Towards a Virtual Ecosystem for K-8 Education

MARK K. GARDNER and WU-CHUN FENG
Virginia Tech
{mkg,wfeng}@vt.edu

In literally all walks of life, computing has become an indispensable tool in enhancing productivity and accelerating discovery and innovation. However, with respect to K-8 education, particularly in rural and under-privileged areas, computing has had limited impact, and arguably, serves as further evidence of the ever-growing digital divide. To address this, we propose to use virtualization to improve the quality of education in the rural and economically disadvantaged areas of southwest and southern Virginia. In addition to the expected benefits of reduced cost and simplified management, our application of virtualization seeks to increase the usability and flexibility of computing resources while simultaneously ensuring the security and robustness of these resources.

Specifically, this paper describes the design, prototyping, and deployment of our virtual ecosystem called SERViCE for a K-8 school in rural southwest Virginia. We pay particular attention in employing security measures that are intended to protect physical assets and private information in the face of increasingly malicious attacks on computing infrastructure as well as safeguarding our youth on the Internet. We then summarize our initial experiences in using our SERViCE ecosystem to teach programming to second- and third-grade students at the school.

Categories and Subject Descriptors: C.2.4 [**Computer Networks**]: Distributed Systems; K.3.1 [**Computers and Education**]: Computer Uses in Education; K.3.2 [**Computers and Education**]: Computer and Information Science Education

General Terms: Design, Human Factors, Languages, Security

Additional Key Words and Phrases: Virtual Computing, Education, Programming, Logo, Storytelling Alice, K-8, K-12

1. INTRODUCTION

With the recognition that computing has become an integral part of our society, access to computing technology is becoming more prominently available in our secondary K-12 schools. However, for every school district like Loudoun County in northern Virginia, where each school seemingly has state-of-the-art computing facilities (e.g., Ball's Bluff Elementary School in Leesburg, Va.) and experience with how to effectively use such facilities, there exist multiple rural and economically disadvantaged school districts such as Lee County in southwestern Virginia, where computing is merely a novel curiosity.[1]

Even when computing technology is available, we argue that the effective utilization of such technology faces many challenges: lack of integration into the curriculum, scarcity of innovative educational software and the management of such software, insufficient training and support for K-12 teachers, and lack of funding and personnel to even attempt to address the above problems. For example, in our K-8 school in rural southwest Virginia, second and third graders only use computers to learn how to type with "Learn to Type Jr." software. Though admirable in intent, the curriculum falls short on many of the aforementioned fronts. First, it is completely decoupled from the rest of the curriculum. Students only work on the

[1]Based on numbers from the 2000 U.S. Census, the per capita income for Lee County was $13,625. Nearly 25% of the population lived below the poverty line.

computers for 20 minutes per week at the end of a Friday. Second, the entire computer lab is maintained by a volunteer parent, and given the difficulty in managing nine heterogeneous (and fairly dated) computers, the "Learn to Type Jr." software only works on five of the nine computers. Third, many of the teachers at the school know little about information technology beyond using commodity software such as Microsoft Word and Learn to Type Jr. Thus, they are *not* empowered to teach more advanced computer skills to students despite their own interest in learning these skills and passing them on to the students.

In summary, we face two major challenges with integrating computing into education: (1) ubiquitous ease of access, thus bridging the digital divide between the *haves* and *have nots*, and (2) effective and transformative use of computing. To address these challenges, we turn to virtualization, as exemplified in part by [NC State 2008]. Although still in its infancy, virtual computing for education promises to dramatically enhance our ability to educate students to meet the challenges of tomorrow. Perhaps more importantly, we see virtual computing as a promising tool to more effectively use technology to educate children in economically disadvantaged areas, particularly in rural Virginia.

2. MOTIVATION

Our pilot for virtual computing — SERViCE: Scalable, Extensible, and Reliable Virtual Computing Environment — has been deployed at a small cooperative in rural Virginia that offers an accredited curriculum for students in grades K-8. Founded by parents who wanted a greater influence in their children's education, the school focuses on meeting the needs of each child while balancing the needs of the community as a whole. Students from the K-8 school, who must transition into the K-12 public school system for high school, have done extraordinarily well, indicating the success of the school's approach.

However, the school faces many challenges. First, as a parent-run cooperative, the school arguably has even fewer funds at its disposal than many of the rural public school districts in the Commonwealth of Virginia, which in turn, generally have less funds than the public schools in the metropolitan areas. As a result, teachers, parents, and students must be more innovative in the use of technology in order to obtain the same benefits as better funded schools. Experience has shown that virtualization can reduce costs while providing the same services.

