, and X.509 certificate. Nowadays because of availability of myriad software tools to generate public-private key pair, it is an easy process. As a result, it is user's responsibility to generate their public-private key and initiate the X.509 certification request process as shown in Figure 7-4. Upon user's X.509 certification arrival, it satisfies all the preconditions for secure communications taking place among the GTC Clients and WS-Tracker. Protection of those credentials is user's responsibility as well.

As it is illustrated in Figure 7-2, following the step 6 or 6', the downloading GTF client (Computer B) asks for and receives the metadata file (.torrent file) of actual content since its owner scheduled that task at step 3 or 3'. Afterwards, it starts establishing connections with other GTF clients whose address information is acquired from the WS-Tracker. At step 8, certificate authentication is performed via certification. Due to security reasons, using a temporary MyProxy [108] certificate

generated from permanent GSI certificate is more preferable method in Grid community; hence, we adopted the same method. In short, basic steps of certificate authentication can be explicated in the following way:

- Computer B sends MyProxy certificate to Computer A.
- Computer A uses the CA certificate in order to check that the Computer B's certification is valid.
- Computer A uses the Computer B's MyProxy certificate in order to check from its ACL file whether content access is allowed or not.

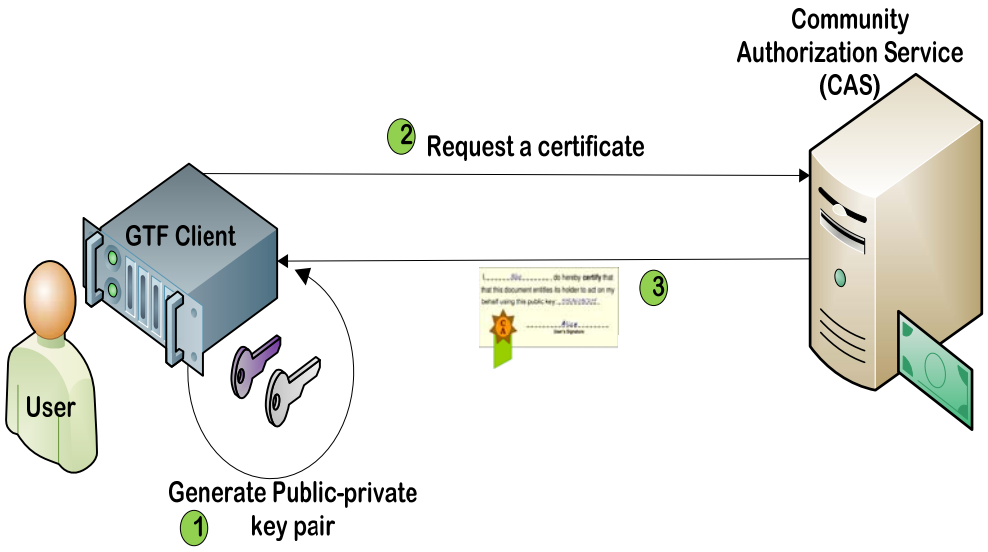

Figure 7-4 Security Credentials obtaining process between GTF and GSI components.

If the Computer B has content download access, after authorization and authentication processes, Computer A allows Computer B to start actual data transferring process occurring at step 9. Finally, all downloaded actual data segments

are coalesced at step 9' after downloading all pieces successfully (Refer Section 4.5). We also summarized the major security concepts used in the GTF in Table 7-2.

Table 7-2 Summary of security issues between GTF components

Security Issues	User → The CCM	The GT Client ←→ The GT Client
Authentication	User uses username and password are to access the CCM.	Credential keys (MyProxy certificate)
Authorization	Content owner decides who is authorized to what	It enforces the authorization settings provided by owner through the CCM and the WS-Tracker service
Message Integrity	SSL/TSL used during the communication	Content data transferred without encryption

7.4 Dealing with Various Attack Scenarios

In this section we explain the various attack scenarios that we try to cope with. As considering cryptographic attacks out of our research, we do not address it.

7.4.1 **Man-in-the-middle Attacks**

In Man-in-the-middle (MITM) attacks, an attacker intercepts and replaces public keys of two communication parties with its own public key, which enables the attacker to decrypt communications using his or her private key. The initial key exchanges between the GridTorrent Clients (GTCs) are vulnerable to this kind of

attack. We address this by requiring that all initial communications with between the GTCs be over SSL, which eliminates MITM attacks. MITM attacks are not a problem for content sharing, since their credentials have already been exchanged over SSL and content data can be encrypted and signed if it is necessary.

7.4.2 **Replay Attacks**

Replay attacks involve the attacker storing network packets and resending them at a later time. SSL/TLS defeats this during initial communications between the GTCs. Since each data chunk has its own SHA-1 hash code in its .torrent metafile, the downloader can easily verify the incoming data integrity. If it is compromised, the GTC just ignores it. Furthermore, if necessary, both parties can send the content data signed with their credentials.

7.4.3 **Denial of Service Attacks**

In this type of attacks, the attacker may try to overload the system resources (CPU and network cycles) by generating a large number of spurious content data packet that are processed by the system. Since each peer needs to authenticate before the actual data transmission, unauthorized entities would be rejected at GTCs that receive them. The WS-Tracker service may be vulnerable to multiple bogus requests originating from a malicious entity. This particular vulnerability may be addressed in the implementation by rejecting socket connections from IP addresses that have made multiple bogus attempts. By their nature, distributed systems generally tend to be less susceptible to denial of service attacks.

7.4.4 **Non-Repudiation**

Non-repudiation is more of a system abuse than an attack. It can be taken place in two ways. In the first one, a user may claim his or her username and password hacked and abused by an attacker . The attacker is only able to change the user settings in the CCM which is mere metadata and does not contain any useful content data. This is defeated by SSL and mutual authentication in the transport layer during the communication between the user and the CCM. If the user's CCM account is stolen by his or her mistake, then it becomes a legitimately stolen username/password and the CCM system admin provides a new username/password to the user. Other abuse is that the user publishes his/her malicious content through the CCM, and informs or allows other people to download his/her content using facilities provided by the CCM and then denies it. Again, the useful content data transferred only from users' machine, so it is impossible for an outside attacker to know what resides on users' machine unless he or she hacks them as well. Thus, the security of the users' machines is the users responsibility and out of our research.

7.5 Summary

In this chapter, we explained the security infrastructure used in the GridTorrent framework and motivations and design decisions behind it. Then, we presented a very detailed description of interactions taking place between all components. Next, we mentioned what type of security issues arises at which step and how we addressed them.

Chapter 8

Performance Evaluation

8.1 Introduction

It is a well-known fact that TCP's window based congestion control mechanism prevents [147] full-scale usage of high bandwidth-delay product. Hence, transferring large data set across high-performance networks is suffering from limitations of the current TCP implementation [147, 148] as it prevents the use of maximum bandwidth. Thus, throughput efficiency is one of the major motivations of GridTorrent, which is supposed to utilize the high bandwidth efficiently, that is, utilize as much bandwidth as possible. GridTorrent accomplishes this goal by aggregating throughput of all concurrent incoming TCP streams from distinct sources. As a matter of fact, using parallel TCP implementations [30, 148] is one common solution utilized by numerous bulk data transfer protocols in order to boost network throughput efficiency at the application level.

Although GridTorrent and parallel TCP have built on different architecture paradigm peer-to-peer and client/server respectively, they share common goal: better bandwidth utilization. However, unlike parallel TCP model, GridTorrent offers, due to its peer-to-peer architecture, other great features such utilizing idle network, computational and storage resources. In other words, in addition to extra features of GridTorrent, if GridTorrent display better or same performance than/as parallel TCP, there would be strong evidence for using GridTorrent for high-performance bulk data transfer in scientific community. Therefore, it is of great importance to compare GridTorrent performance results with that of parallel TCP. In the next section, we are going to present briefly the architecture of our Java-based implemented PTCP [149, 150] data transfer mechanism.

For the sake of fairness, that is, to provide same testing environment for both GridTorrent and PTCP, we used version of GridTorrent without security module, as security component has to send and receive several packets in order to perform handshake, authorization and authentication before starting the actual data transfer process. That overhead could have an adverse impact on actual data transfer performance.

In this chapter, we will investigate and discuss how well our GridTorrent Framework's data transfer mechanism architecture is performing. To observe influence of the underlying networks over its performances, we have set three scenarios and conducted their tests in LAN and WAN type of computer networks. Table 8-1 shows technical features of machines used in different locations.

Table 8-1 Server and client machines' descriptions and their locations

Name	Specifications	Network	Institution	Location
A	Intel(R) Quad-Core Xeon(TM) 4x2.33GHz CPU with 8GB of RAM on Red Hat Enterprise Linux 4.0	Broadcom NetXtreme II BCM5708 1000 Base-T Ethernet	Indiana University	Bloomington, IN
B	Sun Fire V880 8x1.2GHz UltraSPARC III processors with 16GB of RAM on Solaris 9. It has 6x72GB 10K rpm internal HD	Gigabit Ethernet and 10/100-BaseT Ethernet	Indiana University	Indianapolis, IN
C	Dual Pentium III 731MHz CPU with 512MB of RAM on GNU/Linux 2.6.20-1.2316.fc5	Gigabit Ethernet and 10/100-BaseT Ethernet	Florida State University	Tallahassee, FL

In each scenario, to compare the performances of PTCP and GridTorrent, we used both PTCP and GridTorrent test cases. We chose 300 MB for file size because the study [151] has shown that only more than 5% are larger than 1 GB and the mean file size generated in scientific computation community is larger than 300MB.

To measure the practical maximum available bandwidth capacity of the underlying network, we used Iperf, a tool to measure maximum TCP bandwidth, allowing the tuning of various parameters and UDP characteristics. Iperf reports bandwidth, delay jitter, datagram loss. To assess the maximum TCP bandwidth, we tried several TCP window size along with the parallel stream number. In LAN and WAN tests, TCP window size was set to maximum value allowed by the underlying operating system. Note that since the operating systems are not same on test machines, TCP window size used for LAN and WAN are not the same.

8.2 PTCP Architecture

A Parallel TCP stream consists of three basic steps; splitting of data into sub packets at the sender side, sending these sub packets over the network by using multiple streams in parallel, and coalescing of received sub packets at the receiver side. Using multiple parallel TCP streams gives high throughput by aggregating each socket bandwidth, although the default socket buffer size is not set to value of the bandwidth-delay product.

