

Multimedia Session Management for Thin Clients

based on the X Window System

Michael Kropfberger, Hermann Hellwagner
University of Klagenfurt, Institute of Information Technology
mike@itec.uni-klu.ac.at, hellwagn@itec.uni-klu.ac.at

Abstract Since the concern of *total cost of ownership* of personal workstations came up from a study of the Gartner Group [2], server centric computing with network attached thin clients is getting interesting again.

To make remote computing more effective, more and more products offer session management environments, where a user reconnects to an always running server-side session, just redirecting its graphical and acoustic output to his/her local workstation or thin client.

This paper introduces a newly developed X-based session management with sound forwarding, which perfectly integrates into the Unix world and highly outperforms the open source competitor in this field, VNC [3].

Keywords: Session Management – X Window System – VNC – Citrix – SunRay

1. Introduction and Rationale

Following a study of the Gartner Group [2], *total cost of ownership* can be highly reduced by switching to server centric computing and network attached “dumb” – but silent – thin clients with no harddisks and no system administration needed. Those thin clients only show the graphical output of a session, which is completely started on the server.

Modern session management products allow the immediate redirection of a running session to any remote display. This implies user roaming (switching the actual workplace without losing the actual session) and thereby a session roaming eg. from the company workplace back home for teleworking.

In section 2 we outline different existing session management protocols. Section 3 introduces a new protocol based on the X window

system [9]. This work was performed within a project that developed a high-performance, completely noiseless thin client for CAD applications, based on Linux and the quasi-standard X Window System. The protocol is highly efficient (as shown in section 4), is extensible, and is based on a well defined standard. In addition, we describe an already existing open-source program for sound forwarding (discussed in section 5), which was extended to suffice the needs of session management. Those two session management facilities for remote audio and displaying together lead to a multimedia session management ideal for the CAD thin client mentioned above. Section 6 gives an outlook on future work and section 7 gives a final conclusion about this paper.

2. Remote Display Protocols

Because of space constraints we will only show an extensive comparison chart containing AT&Ts VNC [3], Microsoft Terminal Server [5], Citrix Metaframe [6], Tarantella [7], SunRay [8], X [9] and compressed X [12]. For further in-detail comparisons see [1].

Product	Server OS	Client OS	Display En-coding	Screen Updates	Client Caching	Max. Resolution
MS Terminal Services	Win2000, WinNT	Win2000, WinNT	compressed graphics	server frame buffer, pushed adaptively	glyphs, small bitmaps in RAM, large bitmaps on disk (Cache: 1.5 MB RAM, 10 MB HDD)	1024x768 @8bit
Citrix Metaframe	Win2000, WinNT	Win, OS/2, Linux, Unices, DOS, Mac	compressed graphics	server frame buffer, pushed adaptively	glyphs, small bitmaps in RAM, large bitmaps on disk (Cache: 3 MB RAM, 1% of HDD)	no restriction

SCO Taran- tella	Solaris, AIX, SCO, Linux	Java- enabled web browsers (MSIE, Netscape)	compres- sed graph- ics, direc- tives or raw (adap- tive)	server frame buffer, pushed adaptively	glyphs and bitmaps (only in RAM)	no re- striction (depends on middle- ware server memory) @8 bit (newest devel- opment version: 24bit)
SunRay	Solaris	proprie- tary device	compres- sed pixels	update pushed on each window system command	None	no restric- tion
At&T VNC	Windows, X, Mac	Win, X, Mac, Java, WinCE	2D run- length en- coded pixels	screen differences sent on client pull	None	no restric- tion
X	Xlib appli- cations (Win, Unices, Linux)	X- Server (Win, Unices, Linux, Mac, Java)	draw primi- tives	block-wise sending of X com- mands (eg. 5 in a row)	glyphs, fonts, pixmap	no restric- tion
X with ML- View Com- pressor	Unices, Linux	X- Server (Win, Unices, Linux, Mac, Java)	message caching, draw primi- tives with bitmap compres- sion	block-wise sending of X com- mands (eg. 5 in a row), intelligent reordering and prior- itizing	glyphs, fonts, pixmap	no restric- tion

3. X-Ray – X-Based Graphical Session Management

Basically, the aim of this work was to write a graphical session management which is not based on a pixel oriented protocol like VNC's *Re-*

rote Frame Buffer [14] (RFB), but on graphics directives like drawing rectangles and circles. Using the well-defined X Protocol allows connecting to X proxies like *ML-View* [12], which compresses pixmap transfers and optimizes X command queues by their importance. On the client-side, having the uninterpreted X commands allows high optimization to the local hardware graphics acceleration or font and color rendering.

