Optimisation Techniques for Data Distribution in Volunteer Computing

Abdelhamid Elwaer

School of Computer Science & Informatics

Cardiff University

A thesis submitted in partial fulfilment of the requirement for the degree of Doctor of Philosophy

October 2012

Declaration

Abstract

Volunteer Computing is a new paradigm of distributed computing where the ordinary computer owners volunteer their computing power and storage capability to scientific projects. The increasing number of internet connected PCs allows Volunteer Computing to provide more computing power and storage capacity than what can be achieved with supercomputers, clusters and grids. However, volunteer computing projects rely on a centralized infrastructure for distributing data. This can affect the scalability of data intensive projects and when the projects participants increases.

In this thesis, a new approach is proposed to incorporate P2P techniques into volunteer computing projects and apply trust management to optimize the use of P2P techniques in these projects. This approach adopted a P2P technique to form a decentralized data centres layer based on the resources of participants of volunteer computing projects. VASCODE framework is based on Attic File System to enable building the decentralized data centres and makes use of trust framework to provide the necessary data to users to select the optimum data centres for downloading data.

Empirical evaluation demonstrated that the proposed approaches can achieve better scalability and performance as compared to the central server approach used in BOINC projects. In addition, it shows that clients with the support of trust framework have reliable and consistent download times because using trust allows them select the optimum data centres and avoid the malicious behaviour of data centres.

To the memory of my Mother Rabia
who always wanted me to be a ”doctor”

Acknowledgements

Praise to Allah (God) Almighty for providing me with faith, patience and the commitment to complete this research.

I would like to thank my supervisor, Dr Ian Taylor, for his expert guidance and support throughout this research. I would also like to thank Prof. Omer Rana for his support and constructive comments.

Special thanks go to the staff of the Cardiff School of Computer Science and Informatics, especially Robert Evans for his technical assistance and Helen William for her help on administrative issues.

I would like also to thank my colleagues, A. Alsaidi, H. Lenando, Abdalbaset Algreedi, for their support and encouragement.

Last, but certainly not least, I am indebted to my wife, Hayat, for her endurance and unconditional love which provided vital encouragement during my PhD study. Finally, I want to thank my family for their constant support and care throughout my life.

Publications

- **Conferences**
  - Abdelhamid Elwaer, Ian Taylor , *Decentralized Data Centres in Public Resource Computing Based on Trusted Workers and High Credit Scores*, UK e-Science All Hands Meeting, UK.8th-11th September, Edinburgh, UK
  - AbdelHamid Elwaer, Andrew Harrison, Ian Kelley et al. (2011) *Attic: A Case Study for Distributing Data in BOINC Projects*, 1863-1870. In 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum.
  - AbdelHamid Elwaer, Ian Taylor, Omer Rana (2011) *Preference Driven Server Selection in Peer-to-Peer Data Sharing Systems* . In The Fourth International Workshop on Data Intensive Distributed Computing (DIDC 2011).
- **Journals**
  - Abdelhamid Elwaer, Ian Taylor, Omer Rana (2011) *Optimizing Data Distribution in Volunteer Computing Systems using Resources of Participants.* In Scalable Computing: Practice and Experience 12 (2).

Contents

Bibliography **156**

List of Figures

List of Tables

Chapter 1

Introduction

Overview

Volunteer Computing is one of the distributed computing paradigms that has gained attention in recent years. It is used by a number of scientific researchers to perform scientific projects such as climate prediction, search for extraterrestrial live, and protein structure prediction. Volunteer Computing provides scientists with the necessary means for performing projects that require huge resources. However, the increasing number of participants in these projects can exceed the capabilities of the project servers to serve data requests from participants. Moreover, these large numbers of participants may cause a bottleneck on the servers, which may in turn lead to a point of failure and reduce the performance of these projects.

This thesis investigates possible techniques for improving computing's data layer through the use of P2P and trust management technologies. This chapter introduces the research presented in this thesis. The main motivation behind the intended research contribution is discussed. Finally, the organization of the thesis is outlined on a chapter by chapter basis.

1.1 Introduction

The last decades have witnessed a revolution in a wide range of technologies. The computing power of PCs underwent a noticeable improvement and the Internet became available to hundreds of millions of users. This development in technologies has led to the emergence of distributed computing to perform tasks that require more processing power. Distributed computing provides an efficient solution for projects that need intensive computing power by distributing project tasks to a number of machines for execution and returning the results to the main project server to form the final result.

Due to recent advances in science, an increasing demand has arisen among scientists for a means to run their scientific projects, and distributed computing paradigms started to evolve by exploiting the development in new technologies. New distributed paradigms have emerged, such as grid computing and Volunteer Computing.

In grid computing, computing becomes pervasive and users gain access to computing resources (processors, storage, data and applications) without the need to know where these resources are located [31]. Grid computing aims to provide scalable, secure, high performance mechanisms for allocating and acquiring access to remote resources and scientific collaborations can share resources; groups in different locations are also able to work together [57].

Volunteer Computing is a new paradigm within distributed computing where computer owners volunteer their computing power and storage capability to scientific supercomputing projects. The increasing number of internet connected PCs allows Volunteer Computing to provide more computing power and storage capacity than can be achieved with supercomputers, clusters and grids.

A number of different types of middleware [36],[78],[45] are used to create Volunteer Computing projects. BOINC [36] is a popular middleware for building volunteer computing projects. BOINC allows scientists to create scientific projects that utilize the idle CPU time of the participant's computers. The projects that use BOINC are listed in [5] and some of these projects have attracted hundreds of thousands of participants. Table 1.1 shows the top ten projects based on their participants.

Projects	Participants
SETI@Home	1,310,094
World Community Grid	385,807
Rosetta@Home	344,679
Einstein@Home	324,698
Climate Prediction	261,622
MilkyWay@home	143,508
LHC@Home Classic	102,128
ABC@home	65,981
Malaria Control	63,577
Spinhenge@home	58,706

Table 1.1: Top Ten BOINC projects based on their participants

Some of the current scientific projects that use BOINC are data-intensive and have attracted a large number of participants as shown in Table 1.1 For such projects, performance depends on efficient data distribution to its participants when processing their tasks. BOINC projects are based on a centralized architecture, therefore they use a fixed set of data servers to serve data requests received from their participants. Once the number of participants increases, a bottleneck can occur on the main server and therefore it can become a point of failure. To avoid this bottleneck, the BOINC project has to add extra servers, but this solution is unpractical since it is limited by the servers that are available.

This thesis explores an alternative to the centralized approach of data distribution in BOINC projects and this is the emergence of P2P file sharing, which can be used to download files from other BOINC clients instead of using a central server. P2P techniques have received increasing attention in the last decade due to file sharing software such as Napster [21], BitTorrent [4] and KazaA [18]. By using P2P techniques for downloading data in BOINC projects, the download time of data can be improved, thereby the overall time of job processing is reduced and consequently the efficiency of these projects increases. In addition to reducing the total duration of job processing, P2P also avoids bottlenecks since clients always have alternative places to download data from.

The research here focuses on leveraging such P2P techniques by building from the Attic file system[3], which creates a decentralized data centres layer in BOINC projects. Attic is a P2P file system, which has been developed in Cardiff University. It involves using four components for data distributing, data lookup, data publishing and data downloading. This research is extended by introducing a trust framework to reduce the drawbacks of using Attic where peers can have different capabilities and behaviours. A trust management system was researched to avoid malicious behaviour of peers and to allow clients to select the optimum peers for their requirements in downloading data.

This thesis presents a performance study of using P2P for data distribution and studies the effect of trust management optimization. The research toolkit described, VASCODE, was designed during this research and it provides a novel combination of P2P, BOINC, and trust management extensions. VASCODE aims to provide a general solution for volunteer computing projects that use P2P for distributing data.

1.2 Research Problem and Motivation

Volunteer Computing is one of the recent paradigms of distributed computing, where resources belonging to ordinary internet users are utilized for performing scientific projects that need intensive CPU processing power. The main motivation for using Volunteer Computing projects, particularly those that use BOINC middleware, can be summarized as follows:

- CPU intensive applications continue to increase in a wide range of scientific fields such as: protein prediction, climate prediction and search for extraterrestrial life.
- The recent development of the Internet in terms of bandwidth and speed, combined with the continued growth of internet users and the number of computers connected to the Internet, provides a huge range of resources for processing scientific projects.
- Volunteer Computing can leverage such resources to provide a general purpose solution for scientific projects.
- All users should be able to obtain highly scalable and performance systems without the need for expensive hardware.
- Volunteer Computing is an example of location transparency where applications can be run on geographically remote resources.

Since participants execute a scientific application and return results to the project owner and these are typically home Internet users and failure is common; BOINC repeats the jobs across different volunteers to get consistent results, thus, massive duplication is inherent in the use (this typically ranges from 2 to 5). This mechanism to validate results with the increasing number of project participants produces intensive data requests to the data server, which may cause bottlenecks in the data server and reduce project performance. This is because data requestors must wait some time to receive data or in the worst case the server goes down as a result of this congestion.

This thesis is motivated by the need to improve data management in Volunteer Computing projects focusing particularly on BOINC projects. It proposes a new approach to improve data distribution based on volunteer resources and by utilizing P2P file swarming with trust management.

With this goal, an alternative approach is investigated using Attic file system, which uses volunteers resources to share data instead of using a central server. A key advantage of the use of Attic is that the availability of data improves since a number of other volunteers owning it increases. In addition, using concurrent downloading reduces the download time of data and this will lead to reducing the total job processing time since this depends on the download time of input data.

Furthermore, this research aims to enhance the performance of using Attic by using trust management combined with Attic where the trustworthiness of each data resource is calculated to provide users with the most likely efficient resource for downloading data and avoiding any malicious behaviour.

1.3 Research Hypothesis

In existing Volunteer Computing system frameworks, such as BOINC, data is distributed from a centrally managed resource to large numbers of workers. By adopting data distribution techniques based on authenticated decentralised servers and file swarming P2P techniques, it is possible to dynamically select optimal servers using heuristics calculated from several metrics analysed by a self-adaptive trust-based model. These techniques will improve on this volunteer model considerably and offer a more reliable, scalable and efficient framework to enable far more data intensive projects to use volunteer computing within the scientific community.

Thus, the aim of the research is to investigate the use of Attic to extend the data layer in Volunteer Computing using participants resources. In addition, it aims to develop a trust management system using heuristics data for optimum use of this data layer.

1.4 Research Objectives

The aim of this thesis is to address the issues arising from using centrally managed resources for distributing data in Volunteer Computing systems, such as BOINC. More specifically, the aim is to investigate the use of decentralized resources and to develop a method that takes the trustworthiness of resources into account for selection in the downloading of data, assuming that the trustworthy of these decentralized resources is calculated using heuristics trust data. The aim of this research leads to a number of research objectives described as follows:

- **To investigate trust management techniques for establishing trust in decentralized resources.**

  The decentralized data layer is formed using the participants of a Volunteer Computing project. These participants are ordinary internet users who have different resources, different behaviours and different performance These differences introduce a heterogeneous decentralized data layer of data resources with different upload speeds, different availability and there is the possibility of malicious behaviour. Under these conditions, a trust management system is required to make optimum use of the decentralized data layer to allow users to select the best data resources in terms of upload speed and availability and to avoid any malicious behaviour.

- **To gain empirical evidence from Attic operation to feed into the requirements analysis for designing the trust based approach.**

  This research aims to investigate the performance of P2P file swarming where different numbers of chunks and different chunk sizes are used to download data. It will also explore the effects of concurrent download requests using different numbers of clients and data resources. In addition, it will conduct a comparative evaluation of using file swarming to download data in Volunteer Computing projects against the use of the central data server.

- **To investigate the way in which Attic file systems can be integrated into BOINC.**

  The goal of the research task is to provide Attic and BOINC with a new system that allows the existing participants of a Volunteer Computing project to form a decentralized data layer and to allow the new participants to join the decentralized data layer. In addition, this integration must provide a mechanism to use the decentralized data layer to obtain data instead of using the central data server.

- **To design and develop a system for Volunteer Computing projects that utilizes a trust layer.**
  To research and design a system to meet the requirements of the trust-based approach. This system should provide the necessary tools such as:
  Data centre, that allows users to volunteer a part of their resources to become a data distributor.
  Data client, that is used to allow the BOINC client to utilize the decentralized data layer for downloading data to process its assigned jobs.
  Trust service, that is used to provide the trust data about each data centre to the data client to make the selection decision before starting to download data. In addition, it is used by data clients report their feedback about the data centres that are used to download data.

1.5 Research Contributions

The contributions of this thesis are as follows:

- The design and implementation of the VASCODE framework that uses the Attic file system to improve data distribution in Volunteer Computing by utilizing volunteers resources. The design provides a novel combination of the Attic file system, trust management and BOINC middleware, which provides a method for creating Volunteer Computing projects.

- A performance study that shows the effect of using the Attic File System for downloading data compared to the use of a BOINC data server. Furthermore, it shows the performance of the Attic File System using various scenarios such as the effect of the number of data centres, the effect of chunk size and the number of concurrent clients on the download time. To integrate Attic into BOINC a new proxy called the VASCODE proxy was developed. This proxy differs from the proxy that was developed by the Attic team as it uses VASCODE-DW for downloading data and sends the feedback to the trust server. Furthermore, it allows the configuration of download preferences for selecting data centres.

- A trust model based on clients feedback and preferences. This model is used to avoid malicious behaviour of data resources and to select the best ones to download data from.

- A performance study of the system performance that show the impact on performance when the trust framework is used with different weight factors and scenarios.

1.6 Organization of the thesis

- **Chapter 2 Background**

  This presents the background of the research related to Volunteer Computing. The concept of Volunteer Computing is discussed and the BOINC middleware is presented. In addition, other middleware that are related to the research are surveyed, such as Condor, XtremWeb and Entropia.

- **Chapter 3 Related Work**

  This chapter surveys the relevant literature in terms of the current research areas which are related to thesis work. In this chapter, P2P systems and trust systems are also covered.

- **Chapter 4 Evaluation of the current state of the art**

  The aim of this chapter is to evaluate the Attic file system from the requirements analysis perspective. It begins with a description of the Attic file system and its components, its message and how they are used to download data. Following the introduction of Attic, a performance study to compare Attic file system with the performance of BOINC is presented.

- **Chapter 5 System Architecture**

  This describes the architecture and design of the research framework to tackle the problem. Furthermore, it provides the main concept proposed by the thesis to integrate Attic and employ trust within the system to improve the usage of the proposed system by describing and introducing the proposed trust model.

- **Chapter 6 Implementation**

  This chapter covers the implementation of the system and shows how VASCODE is used to integrate Attic into BOINC middleware to create the decentralized data layer. In addition, it presents how the trust framework can be used to improve the performance of the system.

- **Chapter 7 Evaluation of the Hypothesis**

  This presents a quantitative evaluation of the proposed system. Several experiments are conducted to show how the system works. Furthermore, it shows the performance of the system in various scenarios.

- **Chapter 8 Conclusion and Future work**

  This concludes the thesis with a summary of the original research contribution and outlines the scope for future research.

Chapter 2

Background

This chapter presents the idea of Volunteer Computing, which allows supercomputing projects to perform easily and inexpensively by utilizing idle CPU cycles of internet users. It also presents BOINC middleware, which is an open source middleware for Volunteer Computing that enables computer owners to donate their computing resources (such as processing power and storage) to one or more projects. Finally, some of the other systems that use idle CPU cycles are also presented.

2.1 Introduction

Recent decades have witnessed two major advances in computer technology. The first was the rapid growth in processing capability and according to Moore's law [82], this growth doubles every 18 months. The development in microprocessor technology means that nowadays personal computers have the same processing power as older mainframe computers. The second was the development of high speed networks that led to simultaneous connection between millions of computers around the world. This has led to the emergence of distributed systems which Tanenbaum defines as follows [98]:

A distributed system is a collection of independent computers that appear to its users as a single coherent system.

This definition presents a distributed system as consisting of autonomous components, and the complexity of such a system is hidden from its users, who perceive they are interacting with a single machine. The main motivation behind the use of distributed systems is the sharing of resources, including hardware components such as storage capacity, CPU power, printers, and other peripherals, or software components such as files, data objects, and databases. Distributed systems provide a scalable and fault tolerant environment in which to share these resources. In addition, distributed systems often have a better price/performance ratio and computing power than centralized systems.

The emergence of distributed systems has had a significant impact on science. It introduced a new way of conducting computations in which a large computation problem is divided into small parts and distributed to many computers to solve them individually; the result of these parts is combined to form the solution to the problem. The primary advantage of using distributed computing is that it provides super computing power to its users when they cannot afford supercomputers. In research institutions and organizations, researchers can benefit from the distributed computing paradigm by utilizing individual PCs to conduct scientific projects that require large computation power. In the last decade, distributed computing has been used [37],[25],[10]to perform computational tasks on volunteers computers via the Internet. This type of distributed computing is known as Volunteer Computing and has attracted great attention in the last decade due to the popular project SETI@home [37].

2.2 What is Volunteer Computing

In the mid-1990s, the computing power of PCs underwent a noticeable improvement and the Internet became available to hundreds of millions of ordinary users.This led to the emergence of Volunteer Computing. Volunteer Computing is a new paradigm of distributed computing where ordinary computer owners volunteer their computing power and storage capability to scientific supercomputing projects. The increasing number of internet connected PCs allows Volunteer Computing to provide more computing power and storage capacity than can be achieved with supercomputers, clusters and grids [35].

Volunteer Computing has some differences with grid computing and P2P systems. Grid computing involves the sharing of resources within or between organizations, such as universities, research labs and companies. These resources are secure, trusted and centrally managed. On the other hand, Volunteer Computing involves the use of resources donated by ordinary internet users; the Volunteer Computing project has no control over its participants and the participants are not accountable, so the intentional return of incorrect results by a volunteer is overcome by the use of redundancy mechanisms. Volunteer Computing uses central servers and there is no P2P interaction between participants.

In order for Volunteer Computing projects to attract volunteers to join a project they need to create trust in the project in several ways:

- The application provided by the project will not breach their privacy or infect their computers with viruses.
- The project uses proper security measures to prevent the project becoming a vehicle for malicious activities.

- The project is truthful about how the results will be used.

2.3 Scientific and Volunteer Computing

The 20th century saw a revolution in the development of science with computer software accurately simulating the reality of different phenomena; an atomic nucleus, protein molecules, the earths biosphere, or the entire universe. Using these simulations scientists can validate or disprove theories, predict future events and investigate chemical reactions. Scientific projects requiring large computation power stimulated the development of super computers. In the mid 1990s scientists started using Volunteer Computing with two projects: GIMPS [16] and distributed.net [9]. In 1999 the popular SETI@home [26] was launched, and today more than 70 scientific projects use Volunteer Computing.

2.4 Potential of Volunteer Computing

Volunteer Computing has the potential to use the computing power of millions of computers connected to the internet. According to [17], the number of computers connected to the Internet is more than two billion. Even if a fraction of these computers work together it will produce tremendous computing power, more than that generated by any fast super computer. SETI@home project has already gained 575.026 teraflops [6]. One analysis of BOINC [38] has shown it is possible to reach processing at a sustained rate of 95.95 teraflops with 7.74 petabytes of storage and 7.74 terabytes/sec access rate.

These enormous resources can be gained at low cost using Volunteer Computing, whereas super computers can only provide such resources to certain academics or research centres because super computers are very expensive.

A number of systems have been developed to encourage the creation of projects to harness the wasted computing power of idle CPU cycles on desktop computers and PCs that are connected through LAN networks or the Internet, and to encourage Internet users to participate in these projects. In the following sections some of the popular systems are presented.

2.5 BOINC Middleware

BOINC [36] is a platform for Volunteer Computing and it is being developed by the team that developed the popular project SETI@home. BOINC is an open source software and can be downloaded from *http://boinc.berkeley.edu/trac/wiki/SourceCode.*

2.5.1 Goals of BOINC

The general goal of BOINC is to advance Volunteer Computing by encouraging the creation of many projects and to encourage computer owners to participate in these projects; BOINC also has specific goals which can be summarized as follows:

- To reduce the barriers of entry to Volunteer Computing projects.
- To share resources among autonomous projects.
- To support diverse applications.
- To reward participants.

2.5.2 BOINC Architecture

BOINC uses a Client/Server architecture. The server sends tasks to the client who performs the computation and uploads the results. Volunteers can join a BOINC project by downloading and running a BOINC client on their computers. Figure 2.1 shows the architecture of BOINC.

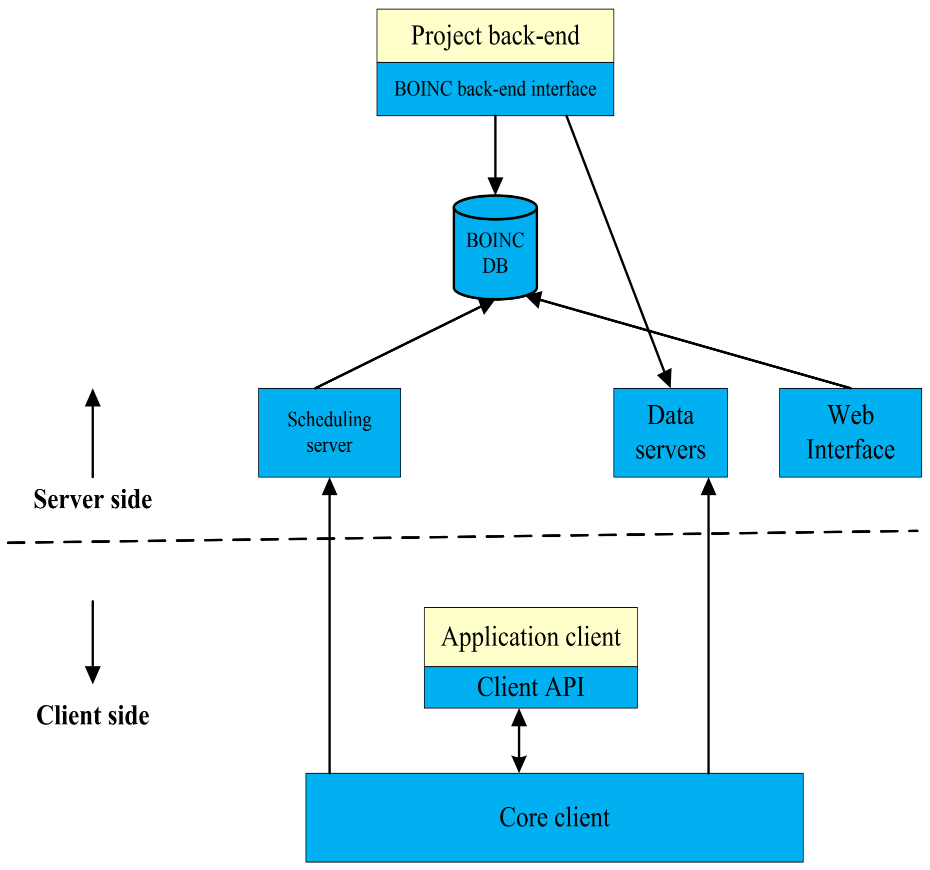


Figure 2.1: BOINC Architecture

2.5.2.1 BOINC Server Side

The server side of BOINC consists of two parts:

- The project back-end is responsible for providing applications and work units and it handles the uploaded results.
- A BOINC server consists of a scheduling server to communicate with participating hosts, a data server to distribute input files and collect output files, and a database for storing information about participants, work units and results. In addition it provides a web interface to interact with those participating in the project.

2.5.2.2 BOINC Client Side

The main component in a BOINC client is the core client that communicates with the scheduler to obtain work units and to upload the results. The core client also uses run-time libraries to interact with the application that is used to execute the work units. The BOINC client uses a graphical interface to allow users to control the computation status. In addition, it uses a screensaver that runs when the participant computer is idle.

2.5.3 Interaction between BOINC Client and BOINC Servers

The BOINC client makes a number of requests to the BOINC servers to obtain work units and report the results. Figure 2.2 shows these requests. The BOINC client starts by downloading the project master page that contains the scheduler URL, which is used to exchange requests and response messages to the scheduler server. The request messages ask the scheduler for work units to be processed and the reply message describes the work units and the download URL for the input files. The client also communicates with the data server to download the application executable file or the input files. In addition the data server is used to upload the output files when the work unit is processed.

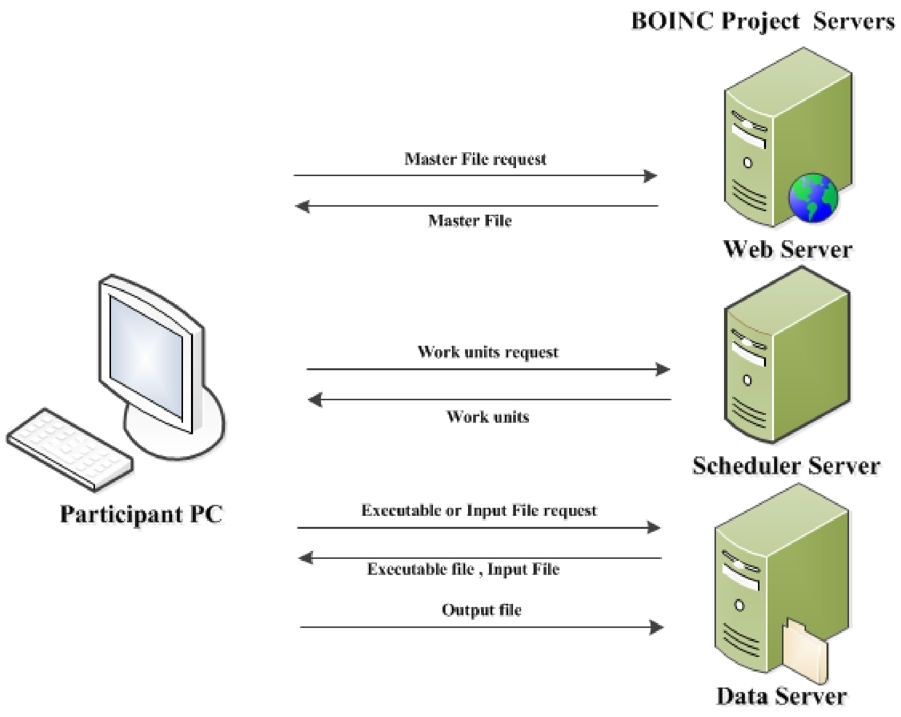


Figure 2.2: Interaction of between BOINC Client and BOINC Server

2.5.4 Projects that currently use BOINC

BOINC has been used by a number of scientific projects which range from biological, medical, and earth sciences, including:

2.5.4.1 SETI@home

SETI@home [37] is developed by a space science laboratory at the University of California and its goal is to analyse radio telescope data from the Arecibo radio observatory. SETI@home currently has 1,296,572 users and has gained 575.026 Teraflops.

2.5.4.2 Climateprediction.net

Climateprediction.net project [7] is used to investigate and reduce uncertainties in climate modelling. It aims to produce a prediction of the Earths climate up to the year 2100 and test the accuracy of current climate models. It has 260,315 users and has gained 36.955 Teraflops.

2.5.4.3 PrimeGrid

PrimeGrid [24] is used to search for prime numbers of world-record size. PrimeGrid aims to bring the excitement of prime finding to ordinary computer users and provide relevant educational material about primes. It currently has 50,318 users and has gained 735.242 Teraflops.

2.5.4.4 Einstein@home

Einstein@home [10] searches for gravitational waves from spinning neutron stars using data from the LIGO [19] gravitational wave detector. it also searches for radio pulsars in binary systems using data from the Arecibo observatory in Puerto Rico [2]. Einstein@home has 321,786 users and has gained 245.333 Teraflops.

2.6 Other Idle CPU Cycle Sharing Systems

This section presents some of the Idle cycle sharing systems that have been developed to utilize idle computing power resources and enable processing projects that require high computing power.

2.6.1 Condor

Condor [78] is one of the first distributed computing middleware systems. It aims to provide management mechanisms, such as scheduling and resource monitoring, and attempts to maximize the utilization of the resources of workstations not being used by their owners. These resources are then used to meet the demands of other Condor clients. Condor uses a central coordinator, which assigns jobs to remote machines and retains information on the status of each job and workstation availability. In Condor, users submit their jobs and the coordinator chooses to assign resources to the jobs based on the scheduling policy. The coordinator informs the user upon completion.

ClassAds language is used to provide a flexible and expressive framework for matching jobs with the available resources. Condor supports application check pointing; this allows the job to resume on a new resource using the checkpoint file. The periodic checkpoint of jobs provides fault tolerance and allows the migration of jobs from one resource to another, using a push mechanism to distribute jobs to the resources. Unlike BOINC, Condor trusts the resources it uses, since they are in the same institutions or academia.

2.6.2 XtremWeb

The XtremWeb Project [45] attempts to meet the large computing requirements of physicists of the Pierre Auger observatory. It aims to build a platform for experimenting with global computing capabilities with scalability, fault tolerance and security which will be able to adopt varying configurations as well as changes in communication latency and throughput. XtremWeb can be used in two ways. Firstly, for a user who downloads worker software, it can be used to process tasks when the user machine is idle. Secondly, a collaborator who downloads the entire XtremWeb software can use it to set up his own global distributed application and the XtremWeb server exploits unused resources gathered by the collaborators.

