Providing Students Universal Access to a Centralized, Graphical Computing Environment

Michael H. Goldwasser
Dept. Mathematics and Computer Science
Saint Louis University
221 North Grand Blvd
St. Louis, Missouri 63103-2007
goldwamh@slu.edu

David Letscher
Dept. Mathematics and Computer Science
Saint Louis University
221 North Grand Blvd
St. Louis, Missouri 63103-2007
letscher@slu.edu

ABSTRACT

We investigate the use of a thin-client based configuration in providing students with universal access to a centralized, graphical computing environment. The primary goal is to enable students to work effectively from arbitrary locations and computing platforms, while always interacting with the consistent environment seen in tightly controlled labs.

Categories and Subject Descriptors

K.3.1 [**Computers and Education**]: Computer Uses in Education; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*client/server*; K.6.4 [**Management of Computing and Information Systems**]: System Management—*centralization/decentralization*

General Terms

Management,Performance,Reliability

Keywords

Computing Environment, Thin-client, Linux

1. INTRODUCTION

Students in today's world have access to highly-responsive, graphical computing environments for use both in and out of the classroom. The most recent EDUCAUSE core data service summary reports that the median institution owns or leases 0.42 computers per student, with this metric even higher at both baccalaureate and doctoral institutions [3]. In addition, the median such baccalaureate or doctoral institution reports that 80% of students possess their own computer. In fact over 8% of those institutions have an across the board policy ensuring that all students possess a computer, with significantly more institutions making such a requirement at least for students in certain departments, "most likely in engineering, technical and business programs." Network connectivity is equally ubiquitous for today's students. Over 95% of baccalaureate and 99% of doctoral institutions provide high-speed network connections in residence halls. To estimate network access for those students living off-campus, we note that an FCC survey reports that 26.0 million U. S. households had broadband access by the end of 2003 [4].

In this paper, our concern is not students' overall access to computing, but rather their access to a single, cohesive environment for use in course work. The combined computing resources available to students might be manifested as a heterogeneous collection of computing devices, encompassing a wide range of platforms, operating systems, software configurations, network connectivity and performance characteristics. Their physical locations are equally heterogeneous, including classrooms, labs, and both on- and off-campus residences. Yet for many disciplines, the computing environment must be highly customized for the required work, so as to include software, libraries and data, some of which may be proprietary and licensed [2, 6]. An academic department may provide this access via classrooms and labs under the direct control of the department. The department likely has only loose influence over additional classrooms and labs administered by a campus-wide IT unit, and no direct control over laptop and personal computers used by students both on and off campus.

A student's learning experience is greatly dependent on whether successful completion of work in a discipline requires direct use of such tightly controlled environments, or conversely is achievable using the broader, available computing resources. With strict reliance on the controlled environment, a student may be limited by the physical location or the available open times for those systems. Yet depending on the computing needs of the discipline, using the broader systems may be infeasible or, at best, inconsistent when compared to the tightly controlled environment.

In this paper, we discuss our experiences in developing and deploying a system, providing a tightly controlled computing environment which is universally accessible from the broader computing resources. Our approach is to use the heterogeneous computing resources as thin clients which connect via the web to provide a consistent, graphical computing environment on a centralized and tightly controlled server. Our students benefit greatly from the flexibility of a computing environment which is consistent both within a closed lab

ITiCSE'05, June 27–29, 2005, Monte de Caparica, Portugal.

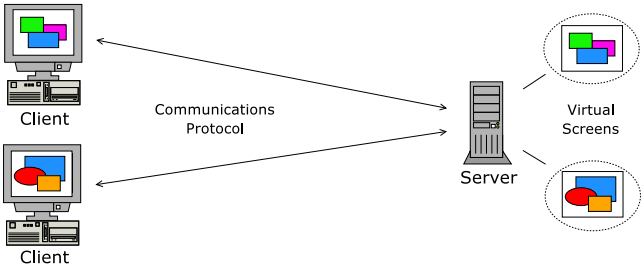

Figure 1: Thin-client computing

setting as well as when accessed from other computers and locations. At the same time, our department and institution benefit from the scalability and cost-effectiveness of the configuration. The entire software suite is free of cost and the existing computing resources on campus can be used as clients, irrespective of their computing platforms.

