Building a Scalable Virtual Community on Commodity Hardware and Open Source Software

by

Andreas Alberts

A dissertation submitted in partial fulfillment
of the requirements for the degree

MAGISTER INGENERIAE

in

COMPUTER AND ELECTRONIC ENGINEERING

in the

FACULTY OF ENGINEERING

at the

NORTH WEST UNIVERSITY

SUPERVISOR: Prof A.S.J. Helberg

November 2008

Abstract

The information era has brought upon us waves of change that brings affordable technology within the reach of the average person. Computers connected to the Internet, part of our daily living, have led to the formation of online communities.

In the spirit of communal efforts, a community cannot be controlled or managed into a specific form or direction. A community has a need to concentrate its efforts towards a common goal or vision, therefore sufficiently nonrestrictive infrastructure is needed to enable the community members to contribute towards their goal.

We design and build infrastructure to support a virtual community, according to the needs of the community. Community members can easily locate and exchange files among each other, interact in private and public chat rooms by means of instant text messages, as well as make announcements and participate in group discussions in a Web based environment. Additional needs are identified and tended to by means of various value adding services. We also formulate a management strategy to lead the community towards self-sustenance.

Key Terms

Virtual Community, Open Source, Direct Connect, Peer to Peer, Offline Search

Opsomming

Die informasie era het golwe van verandering meegebring, wat bekostigbare tegnologie binne bereik van die gemiddelde mens bring. Internetgekoppelde rekenaars wat deel is van ons daaglikse bestaan het tot die ontstaan van aanlyn gemeenskappe gelei.

In die gees van gemeenskaplike pogings, kan 'n gemeenskap nie beheer of bestuur word om 'n spesifieke vorm aan te neem of 'n spesifieke rigting in te slaan nie. 'n Gemeenskap wil hulle gemeenskaplike doel of visie uitleef, dus is vrylik toeganklike infrastruktuur nodig om die lede van die gemeenskap in staat te stel om hulle insette tot die gesamentlike doel by te dra.

Ons ontwerp en bou infrastruktuur om 'n virtuele gemeenskap te ondersteun, aan die hand van die behoeftes van die gemeenskap. Gemeenskapslede kan maklik datalêers tussen mekaar opspoor en uitruil, in publieke en private kuierkamers kommunikeer met behulp van teksboodskappe, asook aankondingings maak en aan groepsbesprekings deelneem in 'n Webgebaseerde omgewing. Addisionele behoeftes word geïdentifiseer en aangespreek met verskeie waardetoevoegende dienste. Ons formuleer ook 'n bestuurstrategie om die gemeenskap te lei na selfonderhouding.

Sleutelterme

Virtual Community, Open Source, Direct Connect, Peer to Peer, Offline Search

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

Virtual communities exist throughout cyberspace, enabling users to exchange information and ideas. A common goal is the key that draws like-minded individuals to specific communities. The nature of the community infrastructure is determined by the activities that need to be supported.

Large corporate networks have a tendency to isolate groups sharing a common interest from the Internet, due to access restrictions and policies. Isolated communities on large corporate networks have a need to collaborate and to exchange information and ideas internally.

We present the evolutionary development of such a virtual community in this dissertation. We proceed to describe the problem space in terms of requirements, limitations, and constants.

1.2 Problem Statement

Infrastructure supporting a scalable, managable virtual community is needed. Our target community has diverse needs that can only be addressed by continuously developing and enhancing our software and systems.

Therefore, we opt from the beginning for a community-driven framework in which each user plays a unique role. We only consider open source software in our community-driven spirit and approach. Using open building blocks, we can build a system that meets the needs and scales to the dimensions of our community by integrating and extending

existing technology.

In order to exchange information and ideas, an environment that supports the typical activities enhancing collaboration is needed. Information is presented in various preferred formats. Plain text is the first and foremost format, used in real time text messaging and Web forums. Information encoded in various binary formats can exchange hands on a file sharing network.

We therefore need three major components, namely an instant messaging collaboration environment, a Web based forum, and a file sharing platform. Additionally, in order to enhance productivity and efficiency, we identified the need for improved search functionality and cooperation incentives on the file sharing platform.

Various platforms and techniques that support these goals and activities already exist, as we will discuss later from the literature. We will add value by strategically implementing, integrating, enhancing, and extending existing technology into a system tailor-made for our community.

The scope of our work will include building a peer to peer network and a Web based forum on a robust, secure platform that enables quick disaster recovery. We will integrate and streamline these services to enable easy user registration and the use of a single account system wide. We will investigate and implement a mechanism to stimulate participation on the peer to peer network, as well as enhance collaboration by providing a mechanism to enable interest groups to communicate in real time. We will implement an advanced search system that is able to perform offline searches on the peer to peer network and automatically notify users of new search results. We will design and implement a robot to automatically crawl the peer to peer network and keep the offline search index up to date.

In practice, many peer to peer networks and Web forums exist to form communities similar to ours. The level of integration we are to perform, however, is rarely seen. Our unique contribution will be the design, implementation and integration of the offline search and notification engine.

1.3 Looking Forward

We discuss the evolutionary development of our virtual community in this dissertation, effectively being a case study.

In the next chapter, we analyze our problem space in more detail, followed by a literature study to provide background information needed for the rest of the dissertation.

We mainly focus on the technical aspects of building community infrastructure in the chapters that follow. We present our design choices, discuss our implementation thereof, and present measured results. We conclude our work by reflecting upon the success of the implementation and provide a view on the future.

2 Requirements and Constants

2.1 Methodology

Building a community-driven infrastructure is inherently an evolutionary process. The infrastructure is built and extended as the community grows and develops. Therefore, it is difficult to follow a traditional top-down engineering approach to design, build, and provision infrastructure that exactly matches the needs of our virtual community. Continuous feedback from the existing community along with guidance from the management team identifies needs and sets the community on a strategic course.

The work presented in this dissertation is the result of the evolution of our virtual community. We initially set out to design and build a hosting platform, a peer to peer network, and a Web forum in order to facilitate community activities. Our literature study and technology evaluation chapters focus on our choices in this regard.

With an established community moving in a strategic direction, we set out to enhance the community infrastructure to improve productivity and enhance user experience.

Our implementation chapter focuses on the work we have done to build our infrastructure, with a majority focus on our own unique work we have done.

Measured results are presented and analyzed in our results chapter, followed by a concluding discussion.

We proceed to present basic requirements and goals we consider key to a successful community deployment.

2.2 Service Requirements

Ultimately, a virtual community is needed where an easily accessible collection of information and ideas are available and where individuals can easily communicate and collaborate. Successfully building such a community will involve successfully integrating several building blocks.

Firstly, a peer to peer network is needed for the exchange of data between users. Several peer to peer applications and ideologies exist today that are very advanced and complex. An important requirement for successfully building a virtual community on top of a peer to peer technology is that it must be simple to use. The client that users use to connect to the network must be freely available and free of spyware and adware. Ideally, a selection of compatible client programs must be available and supported and maintained entirely by their respective authors.

A peer to peer network alone is not sufficient to guarantee the success of a virtual community. An element that contributes towards the conservation of information is needed. Information should be available to any user at any time, and not dependent on the availability of a specific community resource. A Web-based forum or bulletin board system (BBS) where users can post information and discuss ideas is needed, which can act as an information vault and guarantee the longevity of information. The entire community can access information and participate in discussions in an asynchronous fashion on Web-based forums.

Users sometimes need a more direct form of communication than posting messages on a bulletin board. Real time text messaging in private, where two parties are involved, and in public, where the entire community is involved, is a well known and widely used collaboration medium. The success of effective collaboration can be amplified by providing customizable interest groups, which users can utilize to send announcements and important information in real time to other users who joined a specific group.

On a peer to peer network, all available content is not always available online, but dependent on the availability of the specific peers that store the information. Real time searches on a peer to peer network will only yield results of currently available files. A searchable index of all files shared over a relatively long time window can assist users in finding and enqueuing scarce or otherwise hard to find files, so that it automatically

downloads whenever the resource becomes available again. A method of notifying a user when an unavailable file matching specific search criteria first becomes available would also contribute towards the success of a specific community deployment.

2.3 Management and Control Requirements

Imagine a community consisting of knowledge hungry individuals. Each individual with a unique character and personality. Each member contributing new ideas and philosophies, each member asking unique questions derived from their own experience within the community. Imagine this community supported by a flawless infrastructure...

The mental picture of such a community seems to be perfect, but it lacks one core element, the element that will take this community from being successful to becoming self-sustained. That element is guidance. Administration and management of a community is the invisible hand that regulates socially sensitive aspects, be it ethical or moral. Its the hand that protects individualism in terms of culture and religion. Its the hand that guides the community in the direction of pure collaboration.

Strategic implementation and use of infrastructure enables the management team to create a mechanism that concentrates all the energy input from the community towards a unified goal.

2.3.1 Management Goals

The ultimate management goal is to build a self-sustainable virtual community that requires little management intervention, while seamlessly empowering community members with skills and knowledge.

The effective administration of users and their geographical, cultural, and characteristic differences should be a high priority for the management team. Management strategies should enable each user to reach the full potential allowed by the user's unique combination of online behavioural traits and geographical position.

The administration provides infrastructure and tools to guide the community towards

the goal of automation and minimization of administrative input. The administration should keep in mind the long term vision of the virtual community when providing infrastructure and tools.

Community members should feel free to express views and philosophies of any kind within acceptable social norms. A management strategy is needed to minimize the consequences of offensive behaviour.

A strategy for the effective deployment of management human resources is necessary. Administrators with executive skills are necessary to maintain a presence in the community and perform day to day tasks. The community involvement of an administrator with excellent people and communication skills is necessary to guide the community at critical points in time towards a strategic direction. Without being burdened with excessive administration, we can focus on building scalable infrastructure and applications, and continuously develop the community infrastructure and address strategic needs.

Different types of users can be used in different ways to the advantage of the community. Management strategies are needed to leverage the energy within the community to perform certain functions.

2.3.2 Management Tools

Having automation in mind, management features must be as transparent and autonomous as possible. According to this philosophy, administrative tools must enable administrators to perform necessary tasks with as little effort as possible.

Normally, many hub operators are needed to maintain order in the public chatrooms and help users that may experience problems. Socially, hub operators become an administrative burden when they abuse power or when every active community member identifies the need to become a hub operator themselves.

The role of hub operators can be automated to an extent by stealthily filtering offensive messages and providing an interactive help robot that can answer frequently asked questions from a database of known answers.

When management intervention is required for whatever reason, tools are needed to

provide all necessary background information regarding an incident. Indexed connection logs and fulltext searchable public chat logs are needed.

An administrative role that requires responsibility is the registration of user accounts. Ideally, users should be able to email an automated attendant that manages the registration process. In an environment where receiving email messages is not possible, a streamlined user interface is needed where account information can be input and processed with as little effort as possible. Deploying an open web interface for self help user registration defies the purpose of registering users, since it can be easily abused by easily registering another account when the previous account has been disabled for whatever reason.

Disruptive and offensive user behaviour is usually initiated from public network locations, such as computer labs or community networks. Such community resources cannot be banned due to the fact that many legitimate users use the service from the same location. User registration is therefore required to be able to connect to the service from such locations. Abusive users can simply be banned on user account basis. The process of obtaining a new user account for an existing user should therefore be sufficiently cumbersome to prevent such behaviour.

2.4 Looking Forward

We identified key management and control strategies needed to unlock the potential of our community.

The rest of this dissertation will focus on the technical challenges we faced building our virtual community, our design choices and implementation and analysis thereof, as well as a concluding discussion.

However, we are not the first to attempt building a virtual community. We now proceed to study our playground in more detail from the literature, to learn from the efforts of others and strategically locate ourselves in the field.

3 Literature Study

3.1 Virtual Communities

A virtual community, also named an e-community or online community, is a group of people that primarily interact via communication media such as email, online social networks, and instant messages, rather than face to face, for social, professional, educational or other purposes across geographical and organizational boundaries. If the mechanism is a computer network, it is called an online community. Virtual and online communities have also become a supplemental form of communication between people who know each other primarily in real life, but also for those who have not met before and wish to only participate in online interaction. Many means are used in social software separately or in combination, including text-based chatrooms and forums that use voice, video, text, or avatars. Significant socio-technical change may have resulted from the widespread use of such Internet-based social networks [1].

Interacting in a virtual community has many advantages, such as the instantaneous transfer and collaboration between like-minded individuals who seek information, ideas and philosophies within a collective area of interest. Personal relationships develop when people interact via public discussions long enough, with sufficient human feeling involved. These personal relationships create a sense of belonging, making a person feel comfortable [2]. In a virtual community, no person has a face, creating an entirely new identity [3] for each individual upon which a reputation can be built.

Likewise, becoming too comfortable within an online relationship can also pose threats to those who are naively and trustingly entering the community. Malicious people, such as identity [3] thieves and stalkers, abuse this trust to exhibit antisocial behaviour for personal gain. Others fear that spending too much time in virtual communities may have negative effects on real-world interaction with real people [1].

Community members do not necessarily stay involved indefinitely in a specific community, but rather exhibit a pattern of growth during their time of involvement and may leave at any stage due to various reasons. It is not uncommon for a community to renew its user population every few years together with the goals and tasks that these members introduce.

Virtual community members begin their life in a community as visitors or lurkers. These members observe the community and view content, but do not provide content of their own or participate in discussions. After observing the community for some time, the user may tentatively interact in a few discussions or provide some content. Getting comfortable in the community, the user may start to contribute by regularly participating in discussions and providing some unique content. Regular users may then evolve into community leaders, recognized as veteran participants whose opinion is granted greater consideration. After a period of involvement, a user may leave the community for various reasons, including changed interests, lack of time, or evolutionary community changes that do not resonate with the individual's personal beliefs or interests. [4, 1].

3.2 Incentives for Participation in Virtual Communities

A primary objective when building virtual communities is focusing on technology, designing for scalability and efficiency. Active user participation [5, 6] is a key determining factor of a virtual community's success [7]. Various incentives can be leveraged to ensure maximum user participation.

In Web based communities, users primarily post information online when discussing topics and exchanging ideas and information. Active participation in Web communities therefore translates to users frequently posting sensible and relevant information online.

In peer to peer environments where users primarily exchange files, free riding the peer to peer system without contributing your own resources back is a major problem [7]. A small number of altruistic users exist that selflessly provide a lot of files covering a broad spectrum of interests, but most users are self-interested and need some form of incentive to contribute.

Exchange-based currency methods are the most popular, where a user gains currency

when contributing towards the network, and loses currency when network resources are used [8, 9, 10]. Measuring the user's contribution to the entire community as a whole scales better than privately tracking reputation of users on a client to client basis [11].

Kollock [12] describes gifts in an online community as the transfer of goods from one person to another person in the community. The unspoken obligation of giving a gift in return is not necessarily answered by the receiver of the gift, but by anyone in the community. This way, everyone's contributions to the community eventually involves every community member in the process of giving and receiving. Reciprocation of gifts by community members to the community is known as *generalized exchange*. A rough balance of giving and receiving is achieved over time.

Fundamental features of online interaction change the costs and benefits of social action in dramatic ways. Professional people tend to volunteer detailed information in online communities, which they are reluctant to offer for free to somebody in person. [12].

Most contributions to a community is driven by self-interest. The most prominent motivator is reciprocacy, where a community member expects something useful in return from the community in the future. Users regularly making contributions tend to receive help more quickly. The concept of an online reputation is also a motivator for contributing, where users providing high quality contributions earn respect from the community. Users knowing that they provide quality information feel that they make a difference in the group, boosting their own self-image as an efficacious person [12].

Not all contributions are made out of self-interest. In some cases, a user might feel attached to a community and volunteer contributions due to this feeling of commitment. Users might also share unique contributions they have made in private, due to the near-zero cost of sharing it with the community [12].

When building a virtual community, there are key concepts that stimulate contribution. Being involved in the community yourself is important, as it is the most reliable way to poll customer needs and the effectiveness of your strategies. The community must also have a unique aspect that makes it stand out from the rest. Leader involvement in the community must steer it towards being a thought leader in your terrain, where people can come to when they are in need of advice, assistance, or information. It takes time for a community to reach a critical mass, from where it can operate in a sustainable way. The infrastructure supporting the community must be of sufficient quality to facilitate

and stimulate interaction, and effectively scale out as the size of the community increases [13, 14].