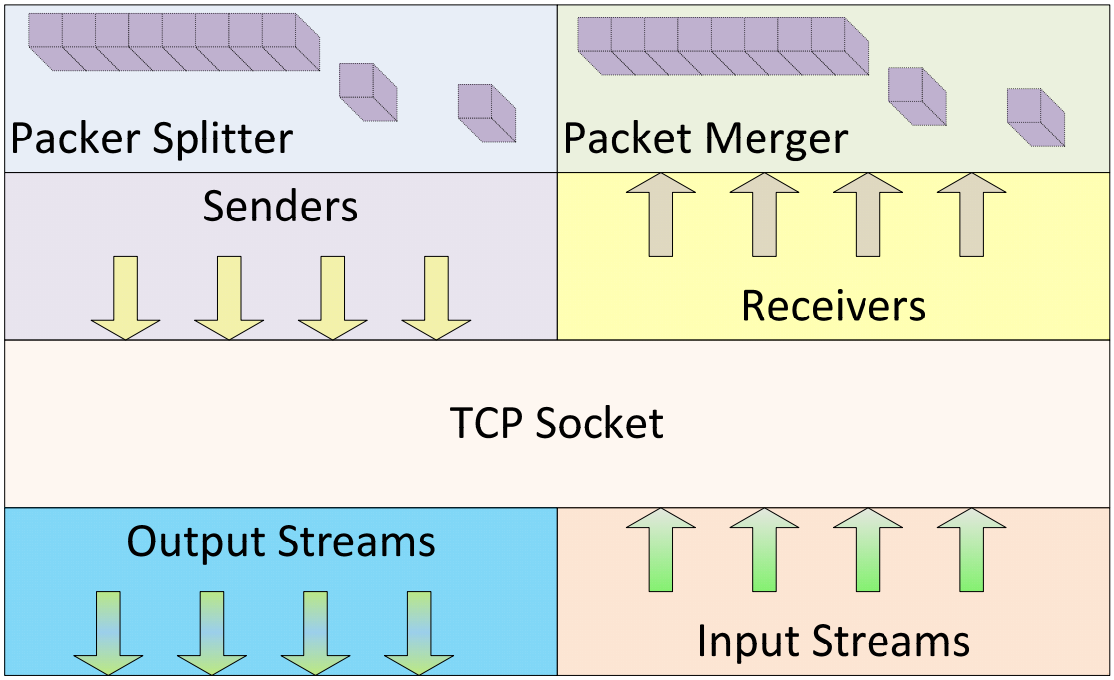

Figure 8-1 A parallel TCP socket architecture

Figure 8-1 depicts the architecture of the Java-based PTCP framework. PTCPSocket derived from Java.net.Socket can handle multiple sockets' input and output streams. It is comprised of packet splitter, packet merger, senders, receivers, and TCP sockets, and has two type of channels; communication and data channels. All

control information and negotiations are sent over the communication channel which stays open till the end of entire data transfer, and actual data are sent over the data channels. For instance, the decision of how many parallel streams are used is determined by the sender and is communicated with the receiver before initiating the actual data transfer through the communication channel.

After the setting the number of parallel streams, the packet splitter divides user's data into smaller sub packets. These sub packets are then passed on by the senders to the receivers while writing out these packets into data channels utilizing TCP sockets. The number of senders and receivers has to be same as the number of parallel streams. Receivers read packets from the data channels and pass them to the upper layer packet merger at the receiver's side. The Packet Merger merges smaller sub packets into one packet. It combines the incoming packets by checking their packet number assigned by the packet splitter. There is no need to check data integrity at the packet merger layer again, since TCP uses a checksum computed over the whole packet to verify that the protocol header and the data in each received packet have not been corrupted.

8.3 LAN Test

It was performed between two Indiana University's machines nearly 50 miles apart. Theoretical available bandwidth capacity is the maximum data transfer rate which the underlying network interface card permits. Measured available bandwidth capacity is assessed by using Iperf with the following parameters.

Theoretical Available Bandwidth: 1000 Mbps

Measured Available Bandwidth: 857 Mbps

Server side: Iperf -s -w 256k

Client side: Iperf -c <hostname> -w 512k -P 50

8.3.1 **Scenario I: Testbed**

The purpose of this scenario is to evaluate the performances of PTCP and GridTorrent in local area network. Therefore, both server and client machines reside on Indiana University (IU) computer network and they are located nearly 50 miles away from each other. For the performance test of PTCP, we used one client and one server. The number of parallel TCP streams between server and client has been increased from one to sixteen in increment of one stream in each step. Figure 8-2 demonstrates the connections diagram of PTCP test case.

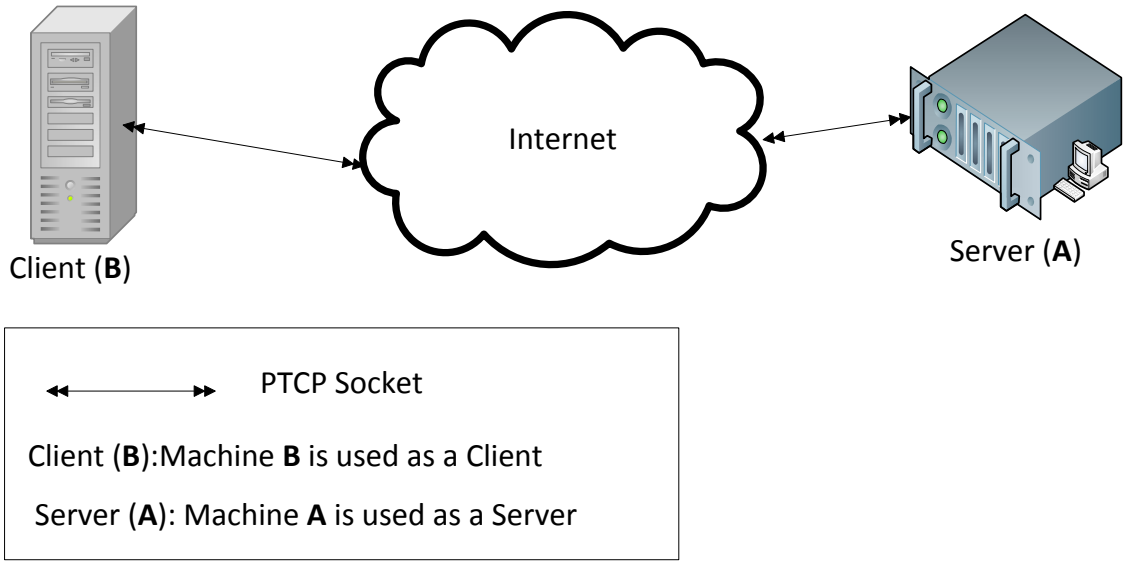

Figure 8-2 Client and server configuration for PTCP test case. Server, machine A, is located at Bloomington, IN, whereas client, machine B, is at Indianapolis, IN.

The connections topology between GTF client and seeders are displayed in Figure 8-3. Similar to PTCP test case, both seeders and client are on IU LAN. In

GridTorrent test case, unlike PTCP test case, one typical Java socket has been used between each seeder and the peer; however, the number of seeders is increased during the testing. Test has been initiated into one seeder and the number of seeders was increased by one in each step, up to sixteen.

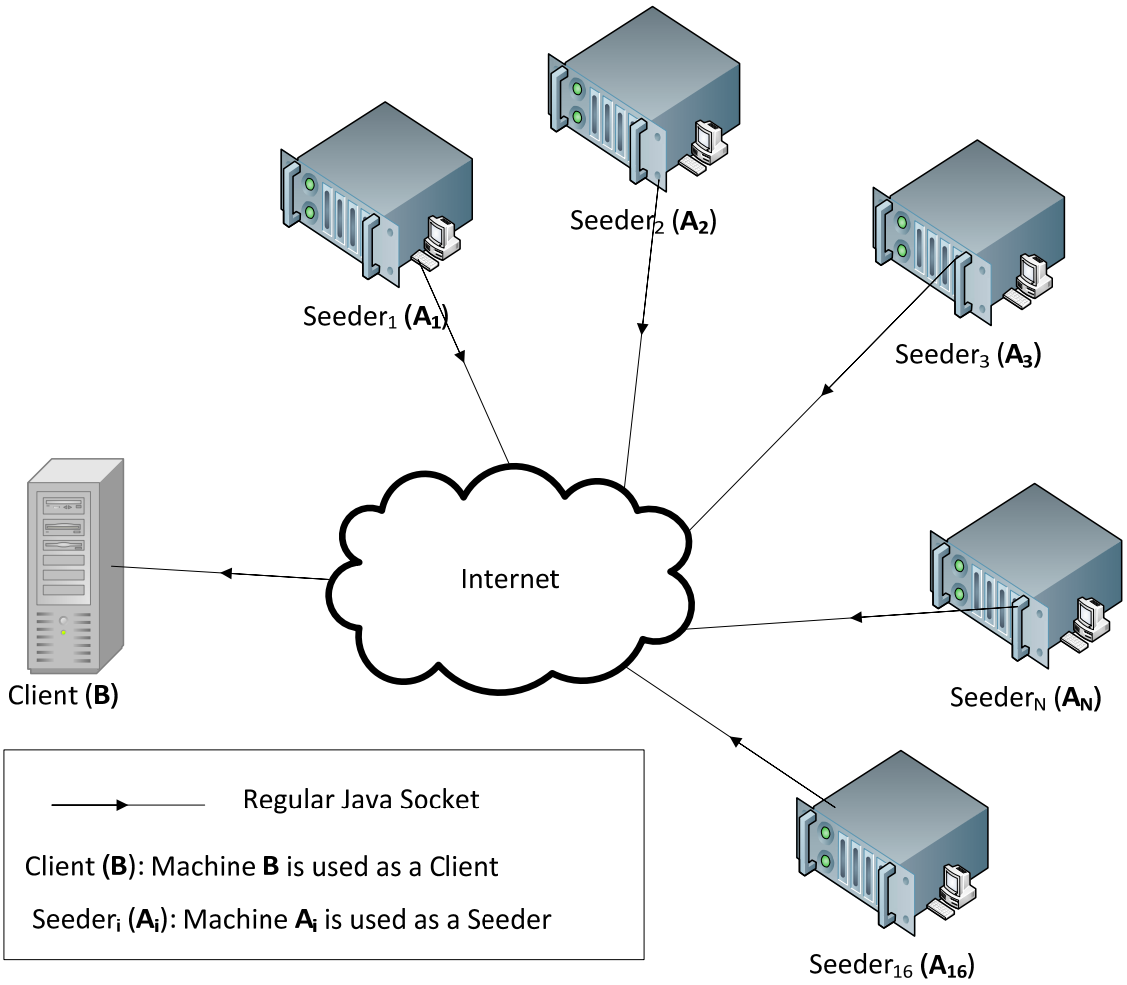

Figure 8-3 GridTorrent test case configuration for LAN test. Unlike PTCP test case configuration, regular single Java sockets are used for data transfer in GridTorrent test case. However, server and client's configuration and location is same as that of PTCP test case.

8.3.2 **Scenario I: LAN Test Result**

Table 8-2 shows average transmission rates between client/peer and servers/sources in IU LAN. These numbers were obtained by transferring files of 300 MB. In LAN test, there is no significant improvement in bandwidth usage while using multiple parallel streams [149, 150] because of today's very fast LAN connection. Furthermore, transmission time is smaller than overhead time in LAN; thus, any overhead process significant impact on data transmission rate, since the experimental data transfer (80-100 Mbps) rate is much lower the theoretical (1000Mbps) and the

Table 8-2 Performance characteristics of PTCP and GridTorrent with various parallel streams or sources on LAN

Streams or Sources #	Mean		Standard Deviation		Standard Error	
	PTCP	GridTorrent	PTCP	GridTorrent	PTCP	GridTorrent
1	90.24	80.53	8.07	9.10	2.55	2.88
2	97.61	81.43	1.28	11.97	0.41	3.79
3	77.13	88.34	9.48	7.94	3.00	2.51
4	79.18	85.95	6.45	6.90	2.04	2.18
5	80.44	79.53	11.18	12.93	3.54	4.09
6	94.81	86.26	4.11	8.05	1.30	2.55
7	84.60	86.59	10.01	6.98	3.17	2.21
8	78.54	83.51	9.69	8.97	3.07	2.84
9	85.22	75.27	8.32	10.07	2.63	3.19
10	90.60	84.01	5.41	9.46	1.71	2.99
11	84.30	87.27	8.81	9.22	2.79	2.91
12	75.14	84.47	10.06	9.47	3.18	3.00
13	76.71	90.81	10.15	7.92	3.21	2.51
14	76.27	86.49	12.37	7.96	3.91	2.52
15	86.90	84.73	7.51	9.87	2.38	3.12
16	74.94	80.06	13.22	8.68	4.18	2.74

measured data transfer rate (857Mbps). As it is seen from Figure 8-4, the deterioration does not have identifiable pattern when the number of parallel streams is increased. The network instability, also, might cause that random fluctuation.

