Performance Measurements of the X Window System Communication Protocol[†]

(To appear, *Software: Practice & Experience*.)

Ralph Droms
Bucknell University
Lewisburg, PA

Wayne R. Dyksen
Purdue University
West Lafayette, IN

October 15, 1990

Abstract

The X Window System[1] is a portable, network transparent window system. An X *server* manages the resources of an X *display* consisting of a screen, keyboard and pointer. An X server distributes user input (*events*) to and accepts output (*requests*) from X *clients* which vie for display resources.

X provides an environment in which servers and clients may be separated by a communications network so that application programs can use graphical interfaces on remote hosts. Servers and clients communicate through a well-defined protocol that in turn uses protocols like the Internet TCP/IP protocol suite as a transport mechanism.

Separation of servers and clients by a network introduces the possibility of performance degradation due to communication delays. These delays may be either due to *latency* resulting from round-trip delays between servers and clients or *transmission* delays resulting from the volume of traffic through the communications network. The X protocol is rigidly defined at a fairly low level, so that some forms of interaction between servers and clients may require the exchange of many events and requests through the underlying transport protocol across the communications network. Thus, there are many opportunities for network communication delays to affect the execution of X programs.

This paper investigates the resources required by the X protocol from the communications network to evaluate the potential for communications delays induced by X protocol traffic. A protocol analysis tool, `xlog`, monitors the use of the communications protocol between a server and clients, and records a log of protocol messages for post-mortem analysis.

[†] This research was partially supported by a grant from Sequent Computer Systems, Inc.

[1] *X Window System* is a trademark of the Massachusetts Institute of Technology.

We present the results of measurements of the X protocol taken with `xlog` for a variety of X clients and servers communicating over a 10Mb per second Ethernet. While the overall network usage observed was not overly high, clients and servers using the X protocol did exhibit bursty behavior in which transitory bandwidth requirements were significant. Some X applications may induce a significant load on the communications resources provided by an Ethernet. Applications developers and network architects must be aware of the load caused by the X protocol when designing networked graphic systems.

1 Introduction

The X Window System [14] provides mechanisms for building portable, network transparent windowing systems. X is based on a client/server paradigm in which an X *server* manages the resources of an X *display* consisting of a screen, keyboard and pointer (mouse), while X *clients* vie for display resources. An X server acts as the intermediary between the display and clients by distributing user input (*events*) to and accepting output (*requests*) from clients.

The network transparency of X results from the separation of servers from clients by a well-defined communication protocol, the *X network protocol* [12]. Requests and events are sent between servers and clients using this X communication protocol interface. The protocol requires the existence of a reliable, duplex byte stream.

The X Window System designers made a conscious decision to define a low level interface so as not to unduly influence or limit the functionality of application programs [13, page 79]. The nature of the interface services is such that certain forms of common interaction may require that large amounts of protocol traffic be sent between servers and clients. For example, while cursor positioning is usually performed by a server, cursor tracking is often performed by clients. More specifically, cursor tracking is needed for line dragging which is implemented and used by many clients including most window managers. Such clients must be sent an X event for each cursor motion; in response, the client immediately issues graphics requests to erase and draw lines.

Characterizing the network traffic generated by X is especially important in light of the mechanisms used. The X protocol uses commonly available interprocess communication mechanisms as the transport mechanism for the communication protocol – most notably, UNIX[2] Domain Sockets (UDS) [15], the Internet TCP/IP protocol suite [9, 10] and the DECNET protocol suite [17]. Thus, for the latter mechanisms, the X protocol traffic between servers and clients represents network communication that requires packet switch resources and network bandwidth.

The resources required by X servers and clients include packet forwarding by routers, packet processing by hosts on which the X servers and clients are executing, and packet transmission by the Ethernet itself. Contemporary routers and hosts are capable of processing packets at the rate

---

[2] UNIX is a registered trademark of AT&T Bell Laboratories.

at which an Ethernet can transmit packets [2, 3]. Thus, the Ethernet itself represents the limiting communications resource.

While the designers of X claim that the display hardware, rather than network communications resources, are the primary bottleneck in X applications [13, page 80], they based their statement on observations of a single X client co-located with an X server, and of that same client running on a remote host and communicating with the server across a network. A communication network may, however, support multiple X protocol connections simultaneously. In our study, we look more carefully at the percentage of communications resources used by the X protocol, to predict the effect of multiple X applications in a typical distributed computing system.

There are two types of delays that might be introduced by network communications between an X server and client. The first is *latency* delay, caused by the effect of round-trip transmission times between a server and a client. The second is *transmission* delay caused by the finite bandwidth available on an Ethernet. Our study examines transmission delay, by determining the percentage of available data transmission bandwidth consumed by the X protocol as used by several specific X applications.

In this paper, we present an analysis of the X protocol in which we measure the number and size of messages passed between a server and clients for a variety of test cases. These data are then compared to the communication resources available to servers and clients in an Ethernet-based inter-network. The resulting bandwidth utilization patterns can help guide X programmers and network architects in the design of systems that use X by pointing out where the X protocol may represent a significant load.

2 The `xlog` Tool

`xlog` is a program that monitors TCP communication between an X server and its clients. `xlog` is an independent process that stands between a server and its clients, logging all X protocol messages sent between them. The effect is much the same as the UNIX utility `tee` which copies its standard input both to standard output and to a file. The data described in Sections 3 and 4 were collected using `xlog`.

In operation, `xlog` receives X protocol messages (events) from the server via TCP/IP, which it records to a log file and forwards to the client. Messages (requests) from clients are similarly recorded and forwarded to the server. Each log entry is tagged with a direction, a connection number and a time of day for post-mortem analysis.

`xlog` is completely transparent to the server and clients whose session is being recorded. No special actions need to be taken on the part of either the server or clients in preparation for logging X protocol traffic data.

To set up a logged connection, the client establishes a TCP connection to the `xlog` process, which then establishes a corresponding TCP connection to the server. By default, `xlog` running on host `hostname` accepts X protocol connections on TCP port 6001 which is specified by a client as server `hostname:1.0`. `xlog` in turn establishes a TCP connection to its local server on port 6000, corresponding to server `hostname:0.0`. `xlog` can be run remotely from the server by explicitly identifying the server to which connections are to be established. This is especially useful when logging interactions with servers that run in environments that cannot support `xlog`, such as dedicated Xstations.

A companion program, `analyze`, extracts byte-level transmission data from the log file generated by `xlog`. `analyze` reads an `xlog` log file and prints the time of the protocol transaction, the direction and the number of bytes transmitted. This data represents a raw count of the bytes passed across the X protocol connection. No attempt is made to interpret the protocol stream itself by extracting information about individual X protocol records[3].

The output of `analyze` is suitable for further post-mortem processing to extract data for subsequent statistical or graphical analysis. One convenient mechanism for such analysis is the `awk` programming language [1]. The data displayed in the graphs and the analysis reported in this paper were generated by running `awk` on the output of `analyze`.

3 Measurements

In this section we report the results of measurements of the X protocol taken with the `xlog` tool. Throughout, the X server is X.V11R4 (X Version 11 Release 4) running on a Tektronix 4319 workstation with a 1280x1024x8 color display and 8Mb of memory. The clients are several X application programs, also running on a Tektronix 4319 attached to the same ethernet network as the server workstation. The `xlog` program recorded the protocol exchanges from a third Tektronix 4319 on the same ethernet network.

Table 1 summarizes the measurements we conducted. Figures 1 through 8 show plots of bytes and packets logged per second for each measurement. Each plot also shows a running utilization of Ethernet bandwidth and a distribution of the sizes of packets logged during each experiment. Because of the amount of data logged in the `xbench` and `session` experiments, the data were plotted as averages across two seconds and ten seconds, respectively. All other data were plotted at an interval of one second.