3.3 Climate and Technology Changes

Virtual communities have evolved since the advance of the information era and are still in an evolutionary process. Howard Rheingold provides detail on the evolution of virtual communities since the first days in [2]. We proceed to mention a few key milestones in the evolution of technologies supporting virtual communities.

The ability to send electronic mail over the ARPANET was received very well by the user community of the day and was adopted very quickly. Soon mailing lists became a popular collaboration medium, establishing virtual communities exchanging information and ideas. [2, 15]

Usenet, a world-wide distributed Internet discussion system, came to light in 1979. Users read and post public messages to a collection of hierarchically organized newsgroups. Usenet operates by means of a loosely knit web of servers that serves local news reader clients, and forwards its messages with a flooding algorithm to all the peer servers it can contact that do not yet have the message. Therefore, Usenet can be regarded as one of the first peer to peer applications, although the peers in this case are servers, not the clients themselves. [16]

Dial-up bulletin board systems popular in the 1980s were replaced with Web-based Internet discussion forums in the 1990s. These forums or boards are Web applications that manage the content provided by users into threads and posts. Regular users on such forums interact with each other frequently and build interpersonal relationships, forming a user community. [2, 17]

Recently, social networking Web sites built specifically for the purpose, allow users to create their profile and link it to the profiles of their friends and acquaintances, forming a social network structure made up of nodes being the individual user profiles. Users can interact in various ways and exchange personal information on events and ideas. [18]

Online communal interaction also extends into the three dimensional world, where users

interact with each other in virtual words in massive multiplayer online role playing games (MMORPGs). Characters controlled by users can interact in real time in life-like ways in a virtual environment. [19]

It is clear that the scope of possibilities in virtual communities and infrastructure supporting virtual communities is ever increasing. A specific virtual community is therefore not tied to a specific technology for an indefinite time, but rather undergoes a process of continuous change. Guidelines for building infrastructure to support a specific community is always subject to review as a community develops additional needs.

We now proceed to discuss virtualization concepts behind platforms hosting services for virtual communities, which may be advantageous in specific circumstances.

3.4 Virtualized Platforms for Service Provisioning

Hosting various applications in a complex environment requires use of different servers, each running their own operating system on their own unique hardware. This leads to server sprawl and underutilization of hardware, as well as increased energy consumption and larger space requirements. Consolidating multiple physical servers onto a virtual platform reduces maintenance cost and effort, simplifies backups and server deployment, and extends the life of legacy applications by enabling the operating systems they run on to run on newer hardware [20, 21, 22, 23, 24].

The most basic approach to virtualize a system is to partition its operating system into virtual servers [25]. Solaris containers [26], Parallels Virtuozzo containers [27, 28], and Usermode Linux [29] are examples of this approach. Isolation between virtual instances in the same operating system is very weak, and a misbehaving virtual machine severely affects the operation of other virtual machines [30].

The most common approach to virtualize systems is to run a virtual machine monitor on a host operating system. Virtual machines run as processes on the host operating system, scheduled by its scheduler and managed by its memory manager. Examples of host based virtualization solutions are VMWare Workstation [31], Sun xVM VirtualBox [32], Linux Kernel Virtual Machine (KVM) [33], QEMU [34], and Microsoft Virtual PC [35]. These desktop based virtualization solutions work well for research and development, or

running different desktop operating systems for application compatibility. Performance, however, is usually significantly lower than the native performance of the underlying hardware, due to virtualization overhead caused by the host operating system and the virtual machine monitor [36].

The best performing approach to virtualize systems is to boot the system directly into a hypervisor that virtual machines are run on. Examples of hypervisor based virtualization solutions are VMWare ESX Server [37, 38] and the open source Xen hypervisor [39, 40, 41, 42, 43, 44]. Microsoft is also working on its Hyper-V [45] virtualization offering that claims to use a hypervisor-based approach. Hypervisors are used in datacenters to create a virtualized platform that virtual servers can be run on, and are optimized to provide performance levels in virtual machines that approach the performance of the underlying physical hardware.

Hypervisors can perform either full virtualization, emulating real hardware thus allowing unmodified guest operating systems to be run, or paravirtualization, running guest operating systems that have been modified to run on a hypervisor. Full virtualization use either binary patching to modify specific unsafe instructions executed by the guest operating system on the fly, or can use hardware assisted virtualization available in modern hardware. The advantages of both full- and paravirtualization can be combined by using paravirtual drivers in a fully virtualized environment [46, 47].

3.5 Looking Forward

We consulted the literature to better understand the dynamics and requirements of virtual communities. We also looked at the concept of server virtualization in general, to provide background for the chapters that follow.

We now proceed to evaluate different technologies that can be used to build our virtual community upon.

4 Design Choices and Technology Evaluation

In this chapter, we present technology choices we are challenged with in designing the infrastructure supporting our virtual community. We specifically look at peer to peer technologies, server operating systems, platform virtualization solutions, and instrumentation tools.

4.1 Peer to Peer Technologies

We present the evolution of peer to peer networks and exercise a design choice after evaluating the advantages and disadvantages of each technology. We then mathematically analyze our design choice to verify its suitability in our application.

4.1.1 Generations of Peer to Peer Networks

Peer to peer networking strategies have developed in distinct generations, each successive generation addressing the shortcomings of and building upon the previous generation.

First Generation Networks

First generation file sharing systems used a central server to either share files or to coordinate the exchange of files between peers [48]. Various hybrid approaches [49, 46] to coordinate searches and connections between peers, exhibiting varying degrees of performance and scalability, are possible.

The fact that every client connected to a hybrid peer to peer network has a connection to the same centralized server, makes searching for files on such a network very fast and efficient. Search queries can be relayed to all the clients connected to the system, or a central directory can be queried for results. The central server can also assist users to establish direct connections with each other to perform transfers. Napster and Limewire [48] are well known hybrid peer to peer services.

Direct Connect [50, 51, 52] has evolved into a hybrid peer to peer system that supports hash-based searching and multi-source downloads. Its functionality and ease of use makes it a widely deployed technology today in relatively small file sharing communities.

The centralized design of hybrid peer to peer networks creates a single point of failure at the center of the network, on which availability the network's existence depends.

Second Generation Networks

Addressing the single point of failure in first generation hybrid peer to peer networks, the second generation strives to decentralize the network by disposing of the central server and building a network of peers that rely on each other to forward searches and connect users [48].

The FastTrack protocol [53, 52] was one of the first second generation peer to peer protocols that emerged. Many peer to peer applications, such as KaZaa [54], Grokster [55] and iMesh [56], are based on the FastTrack protocol.

Gnutella is an alternative, open protocol used by various clients. The Gnutella network is open and entirely decentralized. Many peer to peer clients that formerly used the FastTrack protocol, abandoned it in favour of Gnutella [57, 58, 52]. Chapter 8 of [59].

In order to improve scalability, both FastTrack and Gnutella elect "well connected" peers with lots of bandwidth and processing power to act as message forwarders for other clients connected to them, creating a composite network of leaf nodes and ultra nodes. Queries are forwarded through the connected set of ultra nodes, forming the network.

Third Generation Networks

Third generation peer to peer networks have anonymity [60, 61, 62] features built in [61], by routing traffic through other users' clients based on a Friend-to-Friend topology. Encryption is used to prevent interpretation of traffic by sniffing [48]. Examples of third generation networks are I2P [63], GNUnet [64] and Freenet [65].

GNUnet is a free software framework for decentralized, peer to peer networking. The primary application of GNUNet is anonymous, censorship-resistant file-sharing, allowing users to anonymously publish or retrieve information of all kinds [64, 66].

Freenet uses a decentralized peer to peer architecture to create an uncensorable and secure distributed information storage system [67, 68, 65, 69, 70]. Freenet pools together the bandwidth and storage space of connected nodes to allow users to anonymously publish or retrieve information. Freenet offers anonymity for producers and consumers of information, deniability for storers of information, resistance of censorship attempts, efficient routing of information, and decentralization of all network functions [67]. Chapter 9 of [59]

I2P is an anonymizing network layer upon which anonymity or security conscious applications can operate. Applications covering the full range of typical Internet activities are available, including anonymous web browsing, web hosting, blogging, chat, swarming file transfers and email. Unlike Freenet and GNUnet, services hosted on I2P are entirely interactive [71, 72, 63, 73].

Fourth Generation Networks

The fourth generation of peer to peer technologies send streams instead of files over a peer to peer network. The peer to peer infrastructure is used to deliver streams or multicasts with a BitTorrent-like [52] swarming technology [48]. Miro [74], PeerCast [75], and CoopNet [76] are well known examples.

Peer to peer networks started out as hybrid networks with a central server that all the clients connect to. Coordination of searches and peers connecting to each other is trivial and reliable on such networks, due to the centralized design.

Addressing the single point of failure being the central server in hybrid peer to peer networks, the second generation of peer to peer systems was born. Decentralization is the key design issue, building networks that can exist on their own and cannot easily be denied service. Searching for files and connecting to other users is now a more difficult problem, as the network is decentralized. Clients gossip searches and search results among each other with routing protocols running in the network [77]. Search results are more limited due to the nature of the searching process.

Peer to peer networks in specific applications needed anonymity and deniability for its users. The third generation of peer to peer networks was born, enabling censorship resistant freedom of speech. Networks of trust use encryption to exchange and store data securely. Performance of third generation networks is significantly decreased by the overhead introduced by the encryption and obfuscation of the network topology.

The fourth generation of peer to peer applications leverage the power of peers exchanging data to stream information live over peer to peer networks.

We deliberately choose a first generation peer to peer technology, Direct Connect, due to its ease of use, reliable searching, value adding chat facilities, and its suitability for application in relatively small and private communities.

We now proceed to analyze the scalability of the Direct Connect protocol from first principles, verifying its long term suitability in a community increasing in size over time.

4.1.2 Scalability of the NMDC protocol

Direct Connect clients relay search information and negotiate connections to each other via the Direct Connect hub. Broadcast type messages such as search requests and public chat messages received from clients are sent to all connected users. Unicast type messages

like passive mode search results and connection establishment messages are only sent to a single destination. Unlike TCP-style broadcasting by sending a message once to the subnet broadcast address, broadcasting a message on a Direct Connect hub unicasts the message to each connected user individually. Sending broadcasts is therefore an expensive operation.

Modeling user behaviour statistically, the average user requests the hub to broadcast k messages and unicast l messages to n connected users per unit time. Each user therefore causes $nk + l$ messages to be sent out by the hub per unit time. Multiplying each user's contribution to total traffic by n users, the total amount of messages m sent per unit time is expressed by equation 4.1:

$$m = n^2k + nl \tag{4.1}$$

When a new user connects to the hub, n in equation 4.1 is substituted with $n + 1$:

$$m_{n+1} = (n+1)^2k + (n+1)l = n^2k + 2nk + k + nl + l \tag{4.2}$$

Subtracting (4.1) from (4.2) yields the total amount of additional messages u per unit time the hub transmits for every successive user that connects, given by equation 4.3:

$$u = (2n+1)k + l \tag{4.3}$$

Direct Connect hubs can be linked together. We analyze the effect of splitting a hub into two linked hubs by substituting n with $\frac{n}{2}$ for unicast messages and, taking the hub link into account, $\frac{n+1}{2}$ for broadcast messages in 4.1.

$$m_l = \frac{k}{2}(n^2 + 2n + 1) + nl \tag{4.4}$$

The effective message rate reduction ratio per hub by splitting a hub into two linked hubs, is obtained by dividing (4.1) by (4.4):

$$\frac{m}{m_l} = \frac{n^2k + nl}{\frac{n^2 + 2n + 1}{2}k + nl} \tag{4.5}$$

Taking the limit of (4.5) as n approaches infinity, the upper bound of the message reduction ratio is 2. Scaling out by linking hubs together is possible, allowing a $\sqrt{2}$

times increase in user count for every 2 times increase in linked hub count. Therefore, the NMDC protocol, like any partially centralized approach, inherently scales poorly beyond certain limits.

Peer to peer communities of up to nearly 20,000 concurrent users running on a single Verlihub instance have been recorded [78, 79]. Various compromises have been made by trading off interactivity and search frequency to accommodate such a very large number of users on a single 100Mbps server connection. Moore's law [80] also contributes toward scaling out of hybrid peer to peer network implementations, as Gigabit Ethernet becomes commoditised, and 10 and 40 Gigabit Ethernet is used in backbones and data centers.

4.2 Operating Systems

The total cost of ownership (TCO) of an operating system is a good starting point when evaluating the suitability thereof for a specific application.

In a study comparing the TCO of Linux, Solaris and Windows, it has been found that Linux is the winner by far [81], despite the fear, uncertainty and doubt (FUD) Microsoft is spreading about the subject in their advertising campaigns. This study calculates the total cost implication of an operating system running a unit of work. Linux being the most efficient operating system, can run the same unit of work on much less hardware. The administrative burden of a Linux server is also lower, having a lower administrator to server count ratio.

Technically, Linux is only the kernel of the operating system [82]. A collection of other open source programs bundled with the kernel form a Linux distribution. Many Linux distributions exist, and the option to create a new one yourself also exists.

Bleeding edge developments in open source software may cause instability and incompatibilities between various versions of software. Backporting is a strategy where a distribution standardizes on software versions for a major release, and backport bugfixes, selected enhancements, and updated driver releases to the software versions standardized on. This approach combines the known stability and reliability of a specific configuration with the advances in development that fixes security errata and new hardware support [83].

The best known Linux vendor that backports software is Red Hat. Red Hat Enterprise Linux (RHEL) is a commercial distribution with customers paying for support [84]. Various free, open source clones of RHEL exists that are built from the source packages that are provided by Red Hat, including the Community Enterprise Operating System (CentOS) [85], Scientific Linux [86], and White Box Enterprise Linux (WBEL) [87].

CentOS has the largest and most active community of all the Red Hat clones, with 100% binary compatibility with RHEL [88] and an active community support forum, making it the Linux distribution of our choice.

4.3 Hypervisors and Virtual Infrastructure

Due to their superior performance and manageability, we only consider hypervisor based virtualization solutions to build our virtualized infrastructure upon [89]. Two major players exist in this market, VMWare's proprietary ESX Server product, and the open source Xen hypervisor.

VMWare published a white paper, comparing their ESX Server to Xen [36]. In VMWare's presentation, ESX Server performs significantly better than it's competitor, Xen. Subsequent studies claims [90] to prove the contrary, although the results may not be made public as publishing any benchmark-related information on ESX Server is prohibited by VMWare's end user licence agreement. Evaluating VMWare's test methods shows that Xen has not been configured properly and that the paravirtual Xen frontend drivers were not installed in the guest operating systems. Giving ESX Server an unfair advantage over Xen in their presentation, VMWare's performance difference claims can be disregarded and ignored. Independent studies [41], comparing Xen to the native underlying hardware, shows Xen guest VM's have near-native performance. Properly configured virtual servers running on Xen have a 0.75% overhead on the same SPECJBB tests that VMWare based their benchmarks and performance claims on, actually outperforming VMWare ESX by a considerable margin.