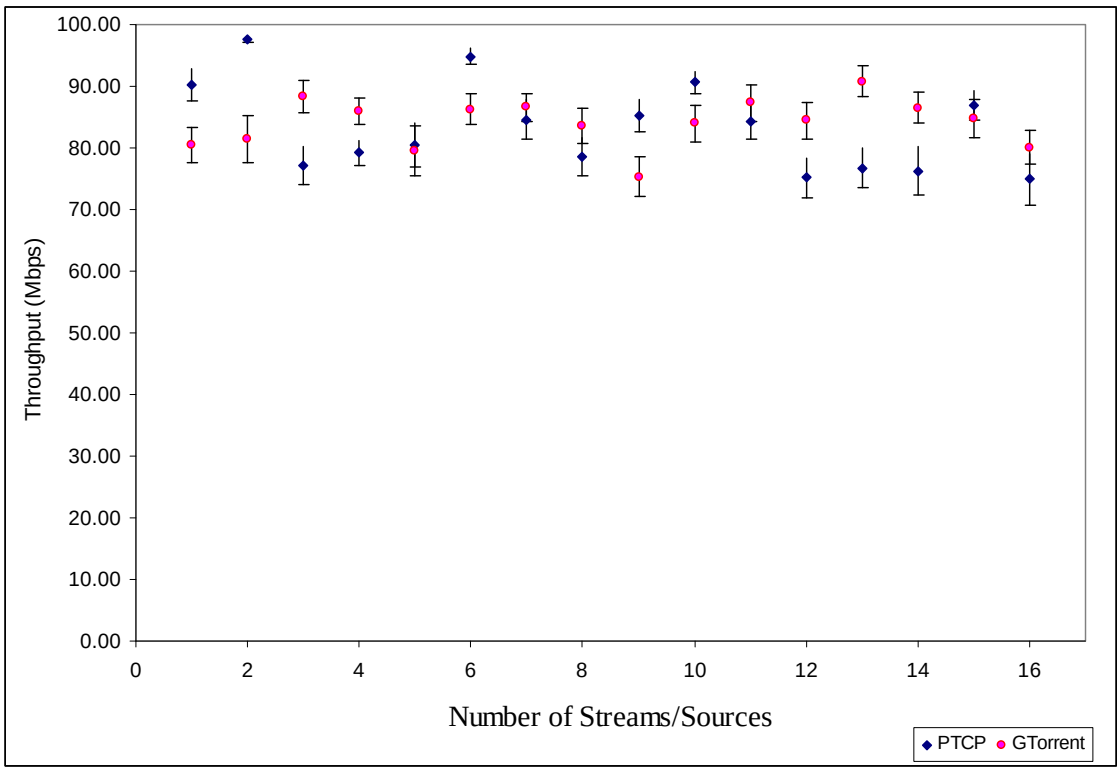

Figure 8-4 Achieved average data transfer rate of PTCP and GridTorrent with various parallel flows or sources on LAN type computer network (IU-IU settings).

8.4 Continental WAN Test

In this test, to evaluate the performance of GridTorrent and PTCP on wide area network, client/peer machine was located at Indiana University at Bloomington and servers/seeders were situated at Florida State University at Tallahassee, and two scenarios were tested.

Theoretical Available Bandwidth: 1000 Mbps

Measured Available Bandwidth: 30.2 Mbps

Instead of 512kB buffer size used in LAN test; we set TCP window buffer size to 256kB because of underlying network characteristic. We used Iperf with the following options to measure the maximum available bandwidth capacity of the underlying network,

For server side: Iperf -s -w 256k

For client side: Iperf -c <hostname> -w 256k -P 50

8.4.1 **Scenario II: GridTorrent Framework Client with One Socket**

The procedures used in this scenario are very similar to scenario I, except the location of servers/seeders and client/peer machines. Similarly, a pair of client and

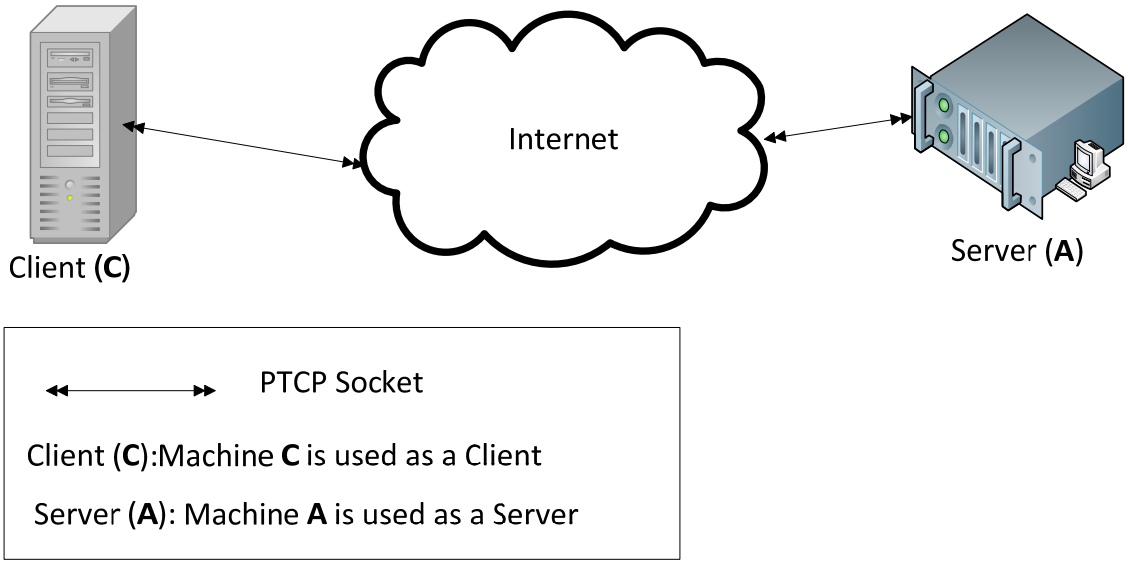

Figure 8-5 Client and server layout for PTCP test case. Parallel TCP streams were used for data transfer. Server is located at Bloomington, IN, whereas client is at Tallahassee, FL.

server has been used for PTCP performance test, and the number of parallel TCP stream was increased from one to sixteen streams in increment of one stream. Figure 8-5 illustrates the connections diagram of PTCP test case. The test configuration of GridTorrent in this scenario, similar to PTCP, is same as that of GridTorrent in the scenario I as depicted in Figure 8-6.

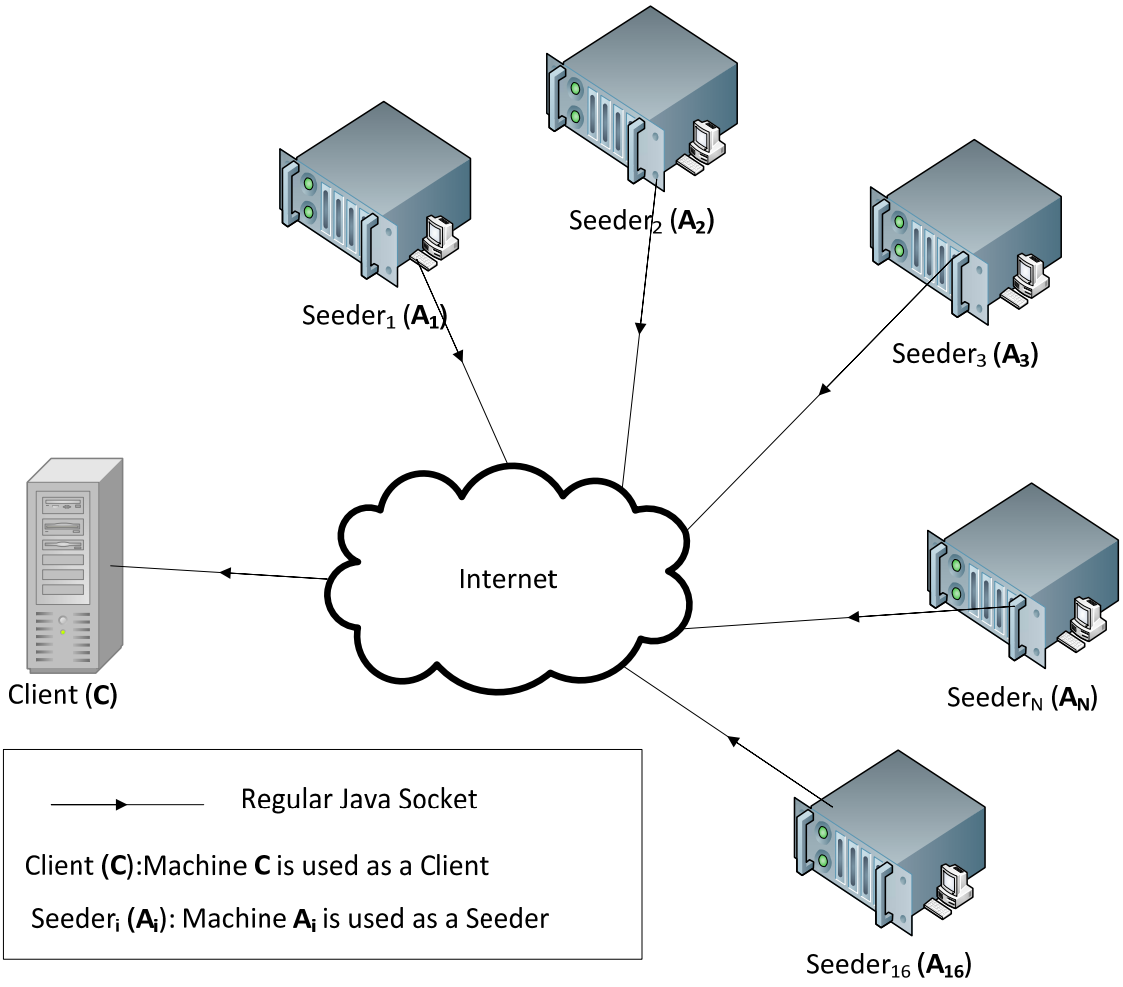

Figure 8-6 GridTorrent test case topology for wide area network test. Unlike PTCP test case configuration, regular Java sockets are used for data transfer in GridTorrent test case. However, server and client's configuration and location is same as that of PTCP test case.

8.4.2 **Scenario II: Test Result**

The gains in terms of accomplished data transmission rate are substantial, when the multiple parallel streams used in long-distance to transfer data. Test results were agreed with the above premise Table 8-3 lists the achieved average data transmission rate of PTCP and GridTorrent, as well as standard error and standard deviation. As seen in Figure 8-7, bandwidth usage is significantly improved in both GTF and PTCP. PTCP's bandwidth utilization rate has risen steadily until fifteen streams. It has its peak value of 118 Mbps. Just after the fifteenth stream, its data transfer rate starts falling.

Table 8-3 Performance characteristics of PTCP and GridTorrent with various parallel streams or sources on WAN. In this scenario, single regular Java socket is used between a GridTorrent peer and sources' connections.

