Proceedings of the Linux Symposium

Volume One

July 20nd–23th, 2005
Ottawa, Ontario
Canada

Conference Organizers

Andrew J. Hutton, *Steamballoon, Inc.*
C. Craig Ross, *Linux Symposium*
Stephanie Donovan, *Linux Symposium*

Review Committee

Gerrit Huizenga, *IBM*
Matthew Wilcox, *HP*
Dirk Hohndel, *Intel*
Val Henson, *Sun Microsystems*
Jamal Hadi Salimi, *Znyx*
Matt Domsch, *Dell*
Andrew Hutton, *Steamballoon, Inc.*

Proceedings Formatting Team

John W. Lockhart, *Red Hat, Inc.*

SNAP Computing and the X Window System

James Gettys
Hewlett-Packard Company
jim.gettys@hp.com

Abstract

Today's computing mantra is "One keyboard, one mouse, one display, one computer, one user, one role, one administration"; in short, one of everything. However, if several people try to use the same computer today, or cross adminstrative boundaries, or change roles from work to home life, chaos generally ensues.

Several hardware technologies will soon push this limited model of computing beyond the breaking point. Projectors and physically large flat panel displays have become affordable and are about to take a leap in resolution[12]. Cellphone-size devices can now store many gigabytes of information, take high resolution photographs, have significant computation capability, and are small enough to *always* be with you.

Ask yourself "Why can't we sit with friends, family, or coworkers in front of a large display with audio system, and all use it at once?"

You should be able change roles or move locations, and reassociate with the local computing environment. The new mantra must become 'many' and 'mobile' everywhere 'one' has sufficed in the past.

Change will be required in many areas from base system, through the window system and toolkits, and in applications to fully capitalize on this vision.

1 Introduction

As much as three quarters of the cost of computing in enterprise environments now goes to system management and support; the hardware and software purchase cost is well under half of the total expense. In the some parts of the developing world, expertise may be in shorter supply than computers. Personally, I now manage three systems at home, in addition to three for work. Clearly something needs to be done.

Project Athena[13], a joint project of Digital, MIT, and IBM in the mid 1980's, had the vision of centrally administrated, personal computing, in which mobile students and faculty could use whichever computer was most convenient or appropriate for their work. Out of this project was born a number of technologies that we take for granted today, including Kerberos[24], the X Window System[31], central administration of configuration information using Hesiod[18] (now mostly supplanted by LDAP), and Zephyr[17], the first instant message system.

Due to the lack of a critical mass of applications, UNIX divisions, and UNIX workstations costing more than PC's, the Athena environment did not reach critical mass in the marketplace, despite demonstrating much lower cost of ownership, due to much easier system management. The Wintel environment has caused almost everyone to become poor system man-

agers of an ever-increasing number of computers, and it is now clear that Athena had more right than wrong about it. The "solution" of having to carry laptops everywhere is poor, at best. Some of Athena's technologies escaped and became significant parts of our computing environment as individual components, but the overall vision was lost.

Athena's vision was right on many points:

- People are mobile, the computing infrastructure is not.
- People should be able to use any computing system in the environment so long as they are authorized.
- There is a mixture of personally owned and organizationally owned equipment and facilities.
- Authentication enables an organization to control its resources.
- Collaborative tools, either realtime or non-realtime, are central to everyone's lives.
- Your information should be available to you wherever you go.

The Fedora Stateless Project[11] is resurrecting most aspects of the Athena environment and extending it to the often connected laptop; and the LTSP project[6] uses X terminal technology for low system management overhead, thin client computing. These technologies reduce cost of ownership due to system management to something much closer to proportional to the number of people served rather than the number of computers. Deployment of systems based on these technologies, the continuing declining cost of hardware, and open source systems' zero software cost, will enable computers to be located wherever may be convenient. We need to go beyond the Athena vision, however, good as it is for centralized system management.

History also shows Athena's presumptions incorrect or insufficient:

- We presumed display technology limited to one individual at a time, possibly with someone looking over the shoulder.
- That users play a single role, where in the adult world we play many roles: job, home life, church, schools, clubs, and often more. Computer systems must enable people to play multiple roles simultaneously.
- That universal authentication was possible. This is probably a chimera despite efforts of Microsoft and Sun Microsystems—it implies universal trust, unlikely between organizations. At best, you may have a single USB fob or wireless device with many keys that authenticate you for your many roles in life; at worst, many such devices, attached to your physical keyring.
- That there would be very small wearable devices, with significant storage and computing power (soon sufficient for most user's entire computing environment).
- That wireless networking would become very cheap and commonplace.
- That the computing environment is a PC, file, compute and print services: today's environments include projectors and large format displays, (spatial) audio systems, display walls, and so on.