A possible disadvantage of a hypervisor based approach in virtualization is driver support, as the hypervisor's microkernel has direct access to the hardware and needs its own drivers just like a normal operating system would. Virtualization vendors cannot maintain driver support for all possible hardware types and models, and hardware vendors

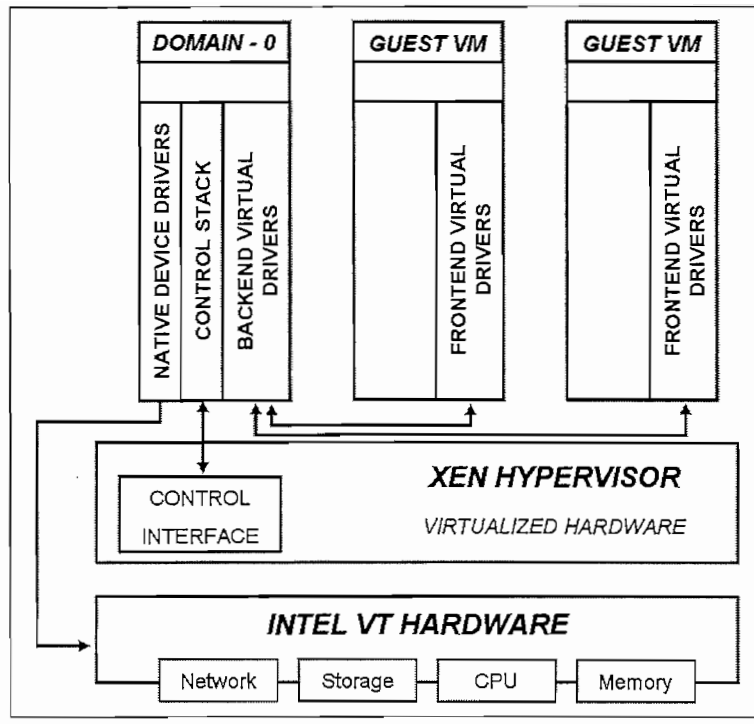

Figure 4.1: Xen Hypervisor Architecture

are reluctant to support yet another platform. Xen mostly overcomes this by booting a privileged virtual domain that runs a modified Linux kernel instead of a proprietary microkernel, that drives peripheral devices and has access to the hypervisor's control interfaces.

The architecture of the Xen hypervisor running fully virtualized guest instances on Intel VT hardware is shown in Figure 4.1. Xen also supports hardware virtualization solutions in AMD hardware. The privileged domain, Domain-0, alleviates the problem of driver support issues of microkernel based solutions by booting a Linux kernel that has access to the real hardware. In case of fully virtualized domains, Domain-0 performs I/O emulation for the platform. Domains using paravirtual I/O frontend drivers connect to the paravirtual backend drivers in Domain-0. The virtual network interfaces of all the domains terminate in Domain-0, where traffic flowing in and out of these interfaces can be bridged, routed, manipulated, prioritized, analyzed, and firewalled with standard features in the Linux kernel running Domain-0.

Virtual infrastructure provides flexibility and manageability for an installation. Live snapshots can be taken of running virtual machines, which can be restored in case of disaster. If centralized network storage is used to host the virtual disk images, the virtual

machine instances can also be live migrated between physical servers to dynamically balance load, recover from failed hardware, or to perform maintenance.

4.4 Instrumentation Solutions

It is important to visualize the day to day operation of production systems to measure utilization, identify bottlenecks, and observe trends. Graphs can be invaluable fault finding tools, and can provide detailed information of a running system to an observer with insight.

Perhaps one of the first widely used monitoring tools is the Multi Router Traffic Grapher (MRTG) [91] by Tobias Oetiker. MRTG primarily uses SNMP to poll targets for data. The author also developed a Round Robin Database (RRD) tool to effectively store the floating point data the graphs are rendered from in a fixed size database, automatically decreasing resolution of weekly, monthly, and yearly information.

Cacti [92] is a complete monitoring solution which also uses RRDTool for data storage. It is highly configurable with templates and plugins. SNMP is primarily used for data collection.

Munin [93] is a lightweight set of Perl scripts that processes data polled from Munin nodes. Munin uses a simple Telnet-like connection to retrieve data from the nodes. Scripts or programs that return data and graphing instructions in plain text format can be written to plug into Munin and generate a new graph.

On the commercial side, Sintrex [94] provides information technology infrastructure management solutions. Custom solutions can be requested, as it is a supported product provided by a commercial vendor.

We choose Munin for our instrumentation purposes, as it is easy to use and can monitor almost anything. The fact that it does not use SNMP makes custom plugin development a quick and simple task. Munin is not as capable as Cacti or commercial solutions with alerts, although basic event support is available and entirely configurable. Munin can output alerts, which can be handled by third party programs.

4.5 Review

In this chapter, we presented some technologies that can be used to support our vision and strategy. We looked at peer to peer technologies and motivated our technology choice after looking at the available options and strategies. We looked at operating systems and platform virtualization solutions and chose technologies that best fit our needs. We also briefly discussed instrumentation solutions and motivated our choice.

In the next chapter, we discuss how we used these metaphorical bricks to build infrastructure in an elegant way, supporting our virtual community.

5 Implementation

In this chapter, we discuss how we designed, implemented, integrated, and enhanced the infrastructure supporting our virtual community. We built a platform to host a collection of open source operating systems and software programs we integrated. We also discuss new software we developed to address specific shortcomings and needs in a unique way.

5.1 Platform Design

A robust, scalable platform is needed to host the services discussed in this chapter. Virtualizing this platform has many advantages, of which server consolidation, consistent image backups, rapid server deployment, secure virtual networking infrastructure, and fast disaster recovery are the most prominent.

In this section, we discuss the software and hardware configuration that makes hosting this plethora of demanding network services possible.

Software

Starting with the virtualization platform, the open source Xen hypervisor is used. Our operating system distribution of choice that ships a Xen kernel is CentOS version 5. The CentOS project builds a free, open source enterprise grade operating system from publicly available source code packages provided by the commercial Red Hat distribution.

Systems running a Xen-based hypervisor do not boot into an operating system. Instead, the system boots into the hypervisor directly. After the hypervisor initializes, a Xen

domain is created in which a kernel boots that has access to the real hardware, as well as privileged access to Xen's control interfaces. The most important ongoing task of this domain is to perform the routing and firewalling of Xen's virtual network and schedule the physical disk access of other domains' virtual block devices. Other unprivileged Xen domains can be configured and started from this domain.

Fully virtualized domains are created for the server instances, allowing unmodified guest operating systems to be run. The performance penalty of using full virtualization is overcome by installing paravirtual Xen frontend drivers in the guest operating systems, significantly optimizing network and disk access by interfacing directly with the hypervisor.

The disk images of the guest domains' virtual disks are managed with LVM (Logical Volume Manager), allowing consistent image backups while the guest domain is running. LVM makes disk management very flexible, allowing dynamic resizing of logical volumes and moving logical volumes between physical block devices on the fly.

Hardware

Designing a hardware specification for a system owned by and doing work for a relatively large community is challenging. A stable and reliable system is needed, providing adequate performance to service the demand while providing good value for money.

Hardware virtualization support for such a system is necessary, as fully virtualized guest domains will be run on the Xen hypervisor. As much as possible physical memory is important, as many virtual servers will be simultaneously needing memory. Networking demand will be very taxing on the hardware, therefore a network interface card with advanced hardware features such as interrupt moderation and TCP checksum offloading is necessary. A fast storage subsystem is needed for all the virtual servers sharing it. Multiple simultaneous operating systems running on a hypervisor can make very effective use of multi core CPUs.

The hardware specification for the machine is shown in table 5.1 on the following page.

Table 5.1: Hardware Specification for Xen Host

Component	Specification
System Board	Intel "Dragontail Peak" DP35DP
Processor	Intel Core 2 Duo E6750
Processor Features	4MB L2 cache, 1333MHz FSB, Virtualization
Processor Speed	2x 2.67 GHz
Memory	4x 1GB Kingston ValueRam DDR2-667 CL5
Memory Configuration	Dual Channel, Bank Interleaved
Network Interface	On-board Intel 82566DC PCI-Express
Network Interface Features	Interrupt Management, TCP Checksum, TSO
Storage Controller	On-board Intel ICH9R SATA-II
Storage Device	2x Seagate Barracuda ST3808110AS
Storage Subsystem Features	SATA AHCI, Native Command Queuing, JBOD

Table 5.2: Operating Systems Used on Community Servers

Server	Operating System	Architecture
Xen Host	CentOS 5.1	x86_64
Verlihub Server	CentOS 3.9	i386
Web Server	CentOS 5.1	x86_64
Management Host	CentOS 5.1	x86_64

5.2 Operating system choices

Of the many mature operating systems that exist for the x86 architecture, Linux is our operating system of choice. Slow moving distributions like CentOS standardize on software versions and interfaces to ensure stability, and backport bugfixes and enhancements from newer releases if necessary. This maintenance is kept up for a long time, as the current CentOS release 5 is under maintenance until the year 2014.

Various database and web servers, scripting languages and data management tools are native to Linux, making it the best operating system choice for mission critical applications where performance and reliable operation is important.

Table 5.2 displays the operating systems used on the various servers.

5.3 Peer to Peer Network Implementation

The peer to peer network is based on the extended Direct Connect protocol, DC++. While being a first generation peer to peer technology, its ease of use, reliable searching, and manageability makes it a very good choice for our purpose. Stable and intuitive client programs exist for Linux, Windows and Mac.

The hub software used is Verlihub, which is an open source Direct Connect hub written in C++ and runs on Linux. Verlihub has a relatively small memory footprint, is entirely scriptable, uses a MySQL database backend and has relatively low processing overhead, being the first hub to break the 10,000 users barrier.

The operating system used to run Verlihub on is the 32-bit x86 version of CentOS 3.9. This specific machine was previously used in peer to peer network experiments on low-spec hardware and proved to be very stable and reliable. Its exact disk image was transferred to the Xen host and booted in a virtual machine without any modifications, with the exception of adding the paravirtual Xen drivers at a later stage.

Verlihub can be extended in many ways using the Lua programming language. In the sections that follow, we discuss the many enhancements made to Verlihub in order to implement the long term administrative vision.

5.4 Peer to Peer Network Enhancements

In this section, we discuss the various enhancements we have designed and implemented in the peer to peer environment. Unless otherwise specified, all enhancements are our own original work.

5.4.1 DC Dollars

Accurately rewarding community members using the peer to peer network for their individual contributions is a very difficult problem. It is virtually impossible to assess the exact nature, significance and quality of each member's contributions. A method

is however needed to motivate passive users to contribute back to the community to stimulate growth, by rewarding individuals who do contribute in accordance with their contributions.

The community perceives a large contributor as an individual sharing a large amount of data. Although this approach may not be entirely correct, using this approach as a metric to calculate rewards is considered to be fair, based on community perceptions in the spirit of stimulating participation. Individuals willing to sacrifice their own comfort by contributing a lot of their own resources for the luxury of other community members, are considered to be big contributors.

DC Dollars is an internal community currency devised to reward contributors for their contributions. DC Dollars can be used to buy various value added services in the community. The basic service is considered to be the chat functionality, basic connectivity to peers and usage of the BBS. Any services other than that, like search, offline search, and search watching, are considered to be value-added.

The primary purpose of the DC dollar currency method is to motivate users to contribute to the community. The economy should therefore be adjusted in such a way that even small contributors are able to use value added services, in order to motivate zero contributors to participate.

DC Dollar profit is calculated with a simple formula that further strives to amplify the purpose of the economic model. Profit aggressively increases in small share sizes increasing, and increases moderately as the share size gets bigger. Formula 5.1 is derived from a statistical regression through target points specified in Table 5.3, and calculates DC Dollar earnings for $BytesShared$ bytes over a period of $DeltaT$ seconds with $Slots$ download slots open.

$$DeltaDCD = (1.018E-7) \times (BytesShared)^{0.255} \times DeltaT \times \frac{Slots}{20},\ 0 \leq Slots \leq 20 \tag{5.1}$$

Graphically, DC Dollar allocation per day is shown in figure 5.1.

An individual can transfer DC Dollars to friends, adding an additional dimension to the interaction between community members. Another possibility that has not yet been implemented is to leverage premium rate SMS service providers to enable users to buy

Table 5.3: DC Dollar Profit per Day per Bytes Shared

GB Shared	DC Dollar Profit per Day
0	0
10GB	3
100GB	5
1000GB	10

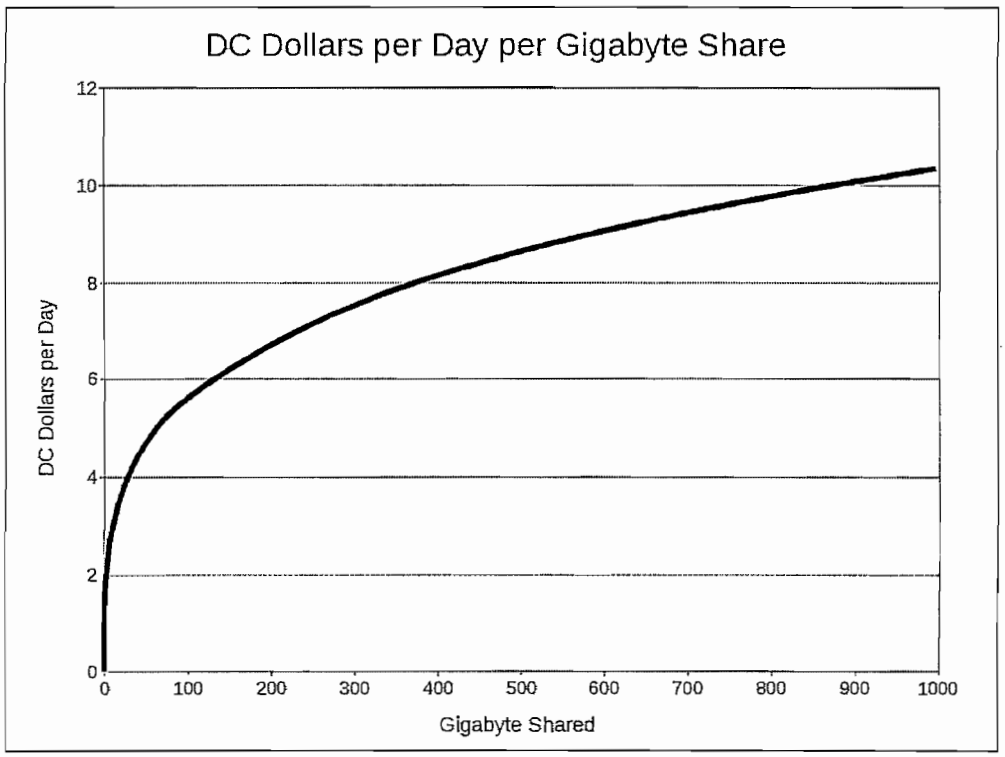

Figure 5.1: DC Dollars per Day per Bytes Shared

DC dollars with a text message from a mobile phone.

The ability to calculate DC Dollar earnings in Verlihub is based on a protocol feature allowing clients to update their status of share size, open slots, user description, hubs connected and the like. Verlihub exposes the reception event of such a message as a trigger that customized scripts can use to perform arbitrary processing of the data.

5.4.2 Announcement Channels

Verlihub has a hub broadcast feature that allows administrators to globally send a private message to all the users connected to the hub in order to make an announcement. The shortcoming of this approach is that the messages are sent to everybody and that only administrators can use the feature.

We developed a finer grained broadcast system in the Lua programming language as an extention to Verlihub, that all registered users can use. Customized announcement channels for common interest groups related to specific academic subjects or certain multiplayer games can be created by an administrator upon request. Any registered user can join such an announcement channel in order to receive messages from it. The user's subscription status remains persistent over disconnects and reconnects until the user decides to resign from the channel. Any registered user can make an announcement in a specific channel, which will be sent to all the users that joined the channel and are online at that stage. Custom user commands that allow a user to join or part from a channel, view available channels, or make an announcement in a channel are provided.

5.4.3 Popular Hash Identification

The Direct Connect network uses the TTH (Tiger Tree Hash) as a digest algorithm to calculate checksums of files, which is used to identify files on the network. All modern clients in use today perform multi source downloading and search for alternative copies of a file to download from in parallel based on the TTH of the file.

When a client begins downloading a file, a TTH search is broadcast via the hub to locate other users that also share the file. Events of TTH searches can also be handled

by Lua scripts in Verlihub. We developed an event handler function that inserts this information into a database table, keyed on TTH value and a hash of the source IP address. A scheduled job performs maintenance on the table on a daily basis by deleting records older than a week, to prevent the table from growing indefinitely.

Information about the data in this table is exposed via a Web service, ordered by TTH popularity and limited to an arbitrary amount. Other programs can use this information to perform data transfer estimations and update the offline search index database, for example. Automated content caches can also use this information to determine which files need caching to speed up the distribution process of popular files.