Streams or Sources #	Mean		Standard Deviation		Standard Error	
	PTCP	GridTorrent	PTCP	GridTorrent	PTCP	GridTorrent
1	10.57	13.58	0.24	0.63	0.07	0.20
2	19.87	26.29	1.08	1.54	0.34	0.49
3	25.19	37.34	1.09	1.91	0.34	0.60
4	30.94	49.08	1.50	2.45	0.48	0.77
5	51.93	53.27	2.25	2.98	0.71	0.94
6	62.07	64.59	3.60	2.09	1.14	0.66
7	71.66	73.55	3.77	3.60	1.19	1.14
8	78.31	82.85	2.53	3.80	0.80	1.20
9	86.76	87.50	4.01	3.72	1.27	1.17
10	96.76	98.61	4.77	4.25	1.51	1.35
11	101.64	104.50	3.08	6.16	0.98	1.95
12	108.83	112.91	4.69	6.40	1.48	2.02
13	108.12	117.60	3.86	3.74	1.22	1.18
14	114.06	104.23	7.02	4.31	2.22	1.36
15	117.99	101.68	4.70	4.28	1.49	1.35
16	112.42	99.94	3.91	4.46	1.24	1.41

GTF has displayed the same characteristic; instead of fifteenth stream, its GTF has displayed the same characteristic; instead of fifteenth stream, its bandwidth usage rate begins to decline right after thirteenth streams. GridTorrent was performing better than PTCP when the number of parallel streams is less than five. Between the fifth and thirteenth streams, it demonstrates that it has slightly better data transfer rate than that

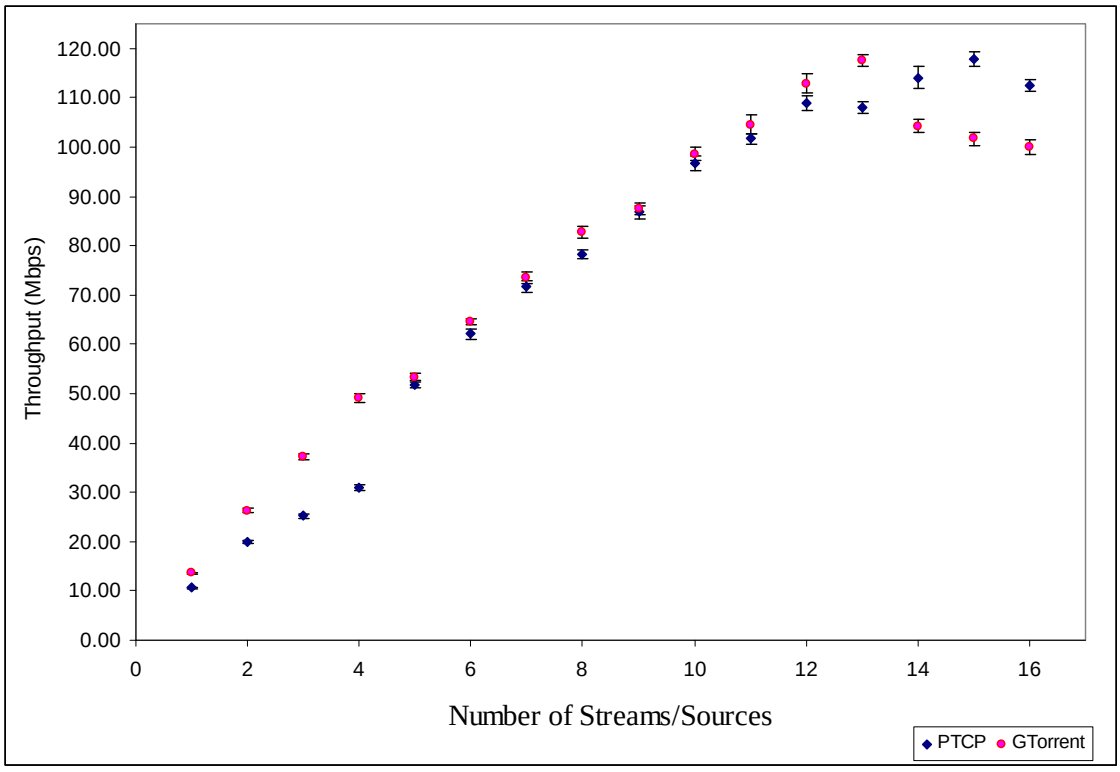

Figure 8-7 Overall data transmission performance for PTCP and GridTorrent on wide area network with various parallel flows or sources and a fixed size file size. In this scenario, GridTorrent uses one socket in each connection for every source.

of PTCP. Another interesting outcome is that the maximum achieved data transfer rate we measured is almost four times higher than Iperf's result because Iperf is used as a standard network bandwidth measurement tool among computer users. Another advantage of GTF is feature of load balancing. Whereas the whole data is sent from a

single source in PTCP setup, the approximate amount of data sent from a single seeder in GridTorrent setup is:

$$\text{AverageAmountofData} = \frac{\text{The whole data}}{\text{Number of seeds}}$$

This feature will help to relieve the bottleneck problem of a single source under a great many requests of data transmission.

8.4.3 **Scenario III: GridTorrent Framework Client with Four Sockets**

As we mentioned in Chapter 4, besides Java socket, other data transfer protocols can be exploited in GridTorrent client. Although we used the same PTCP settings of the scenario II as illustrated in. In order to investigate the performance of the combination of multiple parallel TCP streams and Bittorrent algorithm in wide area network, instead

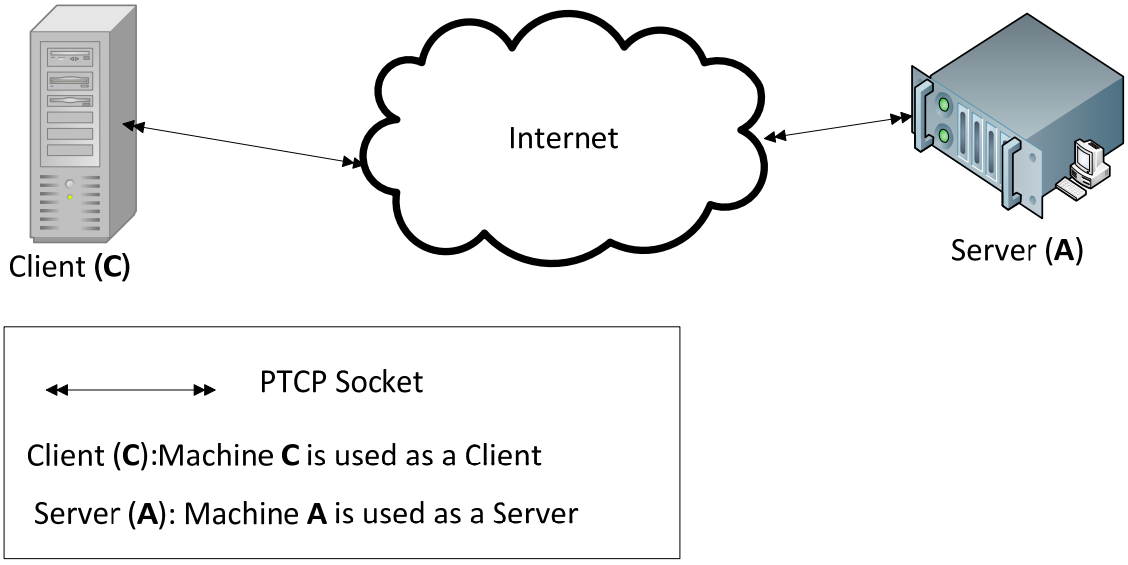

Figure 8-8 Client and server network layout for PTCP test case. PTCP streams are used for data transfer.

of one regular Java TCP socket, as it is seen in Figure 8-9, four parallel regular Java TCP sockets were used between peer and seeders, and number of seeders has commenced from one and increased from one to 16. The increment in number of seeders in each step was one.

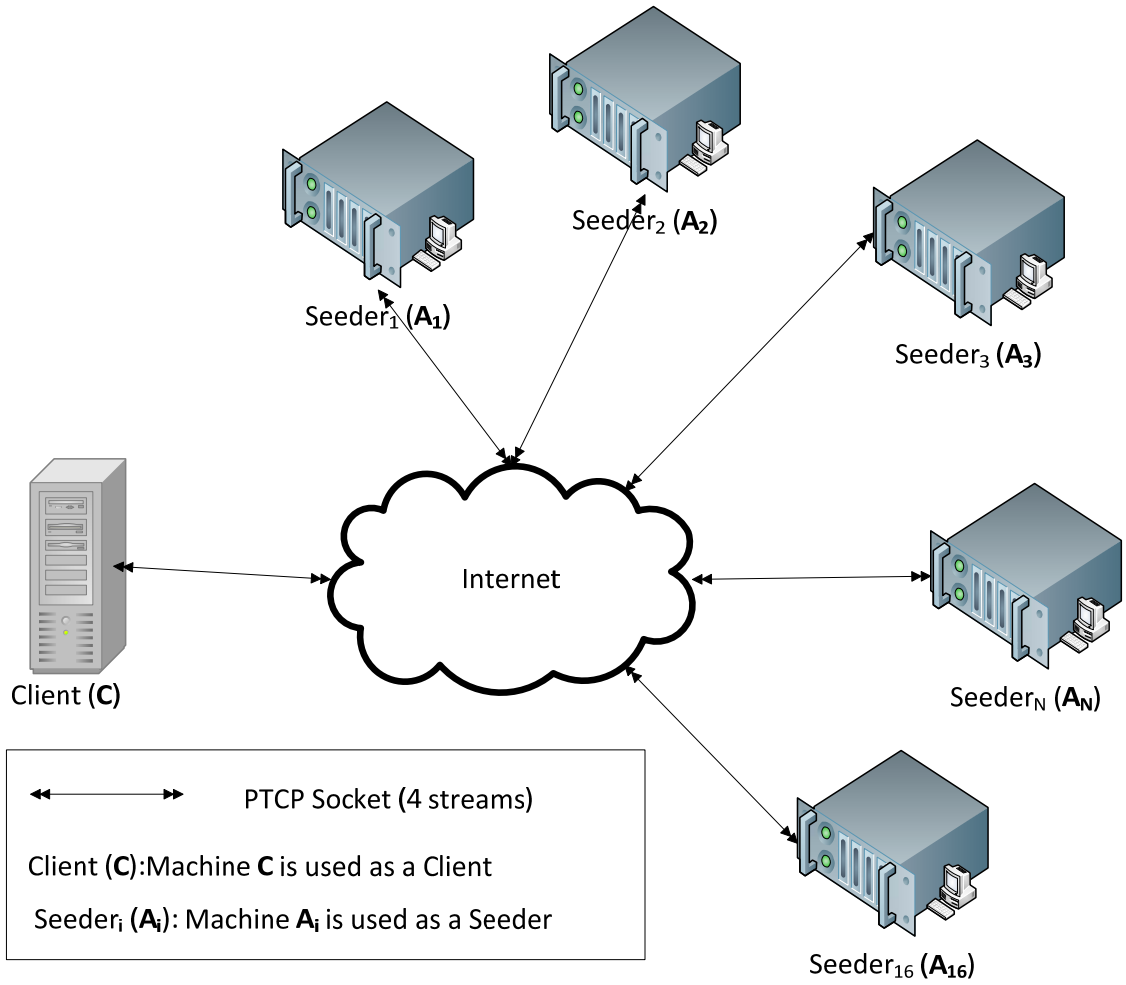

Figure 8-9 GridTorrent test case topology for WAN test. Four parallel TCP sockets are used for data transfer.