So long as large displays are few and far between, and limited in resolution, the pseudo-solution of the laptop VGA connector attached to a projector has been a poor but adequate

solution. Projectors are now cheap and commonplace, but with the imminent advent of 1080i and 1080p large screen HDTV displays and projectors (1080p is 1920x1080 resolution in computer-speak), we face a near future in which we will finally have displays with enough pixels that sharing of the display makes sense. We will soon be asking: "Why can't I use the environment easily? Why can't I combine my 100 gig cell phone with the surrounding environment to always be able to have my computing environment with me? Why can't I easily shift from work, to home, to school, to church, to hobby?"

Computing systems should enable the reassociation of people, any computing devices they have with them, and the computing infrastructure available wherever they meet, work, and play. While many devices can be used by only one person at a time (e.g. keyboards, mice, etc.), others, such as large screens and audio systems can and should be usable by multiple people simultaneously. It is time we make this possible.

2 User Scenarios

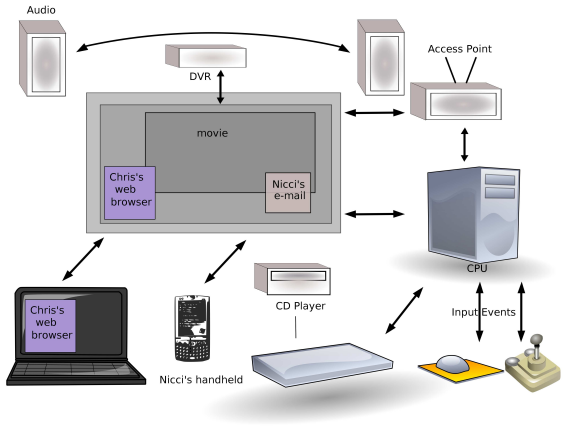
My great thanks to my colleagues Andrew Christian et al. for exploring wider insights into SNAP Computing[15]. Some of the scenarios below are excerpted from that paper. This paper will provide a concrete proposal for work on the X Window System, but without providing background material explaining the SNAP vision, it would be impossible to understand the rationale of the design changes proposed.

2.1 Office

You are sitting in your office. Your incoming frantic call is from your spouse, who is having problems with a complicated formatting problem in the word processor of a document that must be sent before you get home that evening. You ask that the window be cloned to your display, so you can help solve the problem together. When finished, you close the cloned window and the document is finished by the deadline.

2.2 Home

In this example Nikki and her friend Chris are sitting in Nikki's living room watching television on a big, high-resolution video screen, but also doing a little work and web browsing (see below). The living room's personal video recorder (PVR) is playing a movie on the video screen and sending audio to the living room audio system. Nikki has pulled out a portable keyboard, connected to the home office CPU, and pulled up her e-mail on a corner of the living room video screen. As she browses her remote mail store, audio attachments are routed and mixed in the local audio system and played through the living room speakers so that they appear on her side of the room (spatially located so as to not distract Chris).



Meanwhile, Chris has pulled out a wireless handheld computer. Nikki has previously granted Chris some access rights for using the home's broadband connection and living room equipment, so Chris grabs a section of the big video screen and displays output from a web browser running on the handheld computer. Audio output from Chris's web browser is spatially located to help disambiguate it from Nikki's e-mail. Without a keyboard Chris must use the handheld computer for handwriting recognition and cursor control. To speed things up, Chris borrows a wireless keyboard from Nikki's home office. The keyboard detects it is in the living room and bonds automatically to the big screen. Through the handheld computer, Chris assigns the keyboard to work with the web browser and goes back to surfing.

Most of the time Chris and Nikki are working within the confines of the big video screen. For example, both may be driving their own private pointing cursor on the screen. Security policies prevent them from controlling each others' applications; Nikki typing at her e-mail is kept separate from Chris's web browsing. However, the big screen also provides high level services that both can request and access. For example, a screen window manager service positions the individual windows and a screen cut-and-paste service allows data to be shared across users. Should Chris or Nikki wish to change channels or control audio volume in the room, either can ask for video screen control and use the shared, built-in video browser to access the audio volume control or bind it to their local device (Chris' handheld or Nikki's keyboard).