5.4.4 User Authorization and Access Control

Security on the campus network is sufficient to allow anonymous guest access to the peer to peer network from staff buildings and residences. Guest users cannot receive DC Dollars or any of the advantages enabled by it, but the option is still available to those that prefer it. Public locations like computer labs and community networks, however, require a valid user account.

Internal campus network configuration does not allow setting up a mail destination to automatically receive email from the campus email system. An administrative email address is therefore used to which users email registration requests. We developed a customized user registration interface that allows an operator to create a user account and enter information about the origin of the registration request email. The system verifies that the email address in question is used for the first time, before creating and activating the user account. The custom registration process is designed in such a way that a single command is used to create the account, simplifying the process and reducing administrative overhead.

5.4.5 Robots and Crawlers

Automatic Search Robot

An offline index of all available files needs constant updating, which is a cumbersome task if it has to be done by hand. This task can be automated to a large extent by writing intelligent robot scripts.

Information about popular Hub and Web search strings are exposed via Web services. We wrote a PHP Unix-style shell script that logs into the DC hub and searches for these popular search criteria obtained from the Web services. The protocol implementation of the robot is based on reverse engineered protocol information obtained with `tcpdump`. Search responses from currently online clients are parsed, extensively validated with a whitelist-based regular expression filter to ensure clean and valid data, and output in a compatible format to be directly imported by the MySQL database.

In addition to the search function of this robot, we wrote an extention that scans a directory for files and parses any standard DC++-style XML file listings found very efficiently for file information. Instead of incurring the additional parsing overhead of using a standard XML library, we wrote a very simple state machine to walk the directory structure of the data presented in the XML file format and extract all the information needed in a single pass. This program shares the data processing functions of the search robot to ensure clean and valid output results. The file list parser stores the md5 digest of each file list and only refreshes an unchanged file list if it is older than a week. The XML file lists can be obtained by scheduling automatic file list downloads with a normal DC++ client and can be automated.

The volume of data handled by the file list parser can be large, therefore it is computationally expensive to perform bulk updates with file list data too often. The search robot is a very lightweight method to perform frequent updates of only the popular information.

The interfaces and functions of the automated search robot is illustrated in figure 5.2 on the next page.

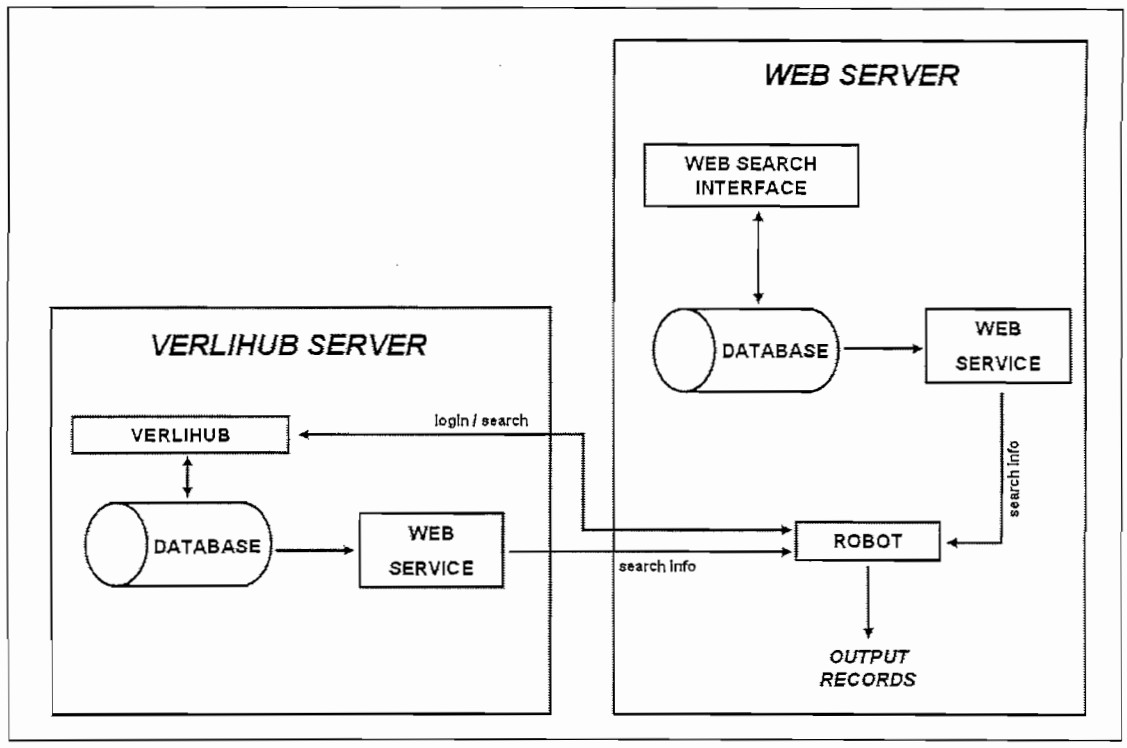

Figure 5.2: Operation of Automated Peer to Peer Search Robot

Automated Hub Assistant

Successfully delivering information on frequently asked questions to a community is important. A user often asks questions and needs an easy way to obtain answers. An interactive chat robot that communicates with a user in private message can unlock and present this information in a usable way.

Information is categorized, and the database can be browsed by sending the category and question numbers to the robot. Questions are grouped in categories like user registration, usage policy, download problems, information access problems, help with the Web tools, and the like. Selecting a category number displays its questions, and selecting a question number displays its answer.

Notifications Service

When a user performs a search on the Web-based search interface, currently and previously available files can be found. When a user is looking for a specific file that is

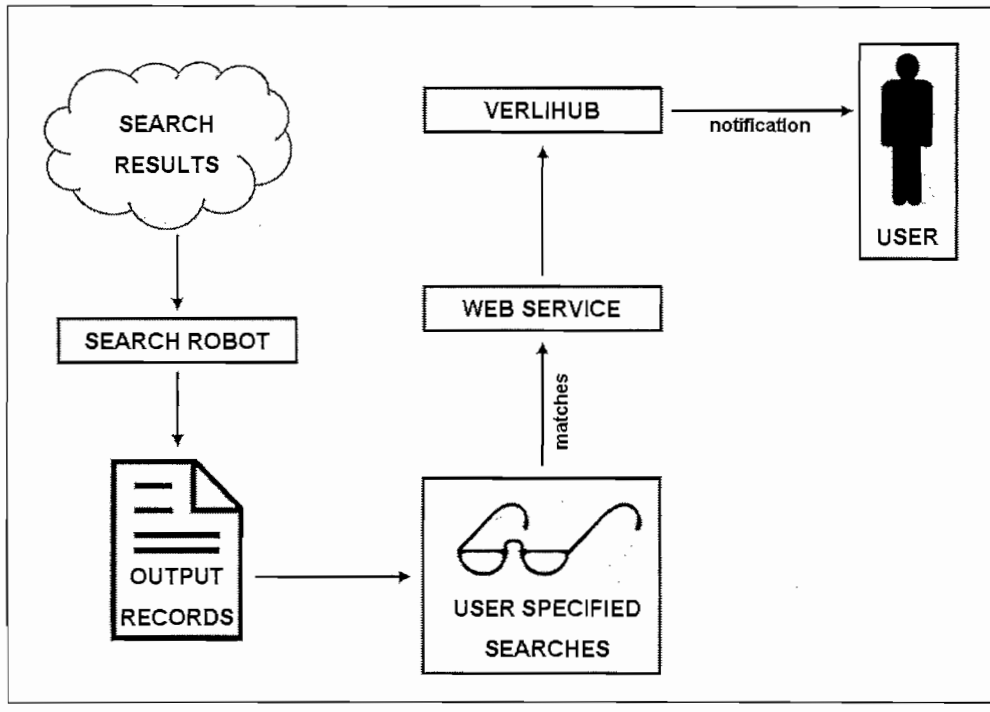

Figure 5.3: Notifications Service

not yet available, productivity is decreased as the user has to periodically check for the availability of the file.

Extending the Web-based search engine's data acquisition programs, we can process data as soon as it is acquired. Users can specify criteria for searches that do not yet yield any results, to be automatically matched against acquired data as soon as it is available to the program. Matching results are pushed via a Web service we developed to Verlihub, where a notification robot we wrote sends it to the specific user as soon as the user is online. The functional components of the notification service are presented in Figure 5.3.

The search matching engine is intelligent and uses the same file version matching engine as the Web-based search engine. If versioning information is detected in an offline search, the search will be automatically adjusted to search for the next available version of the file as soon as a match is found for the current search criteria.

Trivia Bot

A robot asking trivia questions in a private chatroom is provided for recreational purposes. Users compete against each other in real time by being the first to answer every question correctly.

Various trivia robots for Verlihub that are implemented in the Lua programming language can be downloaded. Apart from minor bug fixes and code optimizations, the trivia bot is still in its original form.

Public chat sanitizing

We developed a script that parses every public chat message and calculates a spam probability factor. Repeating patterns and offensive phrases are considered in the calculation. Messages scoring above a certain threshold are considered as unwanted.

Unwanted messages are stealthily killed by sending the message only to the offending user's public chat window instead of the whole community. Users exhibiting antisocial behaviour can spend their energy trying to aggravate other community members until the total lack of response from the community, due to the fact that they did not see the offending messages in the first place, lets them reconsider their strategy.

5.5 Web Environment

5.5.1 BBS

The bulletin board software used to facilitate online discussions, announcements and general exchange of information is based on phpBB version 2. We made extensive modifications to the code to adapt the software to specific community and administrative requirements.

Registration of new user accounts is disabled on the board. Users logging in authenticate against Verlihub's user database with their nickname and password. Upon first login, a new BBS user profile is automatically created and linked against the Verlihub account

in question. Figure 5.4 on the next page outlines the login process.

A photo album modification available on the phpBB community forums has been added to enable users to upload images. We modified the board software's regular expression URL filters to allow links to uploaded images to be displayed in user posts. Images are verified and limited in resolution and size with the `gd2` image processing library, therefore this modification is more secure than simply installing the file upload plugin.

5.5.2 DC Network Offline Content Index

Data collected from the robots crawling the Direct Connect network is stored in a database we designed after being validated and processed. Data is output in a flat file structure in delimited format by the robots, which is later imported by the database on a regular schedule.

Processed information stored in the database include the file name, path, and size, a hashed version of the user name for keying purpose, the file's TTH, the time of indexing, and special file versioning information extracted by an intelligent algorithm from the file name, if available. Files not refreshed after four weeks are considered stale and are deleted from the database by an automated process.

The database structure uses only fixed length records to prevent internal fragmentation after many deletes, inserts, and updates. Columns that will be used in queries are indexed. The file name and path columns are indexed with MySQL fulltext indexes to allow for boolean mode searches, where keywords being searched for can be explicitly included and excluded from the query.

We developed a Web-based search engine that searches the database and presents results in the form of `magnet:` links. Search results are processed and possibly broken file sources within the search result, are detected and flagged for the user. Files that might be corrupt are those with the same versioning information but different TTHes, or files with exact size matches but different TTHes, occurring within a specific search result. The file with the most sources is considered to be correct, and any other copies that differ from the most frequently occurring copy are flagged as corrupted.

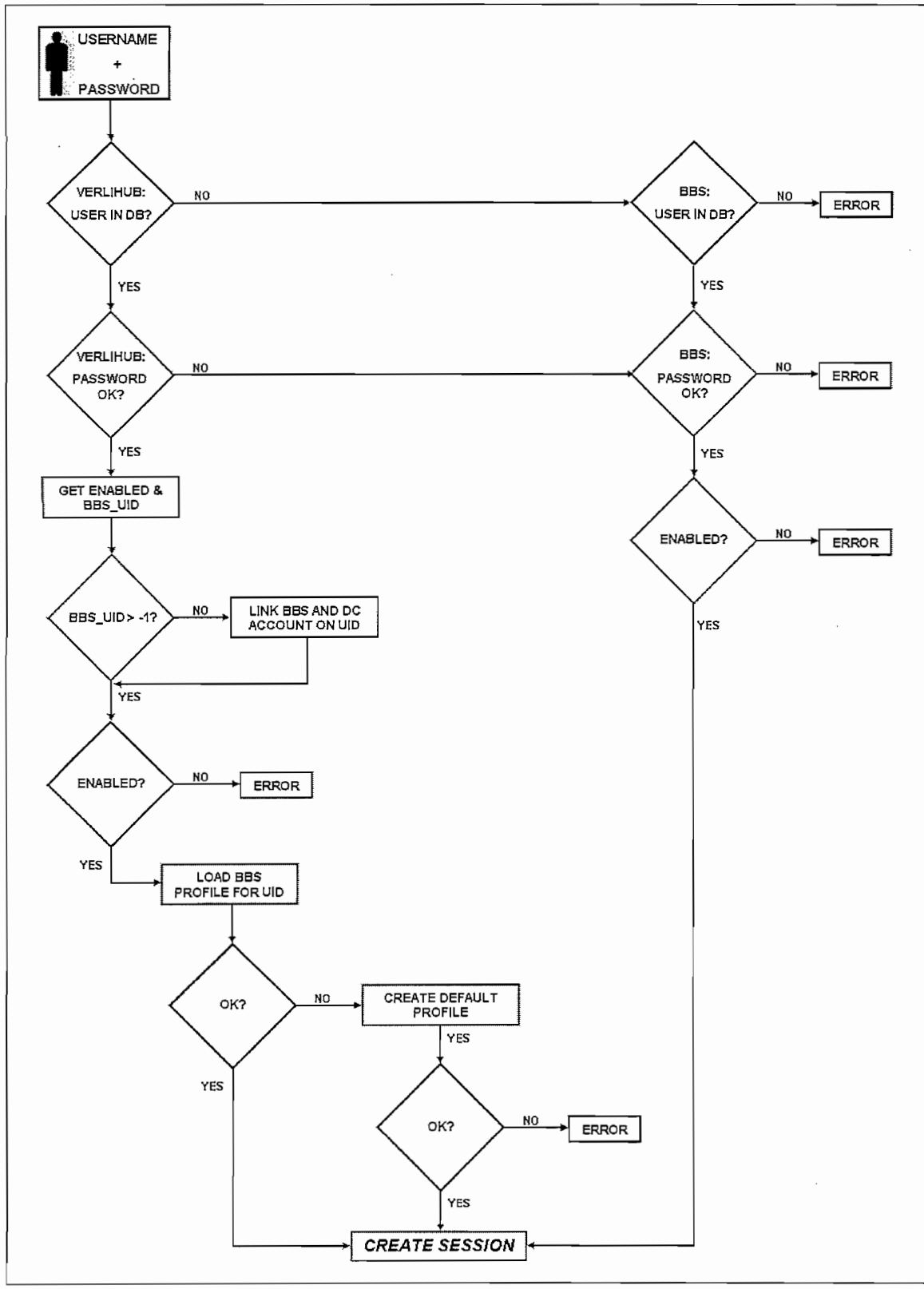

Figure 5.4: BBS Login Process

5.5.3 Web Services

The HTTP protocol can also be used, apart from transferring Web content to browsers, to transfer raw data upon request. A Web service exposes an interface that can be queried and returns data instead of HTML. XML is most frequently used as a container format, although any machine-parsable format can be used.

Internally, Web services are used to exchange data between different programs. Web services provide a high level of isolation between systems that need to exchange data, and compatible interfaces that can be used to construct scalable applications.

We have developed a Web service interface to present information on TTH data. The Web service has access to the databases of the offline content index and the hashes of popular files downloaded. Combining both, an XML download queue compatible with DC++ clients is generated, which can be used by automated content caches connected to the peer to peer network to keep their contents up to date.

We have also created Web service interfaces for the search robot to pull search information from, and a Web service interface for the search notification robot to push announcements to users on the peer to peer network.

5.6 Instrumentation

Monitoring and graphing various measurable metrics of a running system is important to obtain insight in its effectiveness and enables easy identification of bottlenecks and problems. System stability increases with proactive management decisions based on observed trends.