Second, the computing capacity is insufficient to meet the needs of the students. The school has eight Windows desktop computers connected to the Internet via a cable modem to serve the needs of 85 students. The student-to-computer ratio is 10.6-to-1, compared with 3.1-to-1 for the Commonwealth of Virginia and 3.8-to-1 for the nation [Education Week 2008]. As a result, the computer lab is a heavily timeshared resource, where students only get a fraction of the computing time that students from more fortunate schools get on computing resources.

Third, there exists little expertise in computing among faculty and staff to install, configure, maintain, and upgrade computing systems. The teachers are not technologists. If something fails to work in the computer lab, the students have to "do without" until the volunteer system administrator can investigate. Another symptom of the problem is the lack of uniformity in installed software. Teachers and students need to consult messages on the lab's whiteboard in order to determine which machines are operational and which ones have working software installations.

To address the above challenges, we leverage virtual computing for our SERViCE ecosystem, which we discuss in more detail in §4. The server performs the bulk of the computing work by creating virtual machine (VM) instances "on the fly" while less expensive, and even heterogeneous, client machines serve as "virtual computing terminals." Only the virtualization server requires a powerful processor, large amounts of memory, and sufficient storage capacity. The subsequent cost savings for clients can be used to purchase more economical terminals, and ultimately, improve the student-to-computer ratio at the school. Finally, virtualization has the potential to amplify the reach of the few technology-savvy volunteers so that they can more effectively manage the machines. Instead of maintaining individual machines, volunteers maintain small sets of virtual machine (VM) images, which are cloned at run time. And because images are discarded after use, inadvertent changes do not cause unintentional or malicious modifications to the server.

3. CYBERINFRASTRUCTURE ISSUES

While there are many issues to consider in the creation of a virtual computing ecosystem, space only permits us to focus on two: usability and security.

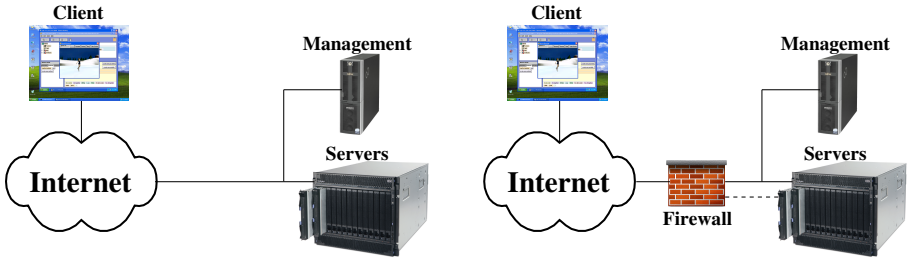
3.1 Usability

The user interface serves as the bridge between the student and the underlying computing technology. So, its design is critical to the acceptance of virtual computing in our rural K-8 school. Virtual computing environments that are targeted at the post-secondary (university) level typically employ a web browser as the user interface. We believe that the obvious choice of encapsulating the user interface inside a web browser unnecessarily increases the visual and operational complexity, thereby increasing cognitive load. This is particularly true for younger students, like the second and third graders that we initially target at the aforementioned rural school in southwest Virginia. However, many students, even at the kindergarten level, are already familiar with a windows-based graphical user interface from experiences with home computers. By implementing the user interface within the native operating system, an additional level of complexity and confusion is eliminated. We are therefore implementing a interface upon the "native desktop" metaphor.

Students initiate a session simply by double-clicking an icon on the desktop, an operation that is already familiar to them. The application connects to the SERViCE server using a secure socket and requests a clone of a pre-built environment containing the already-installed applications. After the virtual machine begins execution, a window containing the desktop of the environment running on the virtual machine opens, and the students begin their lesson.

3.2 Security

Due to the limited expertise and manpower available, the compute machines, particularly the virtual machine (VM) server, must be secured against accidental or malicious disruption. While securing a guest operating system (OS) running in a VM is no more (or less) challenging than securing an OS installed on a physical machine, dealing with the measures put into place to protect the server on which the guest OS executes can be a problem. In this section, we consider issues surrounding the use of a firewall to protect the ecosystem.



(a) Traditional Virtual Computing (b) SERViCE with Firewall

Fig. 1. Architectures of Virtual Computing Environments

A firewall is a software (or hardware) "sheriff" that decides what network packets can pass from the "Wild, Wild West" of the open Internet into the more law-abiding confines of an organization's intranet. Traditionally, machines in a virtual computing environment have been assigned public network addresses on the Internet. That is, they are "directly connected," as shown in Fig. 1(a).[2] This approach presents at least two potential problems: (1) an institution may not have enough public addresses to allocate and (2) an institution's security policy may not allow internal hosts to be directly connected to the Internet. In either case, it results in assigning private addresses to internal machines, whether physical or virtual, and connecting them to the Internet via a firewall, as shown in Fig. 1(b) for our SERViCE architecture. A network address translator (NAT) then handles the process of mapping between private and public IP addresses.