Workers in XtremWeb initiate communications to the server to avoid any firewalls that may block incoming requests from the server. The protocol used between the worker and the server in XtremWeb consists of four requests. Firstly, it uses the *hostRequest* for authentication and to obtain a list of servers that may provide tasks to the worker. Next, the worker asks the server for a job using the *workRequest* and then during the computation the worker periodically sends a *workAlive* to signal its activity to the server. Finally, when the computation is finished, the worker sends back the results to the server using a *workResult* message.

XtremWeb stores some information about each task, such as the host that has performed the task and the client who submitted the task. This information is used to provide feedback to the users about their machines or their tasks. To allow its users, clients, and administrators to interact with the system, XtremWeb provides a Web interface. This can be used to submit tasks, obtain statistics, or to monitor the servers and, when used by an administrator, to perform maintenance.

As with BOINC, XtremWeb allows its users to organize themselves and form teams, and it provides them with statistics such as the rank of the teams based on the time they have spent in computation.

2.6.3 Entropia

The Entropia system [48] is a commercially distributed computing system intended to provide the essential benefits of distributed computing projects such as efficiency, robustness, scalability, and unobtrusiveness. It is used to aggregate raw desktop resources into a single resource; this resource provides high performance for applications through parallelism and is managed through a single administrative console.

In the Entropia system, sandboxing techniques are used to enable applications to be deployed without the need to modify its source code or the use of a special system support. This allows the execution of a large number of applications and supports its executions in a secure manner. Entropia can support applications that are written in different programming languages such as C, C++, Java, and FORTRAN.

The architecture of Entropia is composed of three layers; the bottom layer is the physical layer management that is used to provide resource management, security, application control, communication and naming. The resource scheduling layer in the middle provides scheduling, resource matching and fault tolerance. The top layer is the job management layer and provides the users with access to the system. This facilitates the management and handling of jobs and is used to decompose a single job into a number of sub-jobs to manage the progress of the job, to provide access to the status of the sub-jobs, and to aggregate the results of the sub-jobs. Entropia allows the addition of extra schedulers to the system to provide scalability with increasing numbers of clients and ensure fault tolerance against scheduler failure.

Entropia employs encryption to prevent any unauthorized access to the data files. In addition, it regularly checks the integrity of the application, its data and the results files on the user machine to ensure that there has been no tampering with them.

These techniques used in Entropia and its three layer architecture aim to provide a solid foundation for projects that utilize the idle CPU cycles on the pervasive desktop PC systems.

2.7 Classification of Idle CPU Cycles Systems

In this section, the Idle CPU Cycles Systems discussed in the previous section are investigated and analysed from a system perspective. They are categorized according to two levels. The first level refers to their infrastructure which includes resource types, platforms, scalability, and security; the second level includes the computing model, architecture and data model.

2.7.1 Resource type

Resource types specify how resources are provided to the system. There are types volunteer and enterprise. Volunteer types relie on Internet users who volunteer their resources for these systems, while enterprise types are based on non-voluntary participants usually within a corporation, research lab or university which are connected through a LAN network. Volunteer types are more volatile and fault-prone while enterprise types are more controllable since the resource providers are located in the same administrative domain. Typical examples of a volunteer type are BOINC and XtremWeb, and of an enterprise type are Entropia and Condor.

2.7.2 Scalability

Scalability divides these systems into two groups: Internet-based and LAN-based. Internet-based systems are characterized by anonymous resource providers, connectivity issues (Firewall, NAT, unreliable connection) , malicious resources and security risks. In contrast, LAN-based systems have reliable connectivity and lower security risks and under certain degree of control.

2.7.3 Security

Security deals with aspects of access to the computational resources through authentication and authorization techniques, and access to the computational data and results by providing data integrity and encryption. It is also necessary to protect data integrity and validate results because the computations are run in non-trustable environments.

2.7.4 Computing Model

There are two categories of computing model: the first one is the typical master-worker model, consisting of independent tasks. The other category involves tasks which depend on each other - there is either an execution flow between tasks such that one task needs to be executed only after other tasks are finished, or tasks run in parallel with data communication between each task. Typical paradigms involved are PVM, MPI.

2.7.5 Architecture

This is categorized into centralized and hierarchical according to the architecture of each system. A centralized system consists of a central server to distribute tasks to the available resources based on some scheduling algorithm. Typical examples are BOINC, XtremWeb and Entropia. In a hierarchical system, groups of computing resources can share resources. Condor features a mechanism for sharing resources among condor pools. By using this technique, a condor pool is able to accept job requests that are forwarded from a remote pool.

2.7.6 Data model

A data model is used to describe the transfer of computational data (input/output data) between the components of the system. Two models are identified: the data server model and the network file system model. In the data server model, the job submitter is responsible for uploading the input to the data server and for retrieving the results, while the resource provider is responsible for downloading the input data and uploading results. BOINC , XtremWeb and Entropia use the data server model. In a network file system, each component has access to a common file system by using a distributed file sharing mechanism. Condor applies this model.

System	Infrastructure level				Model level			License	Project Users
	Resource	Platform	Scalability	Security	Computing model	Architecture	Data model		
BOINC	Volunteers	Internet	Authentication, data integrity	Master/worker	Master/worker	centralized	Data server	Open source	Hundreds of Thousands
XtremWeb	Volunteers	Internet	Authentication	Master/worker, MPI	Master/worker, MPI	centralized	Data server	Open source	Few thousands
Condor	Enterprise	LAN	Authentication	Master/worker,PVM, MPI	Master/worker,PVM, MPI	hierarchical	File system	Open source	Few Thousands
Entropia	Enterprise	LAN, Internet	Encryption	Master/worker	Master/worker	centralized	Data server	Closed	Few Thousands

Table 2.1 Classifications of the Idle CPU Cycles Systems

The comparison in Table 2.1 shows the advantages of using BOINC as the volunteer computing middleware in this research because it supports true volunteer computing projects compared to Condor and Entropia which are more suited for enterprise use. In addition, some BOINC projects have attracted hundreds of thousands of participants, who need a better data distribution mechanism to avoid congestion on the central server and improve the project performance. Furthermore, BOINC is an open source project which allows modification to the platform to build our system.

2.8 Summary

This chapter has described Volunteer Computing, in particular its scientific applications where ordinary computer owners volunteer their computing power and storage capability to scientific supercomputing projects. The premise is that if only a tiny fraction of the PCs in the Internet connect and work together, it has the capacity to provide greater computing power than any super computer.

The chapter also presented the BOINC platform for Volunteer Computing as a means of encouraging computer owners to participate in specified projects. The BOINC architecture, its interaction between BOINC clients, servers and projects using BOINC were described.

Condor, one of the first distributed computing middleware systems, was briefly discussed, in particular how it matches jobs with available resources and how much more trusting Condor is than BOINC.

Two other systems were mentioned: the XtremWeb Project, which attempts to meet the large computing requirements of physicists of the Pierre Auger observatory and the Entropia system, which is a commercially distributed computing system that can support applications written in different programming languages, such as C, C ++, Java, and FORTRAN. The three layer management technique used by Entropia to provide a solid foundation for projects that utilize idle time of CPUs of PC systems was also described.

Chapter 3

Related Work

Increasing computing power and the growth of the Internet have led to a number of new paradigms and terms related to distributed computing. This chapter presents those paradigms and terminology that are used in this thesis. In addition, it presents an overview of the trust techniques used in peer-to-peer (P2P) systems. This chapter is organized as follows: Section 3.1 presents grid computing, it begins by defining grid, then the types of grid and Globus toolkit used in grids are listed. Section 3.2 discusses P2P systems and reviews the types and structure of P2P networks. Section 3.3 presents P2P Storage networks. Section 3.5 discusses the concepts of trust and reputation and the different techniques used in P2P networks to establish trust.

3.1 Grid Computing

Grid computing refers to the federation of computing resources from different administrative domains and provides scalable access to those resources. The term grid comes from power grid, since it is similar to the concept of the grid supplying electric power. Grid computing intends to provide an equally consistent, dependable, and transparent collection of computing resources. In grid computing, computing becomes pervasive and users gain access to the computing resources (processors, storage, data, and applications) without the need to know where these resources are located [31]. Grid computing aims to provide scalable, secure, high performance mechanisms for allocating and acquiring access to remote resources, and by using grid computing scientific collaborations, users can share resources and groups in different locations are able to work together [57].

Grid computing has been defined in a number of ways [58, 60, 88]; in 1999 Foster and Kesselman wrote: "*A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities*" [73]. In 2000, Foster, Kesselman and Tuecker [60] redefined grid computing to address social and policy issues, stating that grid computing is concerned with "*coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations*". Foster in [58] suggested a checklist of three points to define grid computing :

1. Coordinates resources that are not subject to centralized domain,
2. Uses standard, open, general-purpose protocols and interfaces, and
3. Delivers nontrivial qualities of services.

3.1.1 Grid Types

Grids are used to provide an integration of heterogeneous computing resources, such as processors, storage, data and applications, and the grid implementation generally focuses on the integration of a specific type of resource [29, 30]. As a result, there are different types of grids, the two primary types of grid being described next.

3.1.1.1 Computational grid

With this type of grid, processing power is the main resource shared amongst users; it provides access to a huge pool of shared processing power, which is suitable for high throughput applications, and performing intensive computing projects. The GridX1 [34] is an example of a computational grid project, it uses shared resources provided by several Canadian institutions. GridX1 has been used by physicists in the Large Hadron Collider (LHC) project [28]at CERN, and in the BaBar experiment at the Stanford Linear Accelerator Center (SLAC) [27].

3.1.1.2 Data Grid

A data grid is a type of grid that uses storage capacity as its main shared resource. It provides its users with access to data across multi-institutional and heterogeneous environments [47]. The DataGrid project [8] is one example of a data grid and it aims to build the next generation of infrastructure to provide intensive computation and analysis of shared large-scale databases across scientific communities.

3.1.2 The Globus Toolkit

As grid computing evolves, it is important to have tools available to create and modify applications to run on grids. The Globus [15, 59] toolkit has been developed at the Argonne National Library Illinois, USA, to support the development of applications for grids; it is a collection of software components, which provides a set of facilities required for grid computing, such as security, execution manager, data management and information services. The core services, interfaces and protocols in the Globus toolkit enable users to access remote resources and preserve local control over who can use these resources and when.

3.2 P2P Systems

P2P systems became more popular in the last decade through the use of file sharing application such as Napster [21], Gnutella [1], eMule [11], and KaZaA [18]. The concept of P2P systems is based on resource sharing in a dynamic environment [83]. P2P systems are used to access various distributed resources (processing power, storage capability and bandwidth) at the edge of the Internet; these resources are shared between users by direct exchange [39].

Two common definitions of P2P systems used by the P2P community are those of Oram:

"P2P is a class of applications that take advantage of resources storage, cycles, content, human presence available at the edges of the Internet" [86],

and Stephenson and Spinellis:

"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" [39].

According to these definitions, the main goal of P2P systems is to provide scalable systems and avoid the weaknesses in central systems for the purpose of resource sharing. P2P systems are classified in the literature into different types.

The following sections present these types.

3.2.1 Classification of P2P Systems

The pure forms of P2P systems are supposed to be totally decentralized, but in practice there are systems with various degrees of centralization [99], so that P2P systems can be classified by their degree of centralization [39], that is the extent to which peers use central servers before they establish connection between themselves. There are two such categories: hybrid and partially centralized.

3.2.1.1 Hybrid P2P Networks

In the hybrid P2P networks [106], peers are connected through a central server, which acts as a lookup server between the peers to identify those peers that store files. This central server maintains metadata describing the files that are currently shared between peers and the metadata contains the address of peers where the file is stored. The central server facilitates the interaction between peers by providing the metadata of the requested file. Using this metadata peers establish a direct connection between themselves to download the requested file. This type of P2P system is simple and efficient for discovery of information as it offers a comprehensive search and provides a guarantee in search, but it is prone to a single point of failure because of the central server.

Napster [21]is one example of hybrid P2P systems. In Napster every peer is connected to a centralized lookup server, which maintains a list of files that peers offer. A peer can issue a request for a file to the lookup server; if the request can be resolved by the lookup server, it returns the address of those peers offering the file. The peer who issued the request uses the address to establish a connection to that peer and downloads the file.

3.2.1.2 Partially centralized P2P Networks

Hybrid P2P networks feature super peers [40, 54], which are dynamically assigned to the task of serving a small group of nodes in the network. Super peers are automatically selected based on their resources (bandwidth and processing power) to provide a file indexing service for their connected peers. When a peer submits a query, it sends it to its super peer, who will respond to the query or it forwards the request to other super peers if it cannot satisfy it and forward the response message it receives back to the peer. In comparison with purely decentralized P2P systems, partially decentralized P2P systems have less discovery time, and the advantage that there is no point of failure such as in Hybrid systems, where if one of the super peers is subject to failure, the peers connected to it open a new connection with another super peer. KaZaA [77] and Morpheus [20] are examples of such P2P systems.

KaZaA is a partially centralized system that uses the concept of super nodes. The nodes with sufficient processing power and bandwidth are selected as super nodes using proprietary algorithms based on FastTrack [13]. Super nodes are used to index files shared by nodes connected to them, and to proxy search requests on behalf of these nodes. In KaZaA the discovery time is reduced in comparison with Gnutella. In addition there is no single point of failure as in Napster; if one or more super nodes goes offline, the nodes connected to them open new connections with other super nodes and the network continues to operate.

3.2.1.3 Pure P2P Networks

In pure P2P networks such as the original Gnutella architecture [1] and Freenet [14] there is no central coordination of the activities in the network, the peers are directly connected to each other; and perform the same tasks by acting as servers and clients. Pure P2P networks are inherently fault-tolerant, since there is no central point of failure. On the other hand these systems are not scalable and have slow information discovery with no guarantee of the quality of service [89].

Gnutella

Gnutella [1] is a purely decentralized P2P network in which the users form a self-organized network and connect to each other directly without any central coordination. At the start up, the Gnutella user connects to a known Gnutella node to obtain a list of existing Gnutella nodes to connect to. Because Gnutella is dynamic and nodes can go offline anytime, once the new node has joined the network it periodically pings its neighbours to stay in contact and prevent it from being disconnected from the network. Since Gnutella is an unstructured network and it is not known where data is located, the Gnutella user uses a flooding scheme [102] to locate a file; the user sends a search query to all its neighbours and these neighbours forward the message to their neighbours until the file is located or the TTL is reached. Gnutella is completely decentralized, there is no single point of failure, such as in Napster. On the other hand the search method used to locate files causes large loads on the network [79].

3.2.2 P2P Network Structure

P2P systems form an overlay on the underlying physical network [101]. When the overlay network is created based on specific rules this type of P2P system is known as structured and when it is created ad-hoc it is called unstructured.

3.2.2.1 Unstructured P2P Networks

Unstructured P2P networks do not use any structure when the overlay network is created; the overlay links are established arbitrarily [103] and the peers organize to form an ad hoc network. When a new peer joins the network, it connects to a set of nodes that can be used to propagate queries to retrieve data. As the data location is unrelated to the overlay network, there is no indication which peers have the requested data. When a peer searches for data, the request message floods through the network to find those peers that share the requested data. Gnutella is one example of this type of P2P systems.

3.2.2.2 Structured P2P Networks

Structured P2P networks [81, 91, 111] were developed to address the scalability issues that face unstructured networks. With structured networks, an overlay network is formed using specific criteria. CAN [91] is a system using n-dimensional Cartesian coordinate space to implement a distributed location and routing table. Each node is responsible for a zone in the coordinate space. CAN uses a distributed hash table (DHT) [52] to map between data and its location, so that a data query can be efficiently routed to the peer that requested the data. Chord [97] is a decentralized P2P network that performs a mapping of keys onto nodes;

it uses consistent hashing [71] to assign keys to the nodes. In Chord, the data location is implemented by associating a key with each data item, and storing the key/data item pair at the node to which the key maps. The nodes in Chord are arranged in a ring and each node maintains a routing table. Using this routing table, the number of nodes that must be contacted to find data in an N-node network is O(logN). For efficient routing, the routing tables are updated when a node leaves or joins the network.

3.3 P2P Storage Systems

One use of P2P systems is to form a P2P data storage system which uses the storage capabilities of peers as the storage space to store and share content. This provides reliable services by guaranteeing availability and durability of shared data. In the past few years many P2P data storage systems have been proposed to take the advantage of the rapid growth of the internet and disk size. The following sections present some of these systems.

3.3.1 Cooperative File System

Cooperative file system (CFS) [51] was developed at MIT - Massachusetts Institute of Technology to provide an efficient, robust and scalable infrastructure for P2P storage systems; it also aims to provide load balance of file storage and retrieval in a completely decentralized architecture that can scale to large systems. CFS uses Chord to maintain routing tables for lookup and query management. In CFS the data is split into several blocks before it is stored on different nodes to avoid the problem that one node is not capable of storing the complete data

[65]. To achieve availability and load balance CFS uses replication and caching of data. The users of CFS can only read data, but the publishers can update their data.

3.3.2 OceanStore

OceanStore [75] has been proposed by the University of California to provide utility infrastructure for distributed access to persistent data. OceanStore is designed to be constructed from an untrusted infrastructure and to support nomadic data. OceanStore protects data through the use of encryption and access lists (ACL) are used to restrict access to data. In OceanStore, objects are identified using a globally unique ID (GUID) and replicated on multiple servers to ensure availability. OceanStore provides two ways of locating a replica for an object: initially, a fast, probabilistic algorithm is used to find the object nearest the requesting node; if this fails, a slower, deterministic algorithm is used.

3.3.3 Malugo

Malugo [46]is a P2P storage system that connects peers and allows them to upload encrypted chunks of their data to other peers. Malugo uses Chord to construct an overlay network; this overlay clusters neighbouring nodes to provide services within a local region. When a new node joins the network, it will be located in an appropriate group, if this group is not close enough it forms a new group. Malugo replicates any new file inserted into the system to all groups of the system. This avoids data retrieval problems when some nodes are offline. Malugo was designed for large-scale collaborative projects, and to achieve load balancing and replicating of data.

3.3.4 BitTorrent

BitTorrent [49] is a file distribution system used for distributing large amounts of data over the Internet. It is used for efficient and scalable replication of large amounts of data. In BitTorrent, the throughput increases with the number of downloads. A file in BitTorent is split up into chunks and its SHA-1 hash value is calculated before it is downloaded. The tit-for-tat mechanism is used to prevent parasitic behaviour of users. BitTorrent is one of the popular P2P systems for transferring files and has 150 million active users according to BitTorrent, Inc.

3.3.5 Attic File System

Attic file system [3, 56]developed at Cardiff University is a decentralized, P2P data sharing software architecture for accessing distributed storage resources available over a network in a similar way to BitTorrent. Attic consists of four main elements: (i) a data serving application that replicates data on the network; (ii) data centres that cache data, providing the distributed data source overlay; (iii) a lookup service that keeps track of which data centres have individual data items, and (iv) client applications that download data from data centres on the network. The primary differences between Attic and BitTorrent are the concept of data centres and the use of HTTP. Attic is discussed in more detail in Chapter 4.

3.4 Comparison of Different Peer-to-Peer File Systems

The attractive properties of P2P systems have generated much research effort in building distributed P2P file systems. This section makes a comparison (shown in Table 3.1) among the P2P storage systems discussed in the previous section using selected criteria. These criteria include: degree of centralization, i.e to what extent do the systems rely to a central server to facilitate the interaction between nodes, the mechanism used for resource location, which is an important P2P design issue, load balancing, security, data persistence, and other criteria that affect the system performance.

System Attribute	Location Scheme	Lookup Time	Load Balance	Encryption	Data Stored	Data Persistence	Read/write	Secure data centre	Http Interface
CFS	Chord	O(log N)	Yes	No	Blocks	Yes	Read Only	No	No
OceanStore	Tapestry	O(log N)	Yes	Yes	Fragments	Yes	Read/Write	No	No
Malugo	Chord	O(log N)	Yes	NO	Complete File	Yes	Read Only	No	No
BitTorrent	Global Components	Deterministic	Yes	NO	Chucks	Yes	Read Only	No	No
AtticFS	Centralized Lookup	O(1)	Yes	NO	Chunks	Yes	Read Only	Yes	Yes

Table 3.1: Comparison of different P2P storage Systems

Table 3.1 shows the advantages of using Attic over the other systems. Attic is an effort to build a P2P software layer that can be used by scientific applications, specifically those engaged in volunteer computing, to distribute, manage, and maintain their data. In addition, Attic introduces the idea of a secure data centre through the use of X.509 certificate. To authorize a group of peers to cache and distribute data, it is possible to centrally issue X.509 certificates to the data caching peers, and the downloading peers verify the certificates against the central certificate authority. Furthermore, Attic uses HTTP for its data transfer layer, leveraging the byte-range attribute of HTTP requests to allow for concurrent downloads from multiple Data Centre hosts. Concurrent downloading helps to maximize client download bandwidth, as well as distribute network load. Additionally, the use of HTTP transactions allows easy integration with existing software and firewall configurations and client applications that choose not to serve data require only an out-going HTTP connection.

3.5 Managing Trust in P2P Systems

P2P systems are often established dynamically and formed from heterogeneous peers that are autonomous and have intermittent presence in the network [74]. In these systems, peers are unknown to each other, and need to make trust decisions regarding peers they will interact with and manage the risk involved in these interactions. Furthermore, P2P systems need to ensure robustness against various attacks that have been made on these systems [67]. To overcome these challenges, peers can use their experience or acquire information from other peers to avoid untrustworthy peers and reduce risk [92]. Existing work [53, 69, 96, 105, 109]using trust models has shown that P2P systems can successfully mitigate this risk by finding reliable trustworthy peers.

3.5.1 Trust Definition

The notion of trust in computer science has been borrowed from human society, where we experience and rely on trust in our daily life. Trust is a multifaceted and context-dependent concept, relating to belief in honesty, competence, and reliability of the trusted person or service.

In the Oxford Reference dictionary [22] trust is defined as "*firm belief in the reliability or truth or strength of an entity*".

Grandison [63] defines trust as "*the firm belief in the competence of an entity to act dependably, securely, and reliably within a specified context*".

These definitions shows that trust is composed of different attributes such as reliability, dependability, honesty, and competence all of which may have to be considered depending on the environment in which trust is being specified.

3.5.2 Reputation Definition

In general, reputation is the opinion of the public towards a person, an organization, or a resource [93]. Reputation is a means of building trust, as one can trust another based on his/her reputation. Therefore, reputation is a measure of trustworthiness, in the sense of reliability. In the Oxford Reference Dictionary reputation has been defined as "*The beliefs or opinions that are generally held about someone or something*". In the research literature, a number of definitions for reputation are given. Abdul Rahman et al. [33] define reputation as "*an expectation about an agents behavior based on information about its past behavior*". Mui et. al. [84] define it as the perception that an agent creates through past actions about its intentions and norms.

3.5.3 Trust and Reputation Technologies in P2P Systems

Various trust systems have been proposed for P2P systems and can be classified into three categories as shown in Figure 3.1

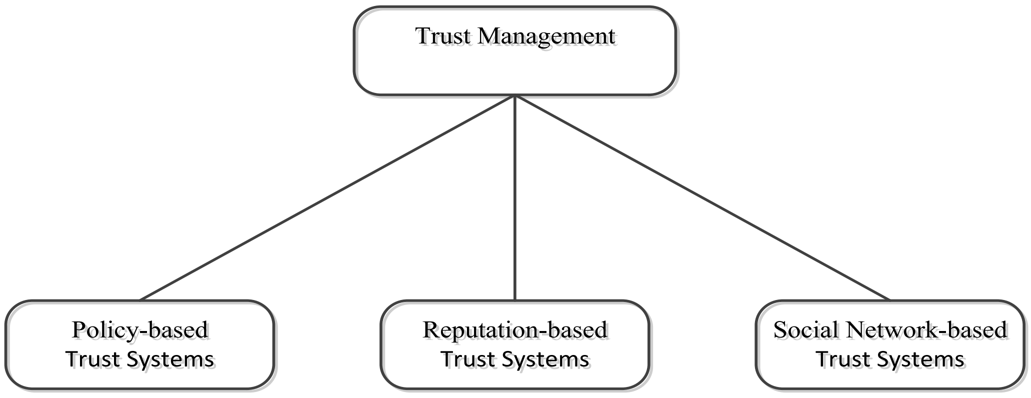

Trust Management
Policy-based Trust Systems
Reputation-based Trust Systems
Social Network-based Trust Systems

Figure 3.1: Trust Management Taxonomy

3.5.3.1 Policy-based trust systems

In credential and policy-based trust management systems credential verification can be used to establish a trust relationship among peers, see for example [41, 68, 76, 107, 110]. The main aim of these systems is to enable access control and trust management is limited to verifying credentials and restricting access to resources according to application-defined policies. A requesting peer obtains access to restricted resources only if its credentials are verified. These policy-based access control mechanisms do not need the requesting peer to establish trust in the resource owner and so do not provide a complete generic trust management solution for all decentralized applications [80].

PolicyMaker

PolicyMaker [41] is a trust management system that is used to provide security features including privacy and authenticity for different kinds of network applications. The PolicyMaker approach to trust management is based on the following principles:

- **Unified mechanism** - the ability to handle trust in a comprehensive, consistent, and transparent manner by expressing policies, credential, and trust relationships as programs or a part of a program using a common language.
- **Flexibility** - the ability of the system to support both complex and simple trust relationships.
- **Local Control** the ability of peers in the network to make local decisions about the authenticity of credential presented by the requesting peers.
- **Separation of mechanism from policy** - this keeps the authentication mechanism application-independent and allows different applications with varying policy requirements to share a single certificate verification infrastructure.

Using PolicyMaker, a peer grants another peer access to its services when it has verified that the credentials of the requesting peer satisfy those specified by PolicyMaker. In PolicyMaker, peers do not query and store trust information, which means bandwidth and storage cost are limited and do not affect its scalability, but its use of a particular language to describe credentials and policies increases the complexity of the system.

3.5.3.2 Reputation Based Trust Systems

Reputation-based trust management systems provide a mechanism by which a requesting peer may evaluate the level of trust it should place in the reliability of a resource and the providing peer. In these systems, peers use information concerning previous interactions between them to establish a reputation measure that will support their trust decisions.

The EigenTrust [70] approach is based on the concept of transitive trust; a peer has a high opinion of those peers who previously had satisfactory interactions with them. Therefore, this peer is likely to trust their opinion about other peers. In EigenTrust each peer stores its trust values for all other peers locally. Thus any one peer can obtain a global trust value for any other peer by asking all peers in the network for their evaluation. The received trust values are aggregated to calculate the global trust value. By using these global trust values to choose peers, the P2P network can efficiently identify malicious peers and isolate them from the network.

Gupta et. al. in [64] proposed a reputation system for decentralized unstructured P2P networks that uses objective criteria to track each peers contribution in the network; the peers capability and behaviour are used to calculate its reputation score in the network. The capability of a peer depends on its processing capacity, memory storage capacity while the behaviour of a peer is determined by its contribution in the content search phase and download phase. In this system, the reputation is computed using either of the two schemes: debit-credit reputation computation (DCRC) and credit-only reputation computation (CORC). This system uses a public key based mechanism that periodically updates the peers reputation.

The PowerTrust [112] model works in a similar way to EigenTrust, focusing on creating overlay hashing functions to assign score managers (i.e. peers who calculate trust values for other peers, thereby preventing peers from maliciously changing their own trust values) for peers in the system and for combining trust values to create a global reputation score.

3.5.3.3 Social network based trust systems

Social network-based trust management systems use social relationships among peers for computing trust and reputation values. These systems evaluate peers reputations based on the social network that represents their relationships in the network.

The trust model used by Marsh [80] is based on the social properties of trust, and attempts to integrate aspects of trust taken from sociology and psychology. In [108], the authors explore the effect of the social relationships of peers belonging to online communities on reputation in decentralized scenarios. REGRET [95] and NodeRanking [90] are other examples of trust management systems that use the social network concept.

In the REGRET system, Sabater et al. [94] adopt the concept that the reputation of a peer is an aggregation of different pieces of information. They proposed a model based on three dimensions of reputation: individual, social, and ontological. The individual dimension relates to direct trust resulting from the outcomes of direct interactions between peers. The social dimension incorporates information on the experiences of other peers with the target peer. Finally, the ontological dimension considers how various types of reputation can be combined to obtain a new type of reputation. For example, the reputation of being a good flying company could be summarized in the reputation of having safe planes, comfortable planes, punctuality, never losing luggage and of serving good food. These dimensions are combined to obtain a single value of reputation.

3.6 Summary

This chapter presented P2P paradigms and terminology used in this thesis together with an overview of the techniques used to determine trust and reputation. It defined and classified grid types, P2P systems and their relative advantages. It described structured and unstructured P2P systems as overlays on an underlying physical network and discussed P2P systems in terms of reliable provision of data storage and sharing.

Trust and reputation in P2P systems were introduced as terms borrowed from human society and then applied to P2P systems: (i) Policy based systems which are limited to verifying credentials and restricting access to resources according to application-defined policies; (ii) Reputation based trust management systems where peers use information from previous interactions to establish a reputation measure to support their trust decisions; and (iii) Social network-based trust management systems which evaluate a peers reputation based on feedback from the social network of which that peer is part.