Munin is a highly customizable, easy to use monitoring solution. Data collection agents are deployed on the systems to be monitored, and are periodically polled by Munin. Data is stored in round robin databases (RRD's) with `rrdtool`. Graphs are rendered on demand by a dynamic web page from the data in the RRD files.

Munin data collection agents use plugins to monitor various measurables. Plugins use a basic text-based format to present data and graphing instructions to Munin. Any script

or program that returns data in Munin's format can be used as a plugin, making the development of custom plugins a trivial task.

Standard Munin plugins used in our case record system load, CPU, memory, and network utilization, MySQL queries per second, disk I/O operations per second, and various system statistics reported by `netstat`, `vmstat`, and `ps`.

We developed additional plugins to monitor the number of online users on the BBS and the peer to peer network, a stacked graph detailing total CPU usage per Xen domain, the total amount of data shared on the peer to peer network, hard disk temperature, and hard disk I/O utilization detailing the percentage of time the disks are busy.

5.7 Security Considerations

Reliable service provisioning requires a secure and stable system. A security-conscious mindset is necessary when designing the system. A new tool or service must always be designed from an attacker's perspective, considering ways to compromise the system using the tool or service in question. Books like Bruce Schneier's *Secrets and Lies* [95] and McClure et al's *Hacking Exposed* [96] provide insight on information security.

Security of the network the servers are connected to, in our case an internal virtual network bridge, has high priority. Netfilter is a very powerful tool in routing, shaping, and manipulating traffic, and can be managed and scripted with the userspace program `iptables`.

Firstly, all external access to the Xen host and all forwarding of traffic by the Xen host is disabled by simply dropping all traffic. Rules are then established to explicitly allow specific traffic needed for service provisioning. Network address translation (NAT) is used to forward connections through the internal network to the Direct Connect and Web servers. The rate at which new connections are allowed to the servers on the internal network is limited to a safe amount, moderating sudden influxes of users and countering specific denial of service attacks.

The web serving environment is optimized to limit the amount of web server and database connection threads and memory used per thread to allocate less memory than the system

has in total, preventing intensive use and denial of service attacks to cause heavy paging to disk or out of memory conditions. The disk image for the web server is stored on its own physical hard disk, protecting other virtual servers from the web environment's intensive disk I/O usage.

Private Web services and instrumentation tools are hosted on virtual servers that are not accessible from the external network in order to prevent abuse, while public Web services are hosted on the Web server itself.

The Xen host and otherwise firewalled ports on the virtual servers can only be accessed from the secured internal network. The internal network can only be accessed via an encrypted virtual private network (VPN) connection running OpenVPN, with a pre-defined static key from allowed, trusted locations. All management traffic to and from the internal network can therefore not be easily intercepted or hijacked.

A separate virtual network bridge that is isolated from the secure internal network is created to connect other, unrelated servers that may also be hosted on the platform, effectively sandboxing them and limiting the extent of damage they are capable of.

5.8 Review

In this chapter, we discussed how we designed, implemented, integrated, and enhanced the infrastructure supporting our virtual community.

We built a virtual platform using the Xen hypervisor. Our virtual infrastructure provides secure internal networking between virtual servers, strong firewalling, and strict network access control, consistent image backups, easy server deployment and configuration on abstracted hardware, and fast disaster recovery.

We built a virtual server hosting the core of our hybrid peer to peer network based on the extended Direct Connect protocol. We designed and coded enhancements to our peer to peer network environment by introducing virtual currency, announcement channels, automatic content search robots, automated searches with notification abilities, popular download identification, and automated public chat sanitation.

We built a virtual server hosting our Web environment, with a community BBS that

has been integrated with the peer to peer network environment's user authorization services. We designed and developed a search engine that indexes the contents of the peer to peer network and has advanced search features, searching on file size requirements or advanced versioning information.

We built a virtual server running instrumentation solutions that continuously monitors the performance of our system. We extended our instrumentation solution to monitor performance and status information of our own customized systems.

The structural overview of the system, detailing the structure of the virtual servers and their roles, is displayed in Figure 5.5 on the following page.

We built our system with security in mind, by thinking from an attacker's perspective and taking steps to minimize the impact of possible attacks.

In the next chapter, we discuss our results on the performance of our system.

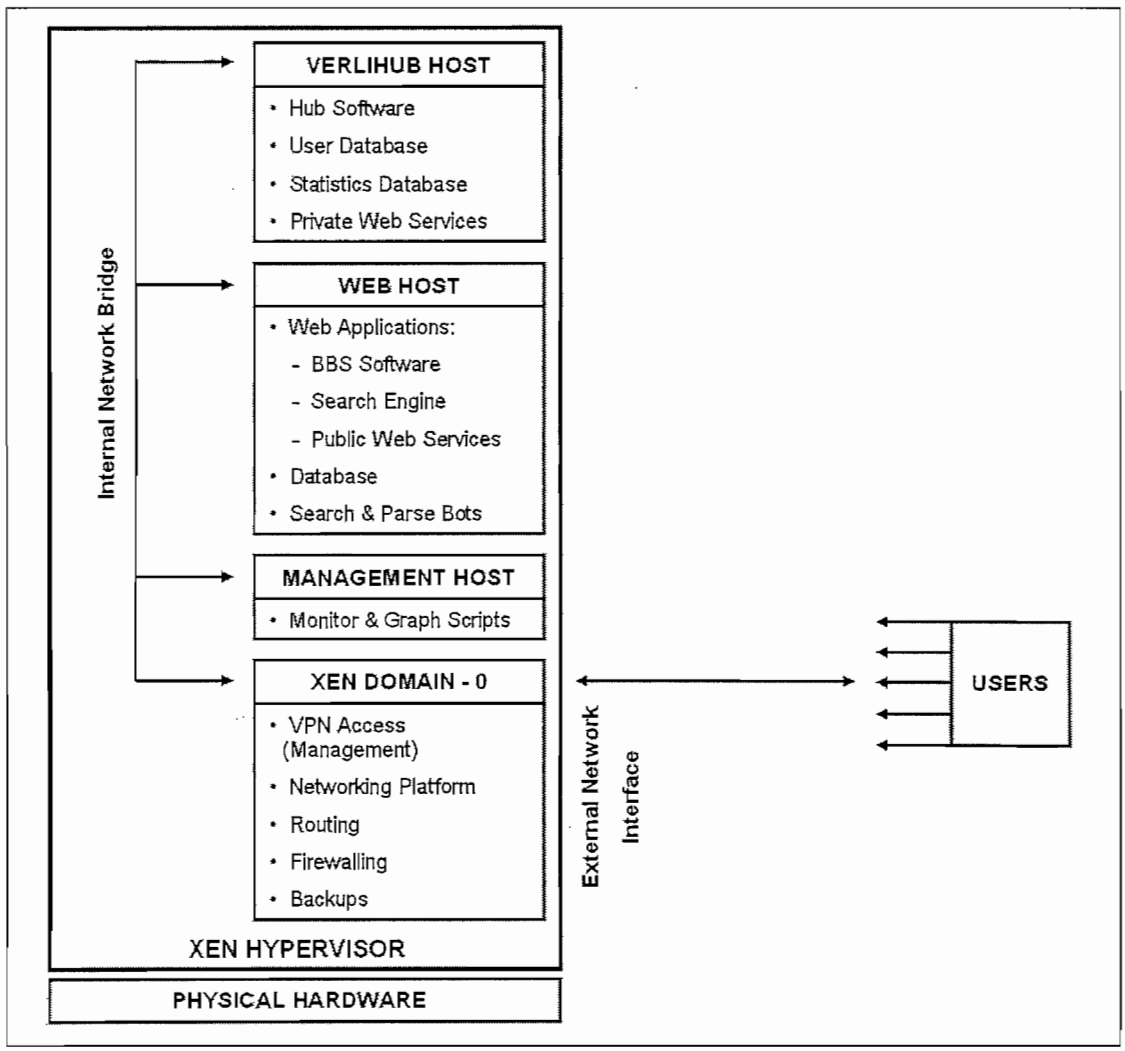

Figure 5.5: Community Servers Structural Overview

6 Evaluation and Results

We present a mostly quantitative view on the inner workings of the infrastructure supporting our virtual community. We discuss the runtime statistics of the virtualized platform, peer to peer network, and Web environment in detail. We also present a qualitative view on some of the strategies and incentive mechanisms we implemented.

6.1 Platform Performance

In this section, we discuss the results of key measurables regarding the platform hosting the virtual servers. The servers present an intensive workload to the platform, as a very busy Direct Connect hub performs thousands of network operations per second, a busy web server rendering dynamic pages is CPU and memory intensive, and a busy database constantly querying and updating a complete offline file index of more than 3 million entries, a few gigabytes in size, is I/O, CPU and memory intensive.

The Xen kernel in CentOS 5 proves to be very stable under such severe workloads. The current platform uptime is 245 days at the time of writing, with no degradation in performance since the day it was first booted.

The stacked graph in Figure 6.1 on the next page details the total CPU usage on the platform by each virtual server running on it. The green section at the bottom of the graph plots CPU usage for `Domain-0`, the privileged domain performing all physical disk and network I/O operations, including network address translation (NAT), connection tracking and firewalling. The dark blue section above it shows CPU usage for `lnx11`, the web server host. `lnx12` runs management tools and the monitoring utilities that renders the graphs and uses very little CPU time, as can be seen from the thin red line. The light blue section above it shows the CPU usage for the DC server, `lnx7`. `win1` shown

Table 6.1: Information on Community Servers

Hostname	Server	Operating System	Architecture
Domain-0	Xen Host	CentOS 5.1	x86_64
lnx7	Verlihub Server	CentOS 3.9	i386
lnx11	Web Server	CentOS 5.1	x86_64
lnx12	Management Host	CentOS 5.1	x86_64
win1	Community Gaming Server	Windows XP	x86_64

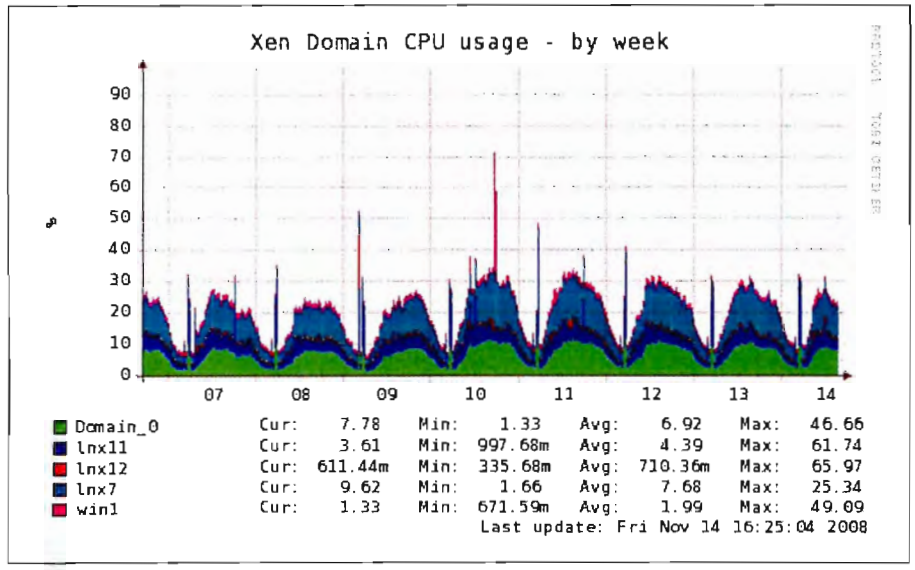

Figure 6.1: Platform: Physical CPU Usage per Xen Domain

in the pink section at the top shows CPU usage for a non-related Windows XP x86_64 SP1 host also running on the platform, acting as a gaming server for the community. Table 6.1 summarizes the host names and roles of the community servers.

The spikes in the CPU graph at 05:15 every morning is caused by the scheduled database import for the offline content index, loading the data acquired by the robots crawling the Direct Connect network throughout the day into the main database. A more detailed overview of the various robots operating in the system is available in Section 5.4.5 on page 33.

The loading of data is done at a quiet time to minimize the impact of the severe disk I/O caused by the process. Most of the work is caused by the rebuilding of the indexes, implemented as a temporary solution to an unresolved bug in MySQL that causes index files to grow indefinitely when data is loaded after an `alter table disable keys`

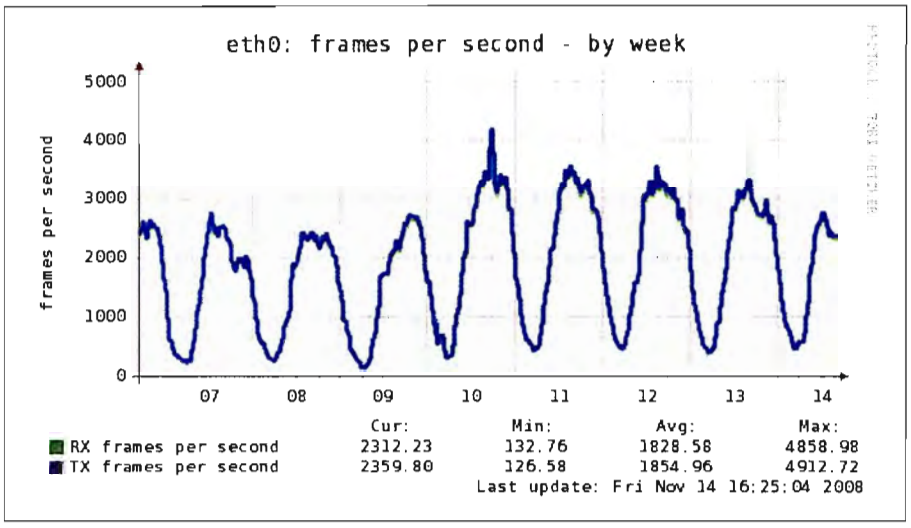

Figure 6.2: Platform: Network Link Layer Frames per Second

statement. Loading data with disabled indexes speeds up the import process exponentially, as the index is only updated after all the data is loaded, and not continuously throughout the entire loading process.

The weekly spike at 04:00 every Sunday morning is caused by scheduled tasks on the operating system, rotating out and compressing log files and performing other routine maintenance tasks.

The large spike in the middle of the graph on the 10th of November was caused by the Windows-based community gaming server that became unstable and crashed. This host has been administratively limited at hypervisor level to 50% of one CPU core, therefore it does not have any measurable effect on the reliability or operation of the platform in general or the rest of the virtual servers. The Microsoft server crashes every week or two, forming a predictable trend in this regard.

The maximum total CPU usage the platform is capable of is 200%, due to the fact that the physical CPU has two processing cores. The credit scheduler of the Xen hypervisor dynamically balances the load of the different virtual machines evenly between all the available physical CPUs. Refer to Table 5.1 on page 27 for the detailed hardware specification of the server.

Figure 6.2 plots the activity on the physical Ethernet interface controlled by `Domain-0`, showing raw Ethernet frames per second flowing in and out of the interface. The transmit

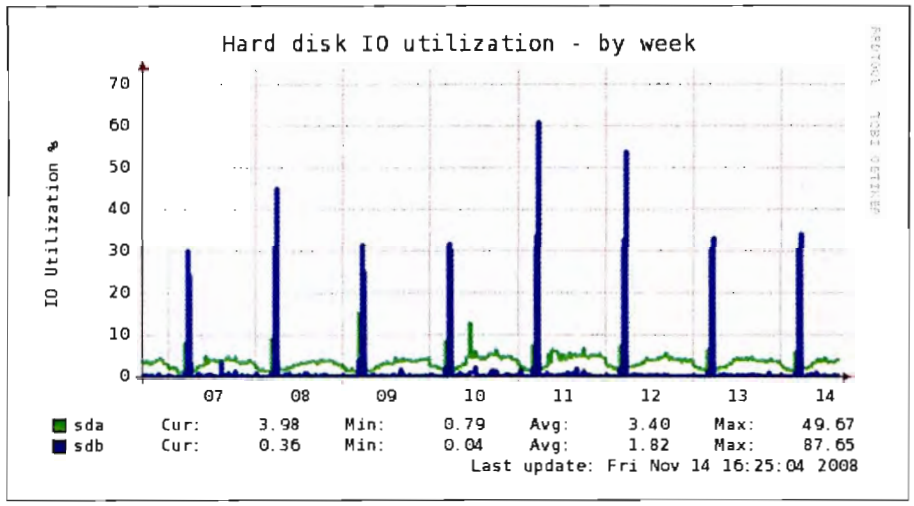

Figure 6.3: Platform: Physical Disk I/O Utilization

rate is only very slightly larger than the receive rate, due to the fact that TCP is mainly used as a protocol requiring `ACK` messages to be sent back by the connected clients. The Direct Connect hub, being responsible for the majority of the traffic, sends relatively short messages at a time, capable of fitting into a single frame. Therefore, an inbound frame carrying a TCP `ACK` message is received shortly after every frame carrying an outbound NMDC protocol chatter packet is transmitted.