2. THIN-CLIENT COMPUTING

Thin-client computing divides the computing environment across two components: a server and a client. In its purest form, the client computer acts solely as a display and input device, with all remaining computation performed on the server. The major benefit of such a configuration is that the majority of the computation is done on the server side, which can be tightly controlled. This also provides consistency since any changes to the user environment can be maintained at the server. At the same time, far fewer assumptions need be made about the client machines because of their limited responsibility as "thin" clients. As a result, client software can often be designed for several operating systems and those clients need not be state-of-the-art machines. In essence, such a configuration provides the ideal combination between the consistency afforded by a single system image, yet extends the universality of access to include loosely controlled or uncontrolled computers.

The key aspect in designing and implementing a successful configuration involves the precise breakdown of responsibilities between client and server and the communication protocol between the two. These design decisions affect how well the system performs in different settings as well as how easily client software can be adapted to varying operating systems.

For example, thin-client systems may differ in the type of information they transmit. Some send high-level graphics primitives to the client, such as "move this window there" or "display this text in that window." Others send low-level primitives like "color this pixel red and that one green." In addition, some protocols incorporate compression of these primitives. Another difference seen in thin-client designs involves the granularity of the communication. Does the protocol use *eager updating* and send all updates immediately or does it conserve bandwidth and use *lazy updating*. Similarly, differences exist based on the timing of the communication. In particular, is updating triggered by a *server-push* or a *client-pull*. These choices can greatly affect bandwidth usage and communication latency, which are crucial factors in the quality of the graphical user environment.

2.1 Other Common Configurations

For comparison, we review other common practices used in providing students with access to computing resources. We note that the responsibility for providing and administering such resources might lie at the departmental level or with a larger institutional IT unit. Furthermore, our discussion will distinguish between what we term *primary usage*, taking place within scheduled class meetings, and *secondary usage*, completed on a students' own time.

Collection of independent computers.

Though physically located in a common place, such as a classroom or a lab, each computer in such a facility operates independently. The collection may contain multiple platforms or one relatively consistent platform. While administration is relatively straightforward for a small collection of machines, running a larger lab or labs can be extremely time consuming. Updating software and maintaining consistency between machines is time-consuming and error prone. Without centralized data storage, students must generally rely on removable media for saving their work and transferring it from one location to another.

In addition, serious limitations can result from any configuration which relies on physical space which must be shared. If multiple classes require access to the same space, they must be scheduled appropriately. Furthermore, if students' secondary usage requires access to this space, that usage must be scheduled around the remaining availability of the space. This may have a detrimental impact.

Independent computers with networked file system.

A networked file system can be employed to provide centralized data storage which is accessible from all of the devices. This approach is used to resolve the lack of such storage in the previous configuration. Generally, a dedicated filesystem is quite reliable, though this configuration reverts to the previous if a failure occurs in the filesystem or on the underlying network.

Collection of computers with a single system image.

Just as a networked filesystem can be used to share data, it can be used to distribute software including much of the operating system. In this way, all computers in a collection can rely on such a centralized "system image" yet run the software locally. When implemented properly, this is an attractive configuration [12]. By centralizing the software distribution, consistency is more readily achievable. Software installations and upgrades need only be performed centrally. At the same time, since software is executed locally, it may take advantage of the local processor and other devices.

The difficulty of properly implementing such a configuration is the greatest challenge. Furthermore, if more than one platform must be supported, a distinct system image must be maintained for each.

Laptop computing.

Having been a source of secondary computing access for students for quite some time, laptops are finding greater use as the *primary* computing platform, with a growing number of institutions requiring them. The major advantages to such a configuration are that the concern over access to shared resources no longer exists, and that for an individual stu-

dent, consistency in computing environment is achieved as the computer travels with the student [9].

The obvious challenge for this configuration is the unavoidable lack of consistency between students' computing platforms. Even if students were required to start with a standard platform, software installations and upgrades would not be uniform. Also, different departments and college may have incompatible requirements for this "standard configuration" which may necessitate the continuing needs of departments to provide specialized labs [15]. Furthermore, software licensing may result in an increased burden on the community. Specialized labs often contain significant proprietary software, which would be prohibitively expensive to install on individual laptops [6].

Text-based remote access to computing.

Remote access to a centralized server can be provided through a text-based interface from another computer, using a common protocol such as telnet or ssh. This approach is a simple example of thin-client computing, though it is clearly limited to text-based applications.

3. OUR INITIAL SYSTEM DEPLOYMENT

Prior to 2003, our students at Saint Louis University used a mixture of all of the access methods mentioned in the previous section. These included a variety of computer labs running Windows XP, Mac OS 9.1, Mac OS X and Red Hat Linux. For remote access students had access to a Unix server via telnet and ssh. The software used and the access methods depended on the course and the student.