2.3 Conference Room

Functionally, a conference room is not greatly dissimilar from Nikki's living room. The conference room provides shared video screens that multiple users can access from their laptop/handheld computers, or via broadband connections back to their desktop machines.

The conference room provides several business-specific services. First, the room itself can provide scheduling and journaling functions. Because the conference room display screens are intelligent—rather than simple projectors—it is easy to allow them to record and store information about what was done in the room. Each user provides authentication before accessing services, so a clean record of users and activities can be journalled and made available to participants later.

Adding video conferencing introduces a second interesting feature: virtual proximity. A video conference establishes a virtual location relationship between people and devices. For example, the remote user may wish to print a file in the conference room, display and control a presentation on the video screen, and play audio through the local speakers.

To make this more concrete, imagine you are at a meeting of a industry working group with representatives from competitors, to work on a standards document. Several of you put up drafts on the conference room display screens to work on from the laptops you brought with you. The computer of one of your working group members has failed entirely, but he has the information cached in his cellphone, so using a spare keyboard in the conference room, he is able to find the needed information using a corner of the screen for the group.

Such conference rooms were described by Isaac Asimov in his *Foundation* series, in which his First Foundation mathematicians work together in rooms whose walls *are* displays. Such conference rooms are no longer science fiction; display wall systems are already being built[9][2], and their cost will continue to fall.

3 Design Requirements of the SNAP Vision

If we are to disassemble, or unsnap, the components of the classic computer and allow the flexible reassociation (or snapping together) of components, while enabling people to reassociate with the computing environment as they move, we require some networking connectors to snap the components back together. I argue that the networking connectors now exist, and if we disassemble our systems and combine the pieces using these connectors, we can then easily snap them back together dynamically at will. I will touch on some of the resulting topics in this section, before diving into X Window System-specific issues.

These software components include:

- distributed caching file systems (e.g. Coda[23])
- encryption of all network communication
- roaming between networks
- software which can easily roam among multiple differing authentication systems
- discovery of network services
- network connectors replacing hard wires to snap the computing components back together
- network audio, so that you can easily use audio facilities in the environment
- the window system that supports multiple people collaborating, and helps protects you from other malicious people

3.1 Service Discovery

People need to be able to discover that facilities are available and take advantage of them. Open source implementations of the IETF Zeroconf[10] protocols are now available such as Howl[4]); zeroconf is also used to good effect as Apple's Bonjour[1] in OSX. We can leverage such protocols to discover file systems, X Window System servers for large displays, scanners, printers, and other services that may be interesting to mobile users in the environment; and zeroconf support is beginning to appear in open source desktop projects.

3.2 Localization

For ease of use, you need to know what devices are in a given physical location. Presenting a user with a long list of devices present in many work environments, even just a local subnet, would result in confusion. Research shows that it may be feasible to localize 802.11[abg] to roughly the granularity of an office or a conference room, but such systems are not generally available at this date. Given these results it is clear that location tracking systems will become available in the near-term future, and there are startup companies working actively to bring them to market.

Bluetooth was intended as a cable replacement, but experience with it has not been very favorable in a SNAP system application. Introducing a new bluetooth device to its controller is involved and time-consuming, not something that is done casually, at least as cumbersome as dragging a cable across a room and plugging it in.

The 801.15.4 ZigBee local wireless technology, just becoming available, does not suffer from these limitations that make Bluetooth so cumbersome. Additionally, IR is ubiquitous can

be used for local line of sight localization, and handheld devices often have consumer IR (intended for remote control use), which has much longer range than that found in laptops.

There are multiple efforts in the research community to provide location based lookup of resources, and this work and expertise should be leveraged.

3.3 Audio

There is a long history of inadequate audio servers on UNIX and Linux.

ESD and NAS are inadequate even for local multimedia use (lacking any provision for tight time synchronization), much less low-latency applications like teleconferencing.

There are a number of possible paths:

- The Media Application Server (MAS)[7] may be adequate.
- We can build a network layer for the JACK[5] audio system.

These possibilities are not mutually exclusive (Jack and MAS could be used in concert), and we can start from scratch, if they will not serve.