To measure datagram counts, we verified that `xlog` trace events can be used to approximate IP datagrams. To understand the relationship between `xlog` trace events and IP datagrams, the `xterm` application was run through `xlog` and simultaneously recorded by the Sun network tool `etherfind` which records ethernet packets. Figure 9 shows the relationship between `xlog`

---

[3] `xlog` provides a record-level trace via a `-verbose` option.

X Application	Description	Figure
`xdvi`	Page through an eighteen page LaTeX document containing text, mathematics and graphics	1
`xbench`	Send an exhaustive, benchmarking sequence of graphics requests [5]	2
`xwd`	Dump an X root window to a file, undump the 1Mb file via `xwud`, iconify and deiconify the `xwud` window	3
`twm`	Select, position and resize an `xterm` window	4
`motion`	Send and receive X protocol requests and events as quickly as possible via cursor tracking	5
`xterm`	Display 1Mb of text	6
`emacs`	Page through a 1Mb file	7
X session	Log all X server activity for 8 hours	8

Table 1: X protocol traffic measured with `xlog`.

and `etherfind` in this case. The `xlog` and `etherfind` logs were then compared. In most cases, each `xlog` event was carried in a separate IP datagram. Some larger `xlog` events required multiple IP datagrams but were recorded as one `xlog` event. In no case were multiple `xlog` events compressed into a single IP datagram. Thus, a count of `xlog` events is a good, lower bound on the number of the IP packets generated by an X client.

The bandwidth utilization was computed by examining the logged data and computing, for unit time, the minimum time required to transmit the logged protocol messages during that time period. The computation included allowing time for 40 bytes of header overhead for the TCP and IP network protocols, time for 26 bytes of header and trailer overhead for Ethernet frame transmission, and time for the 9.6 microsecond inter-frame gap [4]. Small messages were padded out to the 576 bit Ethernet frame minimum. The utilization was then represented as the fraction of time required to transmit the logged protocol messages; i.e., if 200 messages, containing a total of 240,000 bits, and requiring 200 inter-frame gaps were transmitted in a one second period, the utilization was recorded as .026 (or 2.6%) over that period.

`xbench` is a repeatable, well-known benchmark that generates significant X protocol traffic.

Machine	Planes	line	fill	blt	text	arc	complex	xstones
Tek 4319 (no `xlog`)	8	33122	4385	7151	30000	681445	12156	11450
Tek 4319 (with `xlog`)	8	32067	4881	7023	31250	681951	10392	11306

Table 2: Results of running `xbench`

Table 2 shows the results of running `xbench` both with and without X protocol traffic logging, demonstrating that the overhead imposed by `xlog` is minimal.

4 Analysis

Examination of the graphed data recorded from the measurements listed in Table 1 reveals several examples in which the X communications protocol can consume a significant portion of the bandwidth of an Ethernet. Because a typical network may support many simultaneous X protocol sessions, the aggregate throughput to support multiple sessions may begin to cause transmission delays due to bandwidth exhaustion.

Figure 1 shows that `xdvi` requires a transitory transfer rate averaging roughly 60,000 bytes per second and using as much as 6% of the Ethernet bandwidth to display a full page of bit-mapped font text. `xbench` produces three short periods of 70,000 to 80,000 bytes per second, using 6-7% of the Ethernet bandwidth, as shown in Figure 2. Figure 3 shows a short spike of 150,000 bytes per second, with three other spikes of 100,000 bytes per second.

The `twm` window manager, as demonstrated in Figure 4, can generate in excess of 100 packets per second while resizing or positioning a window. However, the plot of packet sizes shows that these packets are quite small, and, even including protocol and Ethernet overhead, only represent a small portion of the Ethernet bandwidth. The packets are used to implement cursor tracking in `twm` via X protocol motion events. The `motion` program uses just those X protocol motion events to determine the speed at which an X client might generate X protocol messages. Figure 5 show that cursor tracking via `motion` generates motion events and draw requests in excess of 150 packets per second, which consume only about 1% of the Ethernet bandwidth.

For a comparison of network resource usage between X applications and other equivalent that did not use the X protocol, we measured bytes and packets per second generated by running `csh` and GNU through both `telnet` and X. Comparing figures 10 and 11 with figures 6 and 7 shows that the overhead associated with running these text applications through X is not significant. The two mechanisms, `telnet` and the X protocol, exchange essentially the same data for both text display and GNU `emacs`. The X protocol does require extra overhead for the protocol headers on both client–to–server and server–to–client traffic. However, the server–to–client traffic (keystrokes) is slow enough that the extra overhead is insignificant and the client–to–server traffic (1Kb blocks of text) is grouped into large enough records that the X protocol overhead is insignificant.

Figure 8 displays a graph of an `xlog` trace of an eight hour X systems programmer session. During this session, all clients run by the user, including a window manager, `xterm` and `emacs`, were directed through `xlog` and the X protocol messages were recorded. During the session, 6,738,400 bytes and 45,108 packets were exchanged, for an average of 236.36 bytes and 1.58 packets per second.

5 Conclusions

While the total throughput required by the X protocol is not overly high, the number of messages and the burst rate can be significant. Because each message requires a fixed overhead to pass through the communication protocol, reductions in the number of transmitted messages may lead to significant performance improvements.

The average network communication requirements of X servers and clients are not significant compared to other resource consumers such as diskless workstations or distributed file systems. In some cases, however, a single client/server X protocol stream can consume up to roughly 15% of the communications resources of an Ethernet. Certain clients, such as the `twm` window manager and `xwd` can generate bursts of network traffic in excess of 100,000 bytes per second or 100 packets per second. This level of traffic suggests that care must be taken when configuring multiple X servers on a single local network, such as when connecting multiple Xstations to a single Ethernet

The traffic generated by some X clients may also represent a significant load on Internet communications resources, so that care should be taken when designing and placing X clients so as to minimize the X protocol traffic. Many of the applications studied in this paper are capable of generating enough traffic to consume the entire bandwidth of a 56Kbyte/second point-to-point connection, and some of the applications consume a significant portion of the bandwidth of a 1.544Mbyte/second (T1) link.

The data presented in this paper may be compared with other network communications resource consumers. For example, a network designer might want to choose between a network of Xstations running applications on a single host or a network of diskless workstations using a single network file server. The first architecture would primarily use the X protocol across the local network, while the second architecture would use a distributed file system protocol like NFS [11, 16]. The average throughput recorded in the 8 hour user session is significantly less than that reported for a diskless workstation by Gusella [7]. The maximum observed packet and throughput rates for the X protocol of 150 packets per second and 150,000 bytes per second, as demonstrated by `xbench`, `twm` and `motion`, is roughly equivalent to the maximum rates of 20 packets per second and (by inference) 160,000 bytes per second reported by Howard, et al. [8]. Thus, while an X server session may represent a lower average load than a diskless workstation, an X server may require as much network bandwidth as a diskless workstation over the short term, so that trading remote file system network traffic for X protocol traffic would likely but not necessarily reduce network traffic.

Another concern of network designers is the relative expense of the X protocol as compared to existing character transmission protocols like telnet. The data from the `xterm`, `emacs` and `telnet` experiments suggest that the X protocol does not impose a significant additional overhead on character-oriented applications.

Finally, X programmers need to be aware of program organizations that might consume unnecessarily large amounts of network communication resources. The `twm` and `motion` experiments demonstrate that some program constructs may, perhaps unexpectedly, generate significant numbers of X protocol messages. The incorporation of a mechanism to move computation from the client to the server, similar to that found in NeWS [6], might contribute to a reduction in network traffic in some applications.