Chapter 4

Evaluation of the Current State of the Art: BOINC and Attic

The aim of this chapter is to evaluate the Attic file system from a requirements analysis perspective to understand how Attic works, how the parameters affect the tuning of the system and to understand the factors affecting the performance of Attic. In this chapter, the Attic file system is compared with the performance of BOINC to identify a baseline for quantifying any improvements the VASCODE framework (discussed in the next chapter) makes over this infrastructure.

The chapter begins with a description of the Attic file system in terms of its components, its messages and how they are used to download data. Following the introduction of Attic, a performance study to compare Attic with the performance of BOINC is presented. The aim of this study is to evaluate the advantages that Attic has over BOINC, to acquire experience in the use of Attic, which will lead to an understanding of how best to optimize Attic to provide a requirements analysis for the VASCODE framework. In addition, it indicates the potential value in using Attic for volunteer computing projects since the use of decentralized data centre layers decreases the download time of users and provides significant gains for BOINC projects as a whole by reducing download time and therefore increasing the throughput of results.

4.1 Attic File System

Attic is a P2P data sharing software system currently being developed as part of the European Union's seventh framework project EU FP7 EDGI [12]. Attic, previously dubbed ADICS [72], was initially developed with support from EPSRC grant EP/C006291/1 and had further support in the EU FP7 EDGeS [55] project. Attic is being developed to take advantage of the network and the storage resources available on the network in a decentralized manner, similar to BitTorrent [49].

The primary differences between Attic and BitTorrent are the concept of data centres, and Attic's use of HTTP. Data centres are interim storage facilities that provide a buffer between the data serving application and client applications. This buffer is particularly important for volunteer computing environments because it ensures that the data sources can be trusted by clients. Trust plays a crucial role in Volunteer Computing environments. If it is broken, then the project will fail to attract volunteers. Therefore, in controlled environments, data centres are typically issued with certificates signed by a trusted certificate authority (which may be the project itself) allowing client applications to verify the identity of the data centre when downloading. Alternatively, clients can be configured with known data centre hosts. However, the Attic architecture allows any client to also serve data (i.e. to become a data centre) and thus the use of trusted data centres is primarily a deployment choice. This flexibility allows projects to conform to the Attic system with regards to the particulars of their security domain. As far as the EDGI is concerned, it makes sense to serve data from a set of known and trusted sources, however, another project could also choose to allow any host to cache and serve data, thereby including participant client machines in data distribution, similar to more traditional P2P systems, such as BitTorrent [49], Freenet [14] and Oceanstore[75].

Attic uses HTTP for its data transfer layer, using the byte-range capabilities of HTTP requests to allow concurrent downloads from multiple data centre hosts. Concurrent downloading helps to maximize client download bandwidth, as well as distribute network load. Additionally, the use of HTTP transactions allows easy integration with existing software and firewall configurations and client applications that choose not to serve data that require only an out-going HTTP connection. The Attic file system consists of four main elements and these are listed below:

- A data serving application that replicates data on the network.
- data centres that cache data, providing the distributed data source overlay.
- A Look-up service that keeps track of which data centres have individual data items.
- Client applications that download data from data centres on the network.

4.1.1 Attic File System Components

The Attic file system consists of different components (Figure 4.1) that are used to play different roles in the system. These components are:

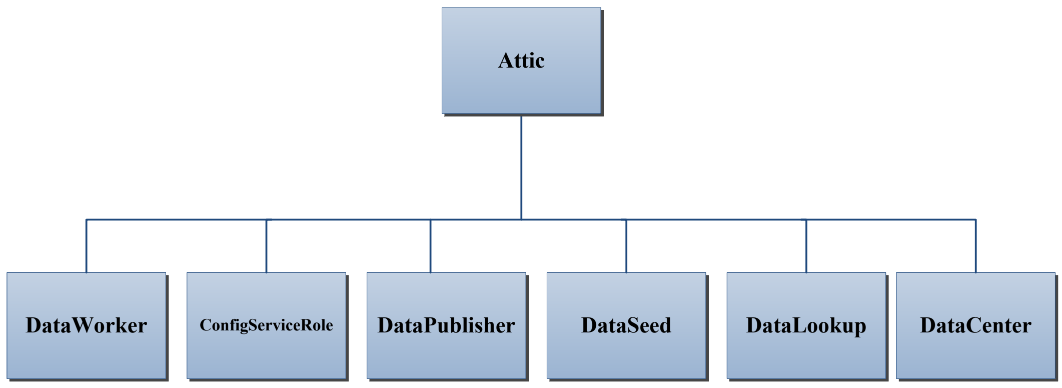


Figure 4.1: Attic File System Components

.

DataWorker is used to pull data when it is given a remote endpoint. The given endpoint should point to a data pointer document.

ConfigServiceRole is used to provide online configuration to an Attic instance.

DataPublisher can index local files, and uses bootstrap endpoint to publish data adverts.

DataSeed extends the DataPublisher to allow remote clients to push data to the seed to publish it.

DataLook-up acts as a look-up service for other nodes. It is used to accept requests to publish data and requests to cache data.

DataCentre acts as (1) a client in that it requests data pointer from a look-up service and uses the endpoints described in data pointers to download data. (2) data centres cache data to provide it to other nodes.

4.1.2 Message Types

The Attic file system uses a number of different messages to describe, publish, and query data. These messages and their uses are described in Table 4.1 below:

Message Type	Definition
DataDescription	Contains name, description, and project associated with the data. It also combines a globally unique ID.
FileHash	Contains the length of the data and an MD5 hash.
FileSegmentHash	Describes a portion of data including its start offset, end offset, and the MD5 hash of the portion.
DataAdvert	Contains a DataDescription.
DataQuery	Is used to query for data.
PointerCollection	Is a list of DataPointers.
DataPointer	Contains a DataDescription and a list of Endpoints.
Endpoint	URI of Data Cache.

Table 4.1: Attic Message Types

4.1.3 Security

Attic uses TLS and mutual authentication with X509 to implement security. During runtime Attic requires a Java keystore containing local keys and certificates to be available. To allow for more fine grained authorization of actions based on the identity in the certificate, Attic uses a mapping between an application level defined action, and a distinguished name (such as common name, organization and country) as defined in a certificate.

4.1.4 Persistence

Attic does not support persistence. It uses in-memory storage for caching identities and various messages exchanged. This is largely suitable for client-side nodes such as DataWorker, or service nodes that are transient. There are various types of storage defined. Table 4.2 lists these types:

No	Store
1	Adverts Store
2	Queries Store.
3	Data Pointers Store
4	Description Store
5	Identities Store

Table 4.2: Attic In-Memory Store

4.1.5 Attic Downloading Mechanism

Downloading from multiple servers using Attic requires the possession of a *DataPointer*. Typically, these can be retrieved from a Look-up server. Data centres may ask for a collection of available pointers because they are interested in caching, as opposed to processing the actual data. *DataWorkers*, on the other hand, will query for a pointer based on the UUID of the *DataDescription* referenced by the pointer, because they require a particular data object to process. A *DataPointer* represents a description of data and a list of endpoints that potentially have some or all of the referenced data. The Endpoint element represents the URL at which the data is available. These endpoints are used to download data. Figure 4.2 shows an example XML serialization of a portion of a *DataPointer*.

```
<DataPointer xmlns="http://atticfs.org">
    <DataDescription>
            ...
        <FileHash>
            <hash>409876cba678d9fe009</hash>
            <size>234834134214</size>
            <Segment>
                <hash>2f4736ba8c0f99f0</hash>
                <start>0</start>
                <end>9234156</end>
            </Segment>
                ...
        </FileHash>
    </DataDescription>
    <Endpoint>
        <url>http://foo.com/dc</url>
        <meta>http://foo.com/dc/meta</meta>
    </Endpoint>
    <Endpoint>
        <url>http://bar.com/dc</url>
    </Endpoint>
</DataPointer>
```

Figure 4.2: Snapshot of Data Pointer

The workflow that describes the download process of a file downloading using the Attic File System is explained in the UML diagram (Figure 4.3) below:

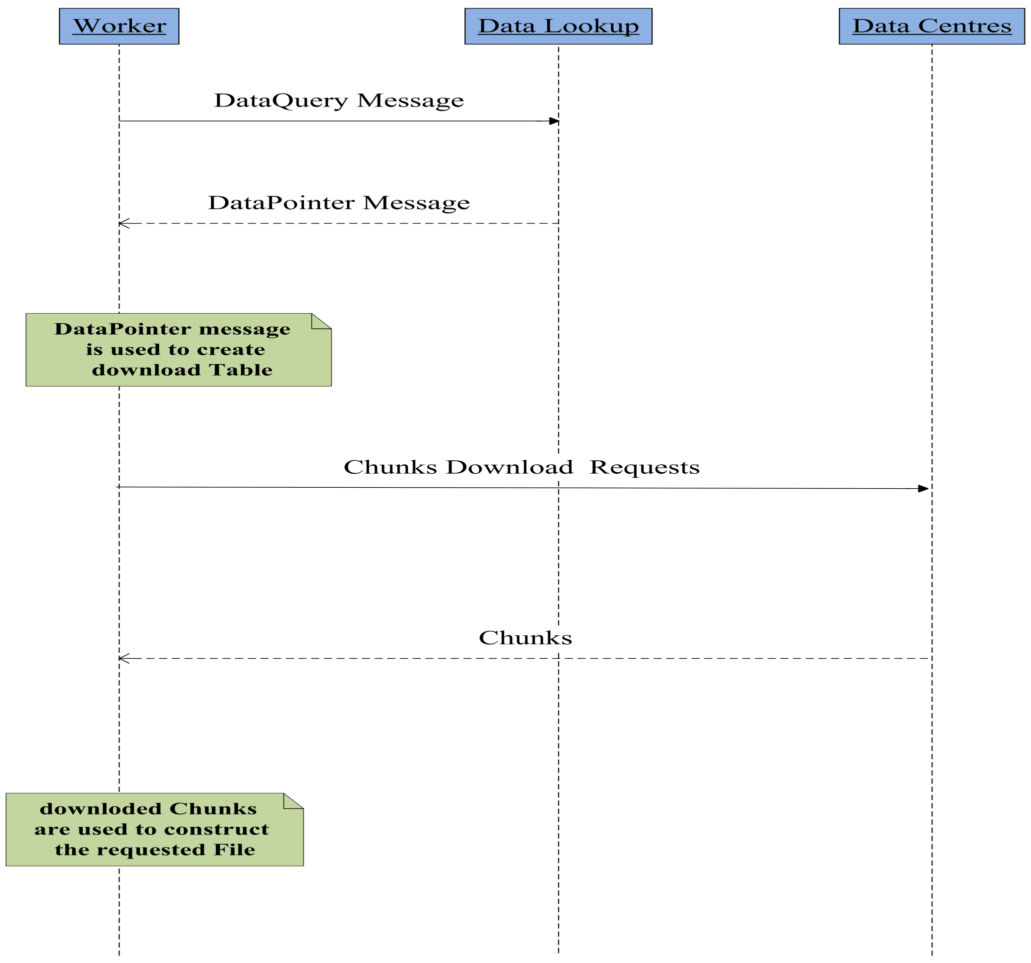

Worker
Data Lookup
Data Centres
DataQuery Message
DataPointer Message
DataPointer message is used to create download Table
Chunks Download Requests
Chunks
downloded Chunks are used to construct the requested File

Figure 4.3: Workflow of File Download Using Attic.

1. The DataWorker sends a DataQuery message to the look-up server.

2. The look-up server replies with a DataPointer message.

3. The DataWorker uses the DataPointer message to configure the downloaded table.
4. DataWorker start request file chunks concurrently from the Endpoints listed in the DataPointer.
5. Data centres send the requested chunks to the DataWorker.
6. Finally, when the DataWorker has received all the chunks, they are used to construct the whole file.

4.2 Experiments Results and Discussion

This section presents a study comparing Attic with the Current State of the Art, to understand the extent to which Attic might improve data distribution in a Volunteer Computing Environment. Projects that employ the use of the BOINC middleware are of particular interest in this thesis. To this end, a number of experiments have been conducted to show how Attic performs using different parameters, namely chunk size and the number of data centres.

4.2.1 Testbed Environment

The experiments were run using networked computers in the Computing Science Laboratory at the School of Computer Science and Informatics, with access to 19 Linux machines. 18 of these machines were used to run various combinations of clients and data centres. These machines each contain a 2.8 GHz P4 processor and 2 GBs RAM. The remaining machine is equipped with a 2.0 GHz Pentium Dual Core processor and 3 GB RAM and was used to run the data look-up server for the Attic experiments and also the BOINC data server in the BOINC experiments for comparison. All machines are connected through a LAN network and the speed of the ports was configured to each of these machines using a Web interface.

4.2.2 Experiment Preparation

The Web interface are used to set the network connection speed of a subset of the machines to 10 Mbps and the others to 100 Mbps. This enabled the simulation of a collection of home users when connected via a broadband internet connection; that is, their upload speed was proportioned as a fraction (such as a tenth) of their download speed. This is the norm, at least in the UK and USA, where an ISP will often restrict a user's upload bandwidth to around one tenth of their download speed. The success of BitTorrent-style protocols capitalises on this download/upload mismatch to maximise download speed by establishing multiple simultaneous download streams from a number of different servers. Since the servers typically have a fraction of bandwidth due to their ISP constraints, BitTorrent allows users to maximise their downloading bandwidth by downloading from multiple servers concurrently. By simulating this mismatch, it is possible to demonstrate the potential speedup by limiting servers in the network so that gains within the bounds of the TCP theoretical limit can be achieved. Otherwise, a one to one connection would always be the best strategy as it is possible to achieve the closest to the theoretical limit of TCP with a one to one connection.

For Internet deployment however, it is never the case that one server can match the bandwidth capability of the Internet because of its inherently distributed nature, so this assumption that one server will not match the theoretical bandwidth of the network as a whole is realistic. There will therefore almost certainly be a bottleneck as users increase because one server will not be able to serve the data as the data clients in the network increase. Therefore, it is believed that deploying multiple distributed servers that have a fraction of the overall network bandwidth is a reasonable approximation to simulating the same effect on a typical Internet deployment. Attic employs the same style of file swarming as BitTorrent to achieve similar results.

4.2.3 The Impact of Chunk Size

As described in Section 4.1 (and in BitTorrent), there are two main factors in the downloading of a distributed file from multiple sources: the availability of servers and the chunk size (called 'piece' in BitTorrent). The chunk size is a critical parameter in achieving optimal utilization of bandwidth across the network and knowledge of the various states of the content and network parameters is required for an optimal setting. For example, the typical size of a file that is downloaded is one key parameter and the number of servers and their bandwidth capability are the others. On the one hand, one would not want to set the chunk size too low because the TCP connection overhead would factor too high in the performance. On the other hand, one would not want to set the chunk size too high because this would limit the number of servers one could utilize in the network.

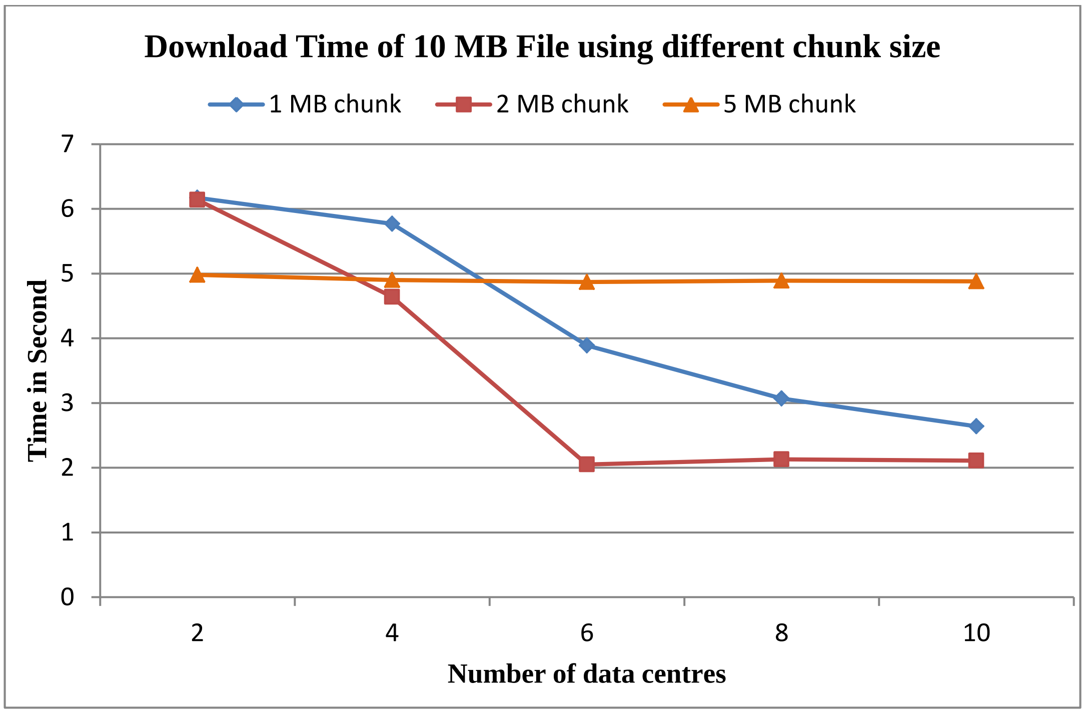


Figure 4.4: The Impact of The Chunk Size for a 10 MB File Using The ATTIC File System.

In this experiment, both these effects are shown to assess how the chunk size impacts on the download time of a 10 MB data file. The 10 MB file is published to Attic for three different chunk sizes (1 MB, 2 MB and 5 MB) and by varying the number of data centres from 2 to 10 (in increments of 2), it is possible to investigate the impact of chunk size on the efficiency of the system. The network speed for all serving nodes in this experiment was 10 Mbps and the client was set to 100 Mbps.

Obviously, different chunk sizes mean a different number of chunks are required to download the 10 MB file. For example, Figure 4.4 shows that for a 5 MB chunk size, it is not possible to do better than to use two servers because the file is twice the size of the chunk size. This was orchestrated to provide the baseline for this experiment.

The use of smaller chunk sizes is more interesting and the results were along the lines expected. For the 2 MB chunk size, 6 servers were the optimal number. The experiment was quantized into increments of two servers (i.e. 2,4,6,8 and 10) with 6 the expected optimal number of servers for the 2 MB size chunk. The finest grained results can be seen for the 1 MB chunk size, where a gradual decrease in download time can be seen as data centres are added. With 10 data centres, a download time of a little over two seconds was achieved. Comparing the results for 2 MB and 1 MB chunks clearly shows that as data centres are added, there is no linear scaling, but the scaling is still reasonably efficient with only a modest difference between five data centres serving 2 MB files and ten serving 1 MB files.

This experiment demonstrates the necessity of choosing informed values for the chunk size. If one knows something about the size of the data being downloaded (which one does in BOINC since this is already defined by the experiment), the average upload speed of the servers, the download rate of the clients and the number of available servers, then much improvement can be gained by choosing an appropriate value for the chunk size. Clearly, the goal here is to choose chunk size to make the ratio between the servers and their upload speed create optimal use of the client's bandwidth which is confirmed by the results of this experiment.

4.2.4 The Impact of Different Numbers of Clients and data centres

In this experiment, information was gathered about how the number of clients impacts on the average file download time for a user. Here, a 10 MB file was published to the Attic using a 1 MB chunk size. The network speed in this experiment was 10 Mbps for the servers and 100 Mbps for the clients. The experiment was run three times setting the number of clients (1, 3 and 9). Figure 4.5 shows that as the number of data centres increases, the average download time for a client decreases.

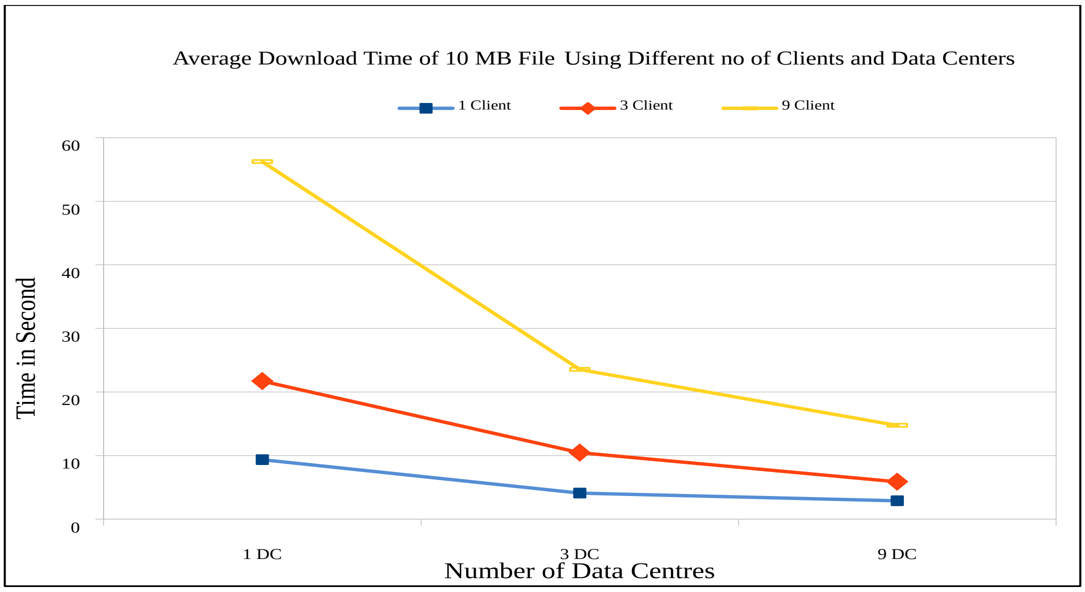


Figure 4.5: The Iimpact of Different Numbers of Clients and Data Centers on The ATTIC File System.

In Figure 4.5, it is clear that the download speed improves with the increasing number of data centres. However, for a 10MB file and a network speed of 1.25 MBytes per second (10 Mbps) it should take a minimum of eight seconds to download the file entirely using one data centre. For three users the traffic increases by a factor of three so the minimum download time should be 24 seconds and finally for nine users there would be a minimum of 54 seconds for all clients to download the file. One client makes good use of the available bandwidth and the download time with nine servers is a little less than 3 seconds, giving a gain of a factor of more than three over the minimum time for a single server. For three users, the download time is reduced from 24 seconds to 5 seconds when the number of data centres increased from one to nine (a gain factor of nearly five). Finally, for nine clients, Attic reduces the time from a theoretical 54 seconds for one data centre to 15 seconds for nine data centres, giving a gain factor of 3.6. The arithmetic average of these three gain factors for nine data centres is 3.9. For three data centres, the gain factors are 2 for one client (8 seconds reduced to 4 seconds), 2.4 for three clients (24 seconds reduced to 10 seconds) and 2.25 for nine clients (54 seconds reduced to 24 seconds). The arithmetic average of these is a gain factor of 2.22.

Clearly, these results indicate the importance of the combined choice of chunk size and number of data centres. For the given size of the file and network parameters three servers are clearly optimal in terms of the efficiency of the Attic files system. However, further gains of up to two are possible if one is prepared to make a sacrifice on the number of data centres utilized. In the Volunteer Computing paradigm where people volunteer such resources, it might well be argued that a high increase in the use of data centres justifies a modest gain in the download times. Again, these are parameters to be tuned for specific deployments.

4.2.5 Comparative Evaluation of BOINC with Attic

In this experiment, a comparative evaluation of BOINC with Attic was made. A BOINC data server was used, running on a 2.0 GHz Pentium Dual Core processor Linux machine with 3 GB RAM. The number of data clients (1, 3 and 9) used to download a 10 MB file were varied as independent variables. The network speed was set to 10 Mbps for the server and 100 Mbps for the clients.

First, the performance of the BOINC server as a baseline was analysed. The results in Figure 4.6 show, as expected, that the average time needed by all clients to download the 10 MB file increased as the number of clients increased.

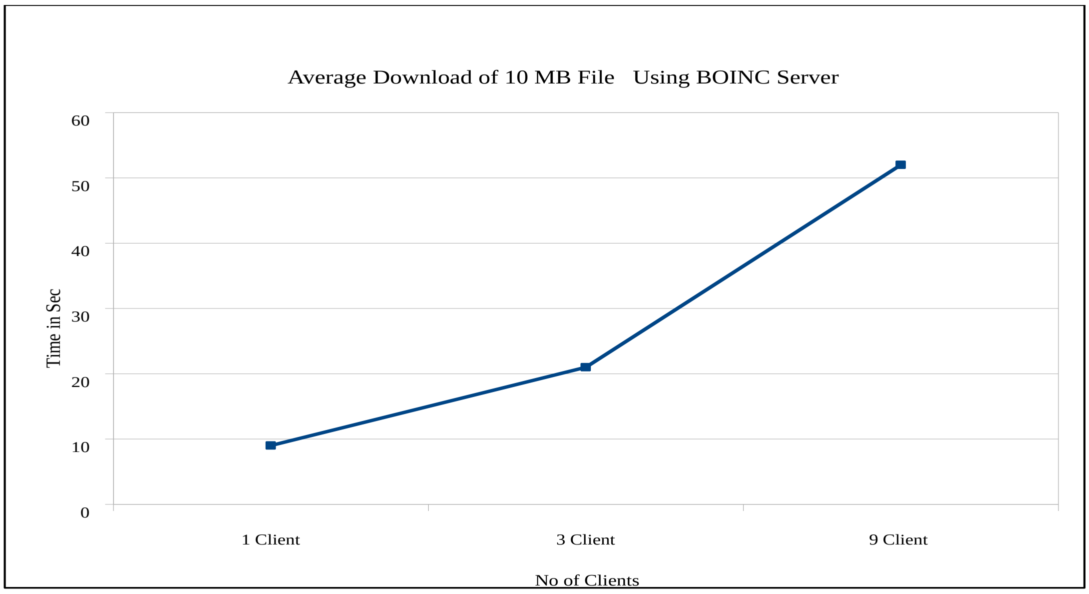


Figure 4.6: Performance of BOINC for Different numbers of clients.

As discussed in the previous experiment, a single BOINC client needed at least 8 seconds to download the 10 MB file, and with more than one client the data simply queues the requests until it has time to serve them.

In Figure 4.7 when the results are compared with the results of the previous experiment (section 4.2.4), it was noted that Attic will provide almost the same results as the BOINC data server when one data centre is used. In fact, BOINC is marginally more efficient which is not surprising because it does not have the message overhead that Attic has, in terms of querying and prioritising

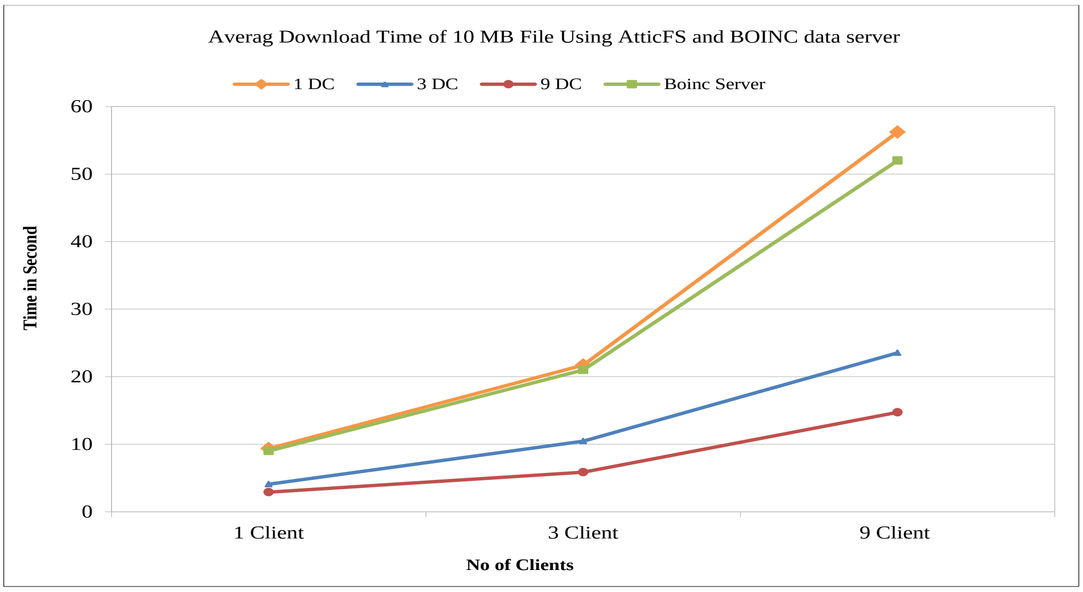


Figure 4.7: Comparing BOINC with the ATTIC filesystem.

endpoints before downloading commences. However, as Attic adds data centres the download time decreases.

4.3 Summary

This chapter described the Attic file system as a P2P data sharing software architecture whose primary differences with, for example, BitTorrent are the concept of data centres and use of HTTP to allow concurrent downloads from multiple data centre hosts. The components and message types of the Attic file system were listed and described. The Attic downloading procedure was explained with the parts played by File System Components and Message types. The procedure for integrating Attic into BOINC projects was also introduced.