The graphs in this section show a week's information and plot 30 minute average values. Therefore, the maximum values claimed in the graph legend are maximum instantaneous values caused by sudden surges in traffic, and are not necessarily displayed on a graph plotting averages.

Figure 6.3 details physical disk activity as a percentage of time the disks are busy. The Serial ATA hard disks are controlled with the `ahci` driver, using the SCSI subsystem of the kernel to take advantage of the command queuing capability of the disks. Therefore, the disks are represented as SCSI block devices `sda` and `sdb`, respectively.

The first hard disk, `sda`, contains the OS installation for the Xen host and all of the virtual disk images, except for the web server's disk image, which is on `sdb`. The combined disk utilization of the virtual machines sharing device `sda` does not frequently saturate the device for I/O, therefore they can all coexist on the same physical device. The web server's virtual disk is stored on its own device, because its heavy disk I/O usage at times can cause I/O starvation for other virtual machines sharing it.

The commodity desktop hard drives used in the server are not designed for enterprise grade workloads. It has been empirically determined that I/O does not degrade gracefully as the workload increases. If a given workload causes the disk to be busy 20% of the time, adding another identical workload does not double disk I/O utilization to 40%. Instead, the device is totally saturated and I/O service times skyrocket to tens of seconds. This fact must be kept in mind when designing systems using this type of hardware. Running the web server off its own spindle isolates the rest of the systems from its impact on the I/O subsystem.

Modern hard disks record in I/O zones, where the first logical addresses have the best transfer rates as they are located on the outer diameter of the platters. In an application where I/O capacity is more important than storage capacity, it is prudent to allocate and use only the beginning of the disk. The web server's virtual hard disk only uses the first 20GB of the 80GB disk it is stored on. Stroking the heads over only a small fraction of the platter surface and using only the fastest sectors on the disk significantly improves I/O response times, throughput, and I/O operations per second sustained by the drive.

6.2 Peer to Peer Network

In this section, we present and discuss the utilization and performance of our peer to peer network environment. We discuss observations on usage patterns and control traffic.

The total amount of data shared on the peer to peer network over time is shown in Figure 6.4 on the next page. The shape of the total share size graph roughly resembles the shape of the graph showing the number of online users in Figure 6.5 on the following page, although the online users graph's line is much smoother. This is partially caused by the large number of users who share nothing, but only log in, consume resources, and then log out. The number of big contributors is small enough that a small amount of contributors logging in or out result in a noticeable change on the total share graph.

A typical day-night pattern is noticeable, with usage sharply decreasing after 00:00 and increasing again after 08:00. The usage pattern follows the shape of the network utilization in Figure 6.6 on page 50. The very low traffic flow rate in the quiet window is attributable to both the quadratic scaling expressed in Equation 4.1 on page 19 and

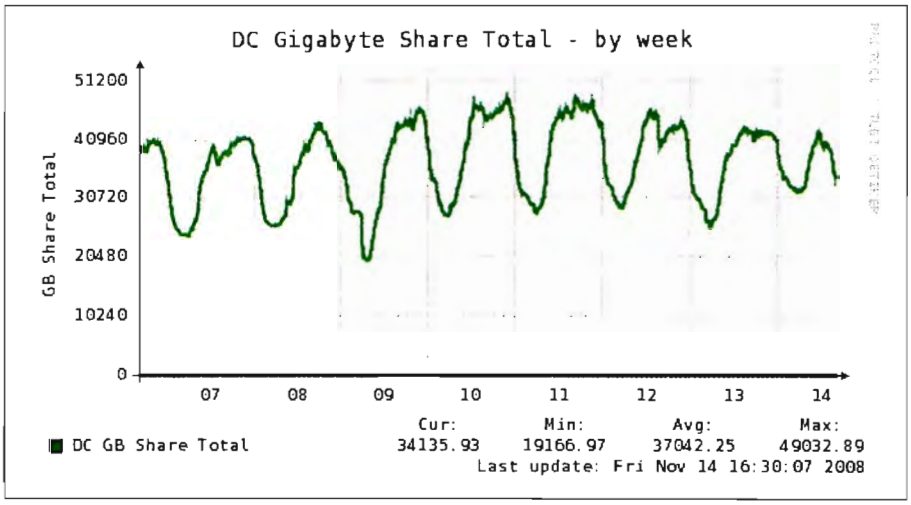

Figure 6.4: Peer to Peer Network: Gigabytes Shared

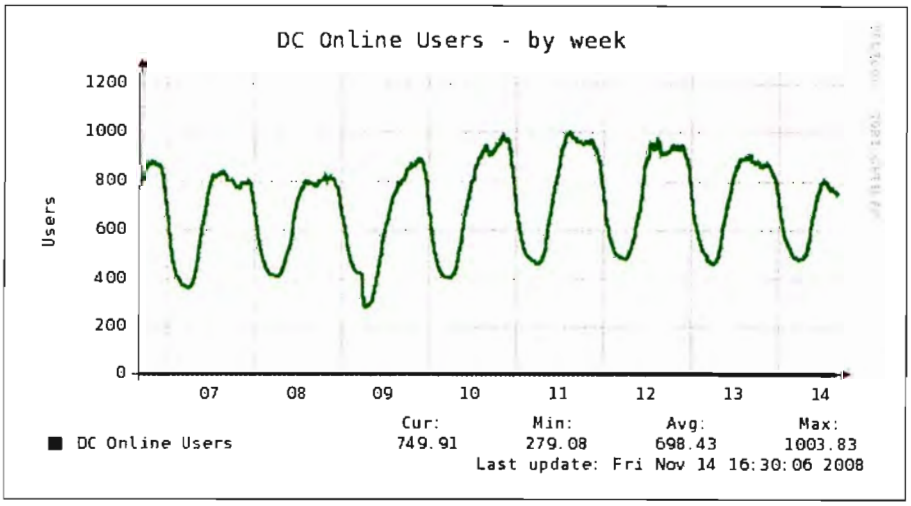

Figure 6.5: Peer to Peer Network: Total Online Users

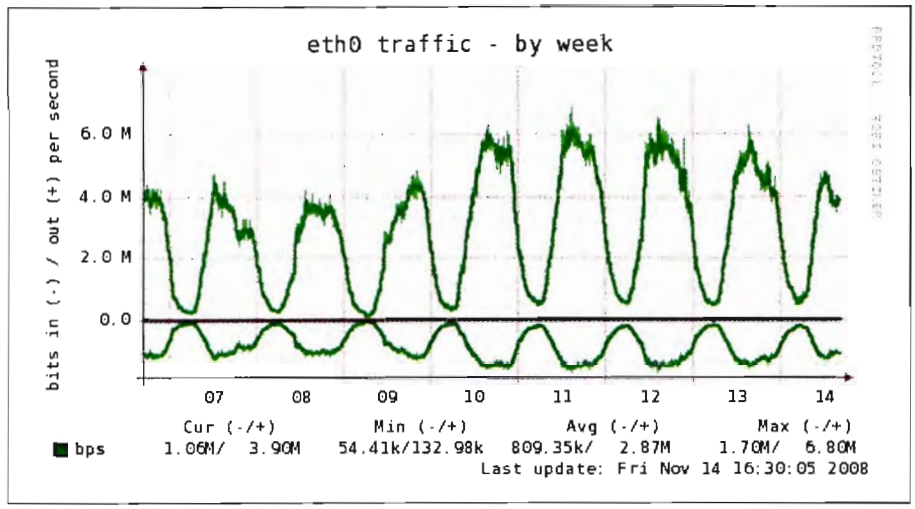

Figure 6.6: Peer to Peer Network: Direct Connect Server Traffic

a lower utilization factor k when most users who are still connected, are asleep.

Abnormalities in the graphs are mostly attributable to failures throughout the network. Power cuts and network outages cause users in the region affected by it to suddenly disconnect. Distinction can be made between the two causes, as clients can automatically reconnect after network service is restored, but clients gradually return after a power failure over a period of a few hours when users restore their connections manually. The effect of a power failure can be seen at 06:00 the morning of the 9th of November.

The Direct Connect network operated in this relatively private community, which is not accessible from the outside Internet, has not nearly reached its practical scalability limits. It is interesting to observe how well the Direct Connect hub functions in a virtualized environment, especially on the network front. Xen virtualizes network I/O extremely well with the paravirtual network drivers interfacing directly with the hypervisor. Virtual network latency remains in the order of a few microseconds even when in the region of a thousand users are connected and the hub is very busy transmitting data to all the connected clients.

An important operating system setting that must be modified is the maximum number of open files allowed per user. Unix-like operating systems allocate a file descriptor for every established network connection. The default maximum number of open files per user is 1024, which pose a problem when more than about a thousand clients want to connect to the hub. The `nofile` limit in `/etc/security/limits.conf` must be

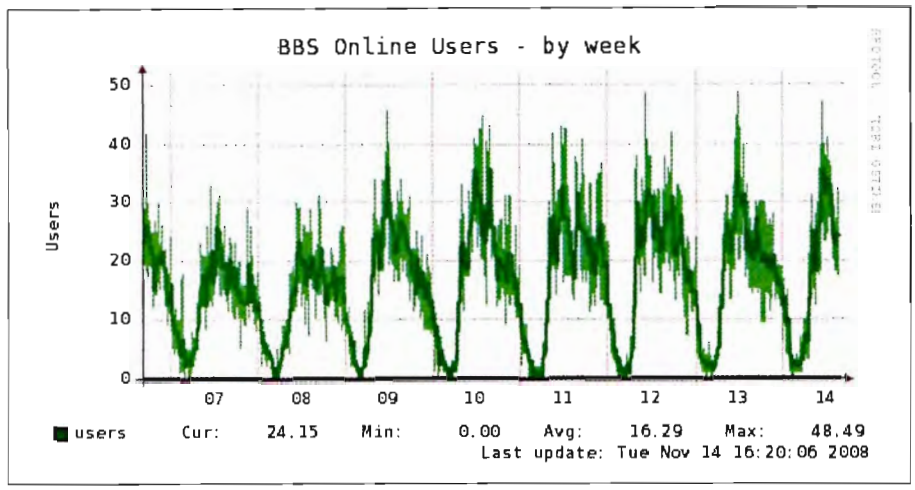

Figure 6.7: Web: BBS Active Users

increased in order to raise the limit with the `ulimit` command.

6.3 Interactive Web

In this section, we discuss usage patterns on our Web environment, including BBS usage and search engine usage. We also discuss database performance and web server traffic.

We graph the number of concurrently active users over a five minute window on the BBS in Figure 6.7. A day-night pattern is also observed, although it is interesting to note that the BBS is busier in the mornings, while the peer to peer network is more active in the evenings. The sudden influxes of users causing spikes in the graph is mostly caused by BBS posts or topics being announced on the public chat on the Direct Connect network. Larger spikes translate to more interesting topics that a larger number of active users simultaneously click on.

The graphing system collects data every five minutes. Half hour averages of the number of searches performed on the offline search Web interface during each graphing period is plotted in Figure 6.8 on the following page. The relatively low usage rate of the Web search interface can be attributed to the fact that most simple, popular search queries can be easily found with the built-in search function of the Direct Connect clients, and that fewer users use the Web community and are technically orientated enough to appreciate the additional functionality the Web search interface provides. The temporal

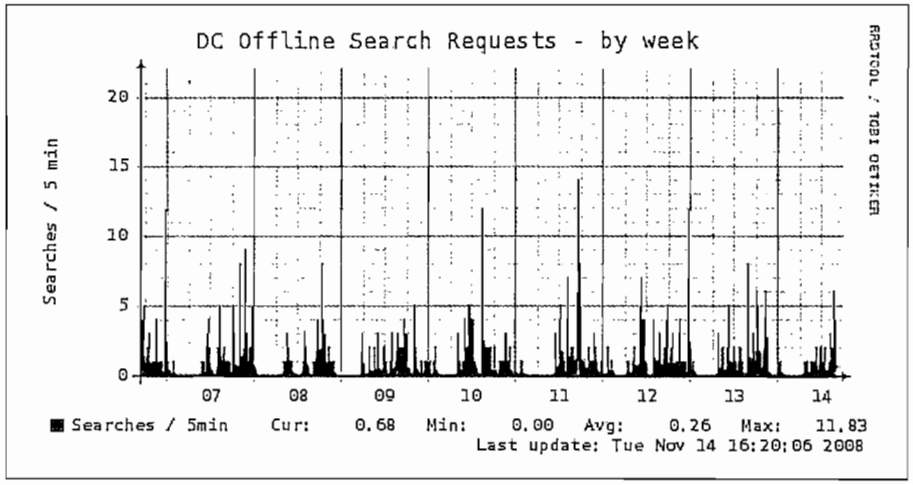

Figure 6.8: Web: Search Engine Searches

distribution of users looking for scarce files not immediately available on the network is also rather widespread.

A breakdown of the type of queries the Web server database is executing is shown in Figure 6.9 on the next page. A query cache has been configured for the database to speed up the processing of popular queries. All the services running on the Web server, of which the BBS and the offline search engine are the most popular, use this instance of MySQL.

The SQL server is the standard 64-bit build of MySQL version 5.0.45 shipping with CentOS version 5. The server has been configured with a large query cache to keep popular results in memory. A very large key cache is configured to keep most of the indexes in memory, in order to reduce I/O operations needed to locate data in the large offline search index database. Care is taken not to use more than about 500MB of the 2GB memory the Web server virtual machine has for the database process, to leave sufficient memory available for multiple Web server processes. The operating system also makes very efficient use of memory not being used by applications by using it for a disk cache. The efficiency of popular file system operations are therefore optimized, reducing I/O load on the storage subsystem.

