Profiling the X Protocol *

John Danskin † Pat Hanrahan

Princeton University
Computer Science Department
Princeton NJ, 08540
jmd@cs.princeton.edu

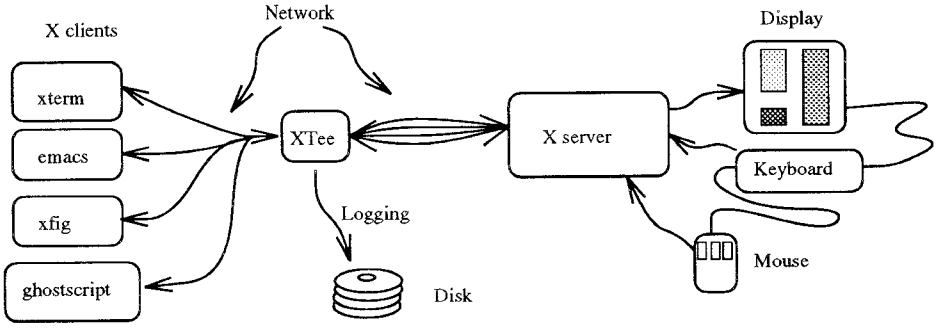

Figure 1: X clients attach to X server through logging process *XTee*

Network transparent window systems, such as X, allow applications to run remotely from their displays. X connections can be made over any network supporting a reliable byte stream interconnect: most commonly ethernet with TCP/IP, but also including telephone lines and other low bandwidth media running reliable protocols. We have gathered traces of complete X sessions and individual applications, and generated statistics on bandwidth and granularity, as well as the relative percentage of bandwidth consumed by different types of X traffic such as geometry and text.

Our traces show that X messages and groups of X messages which could be packed together in a single network packet are fine grained, usually under 100 bytes. This means that the SLIP protocol, which imposes 48 bytes of overhead on each packet, is inefficient for X traffic, but that the CSLIP protocol, which has an average overhead of 3 bytes per packet, is efficient for X traffic. There are also implications for modem design: often the direction of traffic changes twice every 100ms, as shown in Figure 2. Modems which impose delays of tens of milliseconds to reverse the direction of traffic, or to implement compression or error checking will not efficiently transmit X traffic.

To gather traces, we developed *XTee*, a logging program which is interposed between X clients (applications) and the X server, as shown in Figure 1. *XTee* logs the complete X stream in an easy to parse binary format, tagging each X message with a timestamp and a sequence number.

Our 12 traces cover document previewing, 2d line drawing, games, and graduate student activity such as software development, reading mail and writing papers. Most of the traffic in all of these traces was drawing related, not state or window manipulation. In every trace, the top 4 X messages accounted for more than 75% of the bytes transferred. Although X messages can be 64K bytes long, most X messages have 32 bytes or less, although in some of the traces, most of the bytes were in messages of over 1K bytes.

We are using this data to guide the design of compressed protocols for X.

†Funded by an AT&T PhD fellowship and a grant from Apple Computer

*Full paper available from Princeton University Computer Science Department as CS-TR-441-94