8.4.4 **Scenario III: Test Result**

Table 8-4 lists the achieved average data transmission rate of PTCP and GridTorrent, as well as standard error and standard deviation. PTCP test topology in

Figure 8-5 and Figure 8-8 was same and conducted exactly as in previous scenario. When we compared to test results of scenario II and scenario III, we seen that the test results have been very promising. Using parallel TCP with Bittorrent algorithm demonstrates much better bandwidth throughput than standalone GridTorrent and PTCP. The maximum attained bandwidth is around 145 Mbps which is %23 higher than PTCP's result (118 Mbps). Figure 8-10 presents considerable increase in data transfer rate when multiple parallel streams are used in GridTorrent. This result is important, because there is no performance gain anymore after the 15th streams in

Table 8-4 Performance characteristics of PTCP and GridTorrent with various parallel streams or sources on WAN. In this scenario, four parallel streams are used between a GridTorrent peer and sources' connections.

Streams or Sources #	Mean		Standard Deviation		Standard Error	
	PTCP	GridTorrent	PTCP	GridTorrent	PTCP	GridTorrent
1	10.57	47.04	0.24	1.31	0.07	0.41
2	19.87	67.17	1.08	3.86	0.34	1.22
3	25.19	93.82	1.09	4.71	0.34	1.49
4	30.94	101.60	1.50	2.90	0.48	0.92
5	51.93	107.83	2.25	4.74	0.71	1.50
6	62.07	110.41	3.60	4.54	1.14	1.44
7	71.66	116.28	3.77	4.03	1.19	1.27
8	78.31	124.09	2.53	5.89	0.80	1.86
9	86.76	141.87	4.01	4.21	1.27	1.33
10	96.76	144.09	4.77	3.57	1.51	1.13
11	101.64	141.97	3.08	6.04	0.98	1.91
12	108.83	145.15	4.69	5.46	1.48	1.73
13	108.12	143.37	3.86	6.72	1.22	2.12
14	114.06	142.97	7.02	5.66	2.22	1.79
15	117.99	142.10	4.70	7.34	1.49	2.32
16	112.42	141.08	3.91	7.40	1.24	2.34

parallel streams of PTCP; in fact, it deteriorates the data transfer rate. However, we could increase the number of parallel streams in GridTorrent up to 40 while without having any decrease in the data transfer rate.

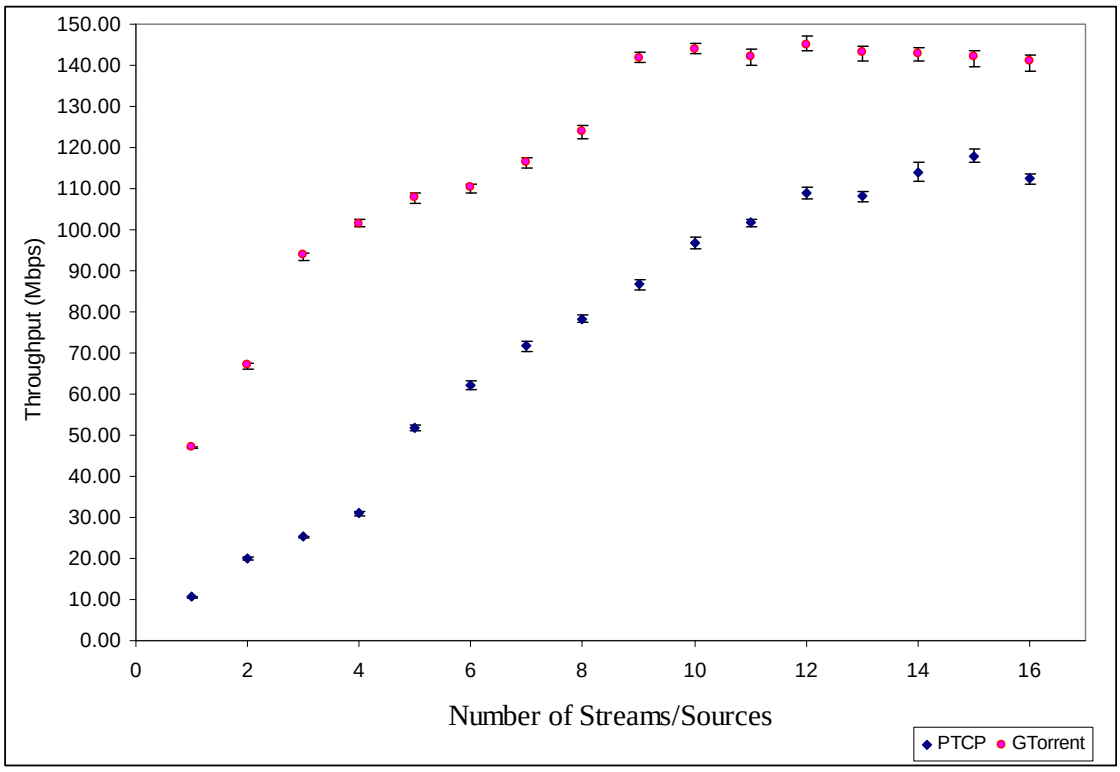

Figure 8-10 Achieved average data transfer rate of PTCP and GridTorrent with various parallel flows or sources on LAN type computer network (IU-FSU settings). GridTorrent client uses four parallel TCP sockets in each connection for every source.

8.5 Multi-nodes

The experimental studies on real networks described in discussed previous sections show that GridTorrent displays better performance or at least similar performance than PTCP on LAN and WAN. In addition, to some extent GridTorrent can overcome the performance barrier which PTCP reaches after a certain number

parallel streams by leveraging parallel streams on top of peer-to-peer connections. In this section, we perform theoretical analysis of GridTorrent's performance with multiple-nodes with the following assumptions:

- Identical resources: All the machines have identical configurations and systems resources such as computation power, memory and storage disk speed, and network bandwidth capacity.
- Dedicated network resources: All network resources are dedicated to single data transmission process at a time. The influences of other networks traffics are ignored.
- Identical half-duplex connections: Each connection allows traffic either way, but only one way at a time. Although, the connections of real networks are full-duplex connections, we chosen this case for worst case scenario since peer-to-peer data transfer performs better then PTCP on full-duplex connections.

One server/seeder and three clients (peers) is illustrated in Figure 8-11. Whereas client machines can only download data from server in PTCP model, clients (peers) can download from other clients as long as they have any data to offer in GridTorrent model. For the simplicity, we assume that the size of file to be downloaded is 3xN Mb since we have only three clients. We list the sequences matrices of data transmission from server to client or client to client in PTC and GridTorrent in Table 8-5 and Table 8-6 respectively. Unlike PTCP clients, if two GridTorrent clients are idle and one of them has data which is required by others, they can send data between them.

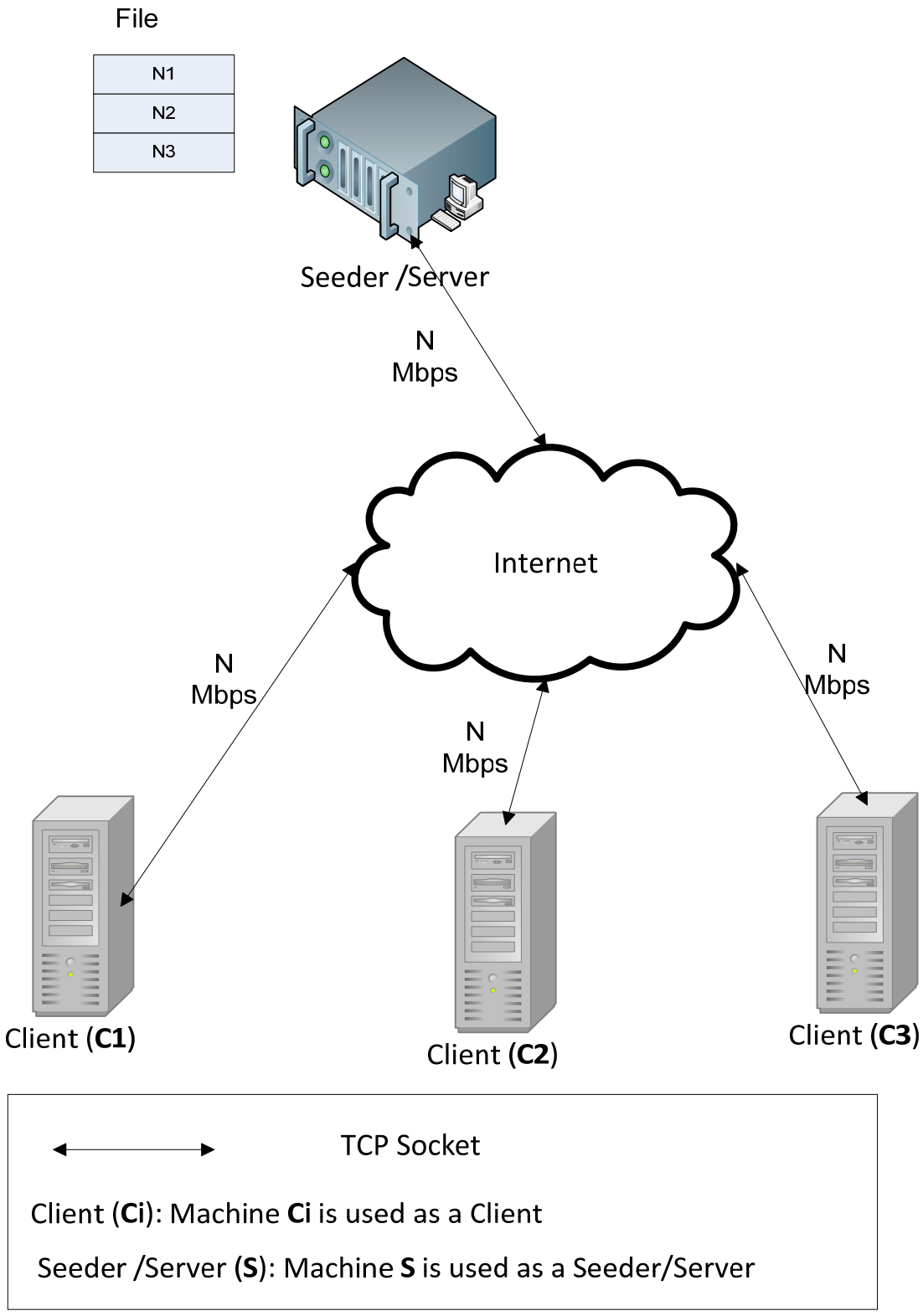

Figure 8-11 Multiple nodes representation for GridTorrent and PTCP

As it is seen in Table 8-5 and Table 8-6, Client1 completes its downloading at the end of 3rd second, Client2 finishes at the end of 6th second and Client3 receives last segment of data from the server at the end of the 9th second. However, the overall downloading completion time for all clients takes only 5 seconds in GridTorrent data transmission since idle clients helping each other whenever they can. As a result of this simulation, when the number of participating nodes increases, GridTorrent can exhibit better performance than other data transfer techniques which use parallel TCP since the data transmission rate of GridTorrent is better or not worse than that of parallel TCP on real networks, for instance WAN.

Table 8-5 Transmission sequence matrix of PTCP

Time (sec)	**S-C1**	**S-C2**	**S-C3**	**C1**	**C2**	**C3**
1	N1			N1		
2	N2			N1,N2		
3	N3			N1,N2,N3		
4		N1			N1	
5		N2			N1,N2	
6		N3			N1,N2,N3	
7			N1			N1
8			N2			N1,N2
9			N3			N1,N2,N3

Table 8-6 Transmission sequence matrix of GridTorrent

Time (sec)	S-C1	S-C2	S-C3	C1-C2	C2-C3	C1-C3	C1	C2	C3
1	N1						N1		
2		N2				N1	N1	N2	N1
3			N3	N1			N1	N1,N2	N1,N3
4	N2				N2		N1,N2	N1,N2	N1,N2,N3
5		N3				N3	N1,N2,N3	N1,N2,N3	

8.6 Overhead

Both parallel TCP and GridTorrent have overhand due to nature of multiple parallel connections. Data splitting and coalescing taking place for the entire data transfer are common overhead processes in both PTCP and GridTorrent. In addition to that, GridTorrent has to fragment files into chuck when the interested file has been created. It is a one-time process, in contrast to data splitting and merging processes.