Traffic flowing through the virtual network interface of the Web server is plotted in Figure 6.10 on the following page. The Web server used is the standard Apache version 2.2.3 shipping with CentOS version 5. Apache has been configured to compress the

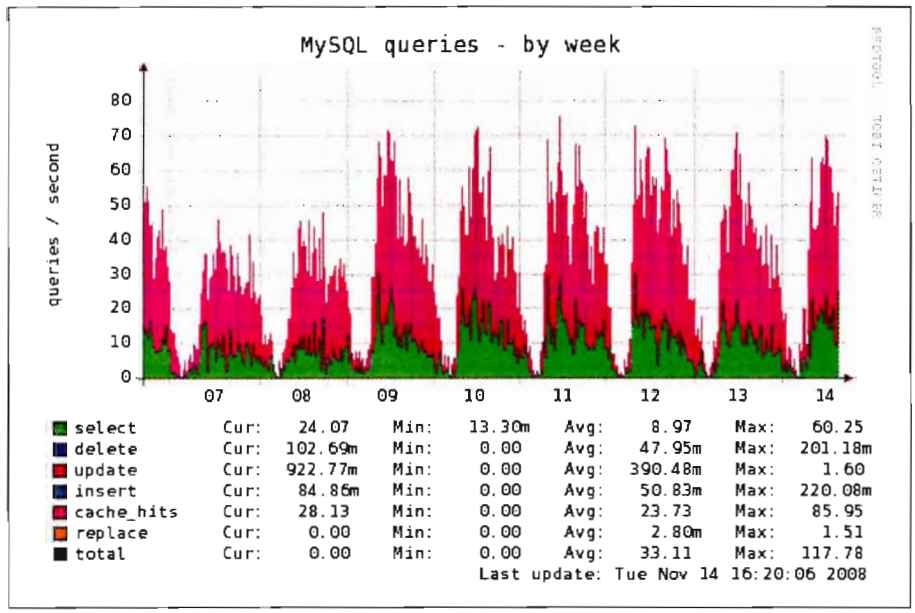

Figure 6.9: Web: Database Queries per Second

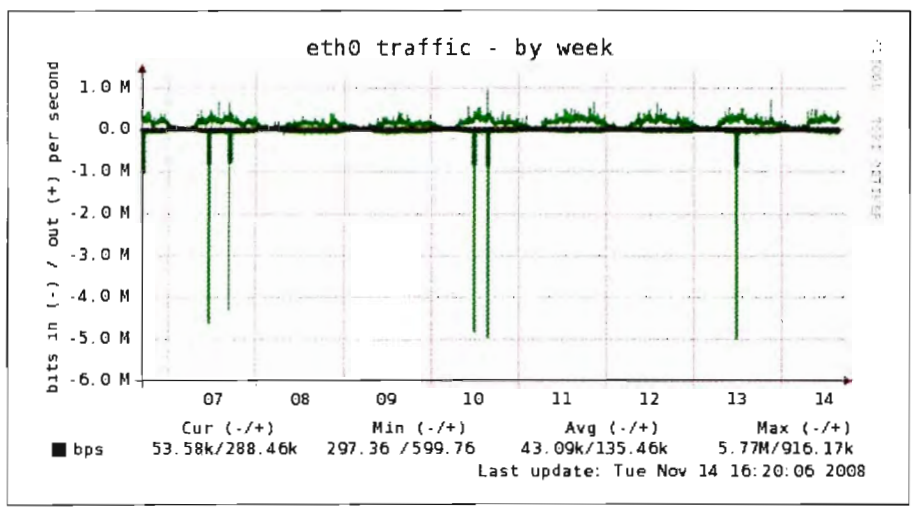

Figure 6.10: Web: Web Server Traffic

HTML it serves in both static and dynamic pages with gzip, resulting in a roughly 200% decrease in the volume of traffic flowing out. The occasional spikes in the traffic flowing in is caused by the uploading of XML file lists to be processed by the offline search index.

6.4 Robots and Automation

In this section, we present a qualitative view on our automated processes, discussing the effectiveness and performance thereof. We also present basic statistics on the performance of the programs we wrote, where applicable.

6.4.1 Offline Search Tools

Data describing files that are being shared is continuously collected by a robot crawling the peer to peer network. This robot pulls information about popular searches from various Web services and stores the results in delimited format, ready to be picked up by the scheduled database import process.

The data acquired by the robot is quickly loaded into a temporary database table just after it landed, to perform offline search queries against for the notifications service. This process runs half-hourly and does not impose any noticeable burden on the system otherwise, as the popular file data returned by the search robot is only a small subset of the total shared data. The search robot typically returns less than 100,000 rows upon every search batch. Multiple queries can be quickly dispatched against this temporary database, as the entire index is cached in memory and results are returned with very little computational effort.

From time to time, the file lists of every connected user are downloaded and batch processed by an XML file list processor sharing the same core logic as the search robot. Processing of this data must be optimized due to the sheer volume of the file list data.

The PHP shell script that processes the XML file lists digests about 200MB worth of bzip2-compressed XML files in just over three minutes, extracting about five million file entries at an effective input rate of about 27,000 file entries per second. Files that

are smaller than 2MB and do not have interesting extentions are ignored in order to filter out most of the useless files some users may share. After filtering, the rate at which records are output is reduced to about 6,500 entries per second. These records are validated and parsed for advanced versioning information with regular expression based filters. All the processing is done in a single pass.

Even when not considering that a high level interpreted language is used to process the data, this program performs extremely well. The workload is mostly CPU bound, therefore it can be very efficiently scheduled by the kernel not to affect the execution of other, more time critical processes.

6.4.2 Chat Filtering Tools

The chat messages in public chatrooms are analyzed in real time and messages passing an unwanted message probability threshold are stealthily filtered, by sending the message in question only to the offending user's chat session. Messages from users spamming the chat by repeatedly copying and pasting text are also detected by scanning for multiple occurrences of patterns.

Filtering out unwanted messages for everybody but the sender of the message works very well to keep the actions of malicious chat users under the metaphorical radar. Nobody is aggravated by the unsolicited chat content, and the source of the messages gets tired after some time. This effect has been successfully observed by viewing the filtered messages in debugging mode.

Another positive outcome of this method is that no user is repremanded in public when the spam filter catches his messages, which eliminates the chain reaction caused by this kind of intervention, all while managing the actual problem.

6.4.3 Community Currency

We have implemented a community currency called DC Dollars as an incentive mechanism to stimulate user participation on the peer to peer network. We are currently evaluating the effectiveness of the currency in the community by running the DC Dollar

system in the background, but not enforcing it. Therefore, users are able to use their DC Dollars into negative balances, in order for us to observe needs and trends.

We have also kindled an active community discussion on the topic of community currency in order to receive direct feedback and suggestions. The majority of users provided positive feedback and expressed valid concerns. Some users initially expressed themselves strongly against the system, but started to like it after they became accustomed with and began to see the value of the concept for themselves.

Valuable feedback raised valid concerns on the ability of obtaining DC Dollars in the first place. Lab users that are offline most of the time do not have the opportunity to obtain as much currency as users with permanent network connections. Users with ancient computers also struggle to contribute a lot to the peer to peer network. Perhaps the best suggestion was to extend the reach of DC Dollars to the Web environment, where a user's contributions on the BBS can be rated by the community and earn DC Dollars that way as well. Another suggested method that might be evaluated is the use of premium rate text messaging services on mobile phones to buy DC Dollars with.

Currently, about 1,500 registered users using the system have a DC Dollar balance. Most of these users have registered accounts to access the system from locations requiring registration, especially computer labs. Therefore, they typically do not contribute to the peer to peer network and most of them have negative balances.

Users with registered accounts who contribute even a little have positive DC Dollar balances, proving that the system will reward small contributors with a totally functional service experience. Large contributors acquire wealth with constantly increasing balances. A community member suggested the trading of a large sum of DC Dollars for a custom online status, as an incentive to make big contributions.

A lot of campus users making most of the contributions on the peer to peer network do not yet have registered accounts and are not eligible for DC Dollars. The onset of DC Dollars and the ability to use announcement channels and log in on the BBS to post messages will motivate these users to obtain a registered account.

The concept of DC Dollars is found to be sound, and has a lot of potential for integration into more systems, tracking and rewarding community participation over a broader spectrum of services.

6.5 User Feedback

We proceed to measure user experience by selecting 10 random users using all of our services to participate in a survey. Our random selection includes a representing collection of a combination of experienced and inexperienced, as well as active and inactive users.

We asked three questions in our survey covering key service aspects. In order to avoid neutral answers, each question has four possible answers on user satisfaction level, ranging from very poor to very good.

Users provided input on their experiences with the notifications service, the DC Dollars concept, and their perception of the overall technical quality of the service infrastructure. Users were also encouraged to share their ideas, views, and suggestions on each question in more detail.

Users are very excited about the notifications service, of which the results are presented in Figure 6.11. No negative feedback was provided. Users emphasize the accuracy and promptness of the service in their responses, indicating that it saves them time and effort.

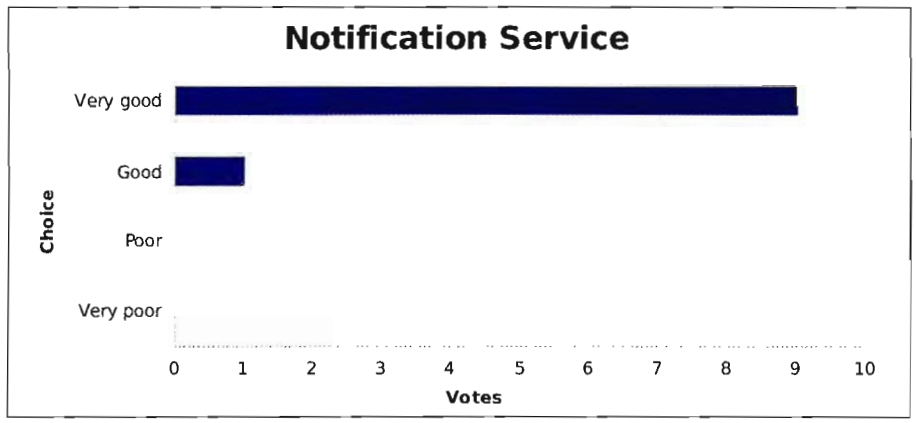

Figure 6.11: Feedback: Notifications Service

DC Dollars shown in Figure 6.12 on the following page, on the other hand, attracted both positive and negative responses. The main concern among users who dislike DC Dollars is that certain geographically disadvantaged users are not able to easily acquire the currency. Users who like the system indicate that it is a good incentive mechanism

to stimulate participation, as it is needed to qualify for the notifications service and causes competition among users.

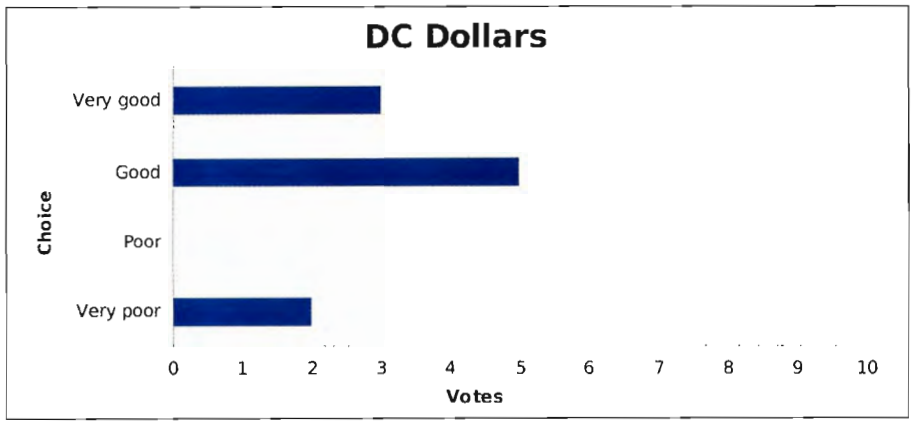

Figure 6.12: Feedback: DC Dollars

Figure 6.13 shows the responses on the perceived infrastructure quality. Users indicate that the service is always available when needed and that the service is fast and stable. Some users even state that our service is more stable than the University's official network services. The reliability of some of the peers connected to our peer to peer network negatively influence the perceived service quality of some users, although this has no relation to the quality of our infrastructure.

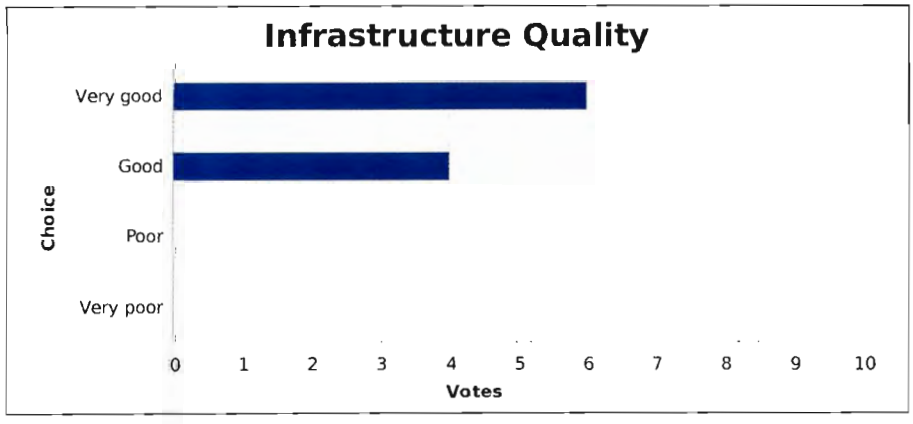

Figure 6.13: Feedback: Infrastructure Quality

6.6 Review

In this chapter, we discussed the inner workings of the infrastructure supporting our virtual community. We discussed the runtime statistics of the virtualized platform, peer to peer network, and Web environment in detail. We also discussed the success of some of the automated processes we implemented in terms of software performance and strategic approach.

In the next chapter, we proceed to conclude our work by evaluating the success of our vision and strategies. We combine the technical successes we discussed here with our management strategies, to evaluate the success of our virtual community.

7 Conclusion

We conclude our work by reflecting upon our established virtual community.

Two main themes contribute towards our success, namely infrastructure and management style. We provide closing thoughts on both, as well as on the road ahead.

7.1 Technology and Infrastructure

Our virtual community is based on Verlihub, a Direct Connect peer to peer network environment, and phpBB2, a Web-based community discussion forum. We have integrated these technologies to enable the use of a single user account system wide. We developed our own incentive mechanism to stimulate participation in the peer to peer network, wrote our own advanced Web-based search engine for the peer to peer network, developed an instant group announcement collaboration system, and added features for effective automated offline searching.

Direct Connect hubs and Web forums can be set up by anyone, there is nothing special about these technologies. The integration of the two systems is also rather simple, due to the open source nature of the software. The element that makes our community infrastructure unique is our advanced search features. The Web-based search engine returns accurate results on file versioning. No other virtual community infrastructure we are aware of has the ability to extend its search features to automated offline searching and notifications.

The servers running the Web and peer to peer applications are hosted on virtualized infrastructure on top of the Xen hypervisor. In our case, it is a sound strategic choice to consolidate our servers onto a virtualized platform. One physical server running a

collection of virtual servers consumes less electricity, requires less space, and allows for flexible virtual infrastructure configuration, consistent server backups even while the virtual servers are running, legacy server hosting on modern hardware, and relatively fast disaster recovery. The combined workload generated by our virtual servers does not exceed the capacity of the physical hardware underneath, therefore the choice of virtualizing our platform does not present trade-offs in terms of performance or scalability.

The inherent poor scalability of the NMDC protocol used by the Direct Connect network places a practical upper bound for the reach of the peer to peer network. Linking hubs together does not scale well beyond a certain point, after it becomes impractical to add more servers. In situations where a Direct Connect network needs to scale faster than natural advances in computing and networking technology, research can be done to determine if star-linking Direct Connect hubs through a dedicated link hub scales better than the mesh-linking that is currently possible. Direct Connect scaling is not yet a concern for our community, since the hub can co-exist along with all the other virtual servers on the same platform while only using a fraction of the available bandwidth and processing power.

Our Web-based search tools are designed with scalability in mind. Should the need arise to split the search engine out to multiple servers to cope with demand, the database can either be replicated entirely and load balanced, or sharded across multiple servers at application level. The Web front end can also easily scale out behind a load balancer. Currently, the size of the database allows the most frequently used parts of the indexes to fit in memory. Should the database size increase significantly, either a storage area network with fast I/O capability will be needed, or the database could be sharded across multiple database servers. The latter scales better, as the database size is divided among the servers in the cluster. Currently, we have not approaced the scalability limit of our Web environment. Apart from needing its own disk spindle, it co-exists well with the other virtual servers on the same platform.

Our Web environment is able to scale out to vast dimensions, while our peer to peer environment will experience scalability problems at a certain limit. Considering the current server utilization, this scalability limit is above the total potential number of users, and therefore not a problem.

7.2 Management Philosophy

Proper infrastructure alone that is able to scale out well beyond the possible size of the community does not guarantee that a community will be successful. Like an empty bee hive, community infrastructure without an active community is worthless.

New users joining the community have their own unique experiences while gaining, discussing, and exchanging information, ideas, and philosophies in their own unique way. If the community is sufficiently large, where the conscience of the community exceeds a critical mass, the community already moving in a direction does not deviate from course due to the addition of a newcomer. The culture and management of such a large community influences the experience of the newcomer, while the experience still remains unique.

From a management perspective, the informative users are a great asset. These users share their knowledge with users asking questions, and play an active role in defining the consciousness of the community. Informative users have the community's best interests at heart and are very interested in new developments and strategies. Keeping these users updated ensures that the consciousness of the community is steered in the right direction and that the correct information filters through to the rest of the community.

Social users keep the chat systems alive, continuously interacting with and making new friends. Active social discussions on topics of common interest to the community gives newcomers an immediate sense of belonging.