There are two additional types of measurements that are not addressed here. First, the experiments and analysis in this paper examine only transmission delays, not latency delays. Latency delays may be significant in highly interactive applications where the client and server are separated by an internetwork connection that spans many intermediate networks. The `xlog` tool could be used to measure latency delays in certain instances where client requests can be matched with server events in the logged data.

The second measurement is in support of the statement that contemporary hosts are capable of generating and receiving packets as fast as an Ethernet can deliver them. Clark, et. al, measured the overhead inherent in processing TCP protocol messages and compute that a 10 MIPS processor can process 25,000 packets per second. Application and network designers might also be interested in the observable maximum number of packets that can be processed through the X protocol as implemented on typical UNIX systems.

References

[1] Alfred Aho, Brian W. Kernighan, and Peter Weinberger. Awk – A pattern scanning and processing language. Technical report, Bell Laboratories, Murray Hill, NJ, 1974.

[2] S. Bradner, August 1990. Private communication.

[3] D. D. Clark, V. Jacobson, J. Romkey, and H. Salwen. An analysis of TCP processing overhead. *IEEE Comm. Mag.*, 26(7):23–29, July 1989.

[4] D. Comer. *Internetworking with TCP/IP*. Prentice–Hall, Englewood Cliffs, New Jersey, 1988.

[5] C. Gittenger. Xbench – A set of benchmark tests for X servers. (Included with the X Window System software distribution).

[6] J. Gosling, D. S. H. Rosenthal, and M. Arden. *The NeWS Book.* Springer–Verlag, New York, 1989.

[7] Ricardo Gusella. The analysis of diskless workstation traffic on an ethernet. Technical Report UCB/UCSD 87/379, Computer Science Division (EECS), University of California, Berkeley, California, 1987.

[8] John H. Howard, Micheal L. Kazar, Sherri G. Menees, David A. Nichols, M. Satyanarayanan, Robert N. Sidebotham, and Michael J. West. Scale and performance in a distributed file system. *ACM Transactions on Computer Systems*, 6:51–81, February 1988.

[9] J. Postel. Internet Protocol. RFC 791, NIC, September 1981.

[10] J. Postel. Transmission Control Protocol. RFC 793, NIC, September 1981.

[11] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon. Design and implementation of the Sun Network File System. In *Proc. of the Summer 1985 Usenix Conf.*, 1985.

[12] Robert W. Scheifler. *The X Window System Protocol.* M.I.T. Laboratory for Computer Science.

[13] Robert W. Scheifler and James Gettys. The X window system. *ACM Trans. on Graphics*, 5(2):79–109, April 1986.

[14] Robert W. Scheifler, James Gettys, and Ron Newman. *X Window System: C Library and Protocol Reference*. Digital Press, 1988.

[15] Sun Microsystems, Inc., Mountain View, California. *SunOS Reference Manual, SunOS Release 4.0*, 1988.

[16] Sun Microsystems, Inc. NFS: Network File System protocol specification. RFC 1094, NIC, March 1989.

[17] S. Wecker. DNA: the Digital Network Architecture. *IEEE Transactions on Communications*, COM-28:510–526, April 1980.

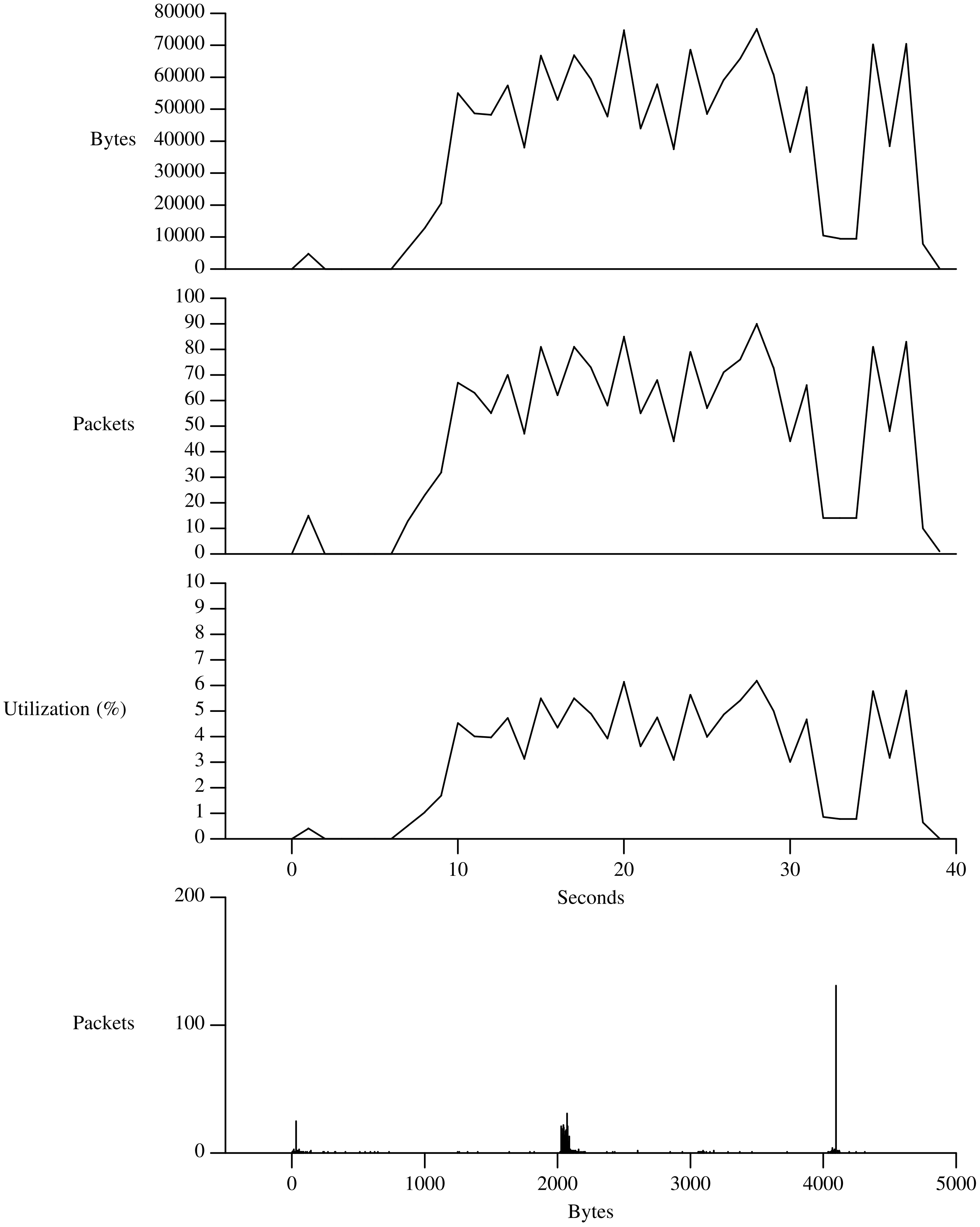

Figure 1: X protocol traffic generated by `xdvi` while paging through an 18 page LaTeX document containing text, mathematics and graphics.