SIGMETRICS 94- 5/94 Santa Clara, CA. USA
© 1994 ACM 0-89791-659-x/94/0005..$3.50

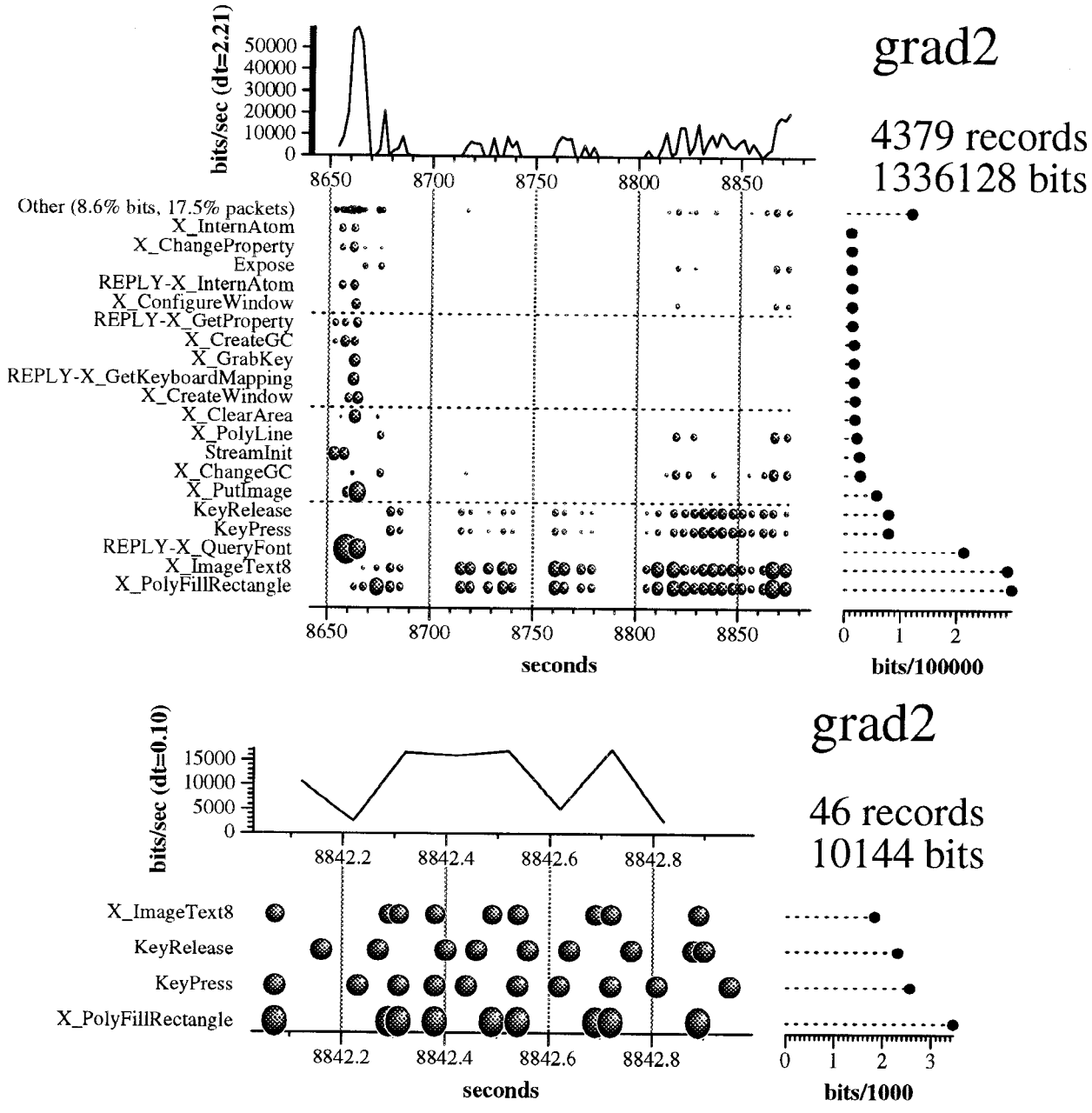

Figure 2: These graphs are excerpts from the grad2 trace showing a terminal emulator (*xwsh*) starting up. The top graph covers almost four minutes, while the bottom graph is a one second detail. The X axis indicates seconds since session startup, while the Y axis shows the types of the messages being sent. Each sphere represents a single message unless it is elongated vertically, in which case it represents multiple messages arriving nearly simultaneously. The volume of each sphere is proportional to the number of bits in the message(s) represented. The dash-dot plot at the right of each graph shows the total number of bits for each type of packet. The line plot at the top of each graph shows the number of bits per second as a function of time. The integration interval is listed in the Y-axis legend.

At the left of the top graph we see *xwsh* starting up. The terminal emulator creates windows, allocates colors, and queries fonts, as we can see by examining the left hand side of the top graph. Looking at the top left, we can see that the instantaneous network load peaks at nearly 60K bps. After about a minute, the variety of different packet types decreases markedly and a repeated pattern involving 4 primitives appears. The pattern is easier to follow in the detail graph below: The server sends a KeyPress event, the client responds with X_PolyFillRectangle (to clear a space) and X_ImageText8 (to draw the character), then the server sends KeyRelease and another KeyPress. The user is typing. The network load frequently reaches 17K bps over 100ms periods, indicating that this level of performance is required to support this typing application without noticable degradation.