As an XFree86 [10] addon, the nested X-Server *XNest* acts as an X command proxy. On a regular X display, *XNest* connects and comes up like a normal application window, but other X applications can direct their output into this *XNest* window. These X applications believe to be started onto a valid, real X server. This includes window managers, background “beautifiers” like *xearth* or screen savers. *XNest* hereby is a kind of layer between applications and another real X-Server. It internally forwards all X commands directly to the hosting X-Server, and is hereby device independent.

X applications (using the X libraries [11]) always need a running X-Server to connect to. At startup, they negotiate about provided color depths, default screens and other X-Server specific features and also during run-time, they need a correctly working output, where they can draw on or read the contents of specific screen areas.

For a session managed environment, this means, we cannot stop offering a fully functional display for running applications, just because there is no connected thin client at any time. The logical answer is to keep two displays, one on the server, and one – if a thin client is connected – on the client-side.

3.1 XRay Internals

X-Ray, highly based on *XNest*, allows all running applications to draw and read screen contents on a dedicated session server (called *X-Ray server-side*) and – if a thin client (called *X-Ray client-side*) is connected – all X commands are also sent to the thin client. The left illustration in Figure 1 shows the general idea of the *X-Ray* session management. With no *client-side* attached, *X-Ray* offers the same functionality as *XNest*.

This double draw mode is called *fulldraw* mode, since we double network load to reach both X-Servers for output and input. To optimize display performance, we also offer a *clientdraw* mode, which reduces all draw primitives (draw lines and circles, put pixels...) to either the *X-Ray server-side* or – if connected – to the *X-Ray client-side*.

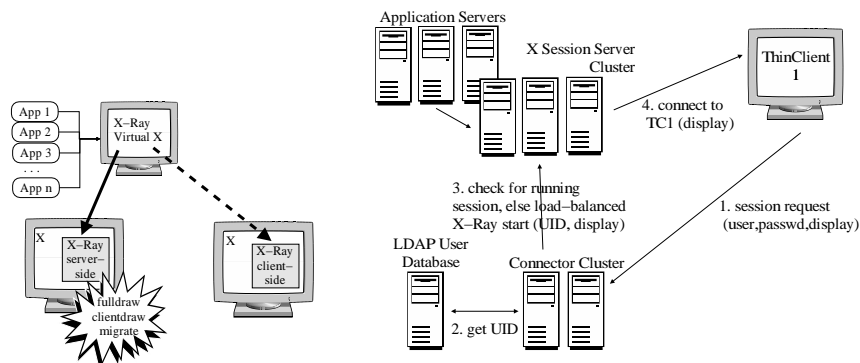


Figure 1. X-Ray Draws on Server-Side and – if Connected – on Client-Side (left); X-Ray Connection Scheme in a Large-Scale Client-Server Environment (right)

3.2 X-Ray in a Large-scale Client-Server Environment

Server centric computing involves many dedicated servers for different applications and tasks. To use X-Ray, a single all-in-one server would be sufficient, but to scale up for many users (which is the intended application environment), we will introduce a more distributed environment.

The illustration on the right-hand side of Figure 1 shows a possible upgrade of an existing distributed computing infrastructure with X-Ray session management capabilities.

We need some *X session servers*, which act as the *X-Ray server-side* session, where one single master X server has to be started to serve all user sessions.

The *session servers* may also be used as the *application servers* for user applications, but it is also possible to out-source this job to a special *application server cluster*.

There is one dedicated server, acting as the *connector*, so all thin clients at the offices are configured with the *connector's* host name for session connection. If high availability and load-balancing of a *connector cluster* are also necessary, the easiest would be the DNS standard feature of round-robin DNS entries.

4. Performance Measurements

In [13] RDP, Citrix, Laplink, VNC and SunRay are compared using the Ziff Davis *i-bench* Web benchmark, which includes displaying text, pixmaps, scrolling a screen and a flash animation clip. All these tests are done with various bandwidths from ISDN up to 100 Mbps LAN.

But because of the totally different approaches and hosting systems of our environment, it is hard to get really good benchmarks. Also a Web benchmark does not really reflect the daily work in an average office.

In this paper, we only compare the free remote display competitors, which are native X, VNC and X-Ray in diverse setups. We also compare all three with the same benchmark. The result is not based on measuring a made-up user scenario but the detailed performance factors of all available X library directives.

4.1 Benchmark Setup

Three computers were connected via a dedicated 100 Mbps line, *svray* as the server machine, *thray* for throttling (traffic shaping) and *clray* as the client machine.

All three are equipped with an AMD Athlon 1.3 GHz with 768 MB of RAM, where the two outer computers, *svray* and *clray*, use a Matrox G-450 graphics card for their output. All tests normally did affect more of *clray's* CPU load because of the X-server displaying the draw primitives on the local display. For the VNC test, CPU power was also needed on the server-side because it had to compress the screen contents before sending them.