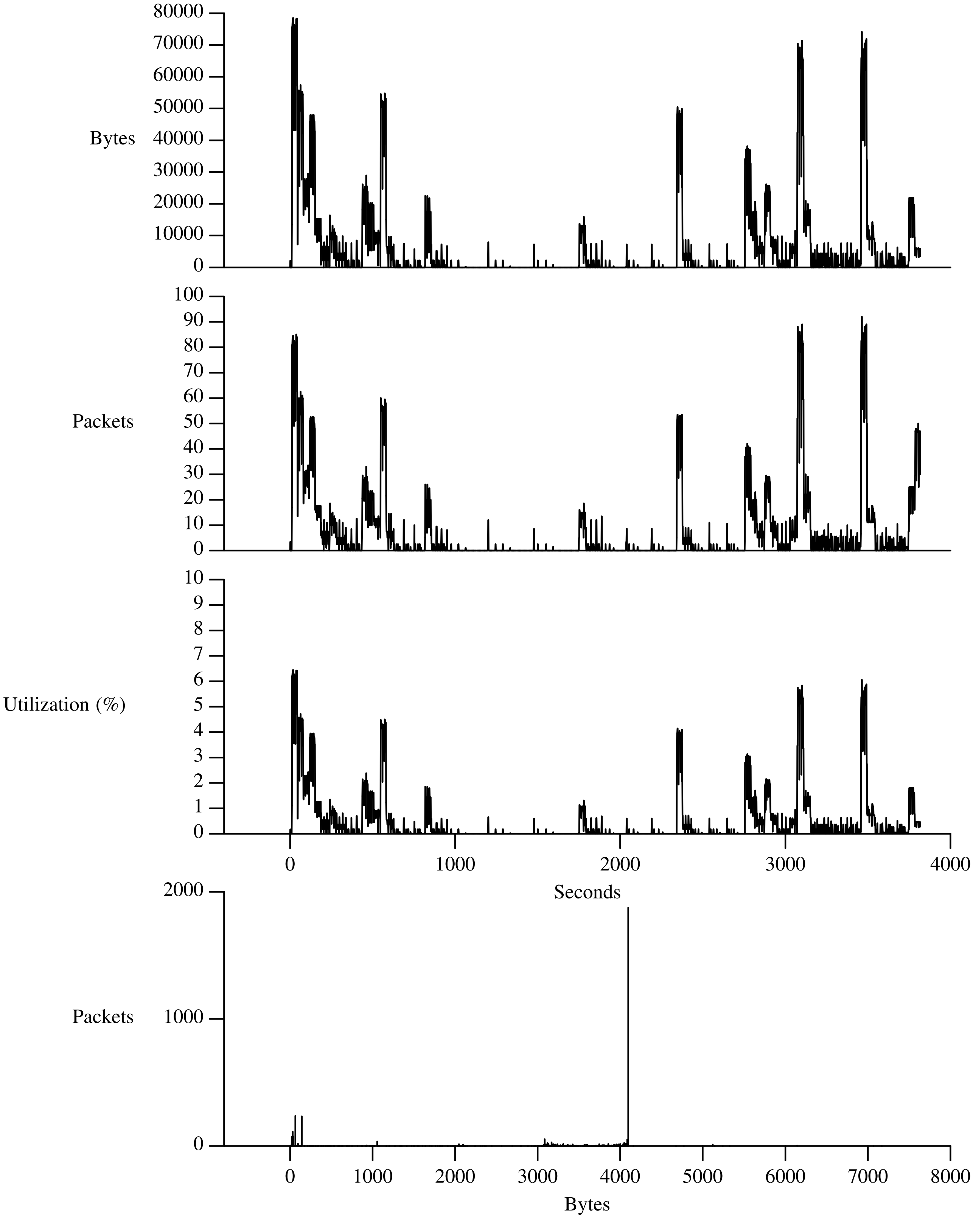

Figure 2: X protocol traffic generated by `xbench` while sending an exhaustive, benchmarking sequence of graphics requests.

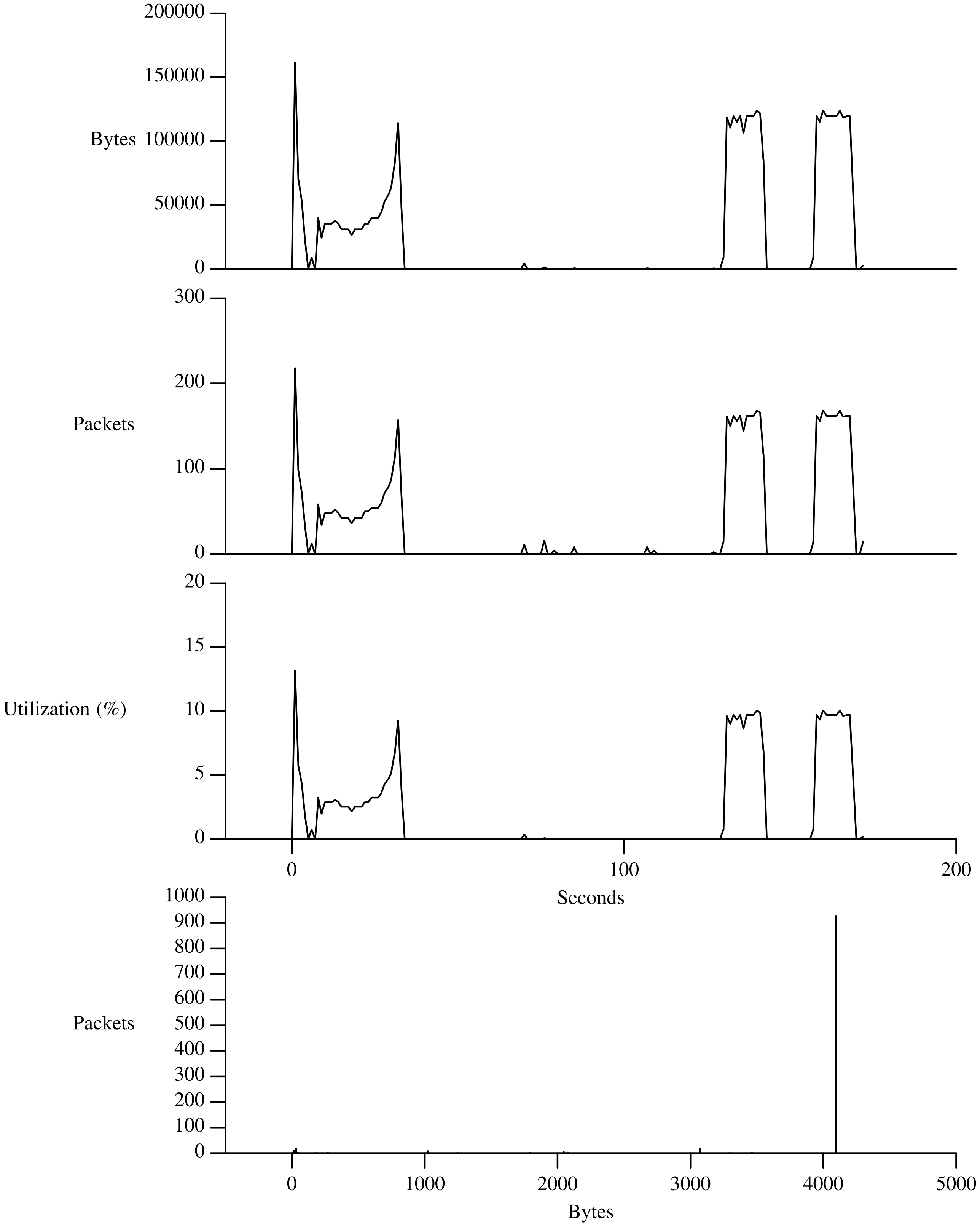

Figure 3: X protocol traffic generated by dumping an X root window to a file, undumping the 1Mb file via `xwud`, iconifying and deiconifying the `xwud` window.