Detailed discussion of the need/requirements for network audio, needed to complement our network transparent window system are beyond the overall scope of this paper. The AF audio server[25] on UNIX of the early 1990's showed that both very low latency and tight synchronization is in fact possible in a network-transparent audio server.

3.4 Network Roaming

There is much work to be done to take what is possible and reduce it to practice. Real-time roaming between networks can be as short as fractions of a second; we should not accept the current delays or manual nature we find today as we DHCP and manually suspend/resume as we transition between networks. Handoff between networks can and should be similar in duration to the cellphone network, so short as to be effectively unnoticed.

4 X Window System

The 'One' mantra is most clearly ingrained in all of today's window systems, where one keyboard, one mouse, one user is the norm. Our base operating system, however, was designed as a multi-user system, with the operating system providing protection between users. The X Window System has at least been, since its inception, network transparent, allowing applications to run on multiple displays, potentially including displays in our environment.

4.1 Multiple People Systems

X's current design presumes a single person using the window system server, and therefore only provided access control. To allow multiple people, particularly in open environments where people cannot trust each other, to use a common screen means that privacy and security problems must be solved.

The core X protocol allows applications to spy on input. Furthermore, cut-and-paste can quickly transfer megabytes of data between applications. Multiple simultaneous users therefore pose a serious security challenge. X needs

better access control to input events, pixmap data, X properties, and other X resources.

During the mid 1990's, there was work to extend X for the requirements posed by military multi-level 'orange book' security. The resulting extension provided no policy flexibility, and still presumed a single user. The resulting X Security extension[35] has remained entirely unused, as far as can be determined.

Recently, Eamon Walsh, an intern at the NSA, implemented an SELinux-style X extension [34] with the explicit goal of enabling multiple possible security policies, that might provide the kinds of policy flexibility. Differing environments, in which different levels of trust between users exist and different sensitivities of information displayed on the screen simultaneously, will clearly need different policies. One policy can clearly not fit all needs. Eamon's work was updated this spring by Bryan Ericson, Chad Hanson, and others at Trusted Computing Solutions, Inc., and provides a general framework that may be sufficient to explore the policies required for this use of the window system.

X has internally no *explicit* concept of a 'user,' without which it is impossible to devise any security policy for systems being used by multiple people. Given good security policies and enforcement, in many environments even unknown people should have unprivileged access to a display. An explicit concept of a user, and the window resources they are using, is clearly needed in X, and once present, policy development using this framework should become feasible. X also lack explicit knowledge of a people's sessions, and since several sessions may be going on simultaneously, I also expect X will require this concept as well.

On Linux and some UNIX systems you can determine the person's identity on the other end of a local socket. We also need the identity of the person's application on the far end of a network connection. In a corporate environment, this might best be served by the person's Kerberos credentials. In other environments, ssh keys or certificate-based authentication systems may be more appropriate. Fortunately, it appears that Cyrus SASL[19] may fill the authentication bill, as it supports multiple authentication families.

Even with this work, there is work remaining to do to define usable security profiles, and work that should take place in toolkits rather than relying solely in the window system. For example, a person cutting from their application and pasting into another person's application does not have the same security consequence as the opposite situation, of others being able to cut from your application into their application: in this case, the person is giving away information explicitly that they already control. It is easier to trust the implementation of the toolkit you are using, than the implementation of a remote X server that you may have much less reason to trust.

More subtle questions arise for which there are not yet obvious answers: How do you know what security profile is currently in force in the X server you are using? Why should you trust that that profile is actually being enforced? These class of problems are not unique to X, of course.

Distinguishing different pointing devices according to the people using them will require an extension to X to support multiple mouse cursors that can be visually distinguished from each other. Since hardware supports a single cursor at most, X already commonly uses software cursors, and by compositing another image with the cursor shape, we can easily indicate whose cursor it is.

4.2 Securing the wire and the SSH trap

At the time of X11's design (1987), and until just a few years ago, the U.S. government actively discouraged the use of encryption; the best we could do was to leave minimal hooks in the wire protocol to enable a later retrofit. Even pluggable architectures allowing the easy addition of encryption were actively opposed and might cause the U.S. Government to forbid export of software. Export of encryption without export control only became feasible in open source projects in the last several years.

In the era of little or no security problems of the 1980's and early 1990's, X was for a time routinely used unencrypted over the network. With network sniffers on insecure systems everywhere today, this usage today is clearly insane.