The overhead of PTCP's communication channel can be compared to the overhead of GridTorrent's WS-Tracker client. It varies between 300 to 600 milliseconds. Another overhead of GridTorrent is that control messages exchanged between peers to ensure Bittorrent protocol rules strictly enforced to all participating peers. Our testing results demonstrated that the total size of overhead messages is between 148KB to 169 KB. This overhead can be ignored when it is compared to file size of 300 MB.

8.7 Summary

In this chapter, we evaluated and compared GridTorrent's performance with that of PTCP both on LAN and WAN. The tests indicate that both GridTorrent and PTCP do not provide any performance gain on LAN type of computer networks. However, on WAN, the performance of GTF is better or not worse than that of parallel TCP. This outcome is important since parallel streaming is used in many scientific computing data transfer tools such as GridFTP. Additionally, using Java socket and parallel TCP indicates that GTF can exploit other high performance data transfer protocols like GridFTP or UDT in traditional low BDP environments at the same time.

Chapter 9

Conclusions and Future Work

9.1 Conclusion

In this thesis, we have presented a novel approach for collaboration framework with high performance data transmission capability for scientific community. Contrary to many existing data transfer solutions, based on client-server paradigm (i.e. FTP, HTTP or other type of client-server solutions), we have chosen peer-to-peer paradigm for data transfer mechanism that enable us to employ a service-oriented architecture and make our framework extensible with Web services features.

A proof of concept implementation on data management and collaboration was demonstrated. Unlike other heavyweight, platform dependent and cumbersome data

catalog and management solutions whose administration tasks require a great deal of the knowledge systems, our prototype of CCM is lightweight, platform independent and it requires minimum system resources, and management and maintenance overhead. In addition, it provides all the basic and major facilities such as search, browse, publish and share capabilities in a simple structure.

Furthermore, on the contrary to existing data movement approaches, GridTorrent client uses data transfer technique built on peer-to-peer architecture, which has two contributions:

1. Performance improvement
2. Efficient utilization of available system resources

Our test results showed that GridTorrent can be faster than or as fast as data movement solutions using parallel TCP streams to improve data transfer speed. Although improving the performance is generally considered the major achievement, we believe that reducing or eliminating waste of available system resources is as important contribution as performance improvement, especially when efficiency and sparingness are becoming a critical issue for every resources of world.

Finally, as WS-Tracker is built on Web service architecture, it brings flexibility, extensibility, and scalability to GridTorrent framework. This feature allows GridTorrent framework to be customizable with regard dynamic needs or emerging requirements.

9.2 Summary of Answers for Research Questions

Here we summarize the answers for the research questions presented in this thesis.

9.2.1 How can we build a peer-to-peer data transfer mechanism which utilizes SOA for scientific community? Which one of available peer-to-peer system is best for this purpose and what type of modifications and new features are needed to be added to it?

There are two distinct types of peer-to-peer system architecture. First type of peer-to-peer system architecture is established on non-modular structure in which they integrate all the facilities such as file searching and peer discovery into one system. Second type of peer-to-peer system architecture is built on modular pattern by establishing a clear borderline between data transport algorithm and other required services (e.g. peer discovery and file search services). The former leads a complicated system which is less modular and customizable. Further, it is very difficult to utilize facilities provided by SOA. On the other hand, the latter enable us not only to harness the benefits of SOA, but also to modify services not pertinent to data transfer algorithm according to our needs. This step was very important since our choice will have impact on all the aspects of our design. Therefore, as BitTorrent was the best candidate that falls into second category, we have chosen it as base peer-to-peer data transfer layer.

9.2.2 How can we provide a medium that allows participants to manage, share, discover, and download their contents and integrate it with data transfer mechanism?

As explained in the previous question, there needs to be a system with modular structure. In BitTorrent, there is no built-in framework which provides content management, sharing, discovery services. However, it provides a simple tracking component used for initial the communication between peers. We developed a separate

component aimed to perform services related to the content management and then we converted the basic tracker into a Web service based tracker which capacitates us to add or remove new services. With the help of WS-Tracker Service, we integrated the separated content management module into data transfer layer by permitting them to communicate each other.

9.2.3 **Is the data transfer mechanism scalable?**

There are two places involved in data transfer process: WS-Tracker Service and GridTorrent client. Since WS-Tracker Service is built on Web service technology and does not participate actual data transfer operation at all, it should not suffer from scalability issues or it has the same problems that a regular Web service provider encounters.

As explained in chapter 4, due to its peer-to-peer nature, GridTorrent client hosting data transfer layer is a basic server with the simultaneous upload and download capability. From, this aspect, therefore, it can be considered as a regular data server such as an FTP server. However, unlike standard FTP server, the downloaders of a GridTorrent client differ from that of FTP server in three respects: downloading data size, bandwidth usage, and session duration. If, therefore, there are any scalability issues, since scalability of the system is by design, they stemmed from characteristics of the data not the design of GridTorrent. An FTP server, for example, may throttle download size to a certain degree in order to increase the number of downloaders. However, unlike an FTP server, the GridTorrent aims to scientific community which generates very large set of data set and is designed to maximize bandwidth usage as reducing data transmission time is the main concern not the number of simultaneous

downloaders. In addition, BitTorrent has proven than it is very scalable and successful data transfer protocol, it was deployed as peer-to-peer file transfer protocol by many commercial enterprises such as Amazon and Warner Brothers.

9.2.4 How is the performance of data transfer mechanism and it is acceptable?

In chapter 8, we evaluated and compared GridTorrent's performance with that of PTCP both on LAN and WAN. The tests indicate that both GridTorrent and PTCP do not provide any performance gain on LAN type of computer networks. However, on WAN, the performance of GTF is better or not worse than that of parallel TCP. This outcome is important since parallel streaming is used in many scientific computing data transfer tools such as GridFTP. Additionally, using Java socket and parallel TCP indicates that GTF can exploit other high performance data transfer protocols like GridFTP or UDT in traditional low BDP environments at the same time.

9.2.5 What is the overhead of this system and is it reasonable?

As explained in chapter 8, our testing results demonstrated that the total size of overhead messages resulted from the communications between the GridTorrent client and WS-Tracker Service is between 148KB to 169 KB. Furthermore, the duration of communications are very short and they do not have any adverse effect on performance even though they takes place during the downloading activity. Therefore, the overhead can be acceptable. Indeed, the overhead size can be ignored when it is compared to file size of 300 MB.

9.2.6 **How can we make it enough secure for scientific community as security is not a concern in peer-to-peer to networks for non-scientific community?**

Similar to content management component, BitTorrent does not provide any security mechanism for authentication and authorization. Therefore, we designed and implemented a security framework with access control list capability (ACL) to meet the moderate security requirements and integrated it with GridTorrent framework. TLS is utilized at Asynchronous Collaboration and Content Management module. There is no security implementation between GridTorrent client and WS-Tracker Service as data exchanged between them has no critical information. ACL is deployed at GridTorrent clients to ensure only allowed clients download the particular content. The unique GridTorrent ID over TLS communication is used for authentication and authorization processes.

9.3 Contributions

Neither high-performance nor peer-to-peer data transfer techniques are not new technologies. They have been around more than couple of decades; however, since techniques used for transferring bulk data in high-speed wide area networks are usually built on FTP or HTTP, utilizing peer-to-peer file transferring protocol for scientific community and integrating it with a framework which enables collaboration and content management is a new approach. This approach has several contributions. The foremost important one is that harnessing power of peer-to-peer file sharing in wide area networks by utilizing unused system resources, particularly network resources.

As GridTorrent has many good features because of its design and underlying peer-to-peer file sharing protocol, it provides many useful and vital services from out-of-box. These features are offered as separate services or products in other widely used high-performance data transfer techniques such as GridTorrent. Checking available disk space before starting download process, reliable file transfer, third party data transfer, for example, are some of those features.

Another contribution is being data structure, system and platform independent. This feature enable our work to be deployable any existing system without changing its data format as an underlying data transfer layer. For instance, one of the goal of the THREDDS (Thematic Realtime Environmental Distributed Data Services) project is to simplify the discovery and use of scientific data between data providers and data users. In this project, data has to be in particular format and are delivered via HTTP. As our work is data format neutral, it can be deployed both as discovery server and as data transfer layer in its existing system.

In addition, GridTorrent components are lightweight processes so that they are easily deployable and can require less system resources contrast to other high-performance data transfer solutions. This is important as there are many scientific communities with diverse resource capacity can use GridTorrent without modifying their hardware structure.

Finally, Web service based WS-Tracker Service not only exploits benefits of SOA, but also makes it adaptable to future requirement change by allowing to add a new service or to remove an existing one a seamless process.

9.4 Limitations and Future Research Direction

9.4.1 **Data Transfer Component**

Data transfer algorithm built on BitTorrent algorithm has demonstrated that it performs well both on LAN and WAN even though the basic BitTorrent algorithm was implemented in our prototype. However, there has been great deal of research and effort on BitTorrent in order to eliminate its overhead and improve its performance. Future work can further investigate the BitTorrent algorithm to deliver a higher level of performance.

Due to performance issues stemmed from very large data, data is sent over unsecured socket connection in our prototype. If a secure data transmission is required, investigating secure data transfer with acceptable/without performance loss would be good research topic.

9.4.2 **WS-Tracker Service**

As WS-Tracker Service exploits benefits of SOA, adding a new service or removing an existing one would be a seamless process. The security framework of GridTorrent framework was designed for moderate security requirements; we have not implemented any security structure for the WS-Tracker Service. However, in particular circumstances where secure communication is necessary, a suitable one among available WS-Security products can be implemented as future work.

9.4.3 **Asynchronous Collaboration and Content Management**

The design concepts of the Asynchronous Collaboration and Content Management are to keep it simple, to make it customizable, and to make it lightweight

framework. Therefore, only basic features are provided. Adding synchronous more and sophisticated collaboration tools, advanced search techniques are left for future work.

Bibliography

1. Bell, G., J. Gray, and A. Szalay, *Petascale Computational Systems.* Computer, 2006. **39**(1): p. 110-112.
2. Foster, I. and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufiuiann, San Francisco, CA. Vol. 211. 1999.
3. Foster, I., et al., *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, June 2002.* Open Grid Service Infrastructure WG, Global Grid Forum, 2002.
4. Foster, I., et al., *A Security Architecture for Computational Grids*, in *Proceedings of the 5th ACM conference on Computer and communications security*. 1998, ACM: San Francisco, California, United States.
5. Foster, I., C. Kesselman, and S. Tuecke, *The Anatomy of the Grid: Enabling Scalable Virtual Organizations.* International Journal of Supercomputer Applications, 2001. **15**(3): p. 200–222.
6. Graham, S.L., M. Snir, and C.A. Patterson, *Getting Up To Speed: The Future Of Supercomputing*. 2005: National Academy Press.
7. Adams, P., et al., *Science-Driven Network Requirements for ESnet*. 2006, LBNL--61832, Ernest Orlando Lawrence Berkeley NationalLaboratory, Berkeley, CA (US).

8. Allcock, W., *GridFTP Protocol Specification (Global Grid Forum Recommendation GFD.20)*. 2003.

9. Dickens, P., *FOBS: A Lightweight Communication Protocol for Grid Computing*. Lecture Notes in Computer Science, 2003: p. 938-946.