`x11perf` was used for benchmarking, which comes with the XFree86 distribution and tests all existing X drawing directives. Graphical tests like drawing filled rectangles are always done with sizes of 1[x1], 10[x10], 100[x100], and 500[x500] pixels. Other directives are eg. window management or color (palette) functions. It is up to the benchmark user to weigh the importance of X commands according to the needed application.

For detailed information about the setup and more result graphs refer to [1].

4.2 Results

All following graphs show the relative performance of the different candidates, where the best candidate has a 100% bar height. The following graphs are generated from cumulated benchmark results like all rectangle operations in different sizes from 1[x1], 100[x100], and 300[x300] up to 500[x500] pixels.

Figure 2(a) shows an overall benchmark on a 100 Mbps network, covering all possible `x11perf` benchmarks. We see the best result for native X directly connecting to *clray*. *X-Ray* in `clientdraw` mode is slightly slower than the equally designed *XNest*, since it introduces a second branch for the second display. As soon as we look at larger rectangles,

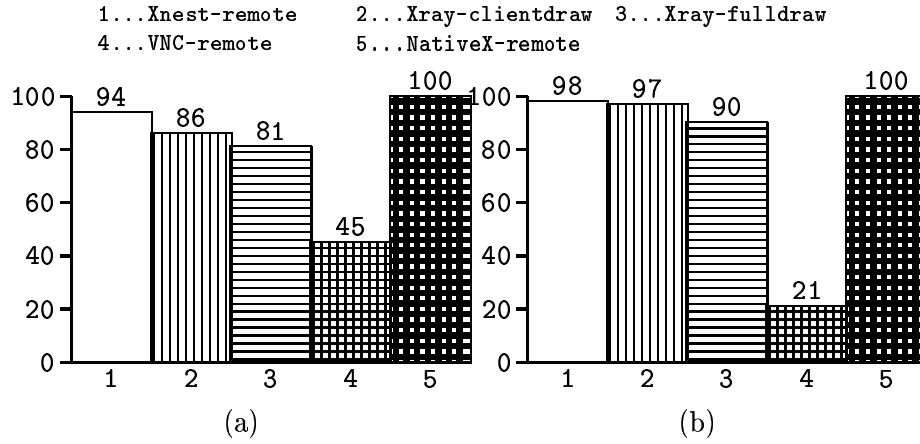


Figure 2. x11perf Benchmark Results on a 100 Mbps Network: (a) All Drawing Operations (b) 100[x100], 300[x300], and 500[x500] Pixel Drawing Operations

lines, circles and image operations, *VNC* is even more outperformed by all X based approaches (Figure 2(b)).

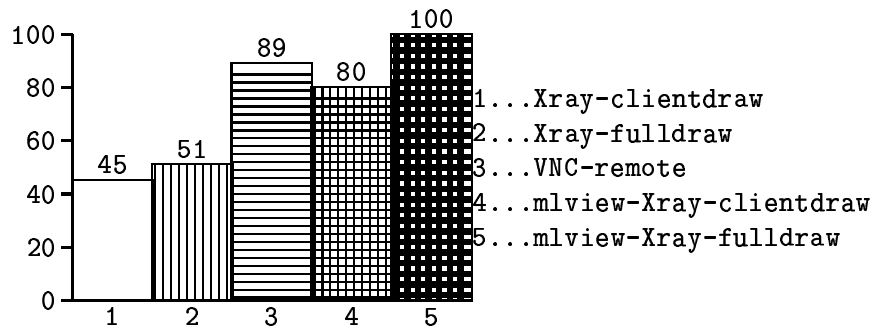


Figure 3. x11perf Benchmark Results on a 10 Mbps Network: *VNC* and X-Ray With and Without X Compression Containing All Drawing Operations

Figure 3 shows the overall benchmark on a 10 Mbps network. *X-Ray* without compression – substitutional for all X based approaches – seems to be outperformed by *VNC*. This is mainly because the network is now the bottleneck and *VNC* has enough CPU power on both sides to (de-)compress the sent framebuffer changes. Further, *VNC* now requests fewer display updates because of the RFB specific pull approach [14]. The *VNC* client queries the *VNC* server only when it fully received and painted the actual screen update. This leads to an extremely

higher performance, since the *VNC* server runs on the *server-side* and paints all output only to a virtual memory framebuffer without any synchronization with the *VNC* client. But this performance gain is highly misleading, since a user doesn't see screen changes at a constantly high frame-rate; this leads to a loss in interactivenss and even a loss of information (eg. a movie is only shown with 10 fps instead of 24 fps).

X-Ray in `fulldraw` mode in connection with the *ML-View* [12] package is the fastest candidate.