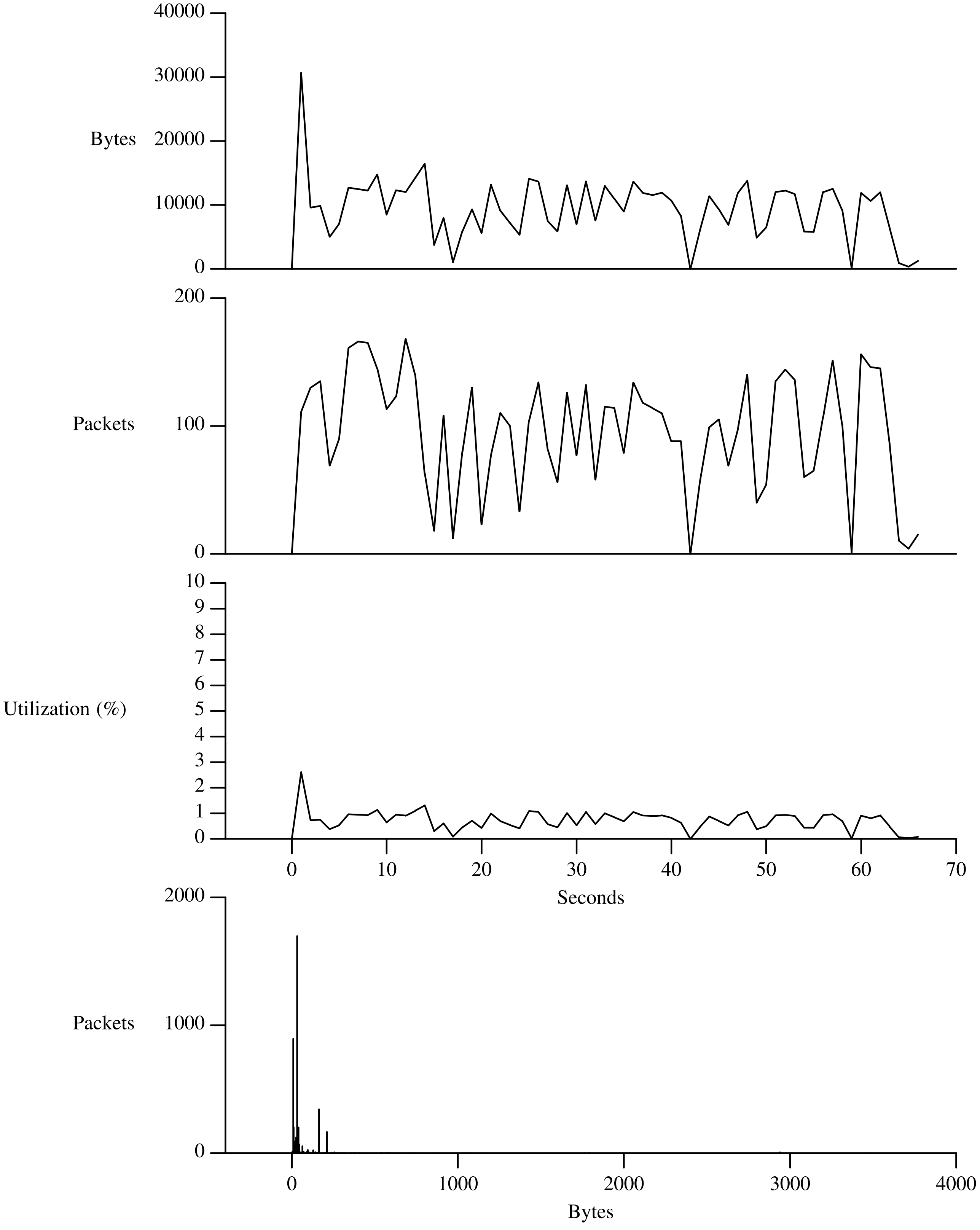

Figure 4: X protocol traffic generated by `twm` while selecting, positioning and resizing an `xterm` window.

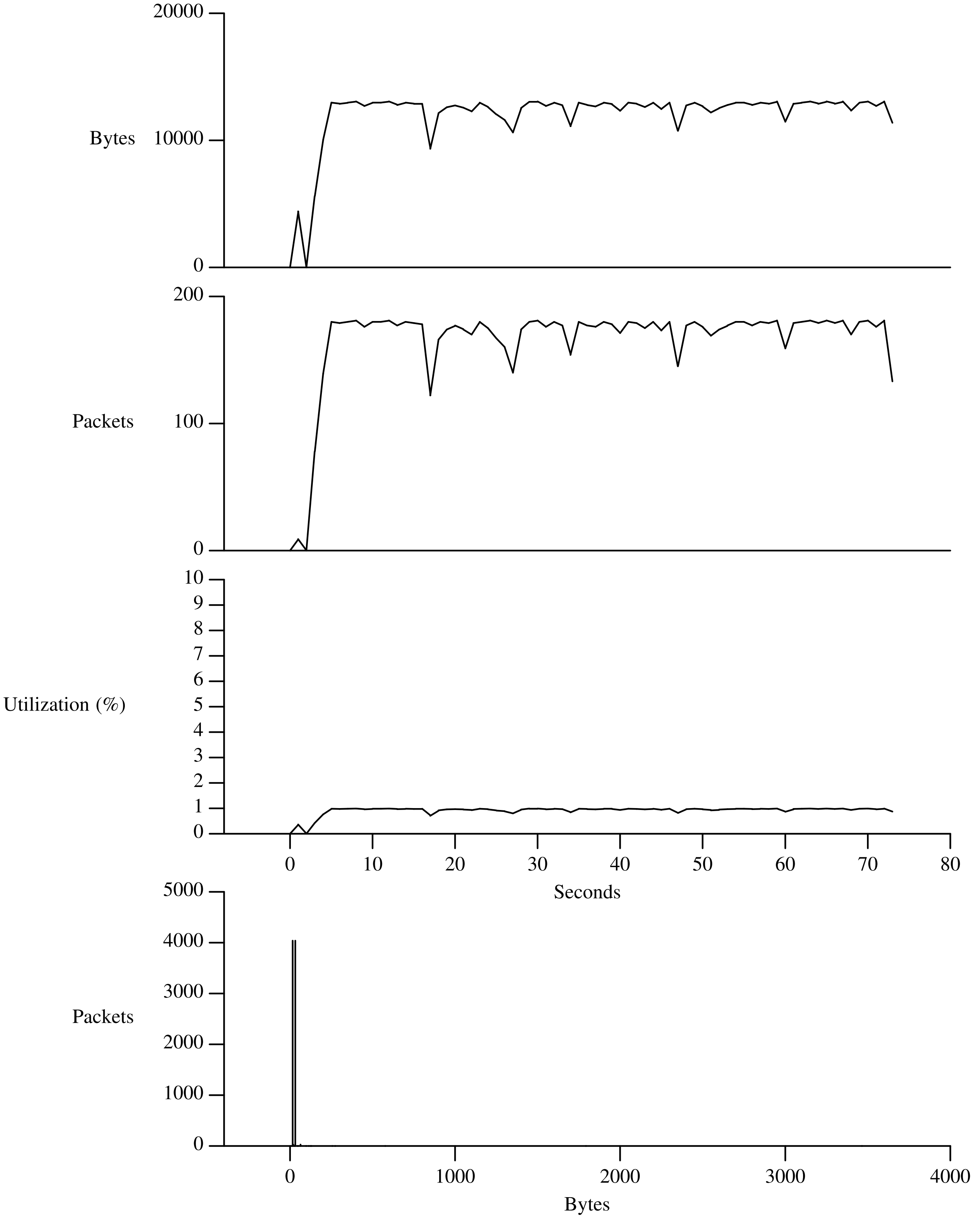

Figure 5: X protocol traffic generated by `motion` while sending and receiving X protocol requests and events as quickly as possible via cursor tracking.

