Network Traffic Measurement of the X Window System

Nipun Basil
Carey L. Williamson
Department of Computational Science
University of Saskatchewan
Saskatoon, Saskatchewan
CANADA S7N 0W0

Abstract

The X Window System is a network-oriented windowing system that runs on a wide range of computing and graphics machines. X servers and clients communicate over a network so that applications running on remote hosts can be displayed locally through graphical interfaces.

This paper presents network traffic measurements of the X Window System. A network traffic monitoring tool, `tcpdump`, is used to collect traces of the packets sent between a server and a client for various X applications. For one X application, X protocol messages are also recorded by another software monitor called `xmon`. The measurements are taken on a 10 Mb/s Ethernet local area network.

Experimental results show that the average packet transfer and data transfer rates between clients and servers are not very high for most of the X applications. Packets are produced in both directions, though the bulk of the bytes are sent from clients to servers. The majority of the applications do not have a high average network utilization. Some applications, however, produce bursty traffic that causes the network utilization by clients and servers to increase significantly. The bandwidth requirements are particularly high for applications that display documents, images, or video.

1 Introduction

The X Window System [8] is a network-transparent window system that provides high-performance, high-level, device-independent graphics. The X interface has become very popular in Local Area Network (LAN) environments. LANs generally consist of diskless personal workstations connected to a shared file server over an Ethernet segment. The X Window System provides a window user interface and facilitates distributed computing in such an environment.

X is based on a client-server model [10]. An *X server* manages various devices including a bitmap display (screen), keyboard, and mouse. The application programs, or *X clients*, exchange high level request messages with the X server to achieve graphical and windowing functions. The essential tasks of the server are to multiplex requests from clients to the bitmap display, and demultiplex keyboard and mouse input back to the appropriate clients. Multiple clients can have connections open to a server simultaneously, and a client can have connections open to multiple servers simultaneously.

The servers and the clients communicate through a *network protocol* that is a simple block-stream protocol [8]. The communication protocol is followed even when the client program runs on the same workstation as the server. The protocol is layered on top of a reliable duplex byte stream. The stream is based on a local interprocess communication mechanism if the client and the server run on the same workstation. A network connection is established if the client and the server exist on different machines. The X Window System is designed to operate in an environment where the interprocess communication round-trip time is between 5 and 50 milliseconds (ms), both for local and for network communication. The X protocol can be built over many environments including TCP [7], DECnet [11], and Chaos [6].

A rich variety of message types are supported by the X protocol. There are four categories of messages: *requests, replies, events,* and *errors.* Request messages are sent by the client program to the server to request various windowing and graphics functions. Replies are sent from the server to the client programs in response to requests that ask for some information from the server. Events are sent from the server to the client programs as a consequence of real-time activities of the user, such as keyboard input (press or release of a key), mouse input (press or release of a mouse button), and mouse movement (enter or leave a window). Finally, errors are warning messages of various types that are sent from the server to the client.

This paper presents measurements of the network packet traffic generated on an Ethernet local area network by a variety of X applications. The packet transfer and data transfer rates between a client and a server are measured and an analysis is performed to study the volume of traffic in each direction. The network utilization is computed from the data to provide a measure of the load generated on the Ethernet by each application. Measurements are also conducted to record requests and events generated by one of the applications. The experiments performed in this paper are important for studying the network requirements of various X applications. The measurements are also helpful in predicting the number of X sessions that could be simultaneously supported on an Ethernet local area network.

The remainder of this paper is organized as follows. Section 2 explains the measurement methodology and describes the measurement tools, the experimental design, the measured X applications, and the performance metrics. The measurement results and analysis are presented in Section 3. Section 4 contains the conclusion.

2 Measurement Methodology

2.1 Measurement Tools

The network traffic measurements for this paper were recorded using `tcpdump` [4]. `tcpdump` is a software tool that monitors the traffic on a LAN and prints out the headers of all the packets observed on the network. The traffic observed by `tcpdump` includes ARP/RARP packets, TCP packets, UDP packets, NFS requests, Fragmented IP datagrams, and Appletalk packets.

`tcpdump` is a passive monitor, i.e., it is entirely transparent to the network hosts and does not affect traffic flow over the network. All operations in `tcpdump` are performed at the kernel level and hence few packets are dropped by this tool.