As defined in RFC 2663 [Uberti and Roussopoulos 1999] and RFC 3489 [Rosenberg et al. 2003], the most common type of NAT is an *outbound* NAT firewall with network address/port translation (NAPT), as shown in Fig. 2. When properly configured, this type of NAT allows hosts on a private network to share the use of a single public IP address so that a client on the Internet and server on a private network can communicate as if they are directly connected. Specifically, the client can initiate communications with the server because the firewall has been configured to translate `Public2` to `Private1`. Conversely, the server can communicate with the client because the firewall keeps information about active connections, i.e., *connection state*, allowing it to perform the same translation for replies.

The next issue is what network address to assign physical and virtual machines behind the firewall. The least-effort approach uses one subnet behind the firewall and gives the VMs network addresses on that subnet. However, this suffers from two serious problems. First, the management and server nodes are on the same subnet as the VMs, and hence, are subject to a greater risk of unintentional or malicious disruption, represented by the dashed line in Fig. 1(b). Second, if all the VMs are on the same subnet, they also suffer increased risk from each other. Not only should the infrastructure be isolated from the VMs, but the VMs should be grouped into sets of machines by class such that they can reach each other (facilitating group work) but not reach unrelated machines.

[2]A firewall may still be present but is transparent to the client and server because the virtual machines have public network addresses.

4. DESIGN, IMPLEMENTATION AND DEPLOYMENT OF SERVICE

In SERViCE, the servers that constitute the core virtual computing ecosystem are located at Virginia Tech and are protected by a NAT firewall with network address/port translation (NAPT). The clients at the rural K-8 cooperative school then access the server via the Internet and are also protected by their own firewalls.

A translation between the external and internal IP addresses allows students' requests for virtual machines to pass through the Virginia Tech firewall. (The firewalls on the client machines do not need to be modified as servers do not initiate communications.) Though configuring the firewall for this translation is straightforward because the addresses of the physical infrastructure do not change, configuring the firewall so that clients can communicate with the virtual machines is *not* as straightforward because each virtual machine is dynamically assigned an internal network address and the firewall must be configured accordingly. A related concern occurs because each translation constitutes another avenue of potential attack. Thus, SERViCE contains mechanisms for dynamically creating and removing translations as VMs are created and destroyed.

Virtual machines execute using the freely available VMware Server product [VMware Inc. 2008]. As discussed earlier, sets of related VMs are isolated from each other and from the infrastructure by placing them on separate virtual subnets. VMware Server provides three options for virtual networking. The first is to bridge the virtual subnet to the physical subnet. The second provides a "host-only" option that allows the VM and the physical machine to communicate but does not allow the VM to reach the Internet. The final option is to implement a masquerading firewall or "NAT" (network address translation) option.

It appears that the bridged and host-only options can both be dismissed since the former does not isolate VMs from each other and the latter only connects to the host. That leaves NAT. Ports can be forwarded from the host to the VMs by editing a configuration file and restarting VMware's network daemon. But this drops any existing connections, including those destined for existing VMs. Can messages be injected onto VMware's NAT network without having to restart? No. Injecting messages behind VMware's back causes dropped replies because the firewall does not know that a connection has been created. Indeed, tcpdump can be used to observe messages reaching and replies being returned from the guest. However, no messages exit VMware's NAT network bound for the Internet. It appears that none of the options will work.

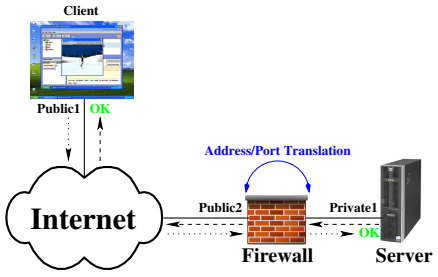


Fig. 2. NAT Firewall with Network Address/Port Translation (NAPT)

This apparent dilemma comes from too quickly dismissing the bridged and host-only options. While neither can be used by themselves, messages can be forwarded with `iptables` to virtual networks created with the Linux `tun` network driver and will appear at the bridged network interfaces in VMware. Alternatively, the host-only option can also be used with a slightly different configuration. In SERViCE, the bridging option is used when groups of VMs need to communicate and the host-only option is used for independent VMs.[3]

5. EXPERIENCES

As discussed earlier, we seek to facilitate computer education in rural schools though the design, implementation, and deployment of SERViCE. The initial deployment of our SERViCE ecosystem occurred at a rural K-8 cooperative school to facilitate the teaching of programming.