An experimental study was carried out to investigate the potential value in using the Attic file system for Volunteer Computing projects. Attic is capable of dynamically adding data centres to the list as and when people volunteer their resources. The results clearly demonstrate that the decentralized data centres approach can significantly decrease download time for users and provides significant gains for the specific BOINC project as a whole by reducing download time and therefore increasing the throughput of results. Even though the scale of the experiments shown here might not reflect a real world trial of the Attic software, as there are tens of thousands of BOINC clients that connect every day to download new input data in production systems, the results are relevant in that they test the Attic architecture and load distribution that takes place within the applied scale.

Chapter 5

System Architecture

In this chapter, the VASCODE framework is presented, which is proposed to provide the necessary infrastructure to improve data distribution in Volunteer Computing projects that employ the use of BOINC middleware. In this chapter, a set of requirements is identified to extend the data layer in BOINC, then define an architecture from these requirements and introduce a trust model that provides a completely dynamic mechanism for allowing participants to act as data centres in BOINC in order to efficiently share data across the distributed resources.

The VASCODE framework (*Volunteered Automated Servers for Data Collection and Optimization in Distributed Environments*) presented in this thesis aims to provide a means to extend the data layer in Volunteer Computing using P2P systems, providing the basis to form a trusted decentralised data centres layer in Volunteer Computing, where, potentially, hundreds of thousands of participants download data to process as part of their jobs. The developed framework integrates Attic into BOINC middleware and applies a plugin trust technique to model, collect and utilise trust information in peers in Attic.

5.1 Design Goals

To demonstrate the benefits of integrating P2P systems in Volunteer Computing projects and, at the same time, address the trust issue when using this. There were three main goals in designing the system:

- To provide the scientific community with an easy to use and flexible tool for building data extensive projects.
- To encourage participants in Volunteer Computing to work as data distributors by allowing them to control the bandwidth they offer.
- To establish trust in participants who act as data distributors by using a trust model.

VASCODE is a Java and C ++ framework that are plugin extensions to Attic and BOINC to provide the necessary infrastructure to achieve these goals.

5.2 System Requirements

A set of requirements for VASCODE was identified that makes use of resources provided by the participants within a Volunteer Computing project. The core aim is to provide a tool based on this framework that is usable by BOINC project participants, allowing them to contribute as a data centre (data distributor) in the decentralized data centre layer, a data worker, or both. Various subsections below identify the components of the framework.

5.2.1 Data Caching

In BOINC projects, clients contact the scheduling server to get jobs to execute on their local resources. They then request input data files from a data server to process the downloaded job and finally upload the results. The input data files on a BOINC client can be obtained by using dynamic caching of the data file on the distributed data centres across the network. This increases data availability and improves fault tolerance because data can now be downloaded from different places rather than just the data server. Furthermore, it improves data download time since clients can parallelize the download of a file by using these different sources. Clients can open multiple connections per file and download a different portion of the file on each connection simultaneously.

Adding caching functionality to the BOINC client enables a BOINC client, previously only capable of processing jobs, to also be able to cache data and provide it to other clients.

5.2.2 Trust

The participants of BOINC projects are ordinary internet users, who have different behaviours and connection capability. Therefore, to utilize their resources effectively for data distribution, optimization or trust mechanisms are needed in order to dynamically re-adjust their behaviour according to the current operating environment. In this research, a trust mechanism is used that makes use of particular properties of a data centre, such as its bandwidth, connection speed and availability, provided that the data associated with these preferences is recorded. The trust mechanism is subsequently used to select one or more data centres from which to download data, based on preferences identified by a client.

5.2.3 Data Management

The formation and use of a decentralized data centre layer requires the consideration of two key issues outlined next.

5.2.3.1 Data Source

For a BOINC client to become a data centre, it has to cache data that is initially provided by the main BOINC data server. When the BOINC client downloads data to process its job, it will cache this data to be available for other clients who process the same job, thereby propagating the dataset on demand. Here, the BOINC data server is made the primary source of data when data is not available on the data centre layer. When data becomes available from other clients this will extend the source of data and the BOINC client can use them to download data and cache it.

5.2.3.2 Data Downloading

As the data centres are ordinary internet users (who may be connected to the network using a variety of connection types, such as dial-up, DSL and wireless), they can frequently become unavailable to provide data to other BOINC clients. This transient connectivity therefore needs to be addressed in the download algorithm and available data centres need to be dynamically updated as the network evolves. It must also deal with the case that no data centres are available, in which case, the BOINC client should switch to the main source, such as the BOINC data server, to get data.

From the requirements, a general scheme is outlined for downloading data in figure 5.1, which shows that the trust framework provides the intelligence for each client to determine the best data centre at that point in time. Each client updates its empirically gathered parameters to feed back into the trust model for the next iteration. In this way, the system can learn to dynamically deal with the changing network conditions.

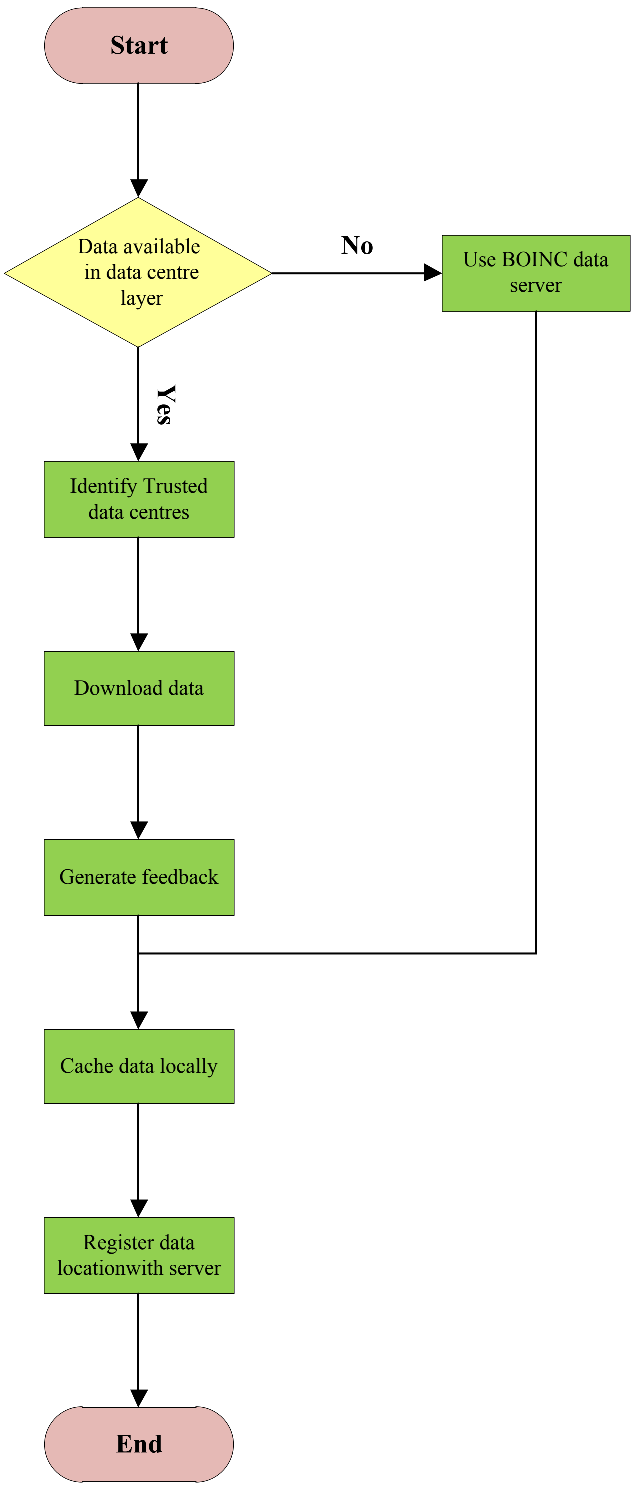

Start
Data available in data centre layer
No
Use BOINC data server
Yes
Identify Trusted data centres
Download data
Generate feedback
Cache data locally
Register data locationwith server
End

Figure 5.1: Download Flowchart

5.2.4 Bandwidth Throttling

Many internet users have limited bandwidth and may not be interested in participating in this scheme for distributing BOINC data for fear that this would slow their own use of the internet. Throttling is a technique that inhibits an applications use of a connection so that it only utilises a certain amount of the overall bandwidth. This framework therefore must offer a throttle capability to a client based on its bandwidth use in order to preserve the clients normal pattern of usage. For example, if it has 1 MB/s connection it can offer 256KB/s to other clients for downloading data, thereby enabling a user to better plan how their capacity will be shared with other users. This enables volunteered resources to continue to participate in the framework, whilst also enabling a resource owner to continue their own work. It is believed that such mechanisms for bandwidth sharing are likely to increase contributions of resources to a project.

5.3 System Architecture

The distributed roles and data distribution requirements outlined above lead into a general four layer architecture, which can be seen in Figure 5.2. The bottom layer represents the participants of scientific projects who provide their resources to Volunteer Computing projects. The next layer provides the P2P network capability; here, Attic is used. This layer provides the core capability for volunteers to share data with other participants.

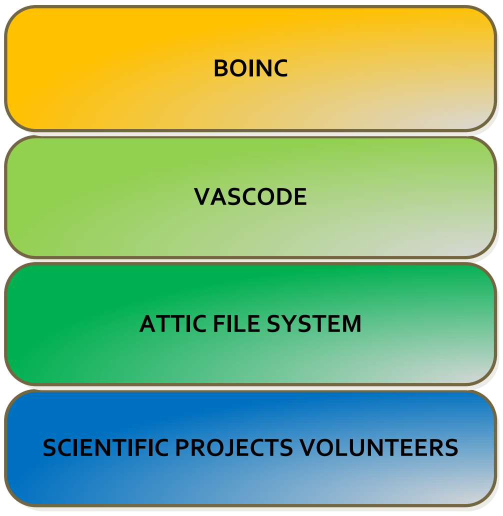


Figure 5.2: Framework Layer

The third layer provides the Volunteered Automated Servers for Data Collection and Optimization in Distributed Environments (VASCODE) framework for choosing the locations of data download at each time step. In layer three therefore, a data collection layer is necessary to provides peers with the necessary tools to select the data server from which to download data, which peers to trust, and the throttling capabilities to manage their bandwidth. This layer therefore has a further inner architecture and is broken down and described in further detail in section 5.3.1.

Finally the last layer represents BOINC, which is the programming and Web interface that a project interacts with in order to the use the system. Since both Attic and VASCODE deal with HTTP endpoints, a simple Attic URL scheme plug-in can be used to switch out the general BOINC URL endpoint with a dynamic Attic one to provide multiple possible endpoints for each dataset. Therefore, a project does not need to be aware of the use of VASCODE and Attic in order for this approach to be enabled as the general data distribution mechanism. This HTTP protocol therefore provides a convenient abstraction to allow Attic and VASCODE to be implemented almost completely on the server side, thereby not needing existing clients to install complicated toolkits in order to make use of this new feature. The integration of Attic and VASCODE into existing projects is possible through a plug in that proxies the HTTP connections and provides a mechanism to resolve multiple endpoints from a single Attic URL, which provides access to the concurrent downloading capabilities in Attic.

5.3.1 VASCODE Layer

VASCODE is a layer built upon the capability provided by Attic to add the necessary functionality to its peers (such as worker or data centre) when they download data or distribute it. Figure 5.3 provides an overview of this functionality. Essentially, VASCODE provides the trust-based mechanisms within Attic, enabling better integration of the various components that make up the Attic system into BOINC. VASCODE enables user defined preferences to be taken into account when selecting data centres from which to download, based on previously recorded usage data about these data centres.

The three user defined preferences used in this research to demonstrate the concept consist of the following:

1. Availability whether the peer is available at that point in time or not
2. Data integrity the establishment of trust in that the data has not changed
3. Connection speed of each data centre the Internet bandwidth allowable for this particular data centre after throttling has been applied.

Each peer in the system provides feedback on each interaction they have had with a data centre. This data is collected (through the data lookup nodes) and used in the trust algorithm to feedback into the selection criteria a client uses, to determine the most appropriate sets of data centres to use. By using this mechanism, the system continuously (on the completion of each interaction with a data centre) updates the aggregate statistics for each data centre

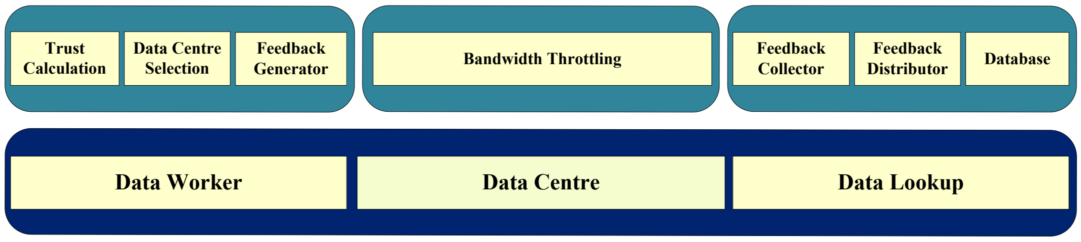


Figure 5.3: VASCODE on top of Attic

Figure 5.3 shows the three possible roles that a peer can perform in the BOINC-VASCODE integration: a conventional BOINC data worker, a data centre and a data lookup server. Furthermore, each of these roles has various properties. For example, when the BOINC client is a data worker, it must also be capable of accessing the data centre layer and needs to make use of VASCODE to calculate the trustworthiness of these data centres and select which data centres to get data from. It needs to then provide feedback to help other participants determine which is the best data centre at this point in time.

A BOINC client can use the VASCODE framework to interact with these functions. If a BOINC client wants to perform data centre capabilities, then for the general configuration of the data centre it needs to provide how much bandwidth it wants to offer for data distribution. Finally, when the BOINC client plays the role of a data lookup server, it is important to have the ability to collect feedback from those clients and distribute these to other clients and for use by the network as a whole. These capabilities are also offered through the VASCODE layer.

5.3.2 The VACSCODE Trust Framework

In this section, the design of the trust model is provided as well as its distributed requirements, which are integral to the architecture of the resulting VASCODE framework. In comparison with the general BOINC system (which uses a pre-defined data server), a client in this system is required to identify a data centre prior to commencing data download, by resolving an Attic URL. When Attic is used in BOINC, Attic URLs are provided as data points, which are abstract identifiers for a data set. When a client receives the URL, it passes the URL to Attic (using the local http proxy) to resolve into an actual http endpoint (or set of http endpoints), which store the locations of the data. Furthermore, the data can be chunked into pieces and distributed across multiple data centres, so it is possible to download the multiple pieces from multiple data centres concurrently. In VASCODE, the final selection in which data centres are used for download is based on their reputation in the system, with reference to one or more metrics.

The concept of trust is key to enable peers in P2P systems to make successful decision-making processes. In this section, it is aimed to develop a computational model of trust that a data centre client can use to make a successful decision when selecting data centres for downloading data. This model aims to determine the trust level in data centres as a data source, considering the behaviour of the data centres in previous download requests.

This model of trust and reputation can be used to support informed decision making to assure improved download time of data in Volunteer Computing projects that use a decentralized data layer.

This use of trust in decision-making can assure reliability on a decentralized data layer by enabling users to reason whether or not to select a data centre. For example, if a data centre layer is formed from a group of data centres, then it is important for a client to select the most appropriate data centres to download data from; this selection is not only based on data centres capabilities, but also on their trustworthiness.

Trust and reputation systems can be used to encourage and promote good behaviour in P2P file sharing systems. Several trust systems have been deployed for practical applications. In this thesis, a new trust model was developed for use in the VASCODE framework to allow clients to select data centres based on their trust value. The aim behind this trust model is to optimize the Attic protocol by incorporating a trust mechanism to allow the clients to make trust-based decisions for selecting a data centre, which improves data distribution in BOINC-like Projects using file sharing protocols.

The trust framework makes use of Attic to support concurrent data downloads from multiple data centres. It utilizes the communication between the clients and the data lookup server to send feedback and receive data on the associated trust metrics.

5.3.2.1 Modelling Trust

The focus of this research on trust in the decentralized data layer is to discover and exclude misbehaving data centres and to minimize the effect of unreliable data centres by selecting the proper data centres during data downloading. This requires the development of a trust model to address these issues.

The general notion of trust is excessively complex and appears to have many different meanings depending on how it is used in electronic service provisioning. There is also no consensus in the computer and information sciences literature on a common definition of trust, although its importance has been widely recognized and the literature available on trust is substantial. Generally, trust may be used as a metric to guide an agent in deciding how, when and who to interact with. An agent in this context refers to either a service user or a provider. Such a metric takes into account the subjective probability with which an agent views its interaction partners, taking into account local state and external recommendations made by other agents. To establish a trust model, agents must gather data about their counterparts. This has been achieved in three ways in the literature: (i) using prior interaction experience: in this context, trust is computed as a rating of the level of performance of the trustee using historical data. The trustee's performance is assessed over multiple interactions to check how good and consistent it is at doing what it says it does. Interactions that have taken place recently are treated preferentially to those that have taken place in the distant past. Witkowski et al. [104] propose a model whereby the trust in an agent is calculated based on its performance in past interactions. Similar to Witkowski et al., Sabater et al. [93](using the REGRET system) propose a similar model but do not just limit the overall performance to the agent's direct perception, but they also evaluate its behavior with other agents in the system; (ii) information gathered from other agents: trust in this approach is drawn indirectly from recommendations provided by others. As the recommendations could be unreliable, the agent must be able to reason about the recommendations gathered from other agents. The latter is achieved in different ways: (1) deploying rules to enable an agent to decide which other agents' recommendation they give greater preference, as introduced by Abdul-Rahman et al. [32]; (2) weighting the recommendation by the trust the agent has in the recommender, EigenTrust [70] and PageRank [87] are examples of this approach. In both of these approaches, the connectivity graph between recommenders is used to infer trust. Generally, an agent that has successfully delivered its advertised capability and recommends another agent will cause some of its trust to be transferred to its recommended agent. Both PageRank and EigenTrust are therefore based on the assumption that a general user, searching over a set of possible service providing peers, will eventually end up finding a more trustworthy peer if they follow the recommendation chain from any point in the network. The PowerTrust [112] model works in a similar way to EigenTrust, focusing on creating overlay hashing functions to assign score managers for peers in the system and for combining trust values to create a global reputation score. Both of these approaches have limited benefit when considering multiple criteria when calculating trust i.e., both EigenTrust and PowerTrust are focused on searching for objects using a single keyword, such as a file name; (iii) socio-cognitive trust: in this context, trust is drawn by characterizing the known motivations of the other agents. This involves forming coherent beliefs about different characteristics of these agents and reasoning about these beliefs in order to decide how much trust should be put in them. An example of this is work by Castelfranchi [44]. Our focus in this work is primarily on characteristics (i) and (ii) defined above.

Modelling trust in general is the process of representing the trust of a client in a service provider. Gambetta [61] defines trust as being a measure that represents the probability of an agent carrying out a particular action. Trust is therefore an indication of the reliability of an agent, which, in the context of this research, is the data centre.

For this reason, it is important to represent and model the behaviour of a data centre before developing a mechanism that allows users to determine the level of trust in a data centre. Since data centres are ordinary internet users, it is likely that they have different behaviours. In this research, trust modelling is the mathematical representation of client opinion in data centres in the context of data distribution. Three metrics that represent the data centre behaviour and affect the data distribution are defined:

1. The upload speed that a client obtained through a connection with a data centre.

2. The availability of a particular data centre.

3. The integrity of data supplied by the data centre.

These metrics are named in the model as ***DCspeed***, ***DCAvailability*** and ***DCHonesty***, respectively; see Figure 5.4 where each metric is independently calculated using feedback from multiple clients using specialist tools.

Speed primarily relates to performance issues such as access time, latency and effective bandwidth. *Availability* relates to uptime and resilience, covering

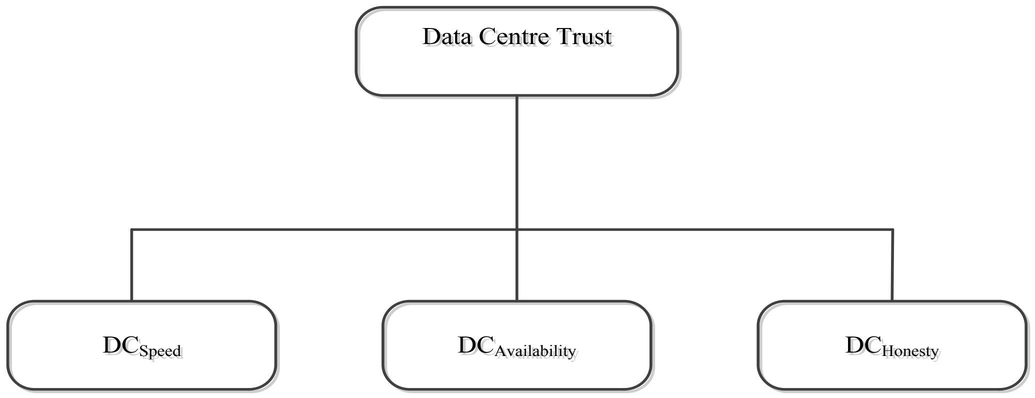


Figure 5.4: Data centre trust

aspects such as downtime, and failure rate. *Honesty* covers aspects such as data integrity and quality, storage reliability and any malicious modification to the data.

Historical data are used to establish a trust model for a given data centre, and use three metrics (*honesty, availability and speed*) to evaluate the level of trust that one can place in a data centre. In this model, reputation is considered to be an aggregated community view about a data provider, i.e., the greater the number of participants who trust a data centre, the greater the reputation the data centre holds.

The trust model also considers the behaviour of a data centre as a probability of a satisfied interaction or a probability of unsatisfied interaction.

This abstraction of data centre behaviour means that the outcome of interaction between client and data centre is a binary value (satisfied or unsatisfied). Binary feedback is used to show whether the client of a data centre was satisfied or unsatisfied based on three metrics used to represent the data centre behaviour; for example, if a client selects a data centre which is expected to be available during the download request. The download progress is deemed satisfied if the data centre responded to the download request as expected by the client. If the data centre did not respond to the download request then the download progress is unsatisfied; this will have a negative impact on the trustworthiness of the data centre.

To model the environment in which VASCODE is applied, the set of data centres is denoted as D=d_1,d_2,..,d_n and the clients of these data centres as C=c_1,c_2,..,c_n. Clients may interact with one or a number of data centres. The outcome of an interaction between d_j and c_i is represented by a binary variable $X_{c_i d_j}$, where :

$$X_{c_i d_j} = \begin{cases} 1 & \text{if Satisfied} \\ 0 & \text{Otherwise} \end{cases}$$

The posterior probability of binary events can be represented by the Beta distribution [62]. The Beta distribution uses two parameters (alpha and beta) to represent binary events; in this case, the binary event is whether the client is satisfied or unsatisfied.

5.3.2.2 Beta Distribution

The Beta model is used in several systems, including Jsang and Ismails Beta reputation system [50], the systems of Mui et al. [85] and of Buchegger [42], the Dirichlet reputation systems [66], TRAVOS [100], and the SECURE trust model [43]. The use of the Beta distribution is a reasonable one, since the history of interactions between clients and service providers can be summarized by the

Beta function with parameters alpha and beta to represent the successful and unsuccessful interactions. Since the beta distribution is conjugated prior to the family of Bernoulli trials it can be made mathematically precise in the language of Bayesian theory.

An important consequence of this representation is that it allows the estimation of the so-called predictive probability, namely the probability of success in the next interaction using the history of previous interactions.

This simple and popular model shows that predictive probability depends on the number of past successful interactions and the number of past unsuccessful interactions.

The Beta distribution (Figure 5.5) is used in a number of different projects to calculate the trust value through iterative calculations based on the outcome of previous trust values. Below, it is specified how trust can be calculated using the Beta distribution for each trust value, using client feedback provided by the client concerning their experience of each trust metric. Each trust measure is calculated from two values specifying whether the client was either satisfied (r) or not satisfied (s). The probability function of predicted outcomes in the future can be expressed as a function of previous observations.

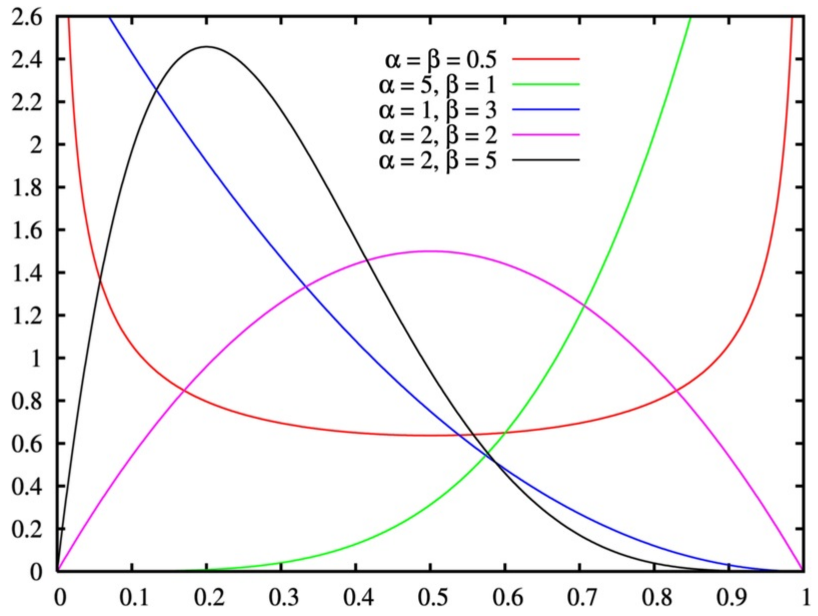


Figure 5.5: Beta Distribution

The Beta distribution can be used in the probability modelling of binary events. Let X be a random variable representing a binary event, X = 0; 1, and p the probability that the event occurs. Then the Beta-family of probability distributions, a continuous family of functions indexed by two parameters α and β, can be used to represent the probability density distribution of p, denoted by Beta(α, β), as shown in equation 5.1

$$\int (p|\alpha,\beta) = \frac{\tau(\alpha+\beta)}{\Gamma(\alpha)\Gamma()\beta)} p^{\alpha-1(1-p)^{3}-1} \tag{5.1}$$

where

$$0 \leqslant p \leqslant 1, \alpha > 0, \beta > 0 \tag{5.2}$$

If the number of outcomes where there are r satisfied and s unsatisfied with the event is observed, then using a Bayesian probabilistic argument, the probability density function of p can be expressed as a Beta distribution, where α = r + 1 and β = s + 1. This probabilistic mechanism is applied to model the reputation of a data centre using feedback on completion of download.

The reputation system counts the number r of successful interactions and the number s of unsatisfied interactions, and applies the Beta probability model. This provides for an easily updatable system, since it is easy to update both r and s in the model. Each new interaction results either in r or s being augmented by 1 and the probability expectation value of the Beta distribution is given by:

$$E(p) = \frac{\alpha}{\alpha + \beta} \tag{5.3}$$

$$\alpha = r + 1 \quad and \quad \beta = s + 1 \quad where \quad r, s \geqslant 0 \tag{5.4}$$

$$E(p) = \frac{r + 1}{r + s + 2} \tag{5.5}$$

5.3.2.3 Calculating Trust

When assessing a data centre, the experience of the other clients provides reliable evidence for predicting its behaviour. Reputation is therefore a useful means of gathering evidence. It involves asking for the opinion of other clients who have interacted with the data centre in the past.

The client c_1 must calculate a single trust value $T_{c_1 d_1}$ for a data centre d_1 by combining all the feedbacks provided by other clients. An elegant and efficient solution to this problem is to enumerate all the successful and unsuccessful interactions from the reports that it receives. The resulting values, denoted $R_{c_1 d_1}$ and $S_{c_1 d_1}$ as follows:

$$R_{c_1 d_1} = \sum_{j=0}^{n} r_{c_j d_1} \quad S_{c_1 d_1} = \sum_{j=0}^{n} s_{c_j d_1} \qquad where n = number\, of feedbacks \tag{5.6}$$

$R_{c_1 d_1}$ and $S_{c_1 d_1}$ are used to calculate shape parameters (see Equation5.7) for a Beta distribution:

$$\alpha = R_{c_1 d_1} + 1 \quad and \quad \beta = S_{c_1 d_1} + 1 \tag{5.7}$$

The trust value $T_{c_1 d_1}$ is calculated by using these parameter values in Equation 5.3.

$$T_{c_1 d_1} = \frac{\sum_{j=0}^{n} r_{c_j d_1} + 1}{\sum_{j=0}^{n} r_{c_j d_1} + \sum_{j=0}^{n} s_{c_j d_1} + 2} \tag{5.8}$$

5.3.2.4 Combining Trust Metrics

The aim of this trust model is to discover and exclude misbehaving data centres and to minimize the effect of unreliable data centres by selecting the proper data centres during data downloading. Depending on the three metrics defined in section 5.3.2.1, a trusted data centre is identified by aggregating the trust value of the data centre based on each metric as follows:

$$w_s.t_s + w_a.t_a + w_h.t_h \geq T_{threshold} \tag{5.9}$$

where $w_s + w_a + w_h = 1$

w_s, w_a, w_h are the weighted factors of each metric and $T_{threshold}$ the threshold for selecting a data centre.