The Swiss army knife of encryption and authentication, "ssh"[14], appeared as a solution, which provides authentication, encryption, and compression by allowing tunneling of arbitrary streams (including X traffic). While it has been a wonderful crutch for which we are very grateful, a crutch it is, for the following reasons:

- SSH requires you to have an account on a machine before tunneling is possible. This prevents the casual use of remote displays, even those we might intend for such use.
- SSH requires extra context switches between the ssh daemon, costing performance, memory, latency, and latency variation, likely an issue on LTSP servers.
- A remote person's identity cannot be determined; only the identity of their local account.

IPSEC might seem to be the easiest solution, and may be necessary to implement as a 'check off' marketing item: however, it does not ensure end-to-end encryption of traffic, and even worse, does not provide user authentication. In IPSEC's use in VPN software, the data is often unencrypted at corporate firewalls and delivered unencrypted, unacceptable for use that involves user keyboard input. It is therefore at a minimum insufficient for SNAP computing, and in some uses, in fact completely insecure.

Therefore authentication, encryption, and compression must be integrated into the X Window System transport to allow for a wider range of authentication and encryption options, to be proxyable to enable secure traversal of administrative boundaries, and to enable use of display resources on displays where you cannot be authenticated. Compression can provide a huge performance benefit over low bandwidth links[27].

4.3 Remote Devices

It has always been trivial for an X application to use a remote display, but when the application is running to a remote X server, there has been a presumption that the input devices are also attached to the remote machine. Having to drape input device cables across the room to plug into the computer driving the display, is clearly ludicrous. We therefore need network transparent input devices.

People may want to use either spare keyboards, a laptop they brought with them, their PDA, or other input devices available in the room to interact with that application. In any case, input events must be routed from the input device to the appropriate X server, whether connected via wires or wireless.

Input devices present security challenges, along with a further issue: we need some way to associate an input device with a particular user. Association setup needs to be both secure and easy

to use, which may present the largest single research challenge; most of the other tasks described in this paper are simple engineering efforts, applying existing technology in obvious ways. One might have hoped that USB's HID serial numbers on devices would help; however, due to the very low cost of many input devices, most manufacturers do not provide actual serial numbers in their hardware.

4.4 Who is in Control?

The X server implementation has presumed that it is in control of all of its input devices, and worse yet, that these do not change during an X session. It uses a static configuration file, only read during server reset (which only occurs when a user logs out). This static model of configuration is clearly wrong, and hotplug is a necessary. The X server needs (as in all good server processes) to react to changes in the environment.

Over the last two years, open source systems have developed extensive infrastructure to support hotplug, with kernel facilities, and the D-BUS[29] and HAL[36] facilities. These should greatly simplify the problem, and allow the policy decisions of whether an input device (local or remote) is connected to a particular X server.

D-BUS can inform the X server of the changes in configuration of input devices. This itself poses a further challenge, as the X server must be able to become a client of the D-BUS daemon. To avoid possible dead-lock situations between X and the D-BUS daemon, some of the internal X infrastructure needs updating.

With only slight care, an interface can be designed that will allow input devices to either use local or remote input devices. Input device association policy should be kept outside of the X server.

4.5 X Input Extension

The X Input Extension[28] provides support for additional input devices beyond the 'core' pointing device (typically mouse) and keyboard. It has a competent design, though it shows its age. XInput lacks:

- Hotplug notification of devices being connected or disconnected.
- The valuator axes should have abstract names (e.g. you would like to know that valuator 0 is the X coordinate, valuator 1 is the Y coordinate, valuator 2 is pressure, and so on).
- Support for multiple users and devices that all users might share.
- A modern reimplementation exploiting the standardization of USB HID (and the `/dev/input` abstraction on Linux); most of the current implementation is supporting old serial devices with many strange proprietary protocols.
- A limit on 255 input devices in the wire encoding (which might become an issue in an auditorium setting); however, if input events are augmented by a identity information, this should be sufficient.

Whether a upward compatible wire protocol version is possible or a new major version of the X Input extension is not yet completely clear, though an upward compatible API looks very likely.

4.6 Toolkits

Replication and migration of running applications has in fact been possible from X's inception: GNU emacs has had the capability to both

share buffers on different X servers, allowing for shared editing of text, and therefore migration of emacs from X server to X server for more than 15 years.