Since the *ML-View* server also runs on the server machine, the acknowledgement for X commands arrives without ever leaving the computer. This is valid for both `fulldraw` and `clientdraw` modes. But `fulldraw` mode is faster in querying X-Server states or getting images, since there is a fully valid instance running on the local machine.

Concluding the latter measurements, *X-Ray* session management offers a much better solution than *VNC* in terms of graphics performance, including user interactivenss and guaranteed displaying of screen changes. This is valid especially on a local area network (LAN) with 100 Mbps or 10 Mbps. On slower lines, this is only dependent on the X compression done by *ML-View*.

ML-View is under development, so X measurements on slow lines will have to be done later, to show *ML-View's* competitive ability versus other low-data-rate protocols like Citrix Metaframe.

5. Sound Forwarding

Since we also want to offer audio features in our session management system, we first have to overcome some severe obstacles. All problems stem from one single fact: All applications run on the server-side but are displayed on a (roaming) client-side. Also audio output is generated on the server-side, but has to be forwarded to the connected thin client. During a session, the user may change his/her working thin client, and hereby require to redirect the sound. This has to happen without user intervention and even without bothering and changing the already running sound applications.

We developed a solution based on *dsproxy* [15], which implements a reconnect feature, allows multiple users on one server to access the same `/dev/dsp` sound device, and blocks running applications on disconnect to a certain degree. It is not capable of sound recording and (adaptive) compression, but these features could be easily added, since the necessary hooks in the source code are available.

6. Future Work

To allow video to be played through the session management, a split-
ted player software is necessary. It locally draws a window with the play,
pause and stop buttons and sends the compressed video stream to the
connected thin client over a parallel connection. The stream is decom-
pressed on the client-side and exactly drawn into the predefined window
coordinates of the player software.

To present an optimal X-based solution, we have to embed the session
management features directly in every hardware-optimized X server, so
no extra layer and no visible server-side connection is necessary. Also
the compression features of *ML-View* for pixmaps and reordering of X
commands by their importance (with respect to network latency) should
be directly included in the XFree86 code. Finally, adaptive compression
for sound forwarding would increase acceptance because by that we could
offer acoustic user feedback also on slow networks.

7. Conclusion

This paper shows the necessity and feasibility of features like multi-
media session management and should help to extend future X protocol
versions to reflect the new needs of server-centric thin client computing.
If we believe in the research of well-known institutes like the Gartner
Group [2], the modern thin client approach will be very successful in
the next years, mostly because of reducing the *total cost of ownership* of
computing infrastructures. Besides, this will also increase user efficiency
because of absolutely silent thin client computers with no hard disks or
CPU fans.

Even in this early stage of these “working prototypes” for proof of
concept, connecting *X-Ray* with *ML-View* X compression and sound
forwarding offers a viable alternative for a Unix-based environment.

References

- [1] M. Kropfberger, “Open-Source Multimedia Session-Management for
Thin Clients based on the X Window System”, September 2001,
<http://www.kropfberger.n3.net/xray.html>
- [2] The Gartner Group, <http://www.gartnergroup.com>
- [3] VNC – Virtual Network Computing, AT&T Laboratories Cambridge,
<http://www.uk.research.att.com/vnc/>
- [4] T. Richardson, K. R. Wood, “The RFB Protocol”, January 1998,
<http://www.uk.research.att.com/vnc/protocol.html>
- [5] Microsoft Corp, Technical White Paper, “Terminal Server Edition, version 4.0:
An Architectural Overview”, Redmond, WA, 1998

- [6] Citrix, "ICA Technology Brief", March 1996, <http://www.kios.de/datenblaetter/ica/icatech.htm>
- [7] SCO Inc., Tarantella White Paper, "Tarantella Enterprise 3 Technical Overview", May 2001, <http://www.tarantella.com/whitepapers/overview/>
- [8] Sun Microsystems Inc., "SunRay 1 Enterprise Appliance Overview and Technical Brief", August 1999
- [9] X Consortium for the X Window system, <http://www.x.org>
- [10] The XFree86 Project, Free X Window System <http://www.xfree86.org>
- [11] James Gettys, Robert W. Scheifler, "Xlib - C Language X Interface Reference", X Consortium Standard
- [12] ML-View by Medialogic s.r.l., <http://www.medialogic.it>
- [13] Jason Nieh, S. Jae Yang, Naomi Novik, "A Comparison of Thin Client Computing Architectures", Technical Report CUCS-022-00, Network Computing Laboratory, Columbia University, November 2000, <http://www.ncl.cs.columbia.edu/publications/cucs-022-00.pdf>
- [14] T. Richardson, K. R. Wood, RFB - Remote Framebuffer Protocol for VNC, "The RFB Protocol", January 1998 <http://www.uk.research.att.com/vnc/protocol.html>
- [15] Tony Bybell, dsproxy, <http://www.linux-workshop.com/bybell/dsproxy>