During the 2003–2004 academic year at Saint Louis University, we undertook the development of a prototype, using thin-client computing to provide students with both primary and secondary access to a graphical computing environment. Our intention was to provide our students with a more uniform computing environment that could expand upon existing facilities in a cost-effective manner.

Lai and Nieh [5] compare existing thin-client technologies, measuring responsiveness and image quality as used over a wide-area network. The studied systems include X Windows, Virtual Network Computer (VNC), Microsoft Terminal Services (Remote Desktop) and Citrix MetaFrame. Both X Windows and VNC are open source software, whereas Remote Desktop and Citrix MetaFrame are proprietary. Lai and Nieh conclude that over a high-performance, wide area network (i.e., Internet2), a thin-client which uses compressed, low-level graphics primitives with eager server-push updating is desirable. They suggest that lazy updating may become desirable over a slower network. In particular, VNC was on par with Remote Desktop and Citrix MetaFrame, and superior in some settings. The original X Windows system was significantly worse than the other alternatives.

We chose VNC as the most suitable basis for our system. A variant of VNC called TightVNC [16] was installed on an existing dual processor Dell workstation running Gentoo Linux. For six small courses, it was used with a maximum of 15 students online at any given time. Class meetings were held in one of two computer classrooms, the first with iMacs running Mac OS X and the second with PowerMacs running Mac OS 9.1. Outside of class, students also accessed the system from open labs running Windows XP as well as from personal computers located both on and off campus.

To begin the 2004–2005 academic year we expanded our deployment, replacing the original server with a dedicated four server cluster. With a hardware expenditure of approximately $16,000, we have a setup with a capacity of 75 to 100 simultaneous users running a graphically desktop remotely. The machines act as their own distributed filesystem. Each server will store a portion of the files and distribute them through NFS [8]. To provide redundancy, each server's files will be mirrored on one of the other servers using DRBD [1]. This setup will ensure that if one of the servers fails, then the server mirroring its files can substitute until the former is back online. Our approximate system configuration is presented in Figure 2. This provides redundancy to ensure that if any one machine fails, the cluster remains available, though with slightly lower user capacity.

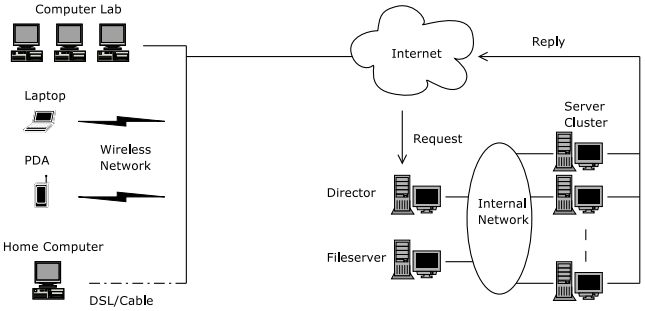

Figure 2: Suggested system configuration

During this same period of time we worked on modifying the VNC server and client software to more closely meet the needs of our students. These changes include improving security, increasing responsiveness and lowering bandwidth requirements. Initial version of our improvement to the VNC server and client are currently available for download [11].

4. PRELIMINARY EVALUATION

Our preliminary experiences with a graphical based, remote login system were extremely positive. At the same time, the off-the-shelf technologies were not quite suitable as a plug-in solution for widespread deployment. In this section, we outline specific benefits, challenges and limitations associated with such a thin-client approach. Benefits are those advantages inherent to thin-client computing and immediately achievable with current technologies. Challenges are not adequately addressed by existing off-the-shelf technologies, achieved in our software or achievable with additional modifications. Limitations appear to be inherent to the thin-client approach, and thus not easily overcome.

4.1 Benefits

Consistent, Universal Access

The highly accessible computing environment enables students to work from any computer with a wireless or broadband network connection. This availability and portability is in stark contrast to a tightly-controlled physical space, which may not be available at all times or which may not be conveniently located. At the same time, the configuration offers consistency in platform, software and access to data. Our hope is that the user experience should be nearly indistinguishable from that of local access.

Scalability

The centralized configuration offers the great benefit of extremely smooth scalability across a wide range of deployments. With an entry level configuration, 10-20 concurrent users can be supported by a single server with purchase cost around $2000-$3000. With the server load distributed across a larger cluster, we estimate that the technique could handle 200-400 concurrent users or more and can be incrementally expanded to match demand.