The value of $T_{threshold}$ in the range of $\in [0,1]$. By using equation 5.8 and equation 5.9 a trusted data centre can be identified using clients feed backs as follow:

$$w_s.\frac{\sum_{j=0}^{n} rs_{c_j d_i} + 1}{\sum_{j=0}^{n} rs_{c_j d_i} + \sum_{j=0}^{n} ss_{c_j d_i} + 2} + w_a.\frac{\sum_{j=0}^{n} ra_{c_j d_i} + 1}{\sum_{j=0}^{n} ra_{c_j d_i} + \sum_{j=0}^{n} sa_{c_j d_i} + 2}$$
$$+ w_h.\frac{\sum_{j=0}^{n} rh_{c_j d_i} + 1}{\sum_{j=0}^{n} rh_{c_j d_i} + \sum_{j=0}^{n} sh_{c_j d_i} + 2} \geq T_{threshold} \tag{5.10}$$

Where r_s and s_s is whether speed are satisfied or unsatisfied , r_a and s_a whether availability is satisfied or unsatisfied, r_h and s_h whether honesty is satisfied or unsatisfied.

Example: Consider a data centre used by three clients to download data, Table 5.1 shows the clients feedback about its availability. Using equation 5.10 the expected trust value that it will provide a satisfied speed is 0.625. Figure 5.6 shows the Beta plots for three feedbacks provided by three separate clients, and the Beta plot from the use of these feedbacks.

Client	Satisfied	unSatisfied
1	7	3
2	8	2
3	4	6

Table 5.1: Feedback of Three Clients

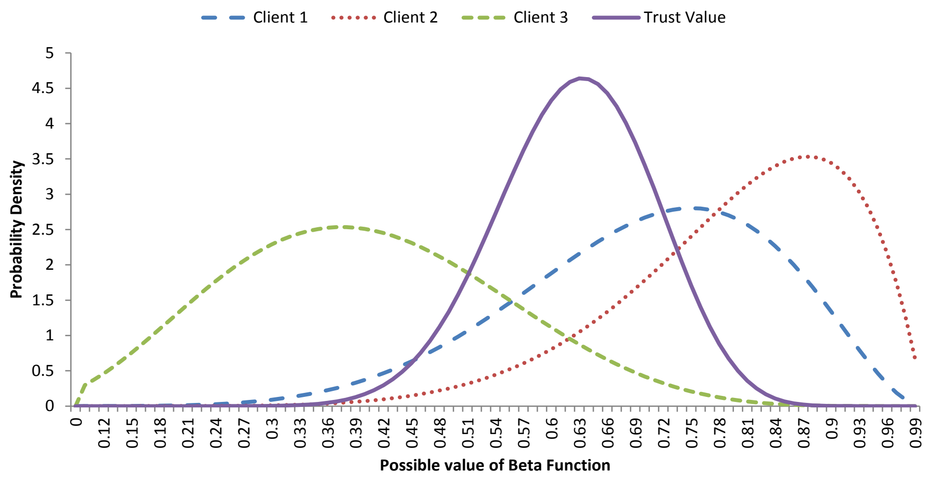


Figure 5.6: Beta function of data centre with feedback of three clients

5.3.2.5 Resulting Trust Framework Architecture

The trust framework (Figure 5.7) includes components that operate on both the client and the server; the client generates feedback, processes trust values and selects data centres based on its preferences; the server collects client feedback, updates the reputation database and provides reputation data to the clients.

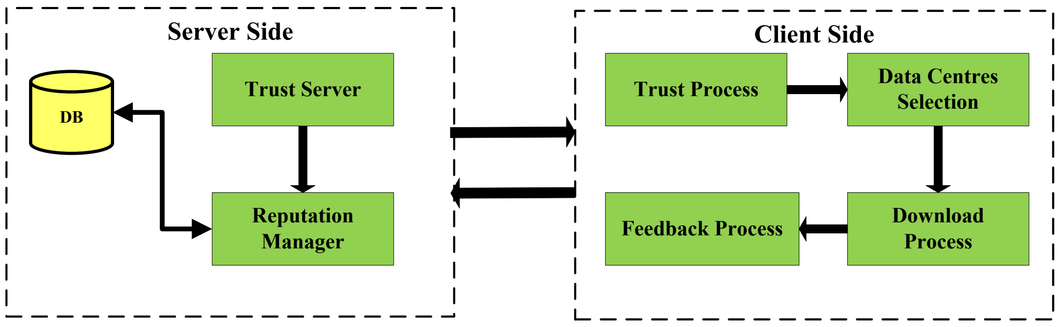


Figure 5.7: Trust Framework extending Attic

On the client side, the following components are identified:

1. **Trust Process**

   This is used by the client to extract the XML message sent by the server to retrieve reputation data for data centres and calculate the trust value of each data centre.

2. **Data Centre Selection**

   Data Centre Selection is responsible for selecting the data centres based on the clients preferences and the trust algorithm.

3. **Download Process**

   When data centres are selected the client uses the download process to generate a download request and send it to the data centres to download data.

4. **Feedback Process** After finishing downloading data the feedback process is used to generate an XML message containing the clients feedback about every data centre used to download that data. This XML message is sent to the server to report client feedback.

The server side has the following components:

1. **Trust Server**

   This server receives the clients request and responds to this request by supplying the reputation data for the data centres.

2. **Reputation Manager**

   This is responsible for updating and retrieving client feedback. It also prepares the XML message that has the data centres reputation data when the trust server is required to respond to a client request.

3. **Database**

   The database is used on the server side to store the reputation data of data centres. This database is updated by the reputation manager as a new client feedback is received.

The general operating procedure and interaction between these components is as follows: trust is calculated using a Beta distribution from the feedback obtained from the multiple clients that interact with a data centre, as outlined in (Equation 5.5 and 5.9 in section 5.3.2.2). Each client specifies whether satisfied (r) or not (s) with the download from the data centre. The Beta distribution was used to take into account this combined assessment by considering both satisfied and dissatisfied clients, rather than considering only positive outcomes (namely the number of times that a client has been satisfied with the download). The clients apply the trust values of data centres in the data selection algorithm to select which data centres will be used for downloading data. After a client completes downloading data, a subjective assessment of each of the three metrics (availability, honesty and speed) for each data centre used by this client is provided. This public feedback can then subsequently be used by other clients to support their decision about which data centres to trust, using Equations 5.3 and 5.9.

5.3.2.6 Data Centre Selection Algorithm

When a client calculates the total trust value of each data centre it uses algorithm 1 which refers to the trustThreshold which is used to limit the number of data centres that have been returned from the data lookup server. The client can either modify this parameter themselves or set the minimum number of data centres (referred to as minDC in algorithm 1) they would prefer to download from (namely the total number of data centres that match their particular trust criteria). In algorithm 1, the threshold value is set by a client to be 1.0. If an automated approach is used where a client does not specify the threshold but instead identifies the minimum number of data centres they would prefer to download from (for example they could set minDC as 3), the threshold value would automatically adjust.

Algorithm 1 Data Centre Selection

```
selectedDataCentres = 0;
trustThreshold = 1.0;
decrement = 0.1;
minDC=3;
loop
for each DataCentre[i] do
    if TotalTrustValue[i] $\geq$ trustThreshold then
        selectedDataCentres = selectedDataCentres + 1;
        return [i] ;
    end if
end for
if selectedDataCentres $\leq$ minDC then
    trustThreshold = trustThreshold - decrement;
    goto loop
    else exit();
end if
```

5.3.2.7 Messaging between components in the framework

In this section, the design of the messages that are transferred between components of the framework using XML is described. XML encoded messages are used for querying, describing data and reporting feedback, which are described in more detail in the following sections.

- **Querying Messages**

  This message (Figure 5.8) is used by the client for querying the data lookup server concerning where data is located and the trust data of data centres.

```
<File>
    <Data identifier>
          Data identifier
    </Data identifier >
 </File >
```

Figure 5.8: Querying Message

- **Describing Data Message**

  This message (Figure 5.9) is initiated by the data lookup server in response to a client request. The data lookup server creates a list of data centres, which have data, retrieves the trust data from these data centres databases and generates an XML message containing the information and sends it to the client.

```
<DataPointerCollection>
      <DataPointer>
      <File>
         <DataIdentifier>
               Data identifier
         </DataIdentifier>
         <DataName>
               Data name
         </DataName>
         <AssociatedProject>
               Associated project
         </AssociatedProject>
         <DataDescription>
               Data description
         </DataDescription>
      </File>
      <Endpoint>
         <url> endpoint-url </url>
         <SatisfiedSpeed> x </SatisfiedSpeed>
         <unSatisfiedSpeed> y </unSatisfiedSpeed>
         <SatisfiedAvialability> x </SatisfiedAvialability>
         <unSatisfiedAvialability> y </unSatisfiedAvialability>
         <SatisfiedHonesty> x </SatisfiedHonesty>
         <unSatisfiedHonesty> y </unSatisfiedHonesty>
       </Endpoint>
</DataPointer>
</DataPointerCollection >
```

Figure 5.9: Describing Data Message

- **Report message**

  This message (Figure 5.10) is generated by the data centre client to report the download process. In this message the client reports whether it was satisfied or dissatisfied regarding the download speed, and the availability and honesty of each data centre used in the download of data.

```
<Endpoint>
    <url> endpoint-url </url>
    <SatisfiedSpeed> x </SatisfiedSpeed>
    <unSatisfiedSpeed> y </unSatisfiedSpeed>
    <SatisfiedAvialability> x </SatisfiedAvialability>
    <unSatisfiedAvialability> y </unSatisfiedAvialability>
    <SatisfiedHonesty> x </SatisfiedHonesty>
    <unSatisfiedHonesty> y </unSatisfiedHonesty>
</Endpoint>
```

Figure 5.10: Report Message

The next UML sequence diagram (Figure 5.11) shows the exchange of these message during the download process.

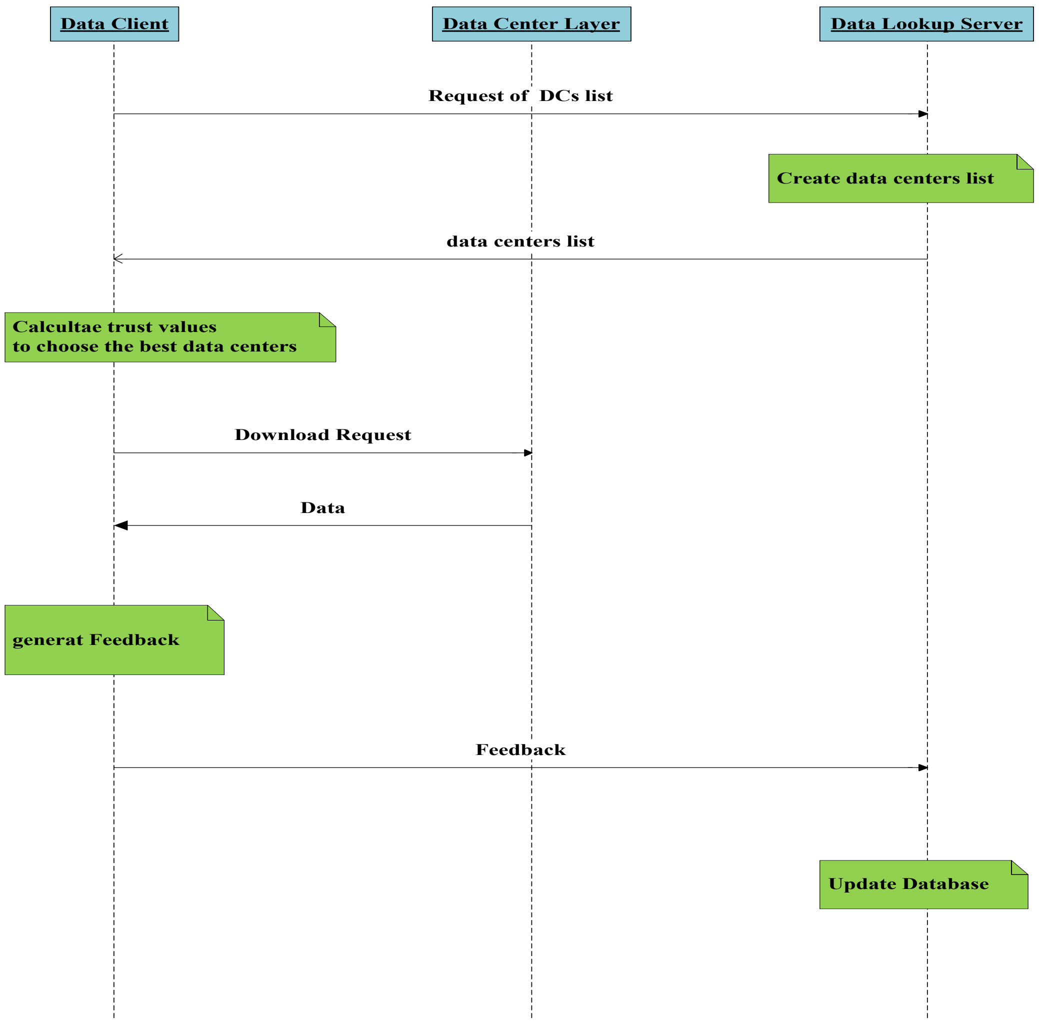


Figure 5.11: Interaction using the Trust Framework and Attic

1. The client of the data centre creates a data centre list request and sends it to the data lookup server.

2. The data lookup server uses its reputation manager to create a list of data centres, retrieves their trust value from the database and then creates a response message and sends it to the client.

3. The client uses data obtained from the XML message to calculate the trust value of each data centre, selects data centres based on its preferences and generates a download request message to the chosen data centres.

4. Data centres reply with the data needed.

5. The client checks the received data and subsequently generates a feedback message and sends it to the data lookup server.

6. The data lookup server uses its reputation manager to extract data from the feedback message and update the data centres trust value in the database.

5.4 Summary

This chapter described the VASCODE framework in terms of its design goals, system requirements (data caching, management, downloading and trust), the system architecture and its four layers, paying particular attention to trust which is modelled using the Beta distribution and an example is provided for illustrative purposes. The chapter ended by listing and describing the different steps in the processes in the interactions between components in the framework.

Chapter 6

Implementation

A novel feature of the VASCODE framework is the use of trust as a service to optimise the use of Attic file system by avoiding data downloads from data centres that behave maliciously or perform poorly in the decentralised data centres layer. Another feature of VASCODE is to allow the volunteers, who are interested in operating as data centres, to control the bandwidth they want to offer for uploading data to the clients. Such features will leverage the performance of volunteered projects that use BOINC middleware. In this chapter implementation details of the VASCODE framework are presented, describing how Attic is extended to meet the requirements discussed in Chapter 5, how the VASCODE trust framework is implemented, and finally how the VASCODE framework is integrated into BOINC middleware.

6.1 Implementation Overview

The implementation uses Java for most components. Java allows for object oriented design, modularity in system design, and easy integration with other Java, C, and C ++ components. The implementation also uses httpeer which was developed at Cardiff University as a very lightweight library that allows easy server-

and client-side HTTP data transfers. C ++ and POCO C ++ Libraries [23] are used to integrate VASCODE into BOINC middleware.

6.2 VASCODE Components

Because VASCODE is based on Attic, its three main components are implemented by extending reusable modules provided by Attic. In addition, new modules are developed for the new features provided by VASCODE. In the following sections we present these components.

6.2.1 VASCODE−DL

Figure 6.1 represents the extended data lookup server used to work as the reputation data service (for example, to provide reputation data and store feedback). This component plays a critical role in the framework because it offers the framework the capability to store and retrieve the reputation data of data centres. The VASCODE-DL receives a clients feedback as an XML message, it extracts this XML message to obtain the feedback about each data centre and uses this to update their reputation data in the database. In addition when the client sends a request to VASCODE to query from where data is available, VASCODE−DL retrieves the reputation data from the database, composes an XML message that contains a list of data centres and their reputation data, and sends it to the client to input it to the decision making process.

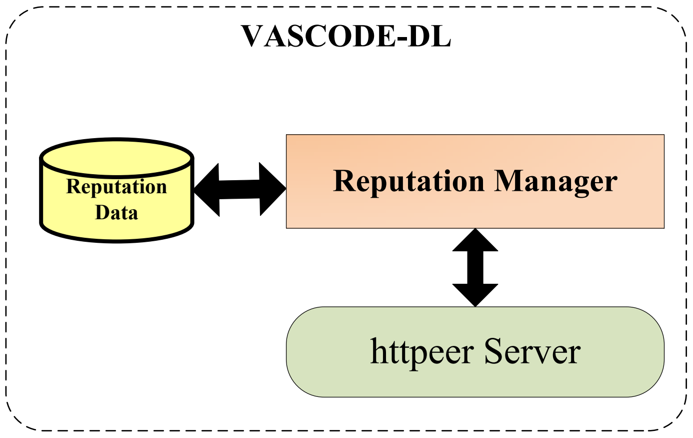


Figure 6.1: Extended Version of Data Lookup in Attic

6.2.1.1 httpeer Server

When a client sends a data download request, the httpeer server is responsible for accepting the clients request and forwarding it to the reputation manager; the reputation manager prepares the reputation data of those data centres which can be downloaded from. When the httpeer server replies to a client, it includes this reputation data in the XML message to the client. Below is a snapshot of a data pointer collection message, which contains a *DataDescription* tag that defines metadata about the data (*file ID, name, and project name*). In addition, it contains a list of *endpoints*. These are the data centres from which data can be retrieved.

```
<PointerCollection xmlns="http://atticfs.org">
    <DataPointer>
        <DataDescription>
            <id>bccb46ba-8d03-4227-b0b5-a115aac72ab3</id>
            <name>10MB.dat</name>
            <project>Test Project</project>
```

```
      <description>A test data description</description>
    </DataDescription>
    <Endpoint>
      <url>
        http://labx01.cs.cf.ac.uk:7777/dc/data/bccb46ba-8d03-4227-b0b5-a115aac72ab3
      </url>
      <meta>http://labx01.cs.cf.ac.uk:7777/dc/meta</meta>
      <SatisfiedSpeed> 9 </SatisfiedSpeed>
      <unSatisfiedSpeed> 1 </unSatisfiedSpeed>
      <SatisfiedAvialability>10 </SatisfiedAvialability>
      <unSatisfiedAvialability>2</unSatisfiedAvialability>
      <SatisfiedHonesty> 8 </SatisfiedHonesty>
      <unSatisfiedHonesty> 2</unSatisfiedHonesty>
    <Endpoint>
      <url>
        http://labx02.cs.cf.ac.uk:7777/dc/data/bccb46ba-8d03-4227-b0b5-a115aac72ab3
      </url>
      <meta>http://labx02.cs.cf.ac.uk:7777/dc/meta</meta>
      <SatisfiedSpeed>3 </SatisfiedSpeed>
      <unSatisfiedSpeed> 7 </unSatisfiedSpeed>
      <SatisfiedAvialability> 10</SatisfiedAvialability>
      <unSatisfiedAvialability> 0 </unSatisfiedAvialability>
      <SatisfiedHonesty>10</SatisfiedHonesty>
      <unSatisfiedHonesty> 0</unSatisfiedHonesty>
    </Endpoint>
    ........
    ........
    ........
  </DataPointer>
</PointerCollection>
```

The endpoint tag includes the URL of a data centre and reputation data of this data centre. For example the first data centre in the list is running on

httpp://labx01.cs.cf.ac.uk:7777/

and its reputation data is described in table 6.1 :

Metrict	Satisfied	unSatisfied
Speed	9	1
Honesty	8	2
Availability	10	2

Table 6.1: Data centre trust data

6.2.1.2 Reputation Manager

VASCODE DL uses the reputation manager process to manage reputation data. The reputation manager performs the following operations:

1. Retrieve Update reputation data of data centres.

2. Prepares an XML message that contains the reputation data of data centres.

3. Provides an XML message to the httpeer server as a response to the clients request.

6.2.1.3 Reputation Database

The system uses MySQL as its relational database. The connection to the database is made through JDBC, and SQL is used to query and update reputation data of the data centres. The VASCODE database is used to store the data centres reputation data, and Table 6.2 provides brief details on information saved in this database

Column Name	Description
dataCentreID	A unique ID for each data centre in the framework
satisfiedSpeed	Number of satisfied speed connections
unsatisfiedSpeed	Number of unsatisfied speed connections
satisfiedHonesty	Number of correct data provided by the data centre
unsatisfiedHonesty	Number of corrupted data provided by the data centre
satisfiedAvailability	How many times the data centre was available when data requested
unsatisfiedAvailability	How many times the data centre was unavailable when data requested

Table 6.2: Data Centres Reputation Data

6.2.2 VASCODE−DC

VASCODE-DC represents the data cache service in the framework and it was developed to offer the possibility of interested participants of Volunteer Computing projects to work as data centres and provide some of their bandwidth. It allows the user to control how much bandwidth can be offered. To achieve this goal, the data centre component in Attic is extended to acquire this function. VASCODE−DC 6.2 has three main components: data cache process to cache data, the httpeer server to accept and reply to clients requests, and the resource manager to manage the resources offered to the framework.

6.2.2.1 Resource Manager

During data sharing, the VASCODE-DC uses the resource manager which is responsible for controlling the bandwidth. For example, using the resource manager, a data centre that has a 10 Mb/s bandwidth can use the throttling option to set the bandwidth to 1 Mb/s for data uploading.

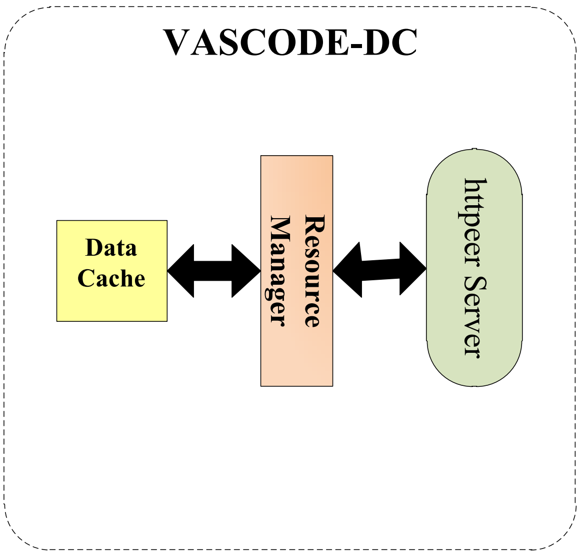


Figure 6.2: Extended version of data centre in Attic

6.2.2.2 httpeer Server

The httpeer server is used to accept the clients request and respond to this request.

6.2.2.3 Data Cache

The data cache is used to cache data using VASCODE-DC and to provide this data to the clients when it is needed.

6.2.3 VASCODE−DW

The VASCODE-DW (Figure 6.3) is used in the framework to download data from the decentralized data centres layer. It is the Attic client with extra new features; these features include data centre selection and sending feedback about each download to the server. The VASCODE−DW has four main components:

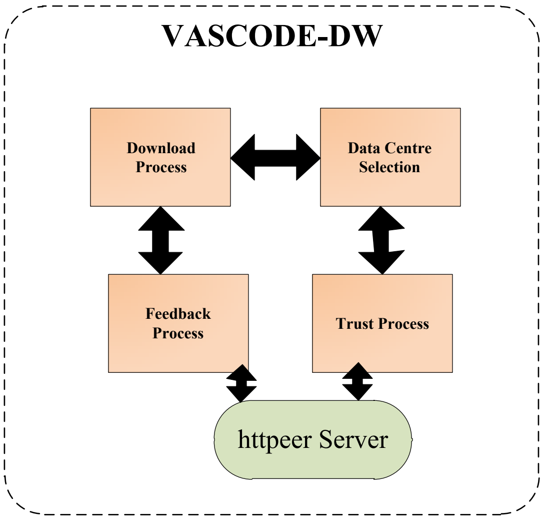


Figure 6.3: Extended Version of Data Worker in Attic

6.2.3.1 Trust Process

The VASCODE implements Trust as a service within the framework. Trust allows the VASCODE framework to optimise the performance of P2P networks and subsequently the Volunteer Computing projects that use these networks. The trust process is used by VASCODE-DW to calculate the trust value of each data centre before it starts downloading data.

6.2.3.2 Data Centres Selection

The data centre selection manager will receive the trust value of each data centre from the trust process, then it applies the data centres selection algorithm using these trust value to select the best data centres from which to download data.

6.2.3.3 Download Manager

The download manager receives a list of data centres that have been selected for data downloading from the data selection manager, then it starts requesting data from the selected data centres.

6.2.3.4 Feedback generator

When the data download is complete, the download manager notifies the feedback generator about the download process. It reports three pieces of information to the feedback generator; which data centres were available; their speed and if any sent corrupted data. The feedback manager then uses this data to generate a feedback message and sends it to the server.

Below a snapshot of the feedback message that contains the URL of the data centre and how many times the client was satisfied or unsatisfied about this data centre using the three metrics. Here, it can be seen that the client made 2 requests using *http://labx01.cs.cf.ac.uk:7777/dc/*. It was satisfied in upload speed for all requests and the data centres was available and honest.

```
<Endpoint>
    <url>
       http://labx01.cs.cf.ac.uk:7777/dc/data/a849534e-1b8b-4e42-9e43-e4677ac973f0
    </url>
    <satisfiedSpeed> 2 </satisfiedSpeed>
    <unsatisfiedSpeed> 0 </unsatisfiedSpeed>
    <satisfiedAvialability> 2</satisfiedAvialability>
    <unsatisfiedAvialability> 0 </unsatisfiedAvialability>
    <satisfiedHonesty> 2 </satisfiedHonesty>
    <unSatisfiedHonesty> 0 </unSatisfiedHonesty>
</Endpoint>
```

6.2.3.5 VASCODE-DW Fault Recovery

The VASCODE-DW uses the download scheme discussed in section 5.2.3.2 to obtain data from the BOINC server when the data centres are unavailable or do not respond to the data request. VASCODE-DW uses the time out option to specify the time needed before the connection to a data centre can be established. When the time out expires before the client can connect to the available data centres it uses the fall-back URL, which is the URL of the BOINC data server. This mechanism allows the BOINC client to get the input data to process its work unit even when the data centres that are selected to download data are not available.

6.3 VASCODE BOINC Integration

Figure (6.4) provides an overview of the implementation architecture of VASCODE with BOINC middleware, showing how the VASODE components are integrated into the BOINC middleware.

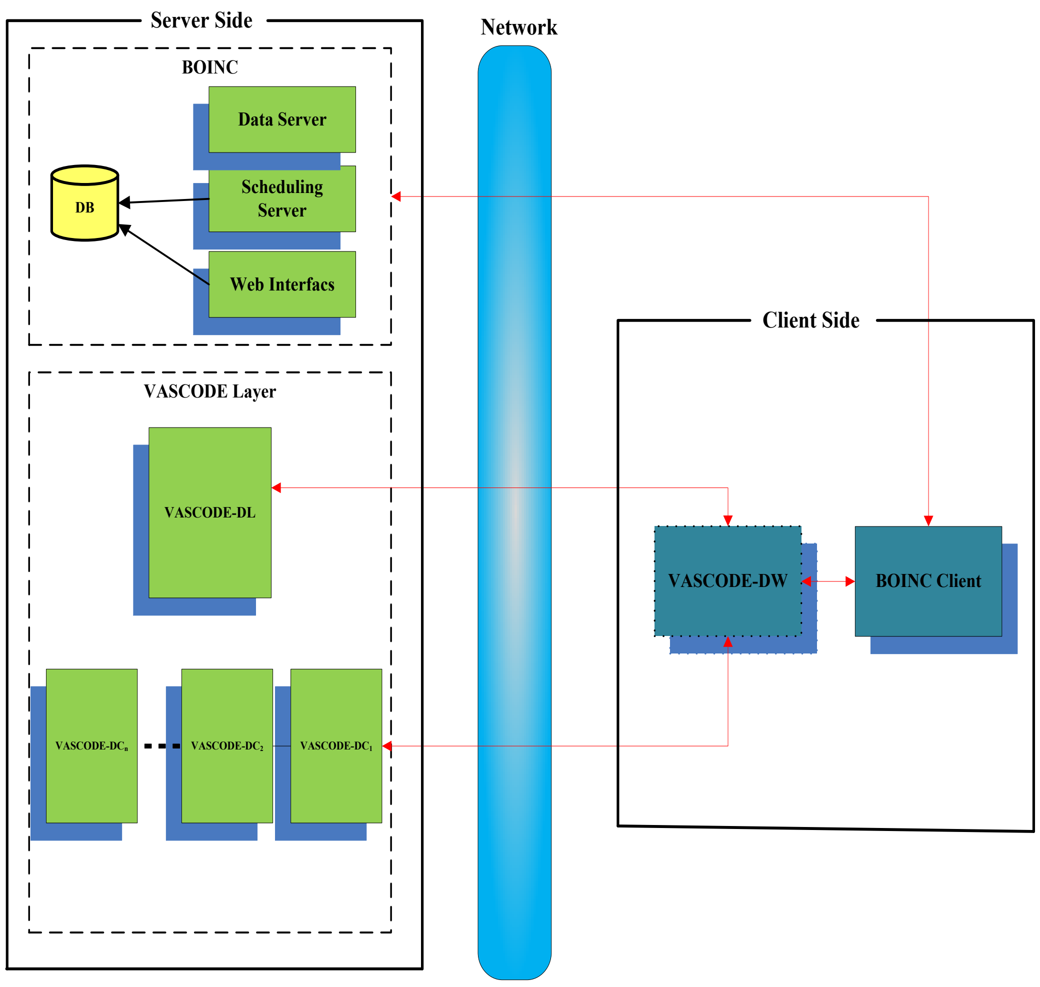


Figure 6.4: BOINC Middleware using VASCODE

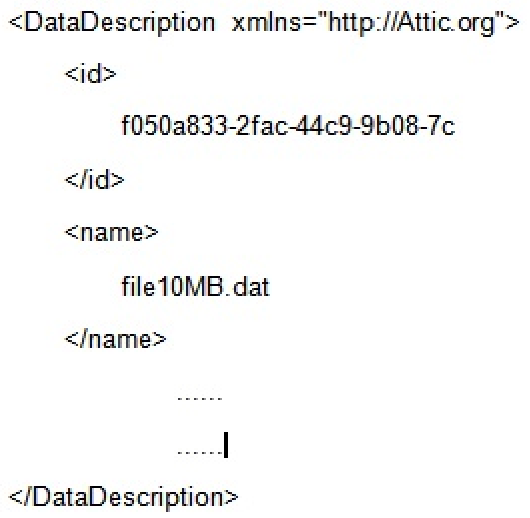
On the client side, the system has two components the BOINC client and VASCODE−DW, while the server side is composed of three components: the BOINC servers, VASCODE−DL, and VASCODE−DC.