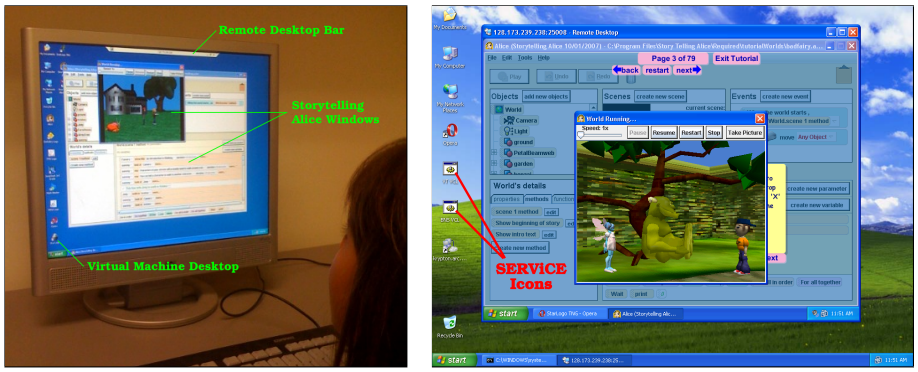
First, we have been teaching a group of 14 second- and third-grade students the Logo programming language [Logo Computer Systems Inc. 1999]. We chose Logo because it was designed as an accessible tool for teaching children how to program. The turtle graphics provided by Logo is a simple yet motivational way to interest the students. Second, we next plan to teach object-oriented programming in a 3-D graphical environment to a group of older student (grades 6-8) using Storytelling Alice [Kelleher 2006; Kelleher and Pausch 2007]. We chose Storytelling Alice, a derivative of the Alice programming environment [Cooper et al. 2003; Rodger 2007], because it is intended to motivate girls to become involved in computer science.

In Fig. 3(a), an anonymous student accesses Storytelling Alice from her client desktop machine at the K-8 school. The student has already clicked the icon on the client desktop to start running a WindowsXP image on the server at Virginia Tech. The remote desktop bar at the top of the screen indicates that the virtual environment has been maximized, and hence, completely covers the desktop of the local machine. At the point shown, the student has started Storytelling Alice from within the virtual environment and selected one of the examples. Fig. 3(b) is a closer look at one of the tutorials. This time, the virtual environment is being displayed in a normal window and the icon on the desktop that brings up the virtual machine is visible.

The response times were surprisingly quite good in spite of the fact that Storytelling Alice is graphically intensive and the graphics are all being rendered on the virtual machine remotely. We worried that keyboard and mouse interactions might be too slow considering the school is connected via a cable modem. While there were occasional lags, none were excessive and could have been caused by lags in the several generations-old client. Animations were generally smooth with only occasional stutters, although fast animation of objects in Storytelling Alice resulted in the objects "teleporting" from one location to another. It appears that the virtual machine is rendering the images but remote desktop is skipping frames to compensate for either the slow client or the slow network.

Figure 4 shows screen captures of two different Logo sessions executing on SERViCE. The first utilizes the MSW Logo environment [Mills 2002] which is built upon Brian Harvey's UC Berkeley Logo implementation [Harvey 1997]. The example shows the commands for drawing a square with the turtle, except that the

[3]Scalability is not a concern in practice since each physical CPU can only support between one and four VMs.

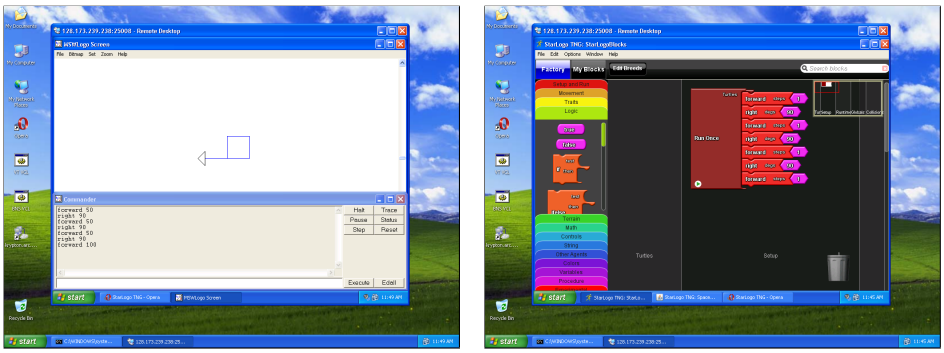


(a) SERViCE at a Rural K-8 School (b) Storytelling Alice

Fig. 3. Teaching Programming in a 3-D Environment

user has specified too many paces for the last side. The second shows an incomplete version of a program to accomplish the same task in the Star Logo TNG environment [McCaffrey 2006; Wang et al. 2006].