Cost Effectiveness

For an institution, there is generally enormous efficiency in the use of thin-client computing [14]. In particular, our configuration uses entirely open source, freely available software for both servers and clients. More so, the existing computing resources can effectively serve as thin clients, and refresh costs can likely be reduced as the clients need not be kept at the state of the art. The only expected capital cost will be in the purchase of the centralized server or servers.

We also note that a centralized computing configuration offers efficiencies involving the licensing of proprietary software. A number of concurrent user licenses for software installed centrally, may be utilized as effective as a larger number of licenses for the same software when installed locally on many machines.

Efficiencies arise from the increased flexibility in scheduling physical spaces, as appropriately sized computer labs become more interchangeable for use as classrooms.

Centralized System Control

As with a single system image, this configuration offers the institution the ability to maintain centralized control over the computing environment, allowing easy installation and updates of software, the operating system, libraries, etc. Yet unlike the intricate administration of a single image system, very little or no control is needed over the client machines. In essence, it extends the sphere of centralized control to reach all the way to a student's laptop or personal computer.

Open Source

Since all of the proposed software will be open source, it can later be adopted and modified by others, with further improvements fed back to the community.

4.2 Challenges

Though the existing VNC software was promising, is was not suitable as a plug-in solution for widespread deployment in an educational setting. In this section, we address some of the shortcomings which are more easily overcome, and outline our own efforts in this regard.

Universality

One of the challenges of the thin-client approach is providing support of client software on different platforms. It is desirable to have native clients for their speed and efficiency. At the same time, students should be able to access the system from anywhere on the internet and from any computer, making a Java-based client a natural choice. Current VNC software has a variety of clients based on different graphics libraries for each supported operating system. No one single VNC client supports all of the major operating systems.

In the future we plan on supporting multiple platforms for native clients using a cross platform GUI and network libraries that will run on Windows, Mac OS X and various UNIX flavors. This eliminates most of the issues with having a native client running on a variety of systems but not the need for a second Java client.

Transparency

A VNC session runs as an application in a separate window on the client desktop, forcing a demarcation between programs running on the remote server and on the local desktop. This hinders being able to cut and paste between programs running on the different systems. While there is a mechanism for handling cut and paste operations in VNC, it requires an additional step. In future versions, we plan on improving upon this cut and paste model to lessen the differences observed between applications running on the local and remote desktop.

Similarly, there is discontinuity between the remote system and the local filesystems and printers. We expect to provide a seamless remedy, through modifications to the client and server software, as well as the writing of Linux kernel modules. Any application running on the server would be able to access the client filesystem and printers as if they were physically part of the server system.

Performance

VNC seems to have a goal of reducing bandwidth usage, yet as noted in [5], it is often more important to reduce response latency to provide a cohesive user experience.

The VNC protocol [10] uses a lazy-update, client-pull mechanism, transmitting pixel data only for the region or regions of the screen that have since changed. Several variants of VNC have been developed, including TightVNC [16], which incorporate additional encodings in an attempt to further reduce the bandwidth usage.

We have begun to add encodings to VNC that allow progressive screen updates, quickly sending a reduced quality image and then sending high-quality refinements if the screen image has not since changed. These encoding use a wavelet-based technique, similar to the algorithms of Shapiro [13]. Bandwidth usage has been further reduced by using image caching to avoid retransmission of common image fragments, such as icons and backgrounds. Preliminary tests indicate that these techniques not only increase the compression ratio by a factor of two to three, but also significantly improve the system's responsiveness.

Session Management

Rather than establishing a new session each time a student connects, a new connection manager might allow each student the choice between establishing a new session or joining an existing session. When used with a server cluster, this manager would be responsible for overseeing existing sessions on any of the cluster nodes.

The ability to join an existing session would allow students to gracefully resume a session if a network connection is unexpectedly disrupted, and will also allow students to suspend a session from one access point and to resume from another without disruption to their work. This feature may allow students to switch from a lab to home, or to provide better support when moving between wireless access points.

Security

From the security perspective, VNC data is sent unencrypted over the network which compromises the security of the computers involved. Advanced users can overcome this by using VNC in conjunction with other application programs for encryption, yet this is not within the reach of introductory students. Instead, we have integrated OpenSSH [7] security directly into the VNC software.

OpenGL Support

The standard variants of VNC could not process OpenGL graphics, which is the standard library for 3D graphics on many platforms. We have incorporated rudimentary support for non-intensive applications of OpenGL, yet providing full support is inherently problematic.