6.3.1 Integrating VASCODE-DW into BOINC Client

When the BOINC client needs an input file to process a job and this input file is available in the decentralized data centres layer, there are two approaches to allow the client to download that data using the VASCODE framework. The first one is by modifying the BOINC client, and the second one is by setting a proxy between BOINC and VASCODE. These two approaches are discussed in the next sections.

6.3.2 Modifying BOINC Core Client

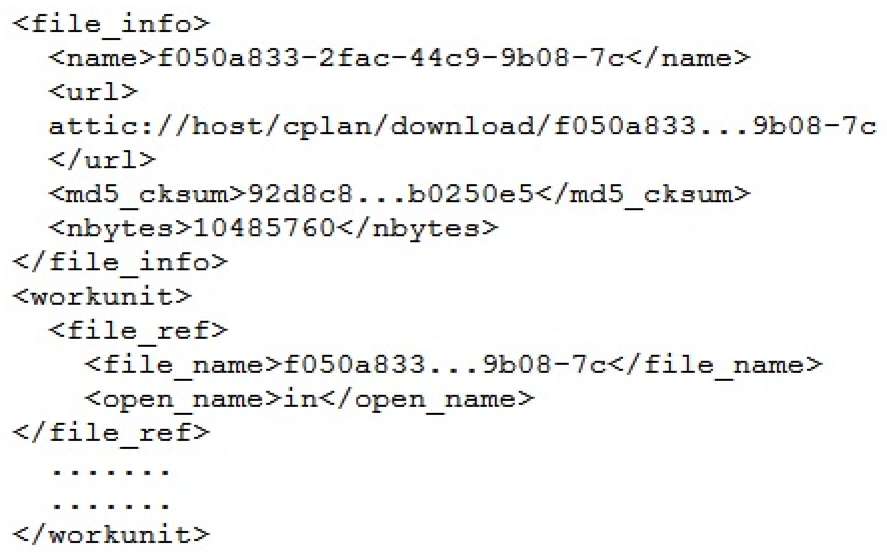
BOINC middleware is an open source software, that provides the capability to modify the BOINC client. The client code responsible for downloading data is modified to allow the BOINC client to use VASCODE for data downloading instead of using the BOINC data server. The following section explains how this approach can be used. Attic uses a unique ID to identify data instead of using the file name. Figure 6.5 shows a snapshot of a Data Description of data published in Attic, stating its ID and name.



```
<DataDescription  xmlns="http://Attic.org">
    <id>
        f050a833-2fac-44c9-9b08-7c
    </id>
    <name>
        file10MB.dat
    </name>
            ......
            ......
</DataDescription>
```

Figure 6.5: Snapshot of a Data Description of Data Published in Attic

When the data is published, a new data identifier is generated and used to generate work units. Figure 6.6 shows a snapshot of a work unit that uses Attic for downloading data. It includes the file name, URL, check sum, and size of data. In addition, there are the file reference and open name, which are used by BOINC client during processing of the work unit. The download URL in a work unit is changed to **attic** instead of **http** and this tells the BOINC client when it parses the work unit XML description to use VASCODE to download data.



```
<file_info>
  <name>f050a833-2fac-44c9-9b08-7c</name>
  <url>
  attic://host/cplan/download/f050a833...9b08-7c
  </url>
  <md5_cksum>92d8c8...b0250e5</md5_cksum>
  <nbytes>10485760</nbytes>
</file_info>
<workunit>
  <file_ref>
    <file_name>f050a833...9b08-7c</file_name>
    <open_name>in</open_name>
</file_ref>
  .......
  .......
</workunit>
```

Figure 6.6: Snapshot of a Work unit using Attic

In Figure 6.7 three stages are indicated: (1) the BOINC client contacts the BOINC Task Server to get a work unit, then it parses the work unit, (2) the BOINC client takes the data identifier and contacts VASCODE-DL to get a list of data centres possessing the data, and (3) the BOINC client contacts the decentralized data centres layer to start downloading the data.

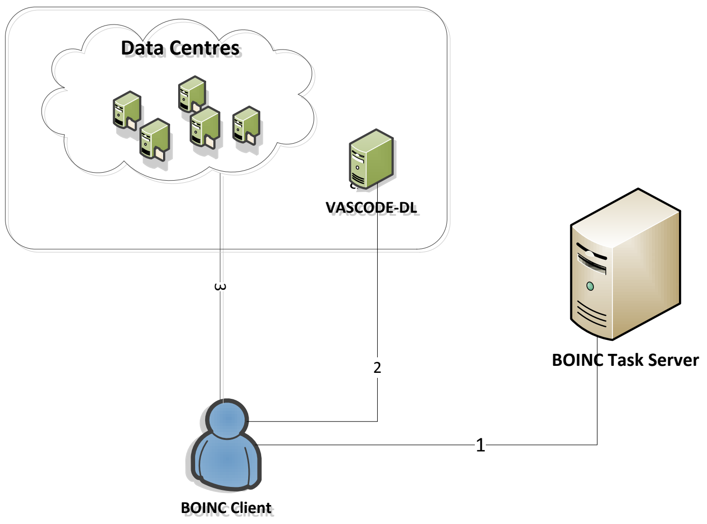
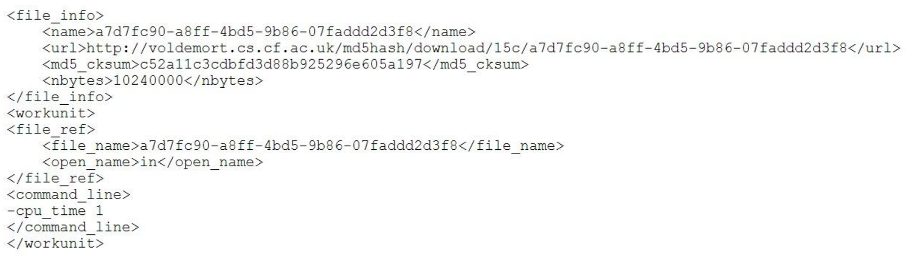


Figure 6.7: Modified BOINC Client uses VASCODE to Download Data

6.3.3 VASCODE Proxy

A potential issue with the previous approach is that a new BOINC client would need to be released for users to take advantage of the VASCODE framework. Therefore, instead a second approach is to have the BOINC client interact with the VASCODE via a proxy. To use this approach, the VASCODE proxy was developed. When the BOINC client requests an input file from the BOINC data server it is redirected to the VASCODE proxy to download data.

The VASCODE proxy is designed to allow BOINC to use the VASCODE framework without the need to modify the BOINC client. It consists of an application that runs in the background for processing data requests locally. This application runs a local server on port 9980. When a data file is published in the Attic file system, Attic assigns a GUID for the file when it is uploaded, which is used to name the data file before adding it to the work unit. Figure 6.8 shows an XML description of a work unit that uses the GUID in the download URL of data. When a BOINC client uses this download URL to obtain data its download request is redirected to the local server which extracts the GUID from the received request and uses VASCODE-DW to download the file from the de-centralized data centres layer. When the data is downloaded the proxy application starts uploading it to the BOINC client.



```
<file_info>
    <name>a7d7fc90-a8ff-4bd5-9b86-07faddd2d3f8</name>
    <url>http://voldemort.cs.cf.ac.uk/md5hash/download/15c/a7d7fc90-a8ff-4bd5-9b86-07faddd2d3f8</url>
    <md5_cksum>c52a11c3cdbfd3d88b925296e605a197</md5_cksum>
    <nbytes>10240000</nbytes>
</file_info>
<workunit>
<file_ref>
    <file_name>a7d7fc90-a8ff-4bd5-9b86-07faddd2d3f8</file_name>
    <open_name>in</open_name>
</file_ref>
<command_line>
-cpu_time 1
</command_line>
</workunit>
```

Figure 6.8: Snapshot of a Workunit used file GUID

In BOINC, to allow the BOINC client to retrieve files from the VASCODE proxy, a redirect rule must be defined for the project download folder. This rule will redirect requests to the following URL.

http://localhost:9980/data/GUID

Figure 6.9 shows the workflow involved in getting the file from the decentralized data centres layer using the VASCODE proxy.

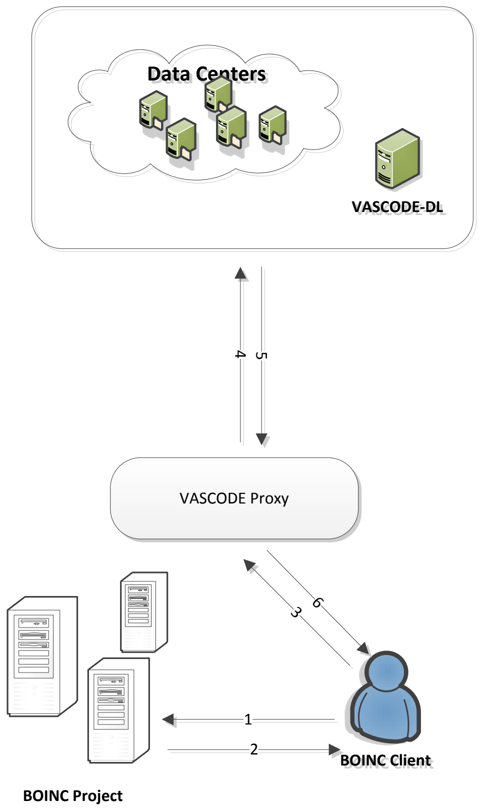


Figure 6.9: BOINC Client uses VASCODE Proxy to Download Data

1. BOINC client requests a file from the BOINC project.
2. TheThe project redirects the request to the VASCODE proxy and returns the resolved Attic URL.
3. The VASCODE proxy extracts the file ID from the URL and uses it to request the file from the decentralized data centres layer.
4. The data centres send the file to the VASCODE proxy.
5. The VASCODE proxy sends the file back to the BOINC client.

6.4 Summary

This chapter presented an overview of the implementation of the VASCODE framework. It introduced the three major VASCODE components: VASCODE−DL, VASCODE−DC and VASCODE−DW. The components and functions of each of these were described: for VASCODE−DL the httpeer server, the reputation manager and reputation database, for VASCODE−DC the resource manager, the httpeer server and the data cache, and for VASCODE−DW the trust process, data centres selection, download manager and feedback generator.

The way the VASCODE trust framework is implemented using VASCODE−DL and VASCODE−DW was described. The two approaches to VASCODE−BOINC integration that can be used when the BOINC client needs an input file which is available in the decentralized data centres layer were also described. The first is to modify the BOINC client and the second is to setup a proxy between BOINC and VASCODE. The chapter ended by introducing the VASCODE proxy whereby a BOINC client can interact with decentralized data centres layer via a proxy.

Chapter 7

Evaluation Of Hypothesis

In this chapter, a performance study of the VASCODE framework is described and presented in order to understand its impact in data distribution when it is used in different environments. Performance experiments have been conducted to show the effect of the trust framework on system performance. In addition, a study is conducted through experiments to show the effect of weighted factors when they are used to address various environments. The described empirical experiments also show the effect of using throttling functionality to form the decentralized data layer and how clients use the trust framework to select the best data centres when throttling is used.

7.1 Objectives

To demonstrate the benefits of the VASCODE framework developed in this thesis, it was necessary to show its effectiveness in improving data distribution in BOINC projects by using the Attic file system for data distribution. The main goal of VASCODE is to provide a means for Volunteer Computing projects that use BOINC middleware, to be extended to allow participants to form and join a decentralized data centres layer by volunteering a proportion of their bandwidth for data distribution. In addition, VASCODE is used to optimize the download time of data to avoid malicious data centres and this goal depends on trust data received from VASCODE to assess data centres and select the best ones that meet the client preferences. A set of experiments have been conducted to evaluate VASCODE using a quantitative approach as follows::

- To compare the benefit of using trust for data centres selection by conducting a number of experiments.
- To show how the throttling feature is used by running different examples using different upload speeds.

7.2 VASCODE Trust framework Evaluation

This section presents an evaluation of the trust framework used in VASCODE; it shows how trust values obtained from VASCODE can help clients of Attic to select the best data centres for downloading data. These experiments explore the different characteristics of a data centre such as its upload speed, availability, and trustworthiness. Also, these experiments show the flexibility of VASCODE when it is used in dynamic environments.

7.2.1 Evaluation Scenarios

To support the argument that the proposed framework leads to an improvement in data distribution by decreasing the download time of data, it is necessary to evaluate its performance using a range of scenarios. These scenarios enable a quantitative evaluation of the proposed framework, and therefore a number of scenarios have been selected to test and demonstrate the research case. These scenarios are derived from the client requirements and data centres behaviours which are discussed below.

7.2.1.1 Data centres behaviours

Since the data centres are ordinary Internet users, who volunteer their resources for a scientific project, three main attributes can be used in the evaluation scenarios.

1. It is expected that some of them may have malicious behaviour.
2. They have either a slow internet speed or are not interested in providing all of their internet bandwidth to serve data for clients requests.
3. Their availability changes over time as they are not dedicated machines for data distribution and they can only voluntarily distribute data when they are online

In this evaluation, in order to mimic the behaviour of data centres, different upload speeds were used and a Poisson distribution was used to mimic data centres availability. In addition, corrupted data was injected to some data centres to make them play the role of malicious data centres.

7.2.1.2 Clients preferences

The preferences of clients can be specified in terms of their expectation of data centres. To make an objective assessment of data centres, a set of client requirements must be identified. These preferences provide important information to

Attribute	Honesty	Speed	Availability	All
Value	1	1	1	1/3 1/3 1/3

Table 7.1: Weight Factors used in the experiments

VASCODE to determine the best data centres that the client can use to download data from. The clients preferences can be considered to have expectations on a single or multiple attributes of the data centre (namely its availability, speed and honesty). It is also possible to differentiate between the preferences of the clients, as some clients prefer to get data from highly available data centres, while others prefer to interact just with the honest data centres to avoid corrupted data and others might prefer to get data from data centres that have high speed internet connections. However, clients can use a combination of these attributes to meet their requirements. Table 7.1 summarizes the weight factors that are used to represent the client requirements in the experiments.

7.2.1.3 Scenarios

The combination of client requirements and data centre behaviors form a set of scenarios. These scenarios are used to demonstrate the effects of using VASCODE in data download. In the experiments, the following four scenarios are used.

- **Scenario 1.** The data centres have different availability, have the same upload speed, and are all honest. The clients are interested in data centres that have high availability.
- **Scenario 2.** Some of the data centres have malicious behaviour, but all are available and have the same upload speed. The clients are interested in data centres that are honest.
- **Scenario 3.** The data centres have different upload speeds, but all are honest and available. The clients are interested in data centres that have the best upload speed.
- **Scenario 4** The data centres have different upload speeds, some of them are malicious, and are not available all the time. The clients are interested in the non malicious data centre that has the highest upload speed and the highest availability.

These scenarios are used to present quantitative measurements that show that VASCODE framework can improve the download time of data and consequently the performance of projects.

7.2.2 Testbed Environment

The Test-bed environment for the experiments includes 33 Linux-based machines. 32 machines have similar specifications with a 2.8 GHz Intel Pentium Processor and 2 GB of Memory, Fedora 7 as an operating system, and Java version 1.6.0.14. These machines where used to run various combinations of clients and data centres. The remaining machine was equipped with a 2.0 GHz Intel Pentium Dual Core Processor and 3 GB of memory, and was used to run the VASCODE-DL server. These machines are connected through a LAN network and a web interface was used to configure the network connections speed of these machines to 10 Mbps or 100 Mbps.

7.2.3 Experimental Hypotheses

The experiments hypotheses are presented and explained in this section to show how they meet the thesis hypothesis.

1. The experiments involve the use of a P2P network (i.e. Attic file system) to form a decentralized data centres layer, and show how VASCODE can be used to form a decentralized data centres layer for data distribution. In addition, the experiments also involve two different clients of data: the basic Attic client; and the VASCODE-DW to show that VASCODE allows the use of different types of clients.

2. The experiments use the VASCODE trust framework to manage the selection of data centres based on their trust values in different environments. Four scenarios were used to represent various environments. The experi-

ments show the ability of VASCODE to improve download time of data in these environments through the use of VASCODE trust framework.

7.2.4 Experimental Error

Error bars were used to compare visually the average download speed of data using Attic-DW and VASCODE-DW. This shows whether VASCODE introduces a statistically significant performance improvement to the distribution of data in BOINC projects through the application of a trust mechanism to select the optimum data centres for downloading data. The error bars are included in all graphs to represent the standard error of the experiments which is calculated using the standard deviation and the average download time of data. Appendix A shows the average download time and standard errors for all experiments.

7.2.5 Effect of Data Centres Availability

In a Volunteer Computing environment peers can enter and exit the network at any time. In Attic, as peers can also play the role of a data centre, the availability of these data centres can change over time. With the distributed data centres appearing and disappearing, a mechanism is required to limit this variability in the network so that a clients download efficiency is maximised. This experiment is conducted to show how the download time is improved when the trust framework is utilised.

7.2.5.1 Experiment Setup

In this experiment, the experiment setup is shown in Figure 7.1. Attic consists of a lookup server and 10 data centres. The 10 data centres have a 10Mb/s connection and are all deemed honest peers (i.e. no malicious behaviour) for this experiment.

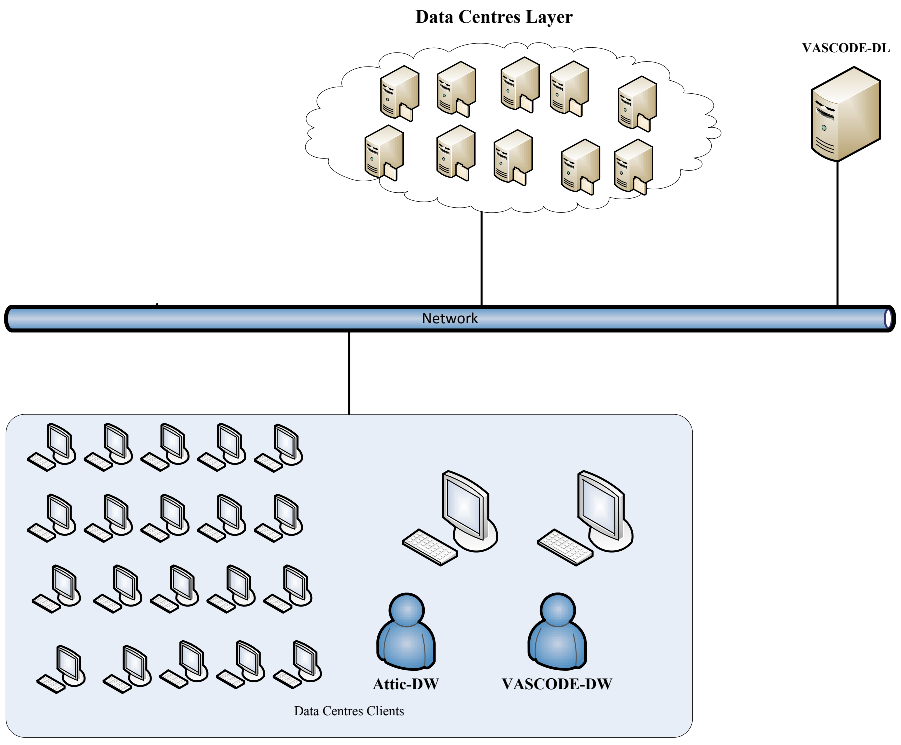


Figure 7.1: Attic File System used with different Data Centres Availability

22 Linux machines were used to run data centres to mimic data requests in all experiments and these were set up to request a 10 MB file at periodic intervals. 20 machines were setup to request the file at periodic intervals, as shown in Table 7.2. They report their download experience to the VASCODE-DL server when they finish downloading data. This feedback to the server contains the status of each data centre and information about whether they were available, whether the download speed was satisfactory and whether they were honest.

Periodic Interval	**Machines**
Every 2 Minutes	M1,M2,M3,M4
Every 4 Minutes	M5,M6,M7,M8
Every 6 Minutes	M9,M10,M11,M12
Every 8 Minutes	M13,M14,M14,M15
Every 10 Minutes	M16,M17,M18,M19

Table 7.2: The Twenty Machines that are Running in Background

The other two machines were setup to request the file every five minutes, one of the machines uses the trust data to select the best data centres before downloading the file, the other machine to mimic the Attic client it was setup to do not use the trust data for selecting data centres.

The data centres can go offline at any time and a Poisson distribution was used to model the behaviour when the data centres are online; it was used to simulate the availability of data centres as this represents a realistic scenario for many existing P2P systems.

The Poisson distribution is a discrete probability distribution and is used to model the number of events occurring within a given time interval. In the experiments in this research, it was used to show when the data centres are on and off during a four hour period. Figure 7.2 shows the distribution of the data centres and when they are online and offline over a four-hour period. The total duration of the experiment was eight hours; only the first four hours have been shown here to demonstrate the overall availability trend.

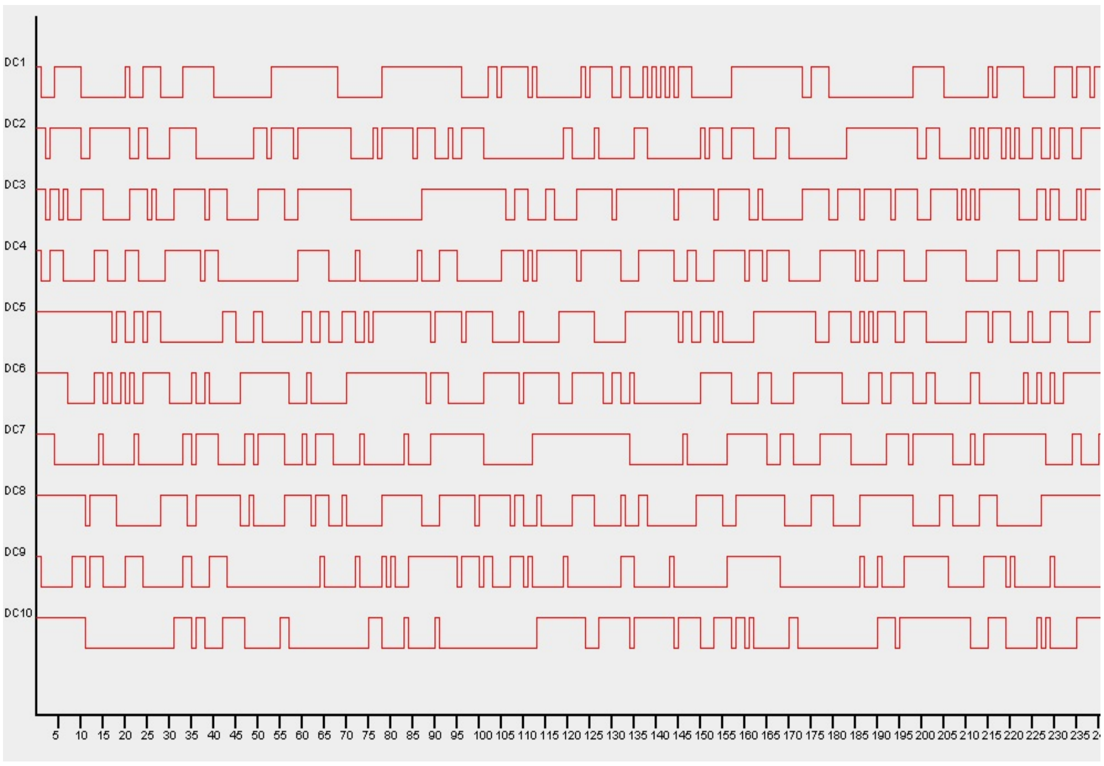


Figure 7.2: Data Centres Availability

The behaviour of the Attic client with VASCODE-DW are compared. One instance of each type of client is used, each requesting data periodically every five minutes during the experiment. VASCODE-DW was configured with the following parameters: Availability weight factor (AWF)=1, Honesty weight factor (HWF) = 0 and Speed weight factor (SWF)=0. As the focus is on the availability of the 10 data centres, the other weight factors are set to zero.

7.2.5.2 Experiment Achievement

This experiment had a duration of eight hours and the data centre availability in the first four hours is the same as in the second four hours as shown in Figure 7.2. It was was found that during the first four hours of the experiment, the download time of both clients is similar Figure 7.3. However, in the next four hours, the trusted data centre has an improved download time as the trust algorithm converges and learns the state of the network, namely VASCOE-DW learns to avoid the unavailable data centres as the experiment progresses. In Figure 7.4, it can be observed that the behaviour of nodes employing the use of the trust algorithm becomes smoother and more predictable during the last four hours of the experiment. This convergence shows promise as the algorithm can adapt to network conditions and variability in node availability, which is a requirement of the Volunteer Computing environment as a whole.

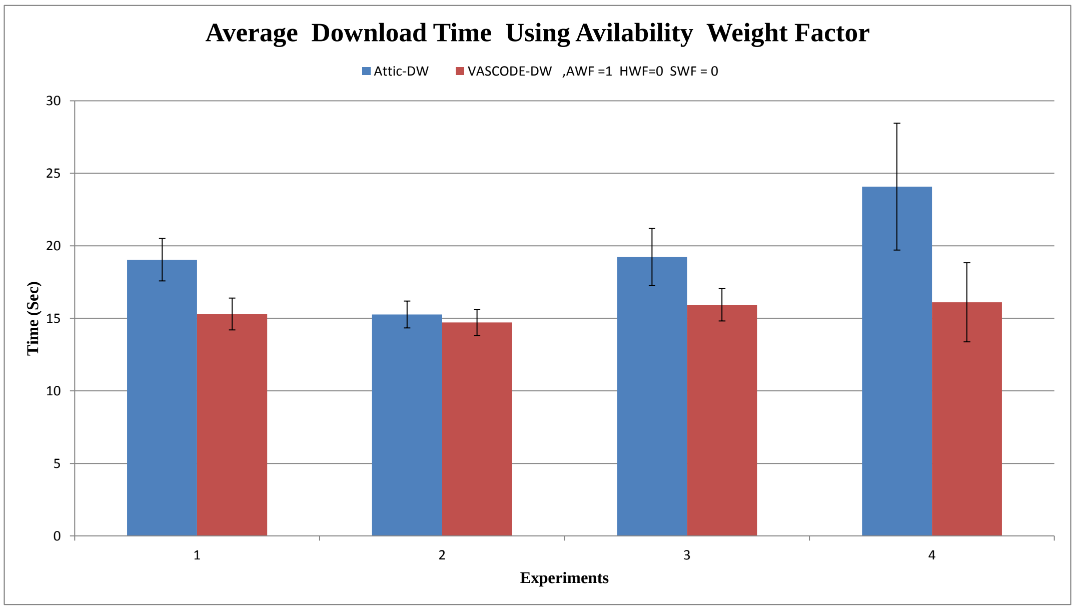


Figure 7.3: Data Download using AWF = 1.0 in the first four hour period

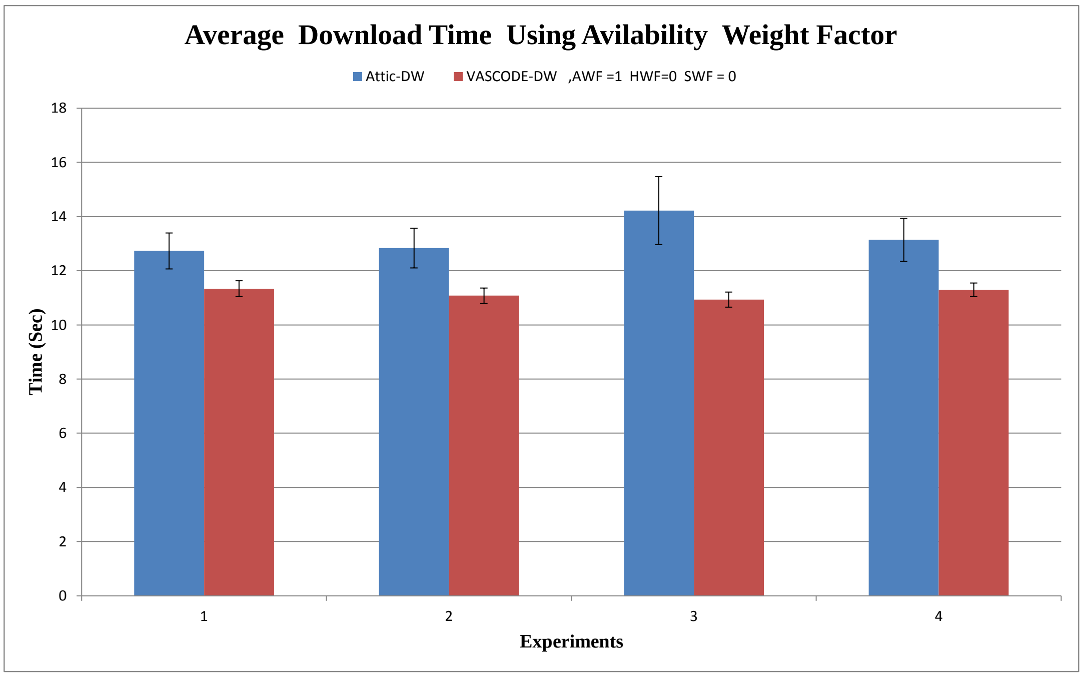


Figure 7.4: Data Download using AWF=1.0 in the second four hour period

7.2.6 Effect of Data Centre Honesty

Since the decentralized data layer consists of ordinary Internet users, it is expected that some of them will exhibit malicious behaviour. A data centre in Attic caches data locally and it has access to this data; a malicious data centre can potentially replace the cached data with corrupted data to disrupt the performance of the system. In VASCODE, to achieve malicious data centre identification, it is the obligation for all data clients to verify the integrity of data. In the event of data integrity failure, the data centre will be identified as a malicious data centre and the data clients will use a different data centre to obtain data.