Contributive users maintain a steady flow of unique information and ideas into the community, which information hungry individuals absorb. While not all contributive users contribute unique information, others contribute by mirroring information to assist in distributing existing information and ideas.

Returning to our bee hive example, self sustenance is only achieved when enough larvae are nurtured to grow into a new generation of adult bees, keeping the swarm alive. The same principle applies to our virtual community: as veteran users move out of the community's geographical reach, fresh newcomers arrive that, through an evolutionary process of continuous interaction with the community, grow into veteran users. The conscience of the community exceeding a critical mass makes this process of constant

renewal possible. It is our management strategy of guiding the conscience of the community that results in self-sustenance, and ultimately makes the community successful.

7.3 The Road to Pure Collaboration

Our work in building our community is not complete, and probably never will be.

The application of our incentive mechanisms can be extended across platforms and integrated into all our current and future services. Diversification on a common incentive mechanism throughout various aspects of the community infrastructure will stimulate participation and amplify the effectiveness of the incentive mechanism.

Our automated systems that index the peer to peer network can be used to automatically feed distributed automated content cache clusters. Such cache clusters can be built from end-of-life commodity computers, and strategically connected to the campus network. Caches mirroring information and ideas ensures the availability of popular data and enhances the rate at which data can be exchanged over the network.

Users on community networks have slow and unreliable network access. Transferring large amounts of data over these networks is not feasible. Browsing our Web environment gives, due to the low volume of traffic generated, an acceptable user experience. A possible solution that can add value to the experience of such users is to enable enqueuing of files from the Web environment to removable drives. An auto-generated file with download instructions can be downloaded from the Web and saved on the removable drive to be downloaded to, which can be plugged into purpose-built downloading stations at strategic locations. This way, users can download information while, for example, enjoying their favourite meal at a restaurant on campus.

The process of empowering the community with knowledge and skills on new technology and infrastructure is also never ending. It is strategically advantageous to transfer skills to a group of veteran users, who not only appreciate the additional functionality, but also transfer it to the rest of the community.

At this stage, our infrastructure is working flawlessly. The community is self-sustainable. Our future management goal is to intensify the community dynamics by transforming

Bibliography

[1] Virtual community. Wikimedia Foundation, Inc. [Online]. Available: http://en.wikipedia.org/wiki/Online_communities

[2] H. Rheingold, *The Virtual Community: Homesteading on the electronic frontier.* Addison Wesley, 1993.

[3] J. Donath, *Identity and Deception in the Virtual Community.* London: Routledge, 1999, ch. 2, pp. 29–58. [Online]. Available: http://smg.media.mit.edu/people/Judith/Identity/IdentityDeception.html

[4] A. Kim, *Community Building on the Web: Secret Strategies for Successful Online Communities.* Addison Wesley, 2000.

[5] S. Rafaeli, G. Ravid, and V. Soroka, "De-lurking in virtual communities: a social communication network approach to measuring the effects of social and cultural capital," in *Proceedings of the 37th Hawaii International Conference on System Sciences*, 2004.

[6] J. Koh, Y.-G. Kim, B. Butlet, and G.-W. Bock, "Encouraging participation in virtual communities," *Communications of the ACM*, vol. 50, no. 2, pp. 68–73, February 2007.

[7] L. Ramaswamy and L. Liu, "Free riding: A new challenge to peer-to-peer file sharing systems," in *Proceedings of the 36th Hawaii International Conference on System Sciences.* Georgia Institute of Technology, 2003.

[8] K. G. Anagnostakis and M. B. Greenwald, "Exchange-based incentive mechanisms for peer-to-peer file sharing," in *Proceedings of the 24th International Conference on Distributed Computing (ICDCS04)*, 2004.

[9] B. Yu and M. P. Singh, "Incentive mechanisms for peer-to-peer systems," in *Proceedings of the 2nd International Workshop on Agents and Peer-to-Peer Computing*, 2003.

[10] T.-W. J. Ngan, A. Nandi, A. Singh, D. S. Wallach, and P. Druschel, "On designing incentives-compatible peer-to-peer systems," in *Proceedings of the 2nd Bertinoro Workshop on Future Directions in Distributed Computing (FuDiCo 2004)*, 2004.

[11] K. Lai, M. Feldman, I. Stoica, and J. Chuang, "Incentives for cooperation in peer-to-peer networks," in *Proceedings of the Workshop on Economics of Peer-to-Peer Systems*, 2003.

[12] M. Smith and P. Kollock, "The economies of online cooperation: Gifts and public goods in cyberspace," *Communities in Cyberspace*, 1999.

[13] C. Elliott. (2008) 5 strategies for building an online community. Microsoft Corporation.

[14] J. Broß, H. Sack, and C. Meinel, "Encouraging participation in virtual communities: The "it-summit-blog" case," in *Proceedings of IADIS International Conference of e-Society 2007*, 2007.

[15] Arpanet. Wikimedia Foundation, Inc. [Online]. Available: http://en.wikipedia.org/wiki/ARPANET

[16] Usenet. Wikimedia Foundation, Inc. [Online]. Available: http://en.wikipedia.org/wiki/USENET

[17] Internet forum. Wikimedia Foundation, Inc. [Online]. Available: http://en.wikipedia.org/wiki/Internet_forum

[18] Social network. Wikimedia Foundation, Inc. [Online]. Available: http://en.wikipedia.org/wiki/Social_network

[19] Mmorpg. Wikimedia Foundation, Inc. [Online]. Available: http://en.wikipedia.org/wiki/MMORPG

[20] A. Achleman. (2008) Virtualization trends, options, and adoption. [Online]. Available: http://images.globalknowledge.com/wwwimages/whitepaperpdf/WP_Achelman_virtualizationTrends.pdf

[21] "Virtualization overview," VMware, Inc., 2006.

[22] M. Peters and S. Duplessie, "Virtualization - management considerations and implications for virtual infrastructures," The Enterprise Strategy Group, Inc., October 2008.

[23] D. L. Shinder. (2008, February) 10 things you should know about virtualization. [Online]. Available: http://i.i.com.com/cnwk.1d/i/tr/downloads/home/dl_10_things_virtualization_deb.pdf

[24] R. Vanover. (2008, February) 10 more things you should know about virtualization. [Online]. Available: http://i.i.com.com/cnwk.1d/i/tr/downloads/home/dl_10_things_virtualization.pdf

[25] S. Soltesz, H. Pötzl, M. E. Fiuczynski, A. Bavier, and L. Peterson, "Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors," in *Proceedings of EuroSys 2007*, 2007, pp. 275–288.

[26] Solaris. Sun Inc. [Online]. Available: http://www.opensolaris.org/os

[27] Top ten considerations. Parallels. [Online]. Available: http://i.i.com.com/-cnwk.1d/html/itp/parallels_Top10_WP_148305.pdf

[28] "An introduction to os virtualization and virtuozzo," SWsoft, July 2006.

[29] Usermode linux. [Online]. Available: http://user-mode-linux.sourceforge.net/

[30] T. Deshane, D. Dimatos, G. Hamilton, M. Hapuarachchi, W. Hu, M. McCabe, and N. Matthews. (2006) Performance isolation of a misbehaving virtual machine with xen, vmware and solaris containers. Clarkson University. [Online]. Available: http://people.clarkson.edu/jnm/publications/isolationOfMisbehavingVMs.pdf

[31] Vmware workstation. Sun Microsystems, Inc. [Online]. Available: http://www.vmware.com/

[32] Sun xvm virtualbox. Sun Microsystems, Inc. [Online]. Available: http://www.sun.com/software/products/virtualbox

[33] Kernel-based virtual machine. KVM-Linux. [Online]. Available: http://www.linux-kvm.com

[34] Qemu. Fabrice Bellard. [Online]. Available: http://bellard.org/qemu

[35] Microsoft virtual pc. Microsoft Corporation. [Online]. Available: http://www.microsoft.com/windows/products/winfamily/virtualpc/default.mspx

[36] "A performance comparison of hypervisors," VMware, Inc., Tech. Rep. PS-004-INF-01-002, 2007.

[37] "Guide to virtual infrastructure implementation," VMware, Inc., 2007.

[38] Vmware esx server. VMware, Inc. [Online]. Available: http://www.vmware.com/products/vi/esx

[39] M. Harringer. (2004) Xen - the art of virtualization. [Online]. Available: http://cs.uni-salzburg.at/ck/teaching/CS-Seminar-Summer-2004/marcus.pdf

[40] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," *SOSP 2003*, pp. 164–177, 2003.

[41] *Toward Xen Virtualization on Every Server*, CTO, XenSource, Inc. Intel Corporation., 2006.

[42] "Virtualization in the data center," Novell, Inc., 2006.

[43] Xen. Citrix Systems, Inc. [Online]. Available: http://www.xen.org

[44] M. Andreou and N. Walji, *Installing and configuring Xen*, Internet Systems and Storage Laboratory, HP Laboratories Bristol, 2005.

[45] Microsoft hyper-v. Microsoft Corporation. [Online]. Available: http://www.microsoft.com/servers/hyper-v-server/default.mspx

[46] J. Nakajima and A. K. Mallick, "Hybrid-virtualization - enhanced virtualization for linux," in *Proceedings of the Linux Symposium*. Intel Open Source Technology Center, June 2007.

[47] K. Adams and O. Agesen, "A comparison of software and hardware techniques for x86 virtualization," *ASPLOS*, October 2006. [Online]. Available: http://www.vmware.com/pdf/asplos235_adams.pdf

[48] File sharing. Wikimedia Foundation, Inc. [Online]. Available: http://en.wikipedia.org/wiki/File_sharing

[49] B. Yang and H. Garcia-Molina, "Comparing hybrid peer-to-peer systems," in *Proceedings of the 27th VLDB Conference*, 2001, pp. 561–570.

[50] Direct connect (file sharing). Wikimedia Foundation, Inc. [Online]. Available: http://en.wikipedia.org/wiki/Direct_connect_file-sharing_application

[51] Advanced direct connect. Wikimedia Foundation, Inc. [Online]. Available: http://en.wikipedia.org/wiki/Advanced_Direct_Connect

[52] C. Soldani, "Peer-to-peer behaviour detection by tcp flows analysis," May 2004.

[53] Fasttrack. Wikimedia Foundation, Inc. [Online]. Available: http://en.wikipedia.org/wiki/FastTrack

[54] Kazaa. Wikimedia Foundation, Inc. [Online]. Available: http://en.wikipedia.org/wiki/KaZaA

[55] Grokster. Wikimedia Foundation, Inc. [Online]. Available: http://en.wikipedia.org/wiki/Grokster

[56] imesh. Wikimedia Foundation, Inc. [Online]. Available: http://en.wikipedia.org/wiki/IMesh

[57] Gnutella. Wikimedia Foundation, Inc. [Online]. Available: http://en.wikipedia.org/wiki/Gnutella

[58] M. Ripeanu, "Peer-to-peer architecture case study: Gnutella network," University of Chicago, Tech. Rep. TR-2001-26, 2001.

[59] A. Oram, Ed., *Peer to Peer: Harnessing the Power of Disruptive Technologies*, 1st ed. O'Reilly Media, Inc., March 2001.

[60] N. Borisov and J. Waddle, "Anonymity in structured peer-to-peer networks," EECS Department, University of California, Tech. Rep. CSD-05-1390, December 2003.

[61] V. Scarlata, B. N. Levine, and C. Shields, "Responder anonymity and anonymous peer-to-peer file sharing," in *Proceedings of the IEEE International Conference on Network Protocols (ICNP)*, 2001.

[62] T. Chothia and K. Chatzikokolakis, "A survey of anonymous peer-to-peer file-sharing," in *Proceedings of the IFIP International Symposium on Network-Centric Ubiquitous Systems (NCUS 2005);, Lecture Notes in Computer Science*. Springer, pp. 744–755.

[63] I2p. Wikimedia Foundation, Inc. [Online]. Available: http://en.wikipedia.org/wiki/I2p

[64] Gnunet. Wikimedia Foundation, Inc. [Online]. Available: http://en.wikipedia.org/wiki/GNUnet

[65] Freenet. Wikimedia Foundation, Inc. [Online]. Available: http://en.wikipedia.org/wiki/Freenet

[66] K. Bennett and C. Grothoff, "Gap – practical anonymous networking," 2002.

[67] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong, "Freenet: A distributed anonymous information storage and retrieval system," in *Proceedings of the Workshop on Design Issues in Anonymity and Unobservability*, 2000.

[68] I. Clarke, S. Miller, T. W. Hong, O. Sandberg, and B. Wiley, "Protecting free expression online with freenet," *IEEE Internet Computing Journal*, pp. 40–49, January-February 2002.

[69] I. Clarke. (2003, July) Freenet's next generation routing protocol. [Online]. Available: http://freenetproject.org/ngrouting.html

[70] ——. (2003, July) The philosophy behind freenet. [Online]. Available: http://freenetproject-.org/philosophy.html

[71] Introducing i2p. I2P. [Online]. Available: http://www.i2p2.de/techintro.html

[72] I2p compared to other anonymous networks. I2P. [Online]. Available: http://-www.i2p2.de/how_networkcomparisons

[73] Introduction to how i2p works. I2P. [Online]. Available: http://www.i2p2.de/-how_intro

[74] Miro (software). Wikimedia Foundation, Inc. [Online]. Available: http://en.wikipedia.org/wiki/Miro_(software)

[75] Peercast. Wikimedia Foundation, Inc. [Online]. Available: http://en.wikipedia.org/wiki/PeerCast

[76] V. N. Padmanabhan, H. J. Wang, and P. A. Chou, "Resilient peer-to-peer streaming," Microsoft Research, Redmond, WA, Tech. Rep. MSR-TR-2003-11, March 2003.

[77] D. Tsoumakos and N. Roussopoulos, "A comparison of peer-to-peer search methods," in *Proceedings of the International Workshop on the Web and Databases (WebDB)*, June 2003.

[78] Dc++ dchublist - the list for nmdc hubs. dchublist.com. [Online]. Available: http://www.dchublist.com/?page=show&id=12636

[79] Verlihub project forum. [Online]. Available: http://forum.verlihub-project.org

[80] Moore's law. Wikimedia Foundation, Inc. [Online]. Available: http://en.wikipedia.org/wiki/Moore's_law

[81] R. F. Group, "Tco for application servers: Comparing linux with windows and solaris," Robert Frances Group, Inc., 2005.

[82] Kernel.org. Linux Kernel Organization, Inc. [Online]. Available: http://www.kernel.org/

[83] Backporting of security fixes. Red Hat, Inc. [Online]. Available: http://www.redhat.com/security/updates/backporting/?sc_cid=3093

[84] Red hat. Red Hat, Inc. [Online]. Available: http://www.redhat.com/

[85] Centos - the community enterprise operating system. [Online]. Available: http://www.centos.org/

[86] Scientific linux. [Online]. Available: https://www.scientificlinux.org/

[87] White box enterprise linux. [Online]. Available: http://whiteboxlinux.org/

[88] About centos. CentOS Ltd. [Online]. Available: http://www.centos.org/-modules/tinycontent/index.php?id=2

[89] "Virtualization: Architectural considerations and other evaluation criteria," VMware, Inc., 2005. [Online]. Available: http://www.vmware.com/pdf/virtualization_considerations.pdf

[90] Linux virtualization - technologies, trends, news and howtos. [Online]. Available: http://linuxvirtualization.com

[91] Mrtg. Tobias Oetiker. [Online]. Available: http://oss.oetiker.ch/mrtg/doc/mrtg-rrd.en

[92] `Cacti. The Cacti Group. [Online]. Available: http://www.cacti.net

[93] Munin. Edgewall Software. [Online]. Available: http://munin.projects.linro.no

[94] Sintrex. Sintrex Integration Services (Pty) Ltd. [Online]. Available: http://www.sintrex.com

[95] B. Schneier, *Secrets and Lies: Digital Security in a Networked World*, paperback ed. John Wiley & Sons, January 2004.

[96] S. McClure, J. Scambray, and G. Kurtz, *Hacking Exposed: Network Security Secrets and Solutions*, 3rd ed. McGraw-Hill, 2001.