4.3 Limitations

Network Dependency

All thin-client solutions are highly dependent on reliability of the network connection. When the network is down no access is possible. On systems that only depend on networked file servers, it is still possible to work on a local version of a file if the network goes down. While there is little that can be done to keep the system running during major network failures, increased support for transient network problems can be made. For example, if a connection is dropped due to networking issues or client errors the connection manager should be able to allow the resumption of a session.

Lack of Local Devices

For some purposes, the lack of local device usage is unacceptable. For example, a computer graphics course may depend on OpenGL for 3D computer graphics support. Yet, those graphics libraries depend of hardware acceleration to operate efficiently. For the networked system, the graphics accelerations must be performed by the CPU, thereby reducing the graphics quality and speed. Similar problems arise in running multimedia applications.

In a networking course, having all of the students using the same system makes writing network software artificial. For such courses, access to additional systems or the creation of virtual machines within the one server become necessary for students to fully appreciate this type of programming.

5. CONCLUSIONS

The use of VNC based student access has significantly improved computer science students' computing access at Saint Louis University. Instructors no longer need to worry about students not having access to the same computing environment in different computer labs and are ensured that students will have high quality access from the dorms or off-campus housing. Students no longer have to worry about having different computer accounts for different classes or carrying their files around with them on ZIP disks and worry about whether they will be able to access computers with the necessary software installed. With the envisioned additions to our software, we expect this student experience to continue to improve.

With the completion of our initial deployment, we expect that we will be able to use our system in new ways both in and out of the classroom. These include sharing a single desktop instantiation from multiple client computers. This may support groups of students working collaboratively; allow an instructor,to broadcast a master desktop to the students; in a closed lab, it would allow an instructor to access a student's desktop from a master computer or to display that desktop to the larger group. The software could also provide access to students in the setting of distance learning.

6. REFERENCES

[1] S. Blackmon and J. Nguyen. High-availabilty file server with heartbeat. *Sys Admin*, 10(9), Sept. 2001.

[2] D. Cherry, P. Phillabaum, and P. Valero. Living on the bleeding edge: creating and managing highly specialized student labs. In *Proc. 28th ACM SIGUCCS Conf. on User Services*, pages 40–49. ACM Press, 2000.

[3] B. L. Hawkins, J. A. Rudy, and J. W. Madsen. *EDUCAUSE 2002 Core Data Service Monograph.* EDUCAUSE, 2003.

[4] Industry Analysis and Technology Division Wireline Competition Bureau. High-speed services for internet access: Status as of December 31, 2003. Technical report, Federal Communications Commission, June 2004.

[5] A. Lai and J. Nieh. Limits of wide-area thin-client computing. In *Proc. 2002 ACM SIGMETRICS Int. Conf. on Measurement and Modeling of Computer Systems*, pages 228–239, 2002.

[6] G. B. Newby. Student laptop ownership requirement and centralization of information technology services at a large public university. *Annals of Cases on Information Technology*, pages 201–212, 2003.

[7] OpenSSH. http://www.openssh.org.

[8] B. Pawlowski, S. Shepler, C. Beame, B. Callaghan, M. Eisler, D. Noveck, D. Robinson, and R. Thurlow. NFS version 4 protocol. In *Proc. Second Int. System Administration and Networking Conference (SANE 2000)*, page 94, 2000.

[9] T. Raich. Treat them like they have laptops. In *Proc. 30th ACM SIGUCCS Conf. on User Services*, pages 106–107. ACM Press, 2002.

[10] T. Richardson, Q. Stafford-Fraser, K. R. Woods, and A. Hopper. Virtual network computing. *IEEE Internet Computing*, 2(1):33–38, Jan. 1998.

[11] Remote linux laboratory(RLL). http://rll.sourceforge.net.

[12] J. Robert C. Heterick. A single system image: An information systems strategy, 1988. CAUSE Professional Paper Series #1.

[13] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Trans. on Signal Processing*, 42(12):3445–3462, 1993.

[14] M. Sheehan. Considering thin client computing for higher education. *Cause/Effect*, 21(3):7–10, 1998.

[15] S. J. Thomas, C. Laxer, T. Nishida, and H. Sherlock. The impact of campus-wide portable computing on computer science education. In *Working Group reports of the 3rd Annulal SIGCSE/SIGCUE ITiCSE Conf. on Integrating Technology into Computer Science Education*, pages 35–40. ACM Press, 1998.

[16] TightVNC. http://www.tightvnc.org.