In this experiment, malicious data centres are injected into the network in order to disrupt the system. These nodes intentionally provide corrupted data to their clients in order to attack the system. An experiment was designed to show

how the VASCODE trust framework can become fault tolerant to this malicious behaviour and avoid the use of malicious data centres to recover and repair the corrupted network. This experiment focuses on the honesty of data centres and how this affects the download time. It aims to show how the client, who uses the VASCODE trust framework, offers better stability and hence increased download efficiency than the ordinary Attic client. Note that since an MD5 hash is taken of the data, malicious peers can only slow down the network because if the hash of the downloaded file does not match the original hash of the data, it will be discarded and downloaded again. The VASCODE trust framework detects these malicious peers and effectively removes them from a clients download list, thereby making significant gains overall.

7.2.6.1 Experiment Set up

Six data centres are used in this experiment (Figure 7.5). Three of them are honest data centres and the other three are malicious and send corrupted data to their clients. Because we are interested in the honesty of these data centres, all have the same speed connection,(10Mb/s), and they are available throughout the duration of the experiment. VASCODE−DW was configured with the following parameters (AWS = 0, HWF = 1, SWF = 0). As the identified in previous experiment for availability, only the honesty weight factor is set to 1 in this experiment.

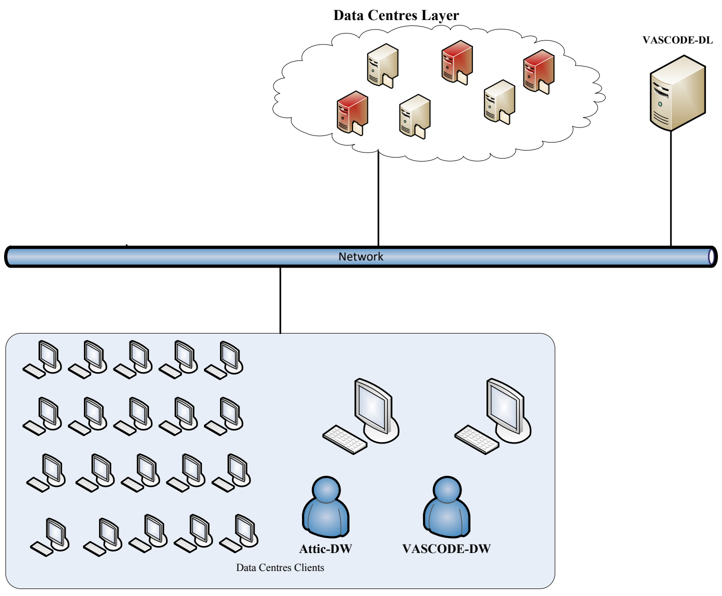


Figure 7.5: Attic File System using a Number of Malicious Data Centres

7.2.6.2 Experiment Achievement

Figure 7.6 shows the result of this experiment. It can be observed that the trusted client has significantly better download time because it avoids the malicious data centres, while the ordinary Attic client uses all the data centres and therefore downloads a number of unnecessary corrupted chunks which need to be reloaded, thereby incurring a download overhead. After a short period of convergence, this system performs on average three times better than the standard Attic approach. This shows huge potential as it addresses one of the fundamental issues in volunteers distributing data, namely the ability to trust third party peers. This experiment indicates that the system can learn to avoid malicious peers and dynamically select more trusted peers in the network, which opens up the possibility for such a dynamic data distribution approach.

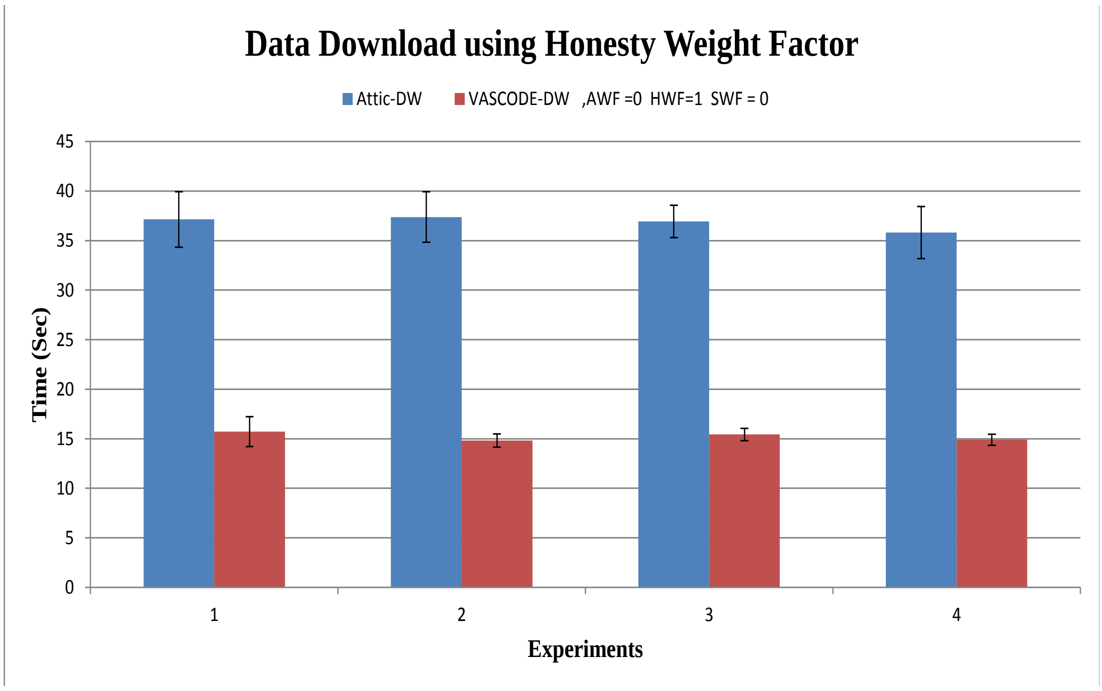


Figure 7.6: Data Download using Honesty Weight Factor

7.2.7 Effect of Data Centres Speed

The data centres used in Attic can have significantly different upload speeds (due to variability in the home users connections, for instance) and this obviously affects the download time of each peer. Bandwidth throttling is also used by many Internet Service Providers, leading to variable download speeds for different users. Since clients are interested in getting data in the fastest possible way, they should obviously choose those data centres with high-speed connections. However, if all peers download from the same fastest peers then there will be a bottleneck, so any algorithm must dynamically optimise the distribution of clients connected to a data centre at each time-step during the operation of the system. The VASCODE trust framework provides such a mechanism by choosing data centres with high bandwidth at that point in time in order to optimise the throughput of the distributed system as a whole.

7.2.7.1 Experiment Setup

This experiment (Figure 7.7) shows how the trust framework is used to choose the data centres which have the highest bandwidth connections at any point in time. For this experiment, 10 data centres are used. Six data centers have 10 Mb/s connections and four data centres have 100 Mb/s connections. The data centres have continuous availability and they all act honestly. VASCODE-DW was configured with the parameters (AWS = 0, HWF = 0, SWF = 1) to configure the system to only focus on the speed of the data centers.

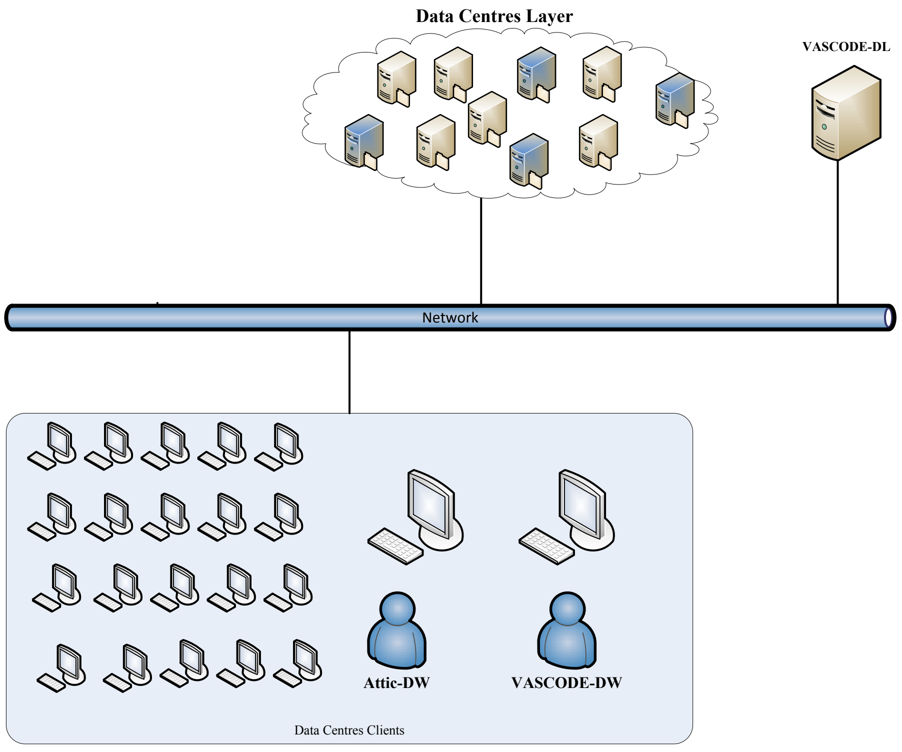


Figure 7.7: Attic File System Using Data Centres with different upload Speed

7.2.8 Experiment Achievement

The results in Figure 7.8 show significant improvements by clients making use of the VASCODE trust framework, compared to the conventional Attic approach, achieving download speeds several times faster overall. These results demonstrate the benefit of the approach.

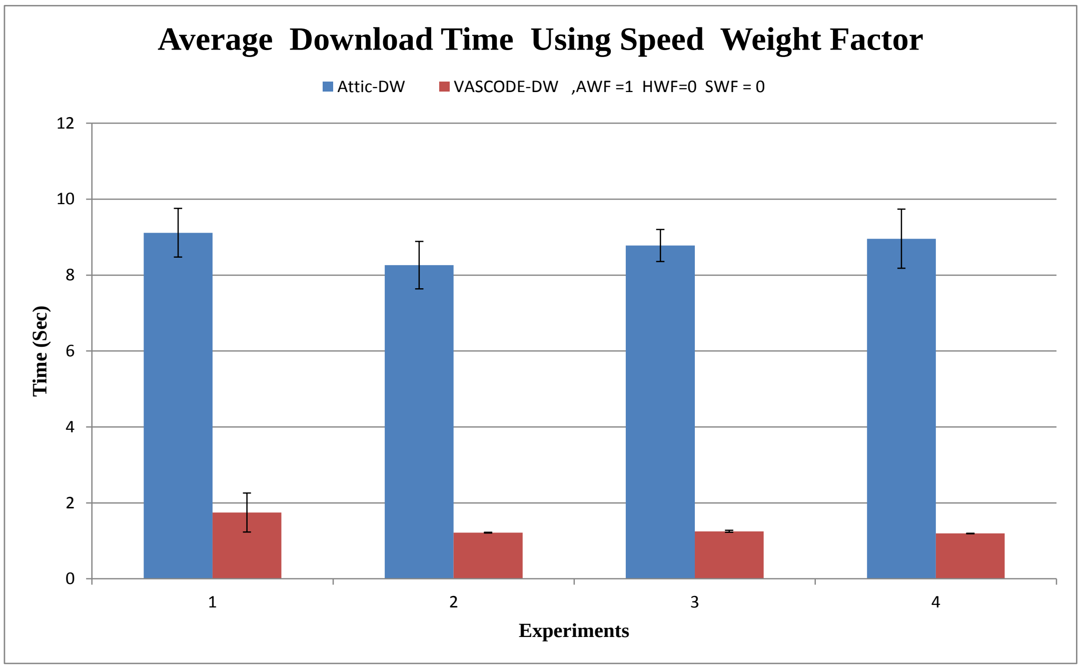


Figure 7.8: Data Download using Speed Weight Factor

7.2.9 Effect of different behaviours of Data Centres

This experiment is designed to show how the trust framework can be used in different environments to improve download time when all three factors (speed, honest, availability) are considered. Therefore, the data centers were configured so that they have different speeds, with some data centres acting maliciously and others with a variable availability over time.

7.2.9.1 Experiment Setup

For this experiment, 10 data centres are used with six data centres having a slow speed connection (10 Mbps) and four data centres having a high speed connection (100 Mbps). Three of the data centres act maliciously and the availability of all data centres changes over time according to the Poisson distribution used in the first experiment. The combination of these three factors creates an extremely hostile network for this size and provides an extreme test for the system in having to deal with a number of complex factors to optimise the clients download capability overall. The VASCODE-DW was configured with the parameters (AWS = 1/3, HWF = 1/3, SWF = 1/3) to provide equal weight to all three factors.

7.2.9.2 Experiment Achievement

Figure 7.9 again shows significant improvement over the standard Attic approach overall, with VASCODE-DW achieving one order of magnitude and more increased performance over a significant duration of the experiment. However, there are spikes in the experiment as the network changes and the trust algorithm has to re-converge. Two explanations are provided for this effect. Firstly, this could simply be a result of the small distribution of the nodes in this experiment and, at certain times, due to the three factors acting together and there simply is not much possibility for achieving improvement. Secondly, a heuristic approach should be taken in optimising the distribution of setting the weights of the parameters of the trust equation. This is a multi-dimensional space in itself and requires further investigation.

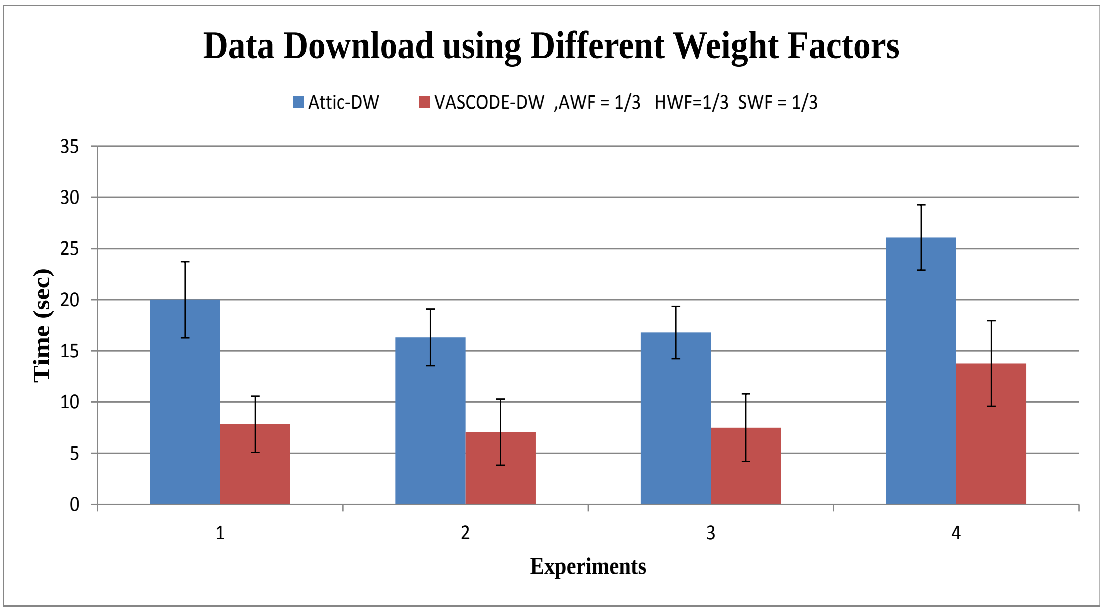


Figure 7.9: Data Download using different Weight Factors

7.3 VASCODE Data Throttling Evaluation

In this section, the data centres will volunteer a part of their bandwidth for data distribution through the use of a throttling option. The upload speed of data centres have an effect on the download time of data and this effect increases when some of these data centres provide corrupted data. VASCODE-DW can deal with this situation by using the trust framework to select the best data centre. Two experiments have been conducted to show how VASCODE-DW can avoid the malicious data centres and select the ones that have high upload speed when these data centres use the throttling functionality.

7.3.1 Data Throttling and SWF

To attract ordinary internet users to participate in data distribution, throttling in VASCODE can be used. This allows the participants of a BOINC project to specify the speed for the distribution of data to prevent any disturbance during their use of the internet. The data centres used in Attic can choose different upload speeds using the throttle functionality and this obviously affects the download time of data. Since clients are interested in getting data in the fastest possible way, they should clearly choose those data centres with high-speed connections.

No of Data centres	Bandwidth
1	1 MB
2	512 KB
3	256 KB
4	128 KB

Table 7.3: Different Upload Speed

This experiment therefore focuses on how the VASCODE trust framework is used to choose the data centres which have the highest bandwidth connections at any point in time. For this experiment, 10 data centres were used. The upload speed of these data centres were configured using throttling functionality to configure the upload speed of each data centre, as mentioned in Table 7.3. The data centres have continuous availability and they all act honestly. VASCODE-DW was configured with the following parameters (AWS = 0, HWF = 0, SWF = 1) to configure the system to only focus on the speed of the data centres

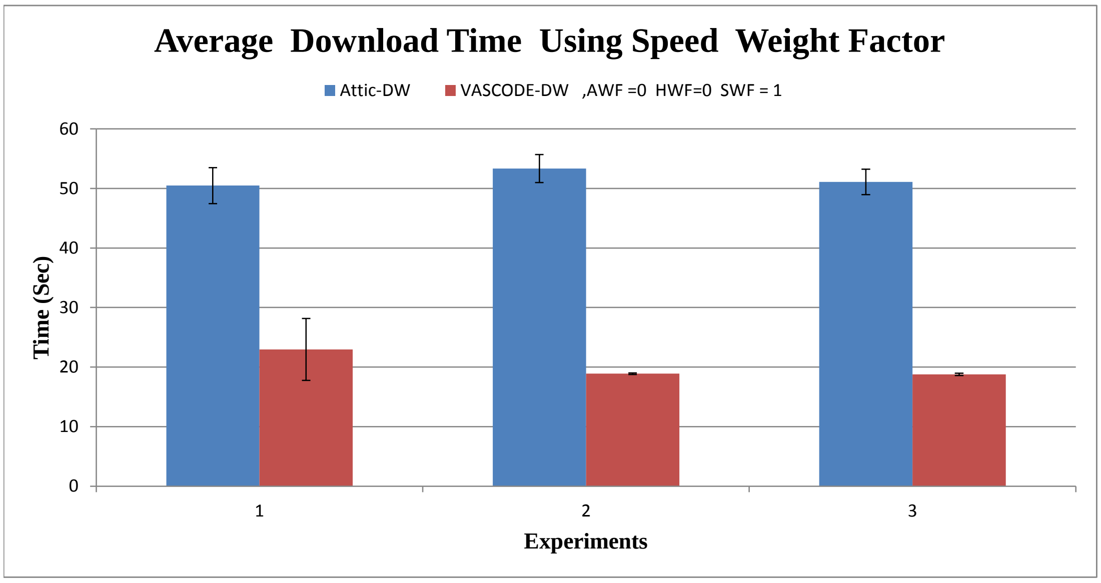


Figure 7.10: Different upload speed

The results in Figure 7.10 show significant improvements by clients making use of the trust framework, compared to the conventional Attic approach, achieving download speeds that are several times faster overall.

7.3.2 Data Throttling and HWF

Seven data centres were used in this experiment. Four of them were honest data centres and the other three were malicious and send corrupted data to their clients. They were available throughout the duration of the experiment and this experiment was repeated three times every time the data centres used the throttle functionality to configure their speed connection to (128 KB, 256 KB and 512 KB). Since the clients were interested in the honesty of these data centres, VASCODE-DW was configured with the parameters (AWS = 0, HWF = 1, SWF = 0) so only the honesty weight factor is set to 1 in this experiment.

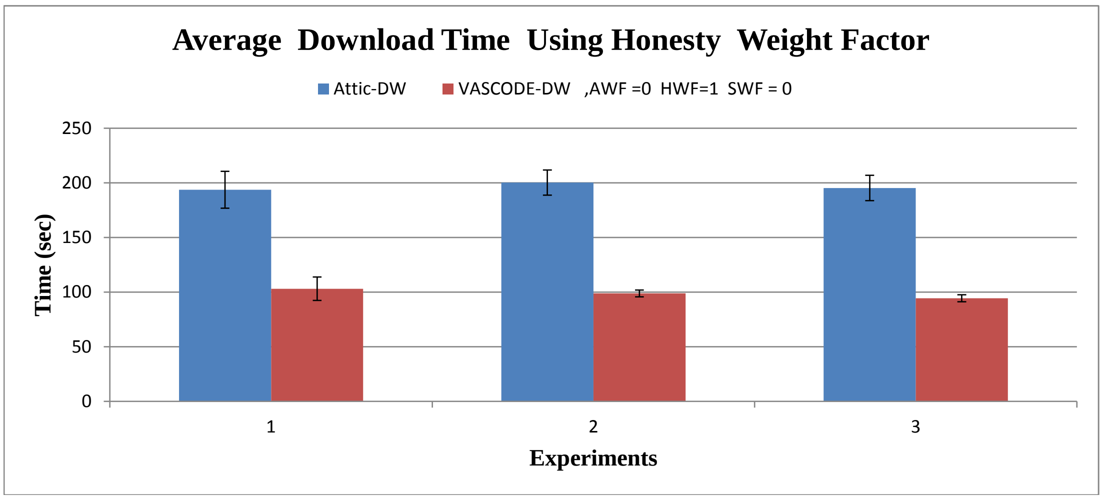


Figure 7.11: 128 KB Upload Speed

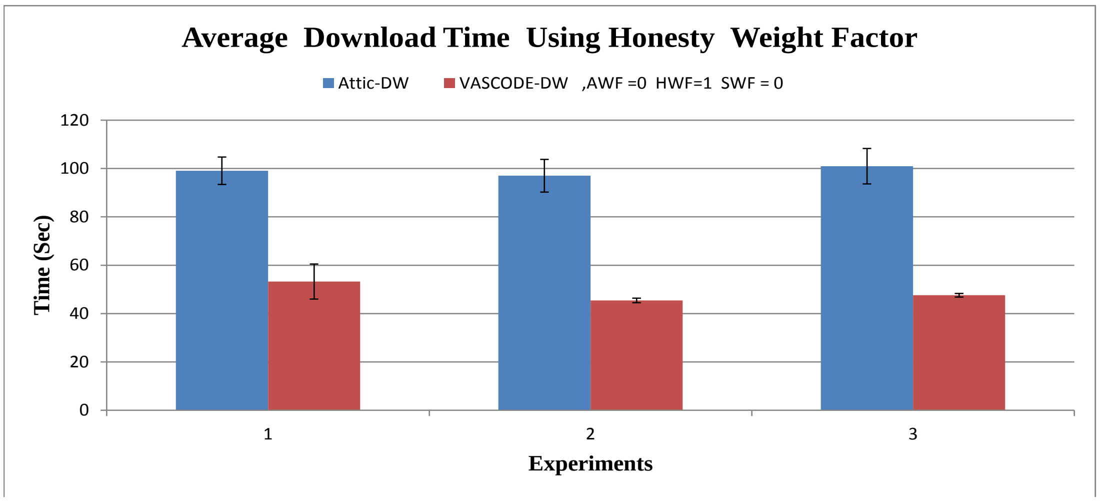


Figure 7.12: 256 KB Upload Speed

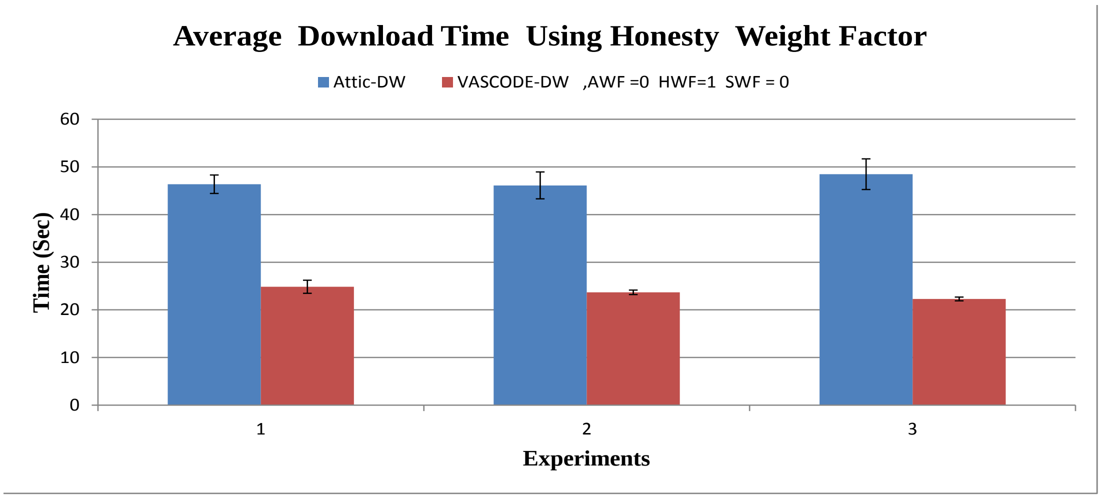


Figure 7.13: 512 KB Upload Speed

Figures 7.11, 7.12and 7.13 show the result of this experiment. It can be observed that VASCODE-DW has significantly better download time because it avoids the malicious data centres, while the ordinary Attic client uses all the data centres and therefore downloads a number of unnecessary corrupted chunks which need to be reloaded, thereby incurring a download overhead. After a short period of convergence, this system performs on average three times better than the standard Attic approach. This experiment shows how the data centres can use the throttle functionality to configure different speeds. In addition, it indicates that this system can learn to avoid malicious peers and dynamically select more trusted peers in the network, which opens up the possibility for such a dynamic data distribution approach.

7.4 Summary

The experiments described in this chapter show how to extend and improve data distribution in Volunteer Computing projects using the VASCODE framework. Furthermore, this chapter shows the performance of VASCODE and its significant improvements with respect to a clients download time when the trust model is used compared to the conventional Attic scheme. Results show that the client with support of the trust framework has a reliable and consistent download time. The experiments also show how weight factors can be used to address various environments and apply client preferences and these weight factors are used to gain the optimum performance of the trust model. Finally, the experiments conducted in this chapter show that using VASCODE can achieve better scalability and performance when the decentralized data centres layer approach with support of trust management system is used.

Chapter 8

Conclusion and Future Work

8.1 Research Summary

Volunteer Computing is a new paradigm of distributed computing where the ordinary computer owners volunteer their computing power and storage capability to scientific projects. The increasing number of Internet connected PCs allows Volunteer Computing to provide more computing power and storage capacity than what can be achieved with supercomputers, clusters and grids. However, the large numbers of participants in Volunteer Computing projects may cause a bottleneck on project servers, which may lead to a point of failure and reduce the projects performance. This thesis presented and investigated a system which provides the necessary mechanisms to improve volunteer computing by extending the data layer in these projects through using P2P techniques and trust management.

P2P systems have become more popular in the last decade through the use of file sharing applications such as Napster, Gnutella and KaZaA. The concept of P2P systems is based on resource sharing in a dynamic environment.

P2P systems are used to access various distributed resources (processing power, storage capability and bandwidth) at the edge of the Internet and these resources are shared between users by direct exchange.

In a P2P system, it may be assumed that a large number of peers may not have interacted before. In these systems there may also be peers that are malicious. For these reasons it is important to select a reliable peer before starting the interaction. A trust indicator, suggesting which peers are more trustworthy than others, would be a useful factor in peer selection. The notion of trust in Computer Science has been borrowed from human society, where we experience and rely on trust in our daily life. Trust is a multifaceted and context-dependent concept and relates to a belief in honesty, competence, and reliability of the trusted person or service.