In practice, due to the level of abstraction of the most commonly used toolkit of X's first era (Xt/Motif[22]), migration and/or replication of windows has been very difficult, as such applications initially adjust themselves to the visual types available on the X server and then draw for the rest of their execution with the same pixel values.

Modern toolkits (e.g. GTK[21] and Qt[16]) operate at a higher level of abstraction, where pixel values are typically hidden from applications, and migration of most applications is feasible[20]: a prototype of migration capability first appeared in GTK+ 2.2.

One to one replication of information is the wrong level of abstraction, since not only is the resolution of different screens extremely wide, but different users on different displays should be able to control the allocation of the screen real-estate. A multi-view approach is clearly correct and to be preferred over the existing server based pixel sharing solutions such as xmove[32], useful though such tools are, particularly for migration of old X applications that are unlikely to be updated to modern toolkits. Work to ease replication of windows for application developers awaits suitably motivated contributors.

Since the resolution between a handheld device and a display wall is over an order of magnitude, applications often need to be able to reload their UI layout on the fly for migration to work really well; again, using the Glade user interface builder[3], libglade and GTK+, this capability is already demonstrable for a few applications.

In the face of unreliable wireless connections, the X library needs minor additions to allow toolkits to recover from connection failures. This work is on hold pending completion of a new implementation of Xlib called Xcb[26], which is well underway and now able to run almost all applications. Testing connection loss recovery may be more of a challenge than its implementation.

Lest you think these facilities are interesting only to SNAP computing, it also aids migration of X sessions from one display (say work) to another (e.g. home). As always, security must be kept in mind: it would not be good for someone to be able to steal one or all of your running applications.

4.7 Window Management and Applications

Besides the infrastructure modifications outlined above, window managers need some modification to support a collaborative environment.

Certain applications may want to be fully aware of multiple users: a good example is an editor that keeps changes that each person applies to a document.

Existing applications can run in such a collaborative environment unchanged. Wallace et al.[33] recently reported experience in a deployed system using somewhat jury-rigged support for multiple cursors and using a modified X window manager on a large shared display at Princeton's Plasma Physics Lab's control room. They report easier simultaneous use of existing applications such as GNU Image Manipulation Program (gimp). They also confirm, as hypothesized above, multiple people working independently side-by-side require sufficient display real-estate to be effective; here they may be looking at different views of

the same dataset using separate application instances. And finally, they report that even sequential use of the display was improved due to less dragging of the mouse back and forth.

5 Summary

Most of the problems SNAP computing pose have obvious solutions; in a few areas, further research is required, but none of the research topics appear intractable.

Network display software systems such as Microsoft's RDP[8] and Citrix and VNC[30] are popular, though by operating at a very low level of abstraction, badly compromise full application integration (e.g. cut and paste, selections, window management meta information) between applications sharing a display from many remote systems. They do, however, do a fine job of simple access to remote applications, but are *fatally flawed* if full collaboration among multiple users is desired.

Open source systems SNAP systems should be able to exist quickly, not only since our technology starts off close to the desired end-state, is more malleable, but also that it does not threaten our business model in the same way that such a shift might to commercial systems.

While this paper has primarily explored X Window System design issues, there is obviously plenty of work elsewhere to fully exploit the vision of SNAP Computing.

References

[1] Bonjour. `http://developer.apple.com/darwin/projects/bonjour/index.html/`.

[2] Distributed Multihead X Project. `http://dmx.sourceforge.net/`.

[3] Glade - a User Interface Builder for GTK+ and GNOME. `http://glade.gnome.org/`.

[4] Howl: Man's new best friend. `http://www.porchdogsoft.com/products/howl/`.

[5] Jack audio connection kit. `http://jackit.sourceforge.net/`.

[6] Linux Terminal Server Project. `http://www.ltsp.org/`.

[7] Media Applications Server. `http://www.mediaapplicationserver.net/`.

[8] RDP Protocol Documentation. `http://www.rdesktop.org/#docs`.

[9] Scalable Display Wall. `http://www.cs.princeton.edu/omnimedia/index.html`.

[10] Zero Configuration Networking (Zeroconf). `http://www.zeroconf.org/`.

[11] Stateless Linux, 2004. `http://fedora.redhat.com/projects/stateless/`.

[12] Will Allen and Robert Ulichney. Wobulation: Doubling the addressed resolution of projection displays. In *SID 2005*, volume 47.4. The Society for Information Display, 2005. `http://sid.aip.org/digest`.