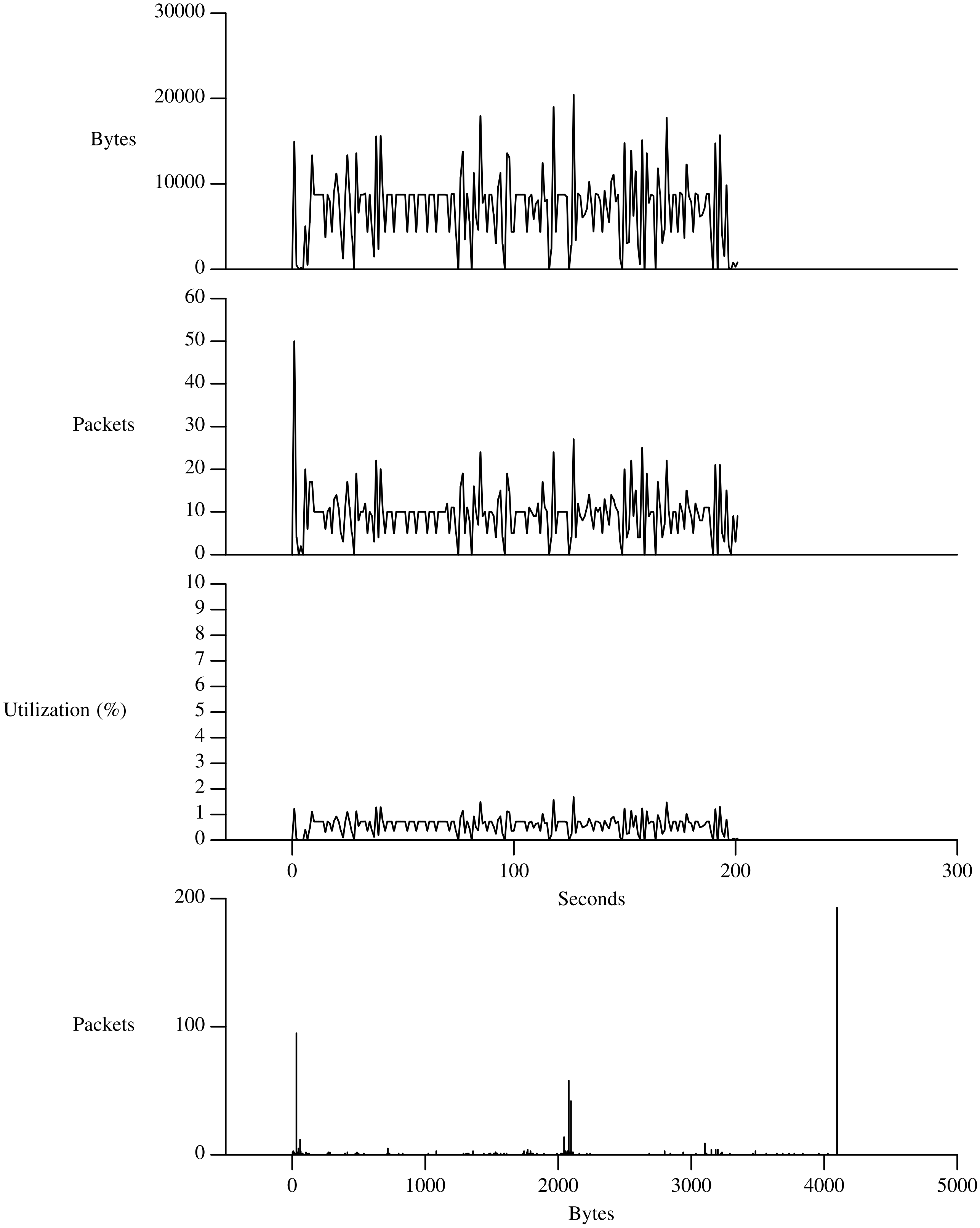

Figure 6: X protocol traffic generated by `xterm` while displaying 1Mb of text.

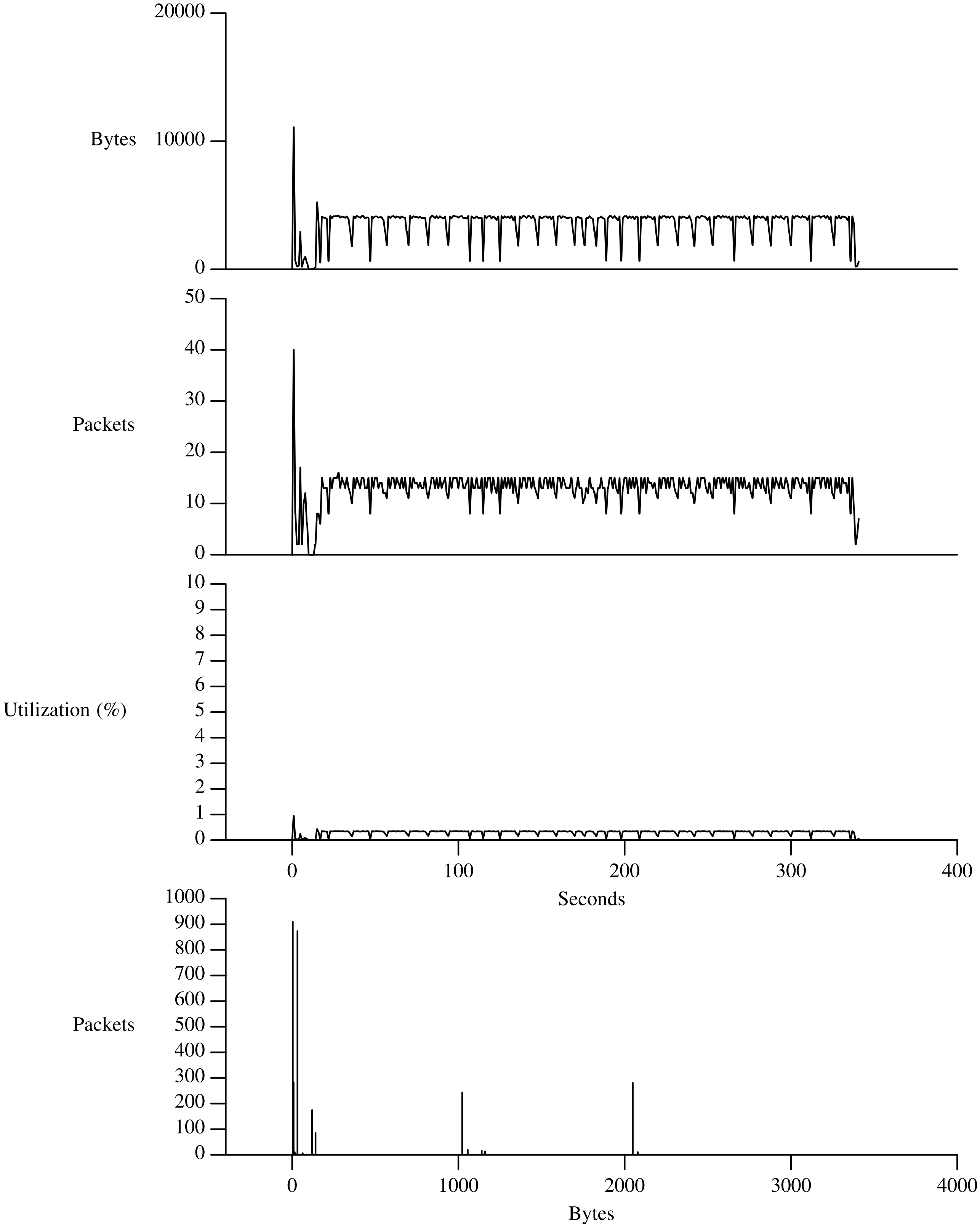

Figure 7: X protocol traffic generated by X11R4 GNU `emacs` paging through a 1Mb file.