10. Booth, D., et al., *Web Services Architecture*. W3C Working Group Note, 2004. **11**: p. 2005-1.

11. Channabasavaiah, K., K. Holley, and E. Tuggle, *Migrating to a Service-oriented Architecture*. IBM White Paper, 2004.

12. Werthimer, D., et al. *A New Major SETI Project Based on Project Serendip Data and 100,000 Personal Computers*. in *Proceedings of the Fifth International Conference on Bioastronomy*. 1997.

13. *The seti@home project web site*. Accessed on-line 2008; Available from: http://setiathome.berkeley.edu.

14. Androutsellis-Theotokis, S. and D. Spinellis, *A Survey of Peer-to-Peer Content Distribution Technologies*. ACM Computing Surveys, 2004. **36**(4): p. 335-371.

15. Larson, S.M., C. Snow, and V. Pande, *Modern Methods in Computational Biology*. 2003, Horizon Press.

16. *The genome@home project web site*. Accessed on-line 2008; Available from: http://genomeathome.stanford.edu/.

17. Zissimos, A., et al., *GridTorrent: Optimizing data transfers in the Grid with collaborative sharing*. 11th Panhellenic Conference on Informatics (Patras, Greece, May 2007). PCI2007, 2007.

18. McNab, A., S. Kaushal, and Y. Li, *Web servers for bulk file transfer and storage* in *CHEP 06 Computing in High Energy and Nuclear Physics (Distributed Event production and processing)*. 2006: Mumbai, India.

19. *Voyager, The Interstellar Mission:*. 2008; Available from: http://voyager.jpl.nasa.gov/.

20. *Voyager I*. Available from: http://en.wikipedia.org/wiki/Voyager_1.

21. *Voyager Program*. Available from: http://en.wikipedia.org/wiki/Voyager_program.

22. *Zhu, Y.*, et al., *Mechanisms for High Volume Data Transfer in Grids*. 2007.

23. Allcock, B., et al., *Data Management and Transfer in High Performance Computational Grid Environments.* Parallel Computing, 2002. **28**(5): p. 749-771.

24. Gibson, G. and R. Van Meter, *Network Attached Storage Architecture.* Communications of the ACM, 2000. **43**(11): p. 37-45.

25. Anglano, C. and M. Canonico, *A Comparative Evaluation of High-Performance File Transfer Systems for Data-Intensive Grid Applications*, in *Proceedings of the 13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*. 2004, IEEE Computer Society.

26. Mathis, M., et al., *RFC2018: TCP Selective Acknowledgement Options.* RFC Editor United States, 1996.

27. Floyd, S., *RFC3649: HighSpeed TCP for Large Congestion Windows.* Internet RFCs, 2003.

28. Kelly, T. *Scalable TCP: Improving Performance in High Speed Wide Area Networks*. in *of First International Workshop on Protocols for Fast Long-Distance Networks*. 2003. CERN, Geneva, Switzerland.

29. Wu, R.X. and A.A. Chien. *GTP: Group Transport Protocol for Lambda-Grids*. in *Cluster Computing and the Grid, 2004. CCGrid 2004. IEEE International Symposium on*. 2004.

30. Zhang, Y., E. Yan, and S. Dao, *A Measurement of TCP over Long-Delay Network.* Proceedings of 6th Int'l Conference on Telecommunication Systems, Modelling, and Analysis, 1998.

31. Postel, J. and J. Reynolds, *File Transfer Protocol (FTP)*, in *RFC Editor United States*. 1985, STD 9, RFC 959, October 1985.

32. Allcock, W.E., *GridFTP: Protocol Extensions to FTP for the Grid*. 2003.

33. *bbFTP --Large files transfer protocol* 2005; Available from: http://doc.in2p3.fr/bbftp/index.html.

34. *Secure Copy-*. 2007; Available from: http://en.wikipedia.org/wiki/Secure_copy.

35. Hanushevsky, A., *Peer-to-Peer Computing for Secure High Performance Data Copying*, in *Computing in High Energy Phyasics*. 2001: Beijing.

36. Allcock, W., et al. *The Globus Striped GridFTP Framework and Server*. 2005.

37. Allcock, B., et al., *Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing.* IEEE Mass Storage Conference, 2001. **20**.

38. Horowitz, M. and S. Lunt, *FTP Security Extensions*, in *RFC Editor United States*. 1997.

39. *What is GridFTP ?* . 2004; Available from: http://it-dep-fio-ds.web.cern.ch/it-dep-fio-ds/Documentation/gridftp.asp.

40. Globus Alliance, *GridFTP: Universal Data Transfer for the Grid,* in *Globus Project White Paper, University of Chicago*. 2000.

41. Globus Alliance. *Reliable File Transfer Service (RFT)*. 2007; Available from: http://www.globus.org/toolkit/docs/4.2/4.2.0/data/rft/.

42. Madduri, R., C. Hood, and W. Allcock. *Reliable File Transfer in Grid Environments*. in *Local Computer Networks, 2002. Proceedings. LCN 2002. 27th Annual IEEE Conference on*. 2002.

43. Allcock, W.E., I. Foster, and R. Madduri, *Reliable Data Transport: A Critical Service for the Grid*, in *Building Service Based Grids Workshop, Global Grid Forum*. 2004.

44. Globus Alliance. *globus-url-copy Documentation*. 2007; Available from: http://www.globus.org/toolkit/docs/4.0/data/gridftp/rn01re01.html

45. Globus Alliance. *Moving Data Fast on the TeraGrid*. 2007; Available from: http://www.globus.org/solutions/tgcp/.

46. Ripeanu, M. and I. Foster, *A Decentralized, Adaptive Replica Location Mechanism*, in *Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing*. 2002, IEEE Computer Society.

47. *MySQL*. 2007; Available from: http://www.mysql.com/.

48. Globus Alliance. *Replica Location Service (RLS)*. 2007; Available from: http://www.globus.org/toolkit/docs/4.2/4.2.0/data/rls/.

49. Bresnahan, J., et al. *Globus GridFTP: What's New in 2007*. in *Proceedings of the First International Conference on Networks for Grid Applications (GridNets 2007)*. 2007.

50. GridSite Team. *Grid Security for the Web, Web platforms for Grids*. 2008; Available from: http://www.gridsite.org/.

51. Sivakumar, H., S. Bailey, and R.L. Grossman, *PSockets: The Case for Application-level Network Striping for Data Intensive Applications using High Speed Wide Area Networks.* Proceedings of the 2000 ACM/IEEE conference on Supercomputing (CDROM), 2000.

52. Mattmann, C.A., et al. *A Classification and Evaluation of Data Movement Technologies for the Delivery of Highly Voluminous Scientific Data Products*. in *Proceedings of the NASA/IEEE Conference on Mass Storage Systems and Technologies College Park*. 2006. Maryland, USA.

53. Web Maintenance Team of cc.jlab.org. *Off-Site File Transfer Facilities*. 2008; Available from: http://cc.jlab.org/docs/services/offsite/off-site-data-transfers.html.

54. Bonachea, D. and S. McPeak, *SafeTP: Transparently Securing FTP Network Services.* Computer, 2001.

55. Bonachea, D. and S. McPeak. *SafeTP: Secure FTP Transparently*. 2008; Available from: http://safetp.cs.berkeley.edu/.

56. Gu, Y. and R. Grossman, *UDT: UDP-based data transfer for high-speed wide area networks.* Computer Networks, 2007. **51**(7): p. 1777-1799.

57. Gu, Y. and R. Grossman, *SABUL: A Transport Protocol for Grid Computing.* Journal of Grid Computing, 2003. **1**(4): p. 377-386.

58. He, E., et al. *Reliable Blast UDP: Predictable High Performance Bulk Data Transfer*. in *Cluster Computing, 2002. Proceedings. 2002 IEEE International Conference on*. 2002.

59. Meiss, M., *Tsunami: A High-Speed Rate-Controlled Protocol for File Transfer*. 2002, Indiana University.

60. Wallace, S. *Tsunami File Transfer Protocol*. in *Proceedings of First Int. Workshop on Protocols for Fast Long-Distance Networks*. 2003. CERN, Geneva, Switzerland.

61. Bush, D. *UFTP - UDP Based FTP with Multicast*. 2001 July 29, 2008; Available from: http://www.tcnj.edu/~bush/uftp.html.

62. Zheng, X., A.P. Mudambi, and M. Veeraraghavan, *FRTP: Fixed rate transport protocol-a modified version of sabul for end-to-end circuits.* Proceedings of the 1 stInternational Workshop on Provisioning and Transport for Hybrid Networks (PATHNETS), in conjunction with the 1 stInternational Conference on Broadband Networks, 2004.

63. Gu, Y., et al., *Using UDP for Reliable Data Transfer over High Bandwidth-Delay Product Networks.* Laboratory for Advanced Computing, University of Illinois at Chicage, 2003.

64. Dickens, P.M. and V. Kannan, *Application-Level Congestion Control Mechanisms for Large Scale Data Transfers Across Computational Grids.* the

Proceedings of The International Conference on High Performance Distributed Computing and Applications, 2003.

65. Ansari, S. *Tsunami—A Study*. Available from: http://www-iepm.slac.stanford.edu/bw/Tsunami.htm.

66. Schoder, D., K. Fischbach, and C. Schmitt, *Core Concepts in Peer-to-Peer Networking*, in *Peer-to-Peer Computing: The Evolution of a Disruptive Technology*, R. Subramanian and B.D. Goodman, Editors. 2005, Idea Group Publishing. p. 300 pages.

67. Schoder, D. and K. Fischbach, *Peer-to-Peer Prospects.* Communications of the ACM, 2003. **46**(2): p. 27-29.

68. *The Kazaa web site*. Accessed on-line 2008; Available from: http://www.kazaa.com/.

69. Oram, A., *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. First ed. 2001: O'Reilly & Associates, Inc. 432.

70. *The Jabber web site*. Accessed on-line 2008; Available from: http://www.jabber.org.

71. Van Renesse, R., et al. *Heterogeneity-Aware Peer-to-Peer Multicast*. in *Proceedings of the 17th International Symposium on Distributed Computing (DISC2003)*. 2003.

72. Castro, M., et al., *Scribe: A Large-Scale and Decentralized Application-Level Multicast Infrastructure.* Selected Areas in Communications, IEEE Journal on, 2002. **20**(8): p. 1489-1499.

73. Stoica, I., et al., *Internet Indirection Infrastructure.* IEEE/ACM Trans. Netw., 2004. **12**(2): p. 205-218.

74. Janakiraman, R., M. Waldvogel, and Q. Zhang, *Indra: A Peer-to-Peer Approach to Network Intrusion Detection and Prevention*, in *Proceedings of the Twelfth International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*. 2003, IEEE Computer Society. p. 226.

75. Keromytis, A.D., V. Misra, and D. Rubenstein, *SOS: Secure Overlay Services*, in *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*. 2002, ACM: Pittsburgh, Pennsylvania, USA. p. 61-72.

76. Vlachos, V., S. Androutsellis-Theotokis, and D. Spinellis, *Security Applications of Peer-to-Peer Networks.* Computer Networks: The International Journal of Computer and Telecommunications Networking, 2004. **45**(2): p. 195-205.

77. Bernstein, P., et al. *Data Management for Peer-to-Peer Computing: A Vision*. in *Proceedings of the Fifth International Workshop on the Web and Databases*. 2002.

78. Huebsch, R., et al., *Querying the Internet with PIER*, in *Proceedings of the 29th International Conference on Very Large Data Bases*. 2003, VLDB Endowment: Berlin, Germany. p. 321-332.