[13] Edward Balkovich, Steven Lerman, and Richard P. Parmelee. Computing in higher education: the athena experience. *Commun. ACM*, 28(11):1214–1224, 1985.

[14] Daniel J. Barrett and Richard Silverman. *SSH, The Secure Shell: The Definitive Guide*. O'Reilly & Associates, Inc., 2001.

[15] Andrew Christian, Brian Avery, Steven Ayer, Frank Bomba, and Jamey Hicks. Snap computing: Shared wireless plug and play. 2005. `http://www.hpl.hp.com/techreports/2005/`.

[16] Matthias Kalle Dalheimer. *Programming with Qt*. O'Reilly & Associates, Inc., second edition, May 2001.

[17] C. Anthony DellaFera, Mark W. Eichin, Robert S. French, David C. Jedlinsky, John T. Kohl, and William E. Sommerfeld. The zephyr notification service. In *USENIX Winter*, pages 213–219, 1988.

[18] S. P. Dyer. The hesiod name server. In *Proceedings of the USENIX Winter 1988 Technical Conference*, pages 183–190, Berkeley, CA, 1988. USENIX Association.

[19] Rob Earhart, Tim Martin, Larry Greenfield, and Rob Siemborski. Simple Autentication and Security Layer. `http://asg.web.cmu.edu/sasl/`.

[20] James Gettys. The Future is Coming, Where the X Window System Should Go. In *FREENIX Track, 2002 Usenix Annual Technical Conference*, Monterey, CA, June 2002. USENIX.

[21] Eric Harlow. *Developing Linux Applications with GTK+ and GDK*. MacMillan Publishing Company, 1999.

[22] Dan Heller. *Motif Programming Manual for OSF/Motif Version 1.1*, volume 6. O'Reilly & Associates, Inc., 981 Chestnut Street, Newton, MA 02164, USA, 1991.

[23] J. J. Kistler and M. Satyanarayanan. Disconnected operation in the coda file system. In *Thirteenth ACM Symposium on Operating Systems Principles*, volume 25, pages 213–225, Asilomar Conference Center, Pacific Grove, U.S., 1991. ACM Press.

[24] J. Kohl and B. Neuman. The kerberos network authentication service. Technical report, 1991.

[25] T. Levergood, A. Payne, J. Gettys, G. Treese, and L. Stewart. Audiofile: A network-transparent system for distributed audio applications, 1993.

[26] Bart Massey and Jamey Sharp. XCB: An X protocol c binding. In *XFree86 Technical Conference*, Oakland, CA, November 2001. USENIX.

[27] Keith Packard and James Gettys. X Window System Network Performance. In *FREENIX Track, 2003 Usenix Annual Technical Conference*, San Antonio, TX, June 2003. USENIX.

[28] Mark Patrick and George Sachs. X11 Input Extension Protocol Specification, Version 1.0. X consortium standard, X Consortium, Inc., 1991.

[29] Havoc Pennington, Anders Carlsson, and Alexander Larsson. D-BUS Specification. `http://dbus.freedesktop.org/doc/dbus-specification.html`.

[30] Tristan Richardson, Quentin Stafford-Fraser, Kenneth R. Wood, and Andy Hopper. Virtual network computing. *IEEE Internet Computing*, 2(1):33–38, 1998.

[31] Robert W. Scheifler and James Gettys. *X Window System*. Digital Press, fourth edition, 1996.

[32] Ethan Solomita, James Kempf, and Dan Duchamp. XMOVE: A pseudoserver for X window movement. *The X Resource*, 11(1):143–170, 1994.

[33] Grant Wallace, Peng Bi, Kai Li, and Otto Anshus. A MultiCursor X Window Manager Supporting Control Room Collaboration. Technical report tr-707-04, Princeton University Computer Science, July 2004.

[34] Eamon Walsh. Integrating XFree86 With Security-Enhanced Linux. In *X Developers Conference*, Cambridge, MA, April 2004. `http://freedesktop.org/Software/XDevConf/x-security-walsh.pdf`.

[35] David P. Wiggins. Security Extension Specification, Version 7.0. X consortium standard, X Consortium, Inc., 1996.

[36] David Zeuthen. HAL Specification 0.2. `http://freedesktop.org/~david/hal-0.2/spec/hal-spec.html`.