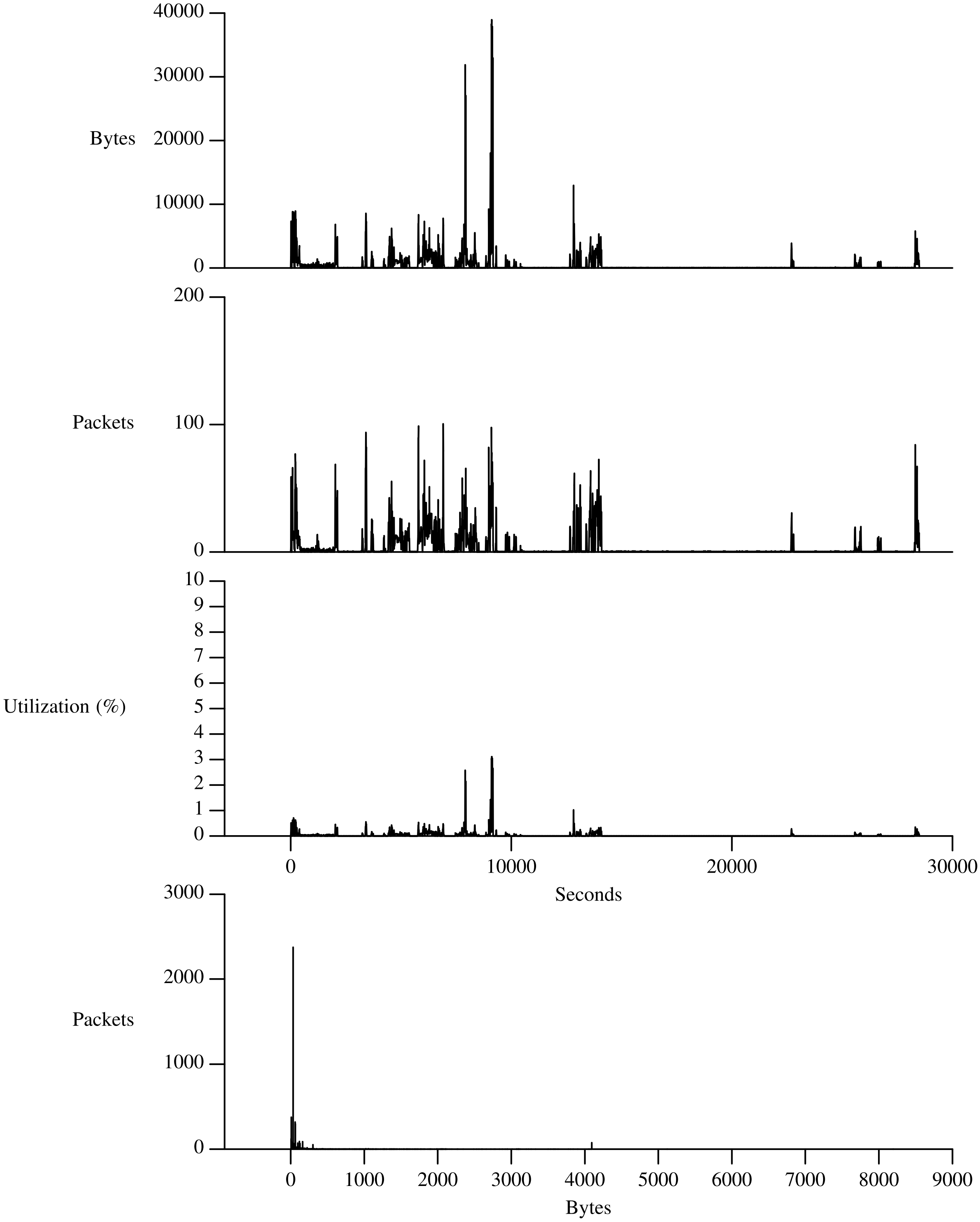

Figure 8: X protocol traffic generated by logging all X server activity for 8 hours.

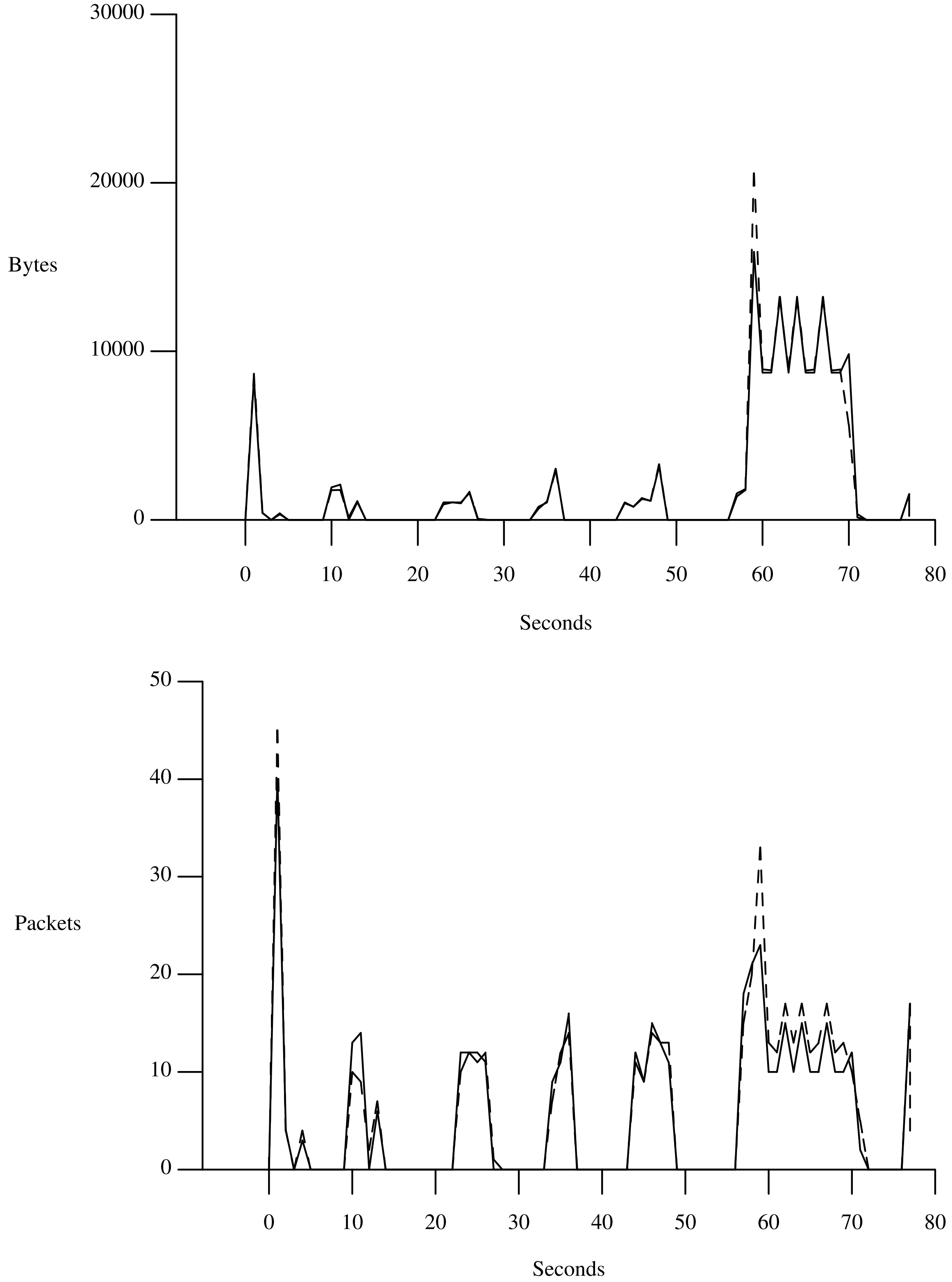

Figure 9: Comparison of the numbers of bytes and packets logged by `xlog` (solid lines) and `etherfind` (dashed lines) while displaying 1 Mbyte of text.