As we have begun to teach Logo to the second- and third-graders, we do not yet have sufficient feedback to evaluate our approach. However, a few observations can be made. First, the students take instructions very literally at this age. Most do not yet have the maturity to readily grasp abstractions. For example, it has proven to be a little challenging for the students to place themselves "in the turtles shoes" and grasp that the Logo command forward moves the turtle in the direction it is facing on the screen. To overcome this, we divided the students into pairs. One student pretends to be the turtle and the other pretends to be the programmer. The programmer gives commands, and the turtle acts out the movements using the linoleum tiles of the classroom as the grid. Then the roles are reversed. Afterwards, the students showed increased ability to predict the behavior of the turtle on the screen in response to their commands. Preliminary results also suggest that the StarLogo TNG environment is too complex for this age group.



(a) MSW Logo (b) StarLogo TNG

Fig. 4. Teaching Programming with Logo

6. CONCLUSION

Virtualization can be used to dramatically improve the educational opportunities in rural communities in the nation. Not only does it reduce costs, an important consideration for schools, but it also increases flexibility and ensures that students have a consistent and safe environment in which to learn online.

In this paper, we described our system, called SERViCE, and how it addresses the goals of usability and security. Unlike extant virtual computing environments, SERViCE was designed to be deployed in an environment in which firewalls are used to protect physical assets and information from accidental, malicious or even criminal attacks on the computing infrastructure.

We have deployed SERViCE as a pilot in a rural K-8 school in southwest Virginia. We have been using the SERViCE ecosystem to teach programming to second- and third-grade students, with plans to teach object-oriented programming in the future.

ACKNOWLEDGMENTS

This work was supported in part by an IBM Faculty Award through Virginia Tech Foundation VTF-873901.

REFERENCES

Cooper, S., Dann, W., and Pausch, R. 2003. Teaching Objects-first in Introductory Computer Science. In *34th SIGCSE Technical Symposium on Computer Science*. ACM, Reno, Nevada, USA, 191–195.

Education Week. 2008. Technology Counts 2008 — STEM: The Push to Improve Science, Technology, Engineering, and Mathematics. Online suppliment containing statistics for Virginia available at `http://www.edweek.org/ew/articles/2008/03/27/30dsr.h27.html` (last checked 2008-05-07).

Harvey, B. 1997. *Computer Science Logo Style, 2nd Ed.* MIT Press.

Kelleher, C. 2006. Motivating Programming: Using Storytelling to Make Computer Programming Attractive to Middle School Girls. Ph.D. thesis, Carnegie Mellon University. CMU-CS-06-171.

Kelleher, C. and Pausch, R. 2007. Using Storytelling to Motivate Programming. *Commun. ACM 50,* 7, 58–64.

Logo Computer Systems Inc. 1999. *Logo Philosophy and Implementation.* Logo Computer Systems Inc.

McCaffrey, C. 2006. StarLogo TNG: The Convergence of Graphical Programming and Text Processing. Ph.D. thesis, Massachusetts Institute of Technology.

Mills, G. 2002. Welcome to MSWLogo. `http://www.softronix.com/logo.html`.

NC State. 2008. Virtual Computing Lab (VCL) Home Page. `http://vcl.ncsu.edu/`.

Rodger, S. H. 2007. An Innovative Approach with Alice for Attracting K-12 Students to Computing. In *First International Conference on the Virtual Computing Initiative (IBM University Days)*. Research Triangle Park, NC, USA, 17.

Rosenberg, J., Weinberger, J., Huitema, C., and Mahy, R. 2003. STUN - Simple Traversal of User Datagram Protocol (UDP) through Network Address Translators (NATs). IETF RFC 3489.

Uberti, J. and Roussopoulos, M. 1999. IP Network Address Translator (NAT) Terminology and Considerations. IETF RFC 2663.

VMware Inc. 2008. Vmware home page. `http://www.vmware.com/`.

Wang, K., McCaffrey, C., Wendel, D., and Klopfer, E. 2006. 3D Game Design with Programming Blocks in StarLogo TNG. In *ICLS '06: Proceedings of the 7th international conference on Learning sciences*. International Society of the Learning Sciences, 1008–1009.