In Chapter 1, it was hypothesized that file swarming and trust management systems could provide a reliable mechanism to improve data distribution in Volunteer Computing. In this thesis, the author presented a novel approach to incorporate P2P techniques into Volunteer Computing projects in applying trust management to optimize the use of P2P techniques in these projects. This approach adopted a P2P technique to form a decentralized data centres layer which overcomes the limitation of centralized data servers. Since this decentralized data centres layer is formed using the resources of the participants of the Volunteer Computing project, this makes it extend dynamically as the number of participants increases.

The use of a decentralized data centres layer raises the issue of data centre selection. Data centre selection concerns which data centres should be selected and used for downloading data. Trust management can be used to address this concern. Different mechanisms were applied to provide trust in P2P systems and this thesis used a mechanism that is based on the reputation of peers in the network to predict its future behaviour. A trust model was developed that provides its users the necessary information for selecting data centres. In addition, it allows the user to apply their preferences to select the optimum data centres.

Another issue addressed in this thesis was the integration of a P2P technique into Volunteer Computing. In this research, two approaches were investigated. Firstly, the BOINC client code was modified to allow participants to use the decentralized data centres layer instead of using the central server to become a data centre. A potential issue with the previous approach is that a new BOINC client would need to be released for users to take advantage of the VASCODE framework. Therefore, instead, a second approach is to have BOINC client interact with the VASCODE via a proxy. To use this approach, the VASCODE proxy was developed to avoid the modification of the BOINC client.

8.2 Conclusions

In this research, it was aimed to extend the data layer in Volunteer Computing to provide another alternative to clients to obtain data instead of using the central data server. The concept of decentralized data centres layer was introduced in Chapter 4, where it is incorporated into a BOINC project to provide an alternative to BOINC clients to obtain data instead of using the central BOINC server.

Since the primary aim was to show how P2P networks could be utilized to optimize data distribution in Volunteer Computing, a comparison study between the use of decentralized data layer and BOINC central server was conducted in Chapter 4. These experiments revealed a noticeable improvement in download time of data and this will lead to an improvement in the project performance. These experiments also introduced a set of requirements to build an optimum approach in the use of P2P techniques. Based on the evaluation experiments in Chapter 4, it was noticed that some characteristics of the data centre affected the distribution of data, such as its availability, behaviour and its upload speed. Three metrics were used to represent data centre characteristics (***availability, honesty, speed***). The concept of trust was adopted to improve the use of P2P networks in Volunteer Computing projects.

In section 5.3.2.1 a computational trust model was proposed based on these three metrics. This model uses the reputation of these data centres, obtained by their clients feedback, to calculate their trust value and the use of trust management was considered to differentiate between data centres.

In chapters 5 and 6, the VASCODE framework was designed and implemented that uses the Attic File System and BOINC middleware for using P2P data sharing within Volunteer Computing projects. This framework makes use of trust management to allow clients to select the data centre. It allows decentralized data centres in Volunteer Computing to be efficient and extends dynamically at run time, giving them the ability to serve a large number of data requestors.

Moreover, the VASCODE framework has been developed to provide a tool to the research community that uses BOINC middleware to build volunteer computing projects and apply the decentralized data centres layer within these projects and utilize trust management.

VASCODE uses different components to allow Volunteer Computing projects participants to become data centres and distribute data. In addition, it utilizes the trust framework to provide the necessary data to clients to select the optimum data centres for downloading data.

Chapter 7 presented a performance evaluation of the VASCODE framework in terms of using the trust value of data centres to select the optimum data centres, which consequently affects the total project performance.

In this chapter, a set of experiments were conducted and these used various scenarios to represent the decentralized data centres layer in different environments. Results showed that clients with the support of a trust framework have reliable and consistent download times. In this context, reliable implies that when a client uses trust values to select the best data centres, it will get an improved download time each time it uses them to download data. This provides significant gains for the BOINC project as a whole by reducing download time and therefore increasing the throughput of results.

The results show also that the clients with the support of a trust framework, compared with ordinary clients, can avoid the malicious behaviour of some data centres by avoiding selecting them.

Performance analysis demonstrated that the proposed approach in this thesis can achieve better scalability and performance compared to the central server approach that is used in BOINC projects.

8.3 Future Work

This research demonstrates the feasibility of using P2P techniques and trust in Volunteer Computing. The promising outcomes using this approach opens several directions for future research:

- **Investigating the use of other P2P networks to build the decentralized data layer**

  One promising extension of this research is to consider how to support different P2P networks. The Attic File System, used in this research, is a centralized P2P system, where peers register their data files to a central data lookup server, so other peers can download it. Future research could explore integrating decentralized P2P networks within Volunteer Computing. Decentralized P2P networks were developed to avoid the use of a central look up server as it is a point of failure but they have less performance due to the exchange of messages before downloading data. It would be interesting to study the performance of Volunteer Computing projects when they use this type of P2P network.

- **Enhancing the trust framework**
  The trust model developed in this thesis is based on client feedback, that is, the model aggregates clients feedback then uses the Beta distribution to calculate the trust value. This method involves the use of the history of each data centre. To avoid using historic data to calculate the trust value, other trust systems could be investigated, such as PolicyMake [41]. In addition, other trust models such as EigenTrust [70] and PowerTrust [112] could be investigated to improve the trust framework.

- **Developing a theoretical model for Attic file system performance**
  The performance of the Attic file system is based on the number of data centres and the chunk size. It would be interesting to develop a theoretical model that provides the system administrator the number of data centres that should form the decentralized data layer. Furthermore, it assumes the best chunk size of the published data which consequently decides the number of chunks a client should download. This theoretical model could

be used to tune the Attic file system and optimize its performance.

- **Optimization of trust metrics in large scale networks** The three metrics used in the trust model define the attributes of the data centre, such as its upload speed, availability and behaviour. When the data centre client reports its interaction with the data centre, it includes whether the data centre was available or not, whether the upload speed was satisfied and if the downloaded data was as expected. In large scale networks, many factors impact on the data centre attributes. For example, the data centre provides a considerable bandwidth for uploading data, but because it serves a number of requests, this causes the clients to be unsatisfied with the upload speed. The clients will report unsatisfied upload speed on their feedback even if the data centre has provided considerable bandwidth for serving data requests. This affects the rate of the data centre during the selection of this data centre by other clients. Therefore, it is desirable to study all factors that affect data centre performance, not just using the clients feedback.

Appendix A

Average Download Time and Standard Error

Experiments	Average Download Tim (Sec)		Standard Error	
	ATTIC-DW	VASCODE-DW	ATTIC-DW	VASCODE-DW
1	19.04508	15.30233	1.472914	1.101168
2	15.26283	14.71117	0.931645	0.911604
3	19.22108	15.93375	1.970007	1.116071
4	24.07683	16.10325	4.365287	2.722969

Table A.1: Standard Error of experiment Data centres availability in the first four hour period

Experiments	Average Download Tim (Sec)		Standard Error	
	ATTIC-DW	VASCODE-DW	ATTIC-DW	VASCODE-DW
1	0.662121	0.295045	0.662121	0.295045
2	0.733194	0.280766	0.733194	0.280766
3	1.254447	0.278048	1.254447	0.278048
4	0.791529	0.252666	0.791529	0.252666

Table A.2: Standard Error of experiment Effect of Data centres availability in the second t four hour period

Experiments	Average Download Tim (Sec)		Standard Error	
	ATTIC-DW	VASCODE-DW	ATTIC-DW	VASCODE-DW
1	37.13292	15.72058	2.801357	1.506626
2	37.37925	14.80817	2.548926	0.664261
3	36.933 15	.42683	1.63865	0.628409
4	35.80608	14.91042	2.626752	0.560343

Table A.3: Standard Error of experiment Data centres Honesty

Experiments	Average Download Tim (Sec)		Standard Error	
	ATTIC-DW	VASCODE-DW	ATTIC-DW	VASCODE-DW
1	9.113667	1.748167	0.64174	0.512152
2	8.26275	1.214917	0.624947	0.010422
3	8.77825	1.252333	0.422131	0.02659
4	8.958583	1.194917	0.779143	0.006215

Table A.4: Standard Error of experiment Effect of Data centres Speed

Experiments	Average Download Tim (Sec)		Standard Error	
	ATTIC-DW	VASCODE-DW	ATTIC-DW	VASCODE-DW
1	20.00317	7.831667	3.72237	2.761008
2	16.33108	7.068917	2.761112	3.228559
3	16.79283	7.499917	2.553147	3.315833
4	26.07425	13.77175	3.189383	4.174713

Table A.5: Standard Error of experiment different behaviours of data centres

Experiments	Average Download Tim (Sec)		Standard Error	
	ATTIC-DW	VASCODE-DW	ATTIC-DW	VASCODE-DW
1	50.4689	22.9582	3.03840906	5.19510238
2	53.3377	18.8797	2.34148943	0.13468251
3	51.084	18.7644	2.13906154	0.17955403

Table A.6: Standard Error of experiment Data Throttling and SWF

Experiments	Average Download Tim (Sec)		Standard Error	
	ATTIC-DW	VASCODE-DW	ATTIC-DW	VASCODE-DW
1	193.7029	103.0719	16.78993	10.76241
2	200.1928	98.802	11.50832	3.050898
3	195.3004	94.4454	11.57912	3.194959

Table A.7: Standard Error of experiment Data Throttling and HWF (128 KB upload speed)

Experiments	Average Download Tim (Sec)		Standard Error	
	ATTIC-DW	VASCODE-DW	ATTIC-DW	VASCODE-DW
1	99.084	3 53.268	5.641064	7.2546
2	97.0235	45.4499	6.787813	0.931566
3	100.942	47.5678	7.33623	0.767633

Table A.8: Standard Error of experiment Data Throttling and HWF (256 KB upload speed)

Experiments	Average Download Tim (Sec)		Standard Error	
	ATTIC-DW	VASCODE-DW	ATTIC-DW	VASCODE-DW
1	46.1312	23.6826	1.950102	1.382111
2	48.4787	22.3005	2.839135	0.47572
3	46.3727	24.8445	3.209677	0.39561

Table A.9: Standard Error of experiment Data Throttling and HWF (512 KB upload speed)

Bibliography

[1] The annotated gnutella protocol specification v0.4 document revision 1.6. http://rfc-gnutella.sourceforge.net/developer/stable/. 33, 36

[2] The arecibo observatory. `http://www.naic.edu/`. 21

[3] Attic file system. `http://www.atticfs.org/`. 4, 40

[4] Bittorrent. `http://www.bittorrent.com/`. 4

[5] BOINC projects list , howpublished=`http://boinc.berkeley.edu/wiki/project_list`,. 3

[6] Boinc projects statics. `http://boincstats.com/`. last accessed May 2012. 16

[7] Climateprediction.net project. `http://climateprediction.net/`. last accessed June 2012. 21

[8] The datagrid project. `http://eu-datagrid.web.cern.ch/eu-datagrid/`. 32

[9] Distributed.net project. `http://www.distributed.net/Main_Page`. last accessed June 2012. 16

[10] Einstein@home project. `http://einstein.phys.uwm.edu/`. last accessed June 2012. 14, 21

[11] emule : Peer-to-peer file sharing application. `http://www.emule-project.net/`. 33

[12] European Desktop Grid Initiative (EDGI). http://edgi-project.eu/. 50

[13] fastrack. `http://developer.berlios.de/projects/gift-fasttrack/`. 35

[14] Freenet. the freenet project. available at. `https://freenetproject.org/`. 36, 51

[15] The globus toolkit. `http://www.globus.org/toolkit/`. 32

[16] Great internet mesenne prome search (gmips) ptojects. `http://www.mersenne.org/`. last accessed June 2012. 16

[17] Internet usage statistics. `http://www.internetworldstats.com/stats.htm`. last accessed May 2012. 16

[18] Kazaa. `http://www.kazaa.com/`. 4, 33

[19] Ligo, the laser interferometer gravitational-wave observatory. `http://www.ligo.caltech.edu/`. 21

[20] morpheus. `http://morpheus.en.softonic.com/`. 35

[21] Napster. `http://www.napster.com`. 4, 33, 34

[22] The oxford dictionary. `http://oxforddictionaries.com/`. 43

[23] The poco c++ libraries. http://pocoproject.org/. 102

[24] Primegrid project. http://www.primegrid.com/. last accessed June 2012. 21

[25] Rosseta@home project. `http://boinc.bakerlab.org/rosetta/`. last accessed June 2012. 14

[26] Seti@home project, howpublished=`http://setiathome.berkeley.edu/`,. 16

[27] The stanford linear accelerator center (slac). `http://www.slac.stanford.edu/`. 32

[28] the large hadron collider (lhc) project. `http://lhc.web.cern.ch/lhc/`. 32

[29] A taxonomy and survey of grid resource management systems for distributed computing. *Softw. Pract. Exper.*, 32(2):135–164, February 2002. 31

[30] *Grid Computing in Research and Education.* IBM reedbooks, 2005. 31

[31] K. Fukui [1] B. Jacob, M. Brown and N. Trivedi. Introduction to grid computing. Technical report, IBM International Technical Support Organization, 2005. `http://www.redbooks.ibm.com/redbooks/pdfs/sg246778.pdf`. 2, 31

[32] Alfarez Abdul-Rahman and Stephen Hailes. Using recommendations for managing trust in distributed systems. In *IN PROC. OF IEEE MALAYSIA*

INTERNATIONAL CONFERENCE ON COMMUNICATION (MICC97), KUALA LUMPUR, 1997. 81

[33] Alfarez Abdul-Rahman and Stephen Hailes. Supporting trust in virtual communities. In *Proceedings of the 33rd Hawaii International Conference on System Sciences-Volume 6 - Volume 6*, HICSS '00, pages 6007–, Washington, DC, USA, 2000. IEEE Computer Society. 43

[34] A. Agarwal, M. Ahmed, A. Berman, B. L. Caron, A. Charbonneau, D. Deatrich, R. Desmarais, A. Dimopoulos, I. Gable, L. S. Groer, R. Haria, R. Impey, L. Klektau, C. Lindsay, G. Mateescu, Q. Matthews, A. Norton, W. Podaima, D. Quesnel, R. Simmonds, R. J. Sobie, B. St. Arnaud, C. Usher, D. C. Vanderster, M. Vetterli, R. Walker, and M. Yuen. Gridx1: A canadian computational grid. *Future Gener. Comput. Syst.*, 23(5):680–687, June 2007. 32

[35] David P. Anderson. Public computing: Reconnecting people to science. Madrid, Spain, Novmber 2003. 15

[36] David P. Anderson. Boinc: A system for public-resource computing and storage. In *5th IEEE/ACM International Workshop on Grid Computing*, pages 4–10, 2004. 3, 17

[37] David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer. Seti@home: an experiment in public-resource computing. *Commun. ACM*, 45(11):56–61, November 2002. 14, 20

[38] David P. Anderson and Gilles Fedak. The computational and storage potential of volunteer computing. pages 73–80, 2006. 16

[39] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv.*, 36(4):335–371, December 2004. 33, 34

[40] B. Beverly Yang and H. Garcia-Molina. Designing a super-peer network. In *Data Engineering, 2003. Proceedings. 19th International Conference on*, pages 49 – 60, march 2003. 35

[41] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *In Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 164–173. IEEE Computer Society Press, 1996. 44, 45, 150

[42] Sonja Buchegger and Jean-Yves Le Boudec. A robust reputation system for p2p and mobile ad-hoc networks. 2004. 84

[43] V. Cahill, E. Gray, J.-M. Seigneur, C.D. Jensen, Yong Chen, B. Shand, N. Dimmock, A. Twigg, J. Bacon, C. English, W. Wagealla, S. Terzis, P. Nixon, G. Di Marzo Serugendo, C. Bryce, M. Carbone, K. Krukow, and M. Nielson. Using trust for secure collaboration in uncertain environments. *Pervasive Computing, IEEE*, 2(3):52 – 61, july-sept. 2003. 84

[44] C. Castelfranchi and R. Falcone. Principles of trust for mas: Cognitive anatomy, social importance, and quantification. In *Proceedings of the 3rd International Conference on Multi Agent Systems*, ICMAS '98, pages 72–, Washington, DC, USA, 1998. IEEE Computer Society. 82

[45] Gilles Fedak Cecile, Gilles Fedak, Cecile Germain, and Vincent Neri. Xtremweb : A generic global computing system. In *In Proceedings of the*

IEEE International Symposium on Cluster Computing and the Grid (CC-GRID01, pages 582–587, 2001. 3, 22

[46] Yu;Wei Chan, Tsung;Hsuan Ho, Po;Chi Shih, and Yeh;Ching Chung. Malugo a peer to peer storage system. *Int. J. Ad Hoc Ubiquitous Comput.*, 5(4):209–218, May 2010. 39

[47] Ann Chervenak, Ian Foster, Carl Kesselman, Charles Salisbury, and Steven Tuecke. The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets. *JOURNAL OF NETWORK AND COMPUTER APPLICATIONS*, 23:187–200, 1999. 32

[48] Andrew A. Chien. Architecture of the entropia distributed computing system andrew a. chien. In *International Parallel and Distributed Processing Symposium*, pages 15–19, 2002. 24

[49] Bram Cohen. Incentives build robustness in bittorrent. *In Proceedings of the 1st Workshop on Economics of Peer-to-Peer Systems*, 2003. 40, 50, 51

[50] Bled Electronic Commerce, Audun Jsang, and Roslan Ismail. The beta reputation system. In *In Proceedings of the 15th Bled Electronic Commerce Conference*, 2002. 84

[51] Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Wide-area cooperative storage with cfs. In *Proceedings of the eighteenth ACM symposium on Operating systems principles*, SOSP '01, pages 202–215, New York, NY, USA, 2001. ACM. 38

[52] Frank Dabek, M. Frans Kaashoek, and C. Smith. A distributed hash table. Technical report, 2005. 37

[53] Ernesto Damiani, De Capitani di Vimercati, Stefano Paraboschi, Pierangela Samarati, and Fabio Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *Proceedings of the 9th ACM conference on Computer and communications security*, CCS '02, pages 207–216, New York, NY, USA, 2002. ACM. 42

[54] Choon Hoong Ding, Sarana Nutanong, and Rajkumar Buyya. Peer-to-peer networks for content sharing. Technical report, Laboratory, University of Melbourne, Australia, 2003. 35

[55] EDGeS Project. http://www.edges-grid.eu/. 50

[56] AbdelHamid Elwaer, Andrew Harrison, Ian Kelley, and Ian Taylor. Attic: A case study for distributing data in boinc projects. In *Proceedings of the 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum*, IPDPSW '11, pages 1863–1870, Washington, DC, USA, 2011. IEEE Computer Society. 40

[57] I. Foster. The grid: A new infrastructure for 21st century science. *Grid Computing: Making the Global Infrastructure a*, 2003. 2, 31

[58] Ian Foster. What is the grid? - a three point checklist. *GRIDtoday*, 1(6), July 2002. 31

[59] Ian Foster and Carl Kesselman. Globus: A metacomputing infrastructure

toolkit. *International Journal of Supercomputer Applications*, 11:115–128, 1996. 32

[60] Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *Int. J. High Perform. Comput. Appl.*, 15(3):200–222, August 2001. 31

[61] Diego Gambetta. Can we trust trust? In *Trust: Making and Breaking Cooperative Relations*, pages 213–237. Basil Blackwell, 1988. 82

[62] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis, Second Edition (Chapman & Hall/CRC Texts in Statistical Science).* Chapman and Hall/CRC, 2 edition, July 2003. 84

[63] Tyrone Grandison and Morris Sloman. A survey of trust in internet applications. *Communications Surveys Tutorials, IEEE*, 3(4):2 –16, quarter 2000. 43

[64] Minaxi Gupta, Paul Judge, and Mostafa Ammar. A reputation system for peer-to-peer networks. In *Proceedings of the 13th international workshop on Network and operating systems support for digital audio and video*, NOSSDAV '03, pages 144–152, New York, NY, USA, 2003. ACM. 46

[65] Ragib Hasan, Zahid Anwar, William Yurcik, Larry Brumbaugh, and Roy Campbell. A survey of peer-to-peer storage techniques for distributed file systems. In *Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume II - Volume 02*, ITCC '05, pages 205–213, Washington, DC, USA, 2005. IEEE Computer Society. 39

[66] Audun Josang and Jochen Haller. Dirichlet reputation systems. In *Proceedings of the The Second International Conference on Availability, Reliability and Security*, ARES '07, pages 112–119, Washington, DC, USA, 2007. IEEE Computer Society. 84

[67] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decis. Support Syst.*, 43(2):618–644, March 2007. 42

[68] Lalana Kagal, Scott Cost, Timothy Finin, and Yun Peng. A framework for distributed trust management. In *In To appear in proceedings of IJCAI-01 Workshop on Autonomy, Delegation and Control*, 2001. 44

[69] S Kamvar. Eigenrep: Reputation management in p2p networks. Technical report, Stanford University, 2002. 42

[70] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web*, WWW '03, pages 640–651, New York, NY, USA, 2003. ACM. 46, 81, 150

[71] David Karger, Eric Lehman, Tom Leighton, Rina Panigrahy, Matthew Levine, and Daniel Lewin. Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the world wide web. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, STOC '97, pages 654–663, New York, NY, USA, 1997. ACM. 38

[72] Ian Kelley and Ian Taylor. A peer-to-peer architecture for data-intensive cycle sharing. In *Proceedings of the first international workshop on Network-aware data management*, NDM '11, pages 65–72, New York, NY, USA, 2011. ACM. 50

[73] Carl Kesselman and Ian Foster. *The Grid: Blueprint for a New Computing Infrastructure.* Morgan Kaufmann Publishers, November 1998. 31

[74] Eleni Koutrouli and Aphrodite Tsalgatidou. Reputation-based trust systems for p2p applications: design issues and comparison framework. In *Proceedings of the Third international conference on Trust, Privacy, and Security in Digital Business*, TrustBus'06, pages 152–161, Berlin, Heidelberg, 2006. Springer-Verlag. 42

[75] John Kubiatowicz, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaton, Dennis Geels, Ramakrishan Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, Chris Wells, and Ben Zhao. Oceanstore: an architecture for global-scale persistent storage. *SIGPLAN Not.*, 35(11):190–201, November 2000. 39, 51

[76] Ninghui Li, John C. Mitchell, and William H. Winsborough. Design of a role-based trust-management framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, SP '02, pages 114–, Washington, DC, USA, 2002. IEEE Computer Society. 44

[77] J. Liang, R. Kumar, and K. Ross. The kazaa overlay: A measurement study. 2004. 35

[78] Michael Litzkow, Miron Livny, and Matthew Mutka. Condor - a hunter of idle workstations. In *Proceedings of the 8th International Conference of Distributed Computing Systems*, June 1988. 3, 22

[79] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and replication in unstructured peer-to-peer networks. In *Proceedings of the 16th international conference on Supercomputing*, ICS '02, pages 84–95, New York, NY, USA, 2002. ACM. 36

[80] S. Marsh. Formalising trust as a computational concept, 1994. 44, 47

[81] Petar Maymounkov and David Mazires. Kademlia: A peer-to-peer information system based on the xor metric. pages 53–65, 2002. 37

[82] G.E. Moore. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82 –85, jan. 1998. 13

[83] Ahmed MEDDAHI Mourad AMAD and Djamil ASSANI. Peer to peer networks management survey. *3IJCSI International Journal of Computer Science*, 2012. 33

[84] L. Mui, M. Mohtashemi, and A. Halberstadt. A computational model of trust and reputation for e-businesses. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)-Volume 7 - Volume 7*, HICSS '02, pages 188–, Washington, DC, USA, 2002. IEEE Computer Society. 43

[85] L. Mui, M. Mohtashemi, and A. Halberstadt. A computational model of trust and reputation for e-businesses. In *Proceedings of the 35th Annual*

Hawaii International Conference on System Sciences (HICSS'02)-Volume 7 - Volume 7, HICSS '02, pages 188–, Washington, DC, USA, 2002. IEEE Computer Society. 84

[86] Andy Oram, editor. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2001. 33

[87] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120. 81

[88] Pawel Plaszczak and Richard Wellner. *Grid Computing: The Savvy Manager's Guide*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005. 31

[89] B. Pourebrahimi, K.L.M. Bertels, and S. Vassiliadis. A survey of peer-to-peer networks. In *Proc. 16th Annual Workshop on Circuits, Systems and Signal Processing*, Veldhoven, The Netherlands, November 2005. 36

[90] Josep M. Pujol, Ramon Sangüesa, and Jordi Delgado. Extracting reputation in multi agent systems by means of social network topology. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, AAMAS '02, pages 467–474, New York, NY, USA, 2002. ACM. 47

[91] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In *Proceedings of the*

2001 conference on Applications, technologies, architectures, and protocols for computer communications, SIGCOMM '01, pages 161–172, New York, NY, USA, 2001. ACM. 37

[92] Paul Resnick, Ko Kuwabara, Richard Zeckhauser, and Eric Friedman. Reputation systems. *Commun. ACM*, 43(12):45–48, December 2000. 42

[93] Jordi Sabater and Carles Sierra. Regret: A reputation model for gregarious societies. In *Proceedings of the fifth international conference on Autonomous agents*, AGENTS '01, pages 194–195, New York, NY, USA, 2001. ACM. 43, 80

[94] Jordi Sabater and Carles Sierra. Social regret, a reputation model based on social relations. *SIGecom Exch.*, 3(1):44–56, December 2001. 47

[95] Jordi Sabater and Carles Sierra. Reputation and social network analysis in multi-agent systems. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, AAMAS '02, pages 475–482, New York, NY, USA, 2002. ACM. 47

[96] Aameek Singh and Ling Liu. Trustme: Anonymous management of trust relationships in decentralized p2p systems. In *Proceedings of the 3rd International Conference on Peer-to-Peer Computing*, P2P '03, pages 142–, Washington, DC, USA, 2003. IEEE Computer Society. 42

[97] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 conference on Applications, tech-*

nologies, architectures, and protocols for computer communications, SIGCOMM '01, pages 149–160, New York, NY, USA, 2001. ACM. 37

[98] Andrew S. Tanenbaum and Maarten van Steen. *Distributed Systems: Principles and Paradigms (2nd Edition).* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006. 13

[99] Ian J. Taylor and Andrew Harrison. *From P2P and Grids to Services on the Web: Evolving Distributed Communities.* Springer Publishing Company, Incorporated, 2nd edition, 2009. 34

[100] W. T. Luke Teacy, Jigar Patel, Nicholas R. Jennings, and Michael Luck. Travos: Trust and reputation in the context of inaccurate information sources. *Journal of Autonomous Agents and Multi-Agent Systems*, 12:2006, 2006. 84

[101] Joseph D. Touch. Overlay networks. *Computer Networks*, 36(2/3):115–116, 2001. 37

[102] Dimitrios Tsoumakos and Nick Roussopoulos. Analysis and comparison of p2p search methods. In *Proceedings of the 1st international conference on Scalable information systems*, InfoScale '06, New York, NY, USA, 2006. ACM. 36

[103] Honghao Wang, Yingwu Zhu, and Yiming Hu. To unify structured and unstructured p2p systems. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Papers - Volume 01*, IPDPS '05, pages 104.1–, Washington, DC, USA, 2005. IEEE Computer Society. 37

[104] Mark Witkowski, Alexander Artikis, and Jeremy Pitt. Experiments in building experiential trust in a society of objective-trust based agents. In *Proceedings of the workshop on Deception, Fraud, and Trust in Agent Societies held during the Autonomous Agents Conference: Trust in Cyber-societies, Integrating the Human and Artificial Perspectives*, pages 111–132, London, UK, UK, 2001. Springer-Verlag. 80

[105] Li Xiong and Ling Liu. Peertrust: Supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Trans. on Knowl. and Data Eng.*, 16(7):843–857, July 2004. 42

[106] Beverly Yang and Hector Garcia-Molina. Comparing hybrid peer-to-peer systems. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pages 561–570, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. 34

[107] Walt Teh-Ming Yao. Fidelis: a policy-driven trust management framework. In *Proceedings of the 1st international conference on Trust management*, iTrust'03, pages 301–317, Berlin, Heidelberg, 2003. Springer-Verlag. 44

[108] Bin Yu and Munindar P. Singh. A social mechanism of reputation management in electronic communities. In *Proceedings of the 4th International Workshop on Cooperative Information Agents IV, The Future of Information Agents in Cyberspace*, CIA '00, pages 154–165, London, UK, UK, 2000. Springer-Verlag. 47

[109] Lan Yu, Willy Susilo, and Rei Safavi-Naini. X2bt trusted reputation system: a robust mechanism for p2p networks. In *Proceedings of the 5th inter-*

national conference on Cryptology and Network Security, CANS'06, pages 364–380, Berlin, Heidelberg, 2006. Springer-Verlag. 42

[110] Ting Yu, Marianne Winslett, and Kent E. Seamons. Interoperable strategies in automated trust negotiation. In *Proceedings of the 8th ACM conference on Computer and Communications Security*, CCS '01, pages 146–155, New York, NY, USA, 2001. ACM. 44

[111] Ben Y. Zhao, John D. Kubiatowicz, and Anthony D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and. Technical report, Berkeley, CA, USA, 2001. 37

[112] Runfang Zhou and Kai Hwang. Powertrust: A robust and scalable reputation system for trusted peer-to-peer computing. *IEEE Trans. Parallel Distrib. Syst.*, 18(4):460–473, April 2007. 47, 81, 150