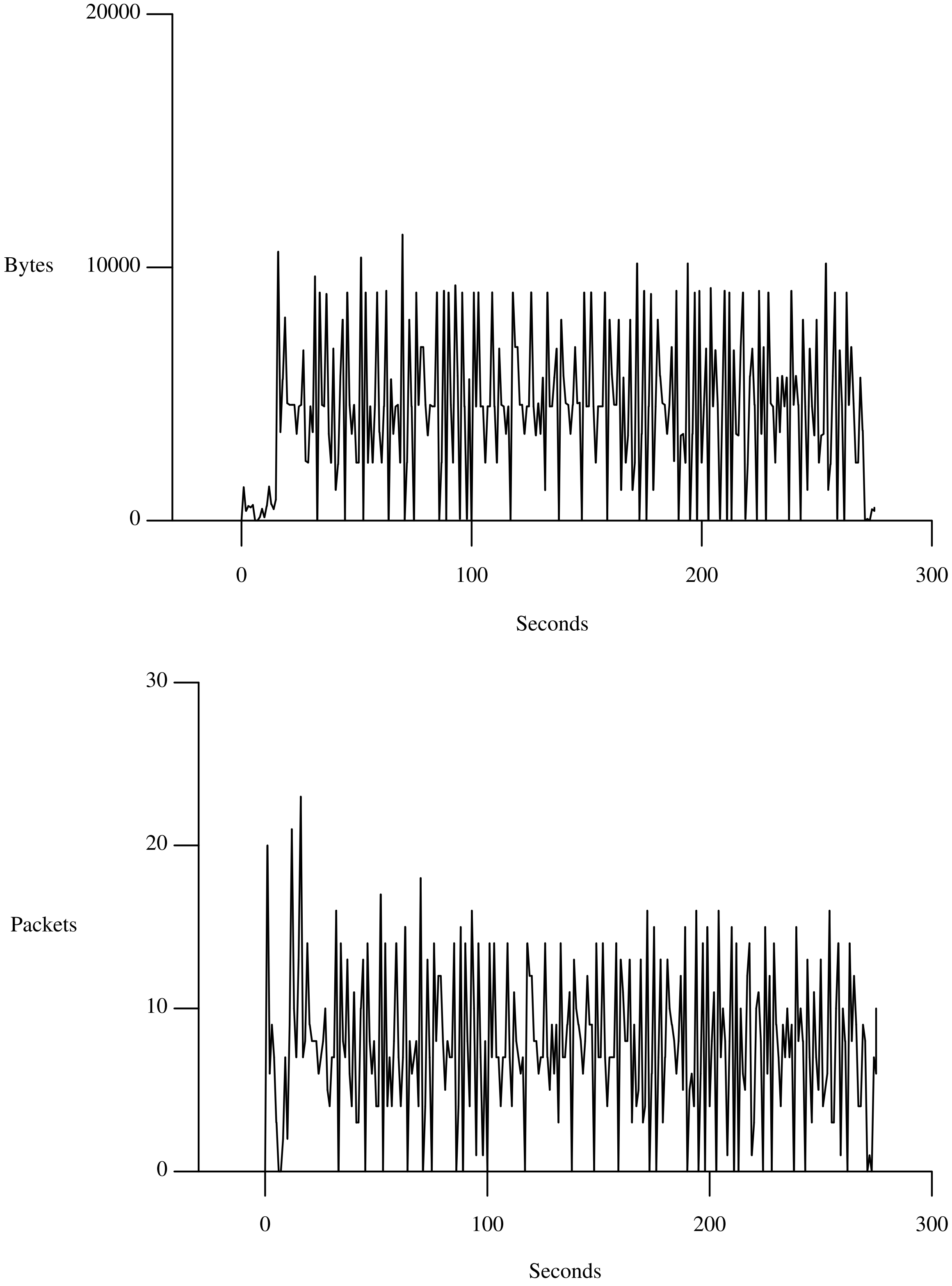

Figure 10: Telnet traffic generated by displaying 1Mb of text in an `xterm` window.

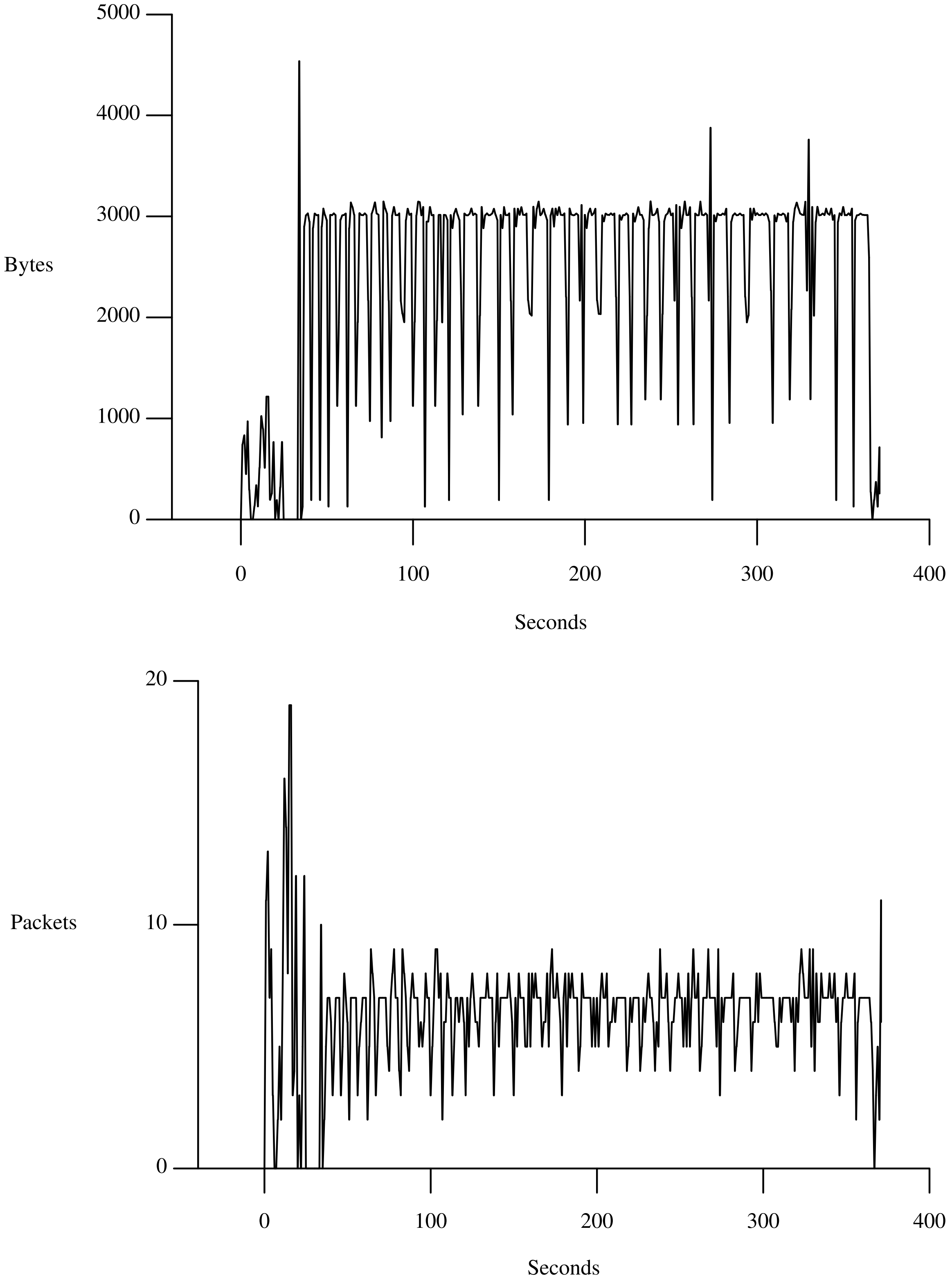

Figure 11: Telnet traffic generated by paging through a 1Mb file with GNU `emacs`.