79. *The Napster web site*. Accessed on-line 2008; Available from: http://www.napster.com.

80. Waldman, M., A.D. Rubin, and L.F. Cranor, *Publius: A Robust, Tamper-Evident, Censorship-Resistant Web Publishing System*, in *Proceedings of the*

9th Conference on USENIX Security Symposium. 2000, USENIX Association: Denver, Colorado. p. 5-5.

81. Ripeanu, M. *Peer-to-Peer Architecture Case Study: Gnutella network*. in *Proceedings of First International Conference on Peer-to-Peer Computing*. 2001.

82. Clarke, I., et al. *Freenet: A Distributed Anonymous Information Storage and Retrieval System in Designing Privacy Enhancing Technologies*. in *Proceedings of ICSI Workshop on Design Issues in Anonymity and Unobservability*. 2000.

83. Druschel, P. and A. Rowstron. *PAST: A Large-Scale, Persistent Peer-to-Peer Storage Utility*. in *Hot Topics in Operating Systems, 2001. Proceedings of the Eighth Workshop on*. 2001.

84. Stoica, I., et al., *Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications.* Networking, IEEE/ACM Transactions on, 2003. **11**(1): p. 17-32.

85. Dingledine, R., M. Freedman, and D. Molnar. *The Free Haven Project: Distributed Anonymous Storage Service*. in *Proceedings of the Workshop on Design Issues in Anonymity and Unobservability*. 2000.

86. Cohen, B. *Bittorrent*. 2007 [cited 2007]; Available from: http://www.bittorrent.org/index.html.

87. *The JXTA project web site*. Accessed on-line 2008; Available from: https://jxta.dev.java.net/.

88. Brian, D. *Brian's BitTorrent FAQ and Guide*. Accessed on-line 2008; Available from: http://dessent.net/btfaq/.

89. Adar, E. and B. Huberman, *Free Riding on Gnutella.* First Monday, 2000. **5**(10): p. 2-13.

90. Ramaswamy, L. and L. Ling. *Free Riding: A New Challenge to Peer-to-Peer File Sharing Systems*. in *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*. 2003.

91. Feldman, M., et al., *Free-Riding and Whitewashing in Peer-to-Peer Systems.* Selected Areas in Communications, IEEE Journal on, 2006. **24**(5): p. 1010-1019.

92. *Amazon Simple Storage Service (Amazon S3):*. Accessed on-line 2008; Available from: http://aws.amazon.com/s3/.

93. *SRB- "The Storage Resource Broker"*. Available from: http://www.sdsc.edu/srb/index.php/Main_Page.

94. Tierney, B.L., et al. *A Network-Aware Distributed Storage Cache for Data Intensive Environments*. in *Proceedings of The Eighth International Symposium on High Performance Distributed Computing* 1999.

95. Watson, R.W., et al. *The Parallel I/O Architecture of the High-Performance Storage System (HPSS)*. in *Proceedings of the Fourteenth IEEE Symposium on Mass Storage Systems (MSS'95)*. 1995.

96. *HDF5- "Hierarchical Data Format 5"*. Available from: http://www.hdfgroup.org/.

97. Barkes, J., et al., *GPFS: A Parallel File System.* IBM Redbook SG: p. 24-5165.

98. Tate, J., F. Lucchese, and R. Moore, *Introduction to Storage Area Networks*. 2006: IBM Corp.

99. *Green Bank Telescope*. 2008 [cited 2008 March]; Available from: http://en.wikipedia.org/wiki/Green_Bank_Telescope.

100. *Hubble Space Telescope*. 2008 [cited 2008 March]; Available from: http://en.wikipedia.org/wiki/Hubble_Space_Telescope.

101. Stewart, G.A. and G. McCance. *Grid Data Management: Reliable File Transfer Services' Performance*. in *Computing in High Energy and Nuclear Physics (CHEP'06)*. 2006. Mumbai, India.

102. Sotomayor, B. *The Globus Toolkit 4 Programmer's Tutorial*. 2005; Available from: http://gdp.globus.org/gt4-tutorial/.

103. Welch, V., *Globus Toolkit Version 4 Grid Security Infrastructure: A Standards Perspective*. 2005.

104. Mahemoff, M., *Ajax Design Patterns*. 2006: O'Reilly Media, Inc.

105. Kaplan, A., G.C. Fox, and G. von Laszewski, *GridTorrent Framework: A High-performance Data Transfer and Data Sharing Framework for Scientific Computing*, in *GCE07 Workshop*. 2007: Reno Nevada.

106. Gu, Y., *UDT: A High Performance Data Transport Protocol*. 2005, University of Illinois.

107. *Service-oriented architecture (SOA) definition*. 2007; Available from: http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html.

108. Novotny, J., S. Tuecke, and V. Welch, *An Online Credential Repository for the Grid: MyProxy*, in *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing*. 2001, IEEE Computer Society.

109. *Plos One: Publishing science, accelerating research*. 2008; Available from: http://www.plosone.org/.

110. Waldrop, M.M. *Science 2.0 -- Is Open Access Science the Future?* . April, 2008; Available from: http://www.sciam.com/article.cfm?id=science-2-point-0.

111. Rizzo, T., *Programming Microsoft Outlook and Microsoft Exchange*. 1999: Microsoft Press Redmond, Wash.

112. Antonovich, M., *Office and SharePoint 2007 User's Guide: Integrating SharePoint with Excel, Outlook, Access and Word*. 2008: Apress.

113. Vandyk, J. and M. Westgate, *Pro Drupal Development*. 2007: Apress.

114. Farmer, J. and I. Dolphin, *Sakai: eLearning and More*, in *EUNIS 2005-Leadership and Strategy in a Cyber-Infrastructure World*. 2005: Manchester, UK.

115. Gilster, R., *Microsoft® Office SharePoint® Server 2007: A Beginner's Guide (Beginner's Guide (Osborne Mcgraw Hill))*. 2007.

116. Mercer, D., *Building Powerful and Robust Websites with Drupal 6: Build your own professional blog, forum, portal or community website with Drupal 6*. 2008: Packt Publishing.

117. *Sakai Project*. 2008; Available from: http://sakaiproject.org.

118. *OpenWetWare*. 2008; Available from: http://openwetware.org/.

119. Lampson, B., *Protection.* ACM SIGOPS Operating Systems Review, 1974. **8**(1): p. 18-24.

120. Bishop, M., *Introduction to Computer Security*. 2004: Addison-Wesley Professional.

121. Ferraiolo, D.F., R.D. Kuhn, and R. Chandramouli, *Role-Based Access Control, Second Edition*. 2007: Artech House, Inc.

122. Ferraiolo, D., J. Cugini, and D. Kuhn. *Role-Based Access Control (RBAC): Features and Motivations*. in *Proceedings of the Eleventh Annual Computer Security Applications Conference*. 1995.

123. Sandhu, R.S., et al., *Role-Based Access Control Models.* Computer, 1996. **29**(2): p. 38-47.

124. Chadwick, D.W. and A. Otenko, *The PERMIS X. 509 role based privilege management infrastructure.* Future Generation Computer Systems, 2003. **19**(2): p. 277-289.

125. Dommel, H. and J. Garcia-Luna-Aceves. *Design Issues for Floor Control Protocols*. in *Proceedings of SPIE Multimedia and Networking*. 1995: SPIE.

126. Dommel, H. and J. Garcia-Luna-Aceves, *Floor control for multimedia conferencing and collaboration.* Multimedia Systems, 1997. **5**(1): p. 23-38.

127. Koskelainen, P., et al., *Requirements for Floor Control Protocols*, in *RFC Editor United States*. 2006, RFC 4376, February 2006.

128. *W3C World Wide Web Consortium*. 2008; Available from: http://www.w3.org/.

129. Bellwood, T., L. Clement, and C. von Riegen, *UDDI Version 3.0. 1: UDDI Spec Technical Committee Specification.* 2003.

130. Curbera, F., et al., *Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI.* IEEE Internet Computing, 2002. **6**(2): p. 86-93.

131. Erl, T., *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*. 2004: Prentice Hall PTR Upper Saddle River, NJ, USA.

132. *Corba: A Guide to Common Object Request Broker Architecture*, ed. R. Ben-Natan. 1995: McGraw-Hill, Inc. 353.

133. Redmond, F., *Dcom: Microsoft Distributed Component Object Model with Cdrom*. 1997: IDG Books Worldwide, Inc. Foster City, CA, USA.

134. Sun Microsystems, *Java Remote Method Invocation Specification.* Sun Microsystems, Palo Alto, CA, 1997. **30**: p. 31.

135. Kirtland, M., *A Platform for Web Services.* Microsoft Developer Network, 2001.

136. *List of Web Service Specifications*. 2008; Available from: http://en.wikipedia.org/wiki/List_of_Web_service_specifications.

137. Pearlman, L., et al., *A Community Authorization Service for Group Collaboration*, in *Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY'02)*. 2002, IEEE Computer Society.

138. Naedele, M., *Standards for XML and Web services security.* Computer, 2003. **36**(4): p. 96-98.

139. Lang, B., et al., *A Multipolicy Authorization Framework for Grid Security.* Proc. Fifth IEEE Symposium on Network Computing and Application, 2006.

140. Neuman, B.C. and T. Ts'o, *Kerberos: An Authentication Service for Computer Networks.* Communications Magazine, IEEE, 1994. **32**(9): p. 33-38.

141. Thompson, M.R., A. Essiari, and S. Mudumbai, *Certificate-based authorization policy in a PKI environment.* ACM Transactions on Information and System Security (TISSEC), 2003. **6**(4): p. 566-588.

142. *Shibboleth*. Available from: http://shibboleth.internet2.edu/.

143. Welch, V., et al., *Attributes, Anonymity, and Access: Shibboleth and Globus Integration to Facilitate Grid Collaboration.* 4th Annual PKI R&D Workshop, 2005.

144. Barton, T., et al. *Identity Federation and Attribute-based Authorization through the Globus Toolkit, Shibboleth, Gridshib, and MyProxy*. in *Proceedings of 5th Annual PKI R&D Workshop*. 2006.

145. Alfieri, R., et al., *VOMS, an Authorization System for Virtual Organizations.* European Across Grids Conference, 2003. **2970**: p. 33–40.

146. Globus Alliance. *The Globus Project*.]; Available from: http://www.globus.org/.

147. Katabi, D., M. Handley, and C. Rohrs, *Internet Congestion Control for Future High Bandwidth-Delay Product Environments.* Dina Katabi, Mark Handley, and Charles Rohrs, Internet Congestion Control for Future High Bandwidth-Delay Product Environments. ACM Sigcomm 2002, August 2002. URL http://ana.lcs.mit.edu/dina/XCP/. 2002.

148. Floyd, S., et al., *Equation-based congestion control for unicast applications.* Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, 2000: p. 43-56.

149. Lim, S.B., et al., *GridFTP and Parallel TCP Support in NaradaBrokering.* Distributed And Parallel Computing: 6th International Conference on Algorithms and Architectures for Parallel Processing, ICA3PP, Melbourne, Australia, October 2-3, 2005: Proceedings, 2005.

150. Burnap, P., Bulut, H., Pallickara, S., Fox, G., Walker, D., A. Kaplan, B. Yildiz, and Nacar, M. A. *Worldwide Messaging Support for High Performance Real-time Collaboration*. in *The UK e-Science All Hands Meeting*. 2005. Nottingham, UK.

151. Al-Kiswany, S., et al., *Are P2P Data-Dissemination Techniques Viable in Today's Data-Intensive Scientific Collaborations?* Lecture Notes in Computer Science, 2007. **4641**: p. 404.