`tcpdump` allows the collected traces to be stored in a file for later analysis. Each entry recorded by `tcpdump` corresponds to one network packet and is tagged with a timestamp, a source host, a destination host, sequence numbers, packet size (data bytes transmitted), protocol used, and other protocol dependent information. The timestamp reflects the time that the kernel first saw the packet and is as accurate as the kernel's clock. `tcpdump` permits selective collection of packets. It is thus possible to filter the packets and log only those packets between a particular set of hosts or belonging to specific protocols.

Another software monitor that was used in the experiments is `xmon` [5]. `xmon` interactively monitors the X protocol between an X server and a number of X clients. It recognizes all X requests, events, errors, and replies sent between the clients and the server that are part of the X protocol. A facility in `xmon` provides statistics on the number of requests, events, and errors generated between the clients and the server.

`xmon` is made up of two separate processes. The core of `xmon` is `xmond` which monitors the X protocol streams between the clients and the server and displays the protocol messages on the standard output. The interactive interface is managed by `xmonui`. These two processes communicate via a pipe from the standard output of `xmonui` to the standard input of `xmond`. `xmond` sits transparently between the clients and the server. To the clients `xmond` behaves just like an X server and to the server it behaves just like a number of X clients.

2.2 Experimental Design

This section explains the experimental design for measurements of the X network protocol. The experimental setup is shown in Figure 1.

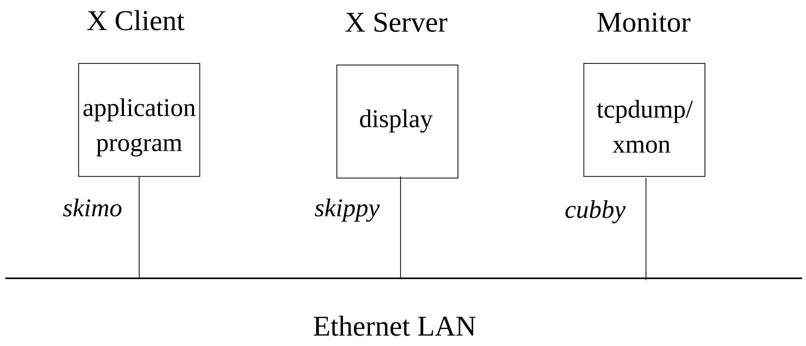

Figure 1: Experimental Setup

The network traffic measurements were conducted on a 10 Mbits/second (Mb/s) Ethernet LAN. The X server for the experiments is X11R5 (X Version 11, Release 5) running on a local UNIX workstation *skippy* (a SPARCstation IPC with 8 Mbyte RAM). The X clients are several application programs running on a remote UNIX workstation *skimo* (a SPARCstation IPX with 16 Mbyte RAM). The protocol exchanges between the X client and the X server were recorded from a third UNIX workstation *cubby* (a SPARCstation 1+ with 8 Mbyte RAM) connected to the same Ethernet network. During the measurements, only the X application under study exchanged messages between *skippy* and *skimo*.

`tcpdump` was run on *cubby* to collect traces of the packets exchanged between *skippy* and *skimo*. The following options were used with `tcpdump` to selectively collect the desired packet traces:

```
tcpdump -w logfile tcp and host skippy and host skimo
```

The `-w` option writes the raw packets to a log file rather than parsing and printing them out. The packets were later printed from the log file using the `-r` option (for reading packets) and the `-q` option (for concise protocol information). The X clients for this experiment used the TCP transport protocol. Thus the `tcp` option was specified with `tcpdump`. To prevent extraneous packets from other hosts in the collected traces, the hosts *skippy* and *skimo* were explicitly specified in the command line.

`xmon` was used to gather statistics on the number of requests and events received for an X application. The following command was used to run `xmon` on *cubby*:

```
xmonui -display cubby:0 | xmond -port 1 -server skippy:0
```

The interactive interface `xmonui` was displayed on *cubby*. The `-port` option specifies the port (1 in this case) on which TCP socket connections are accepted by `xmond` from the X clients (running on *skimo*). A TCP socket connection is in turn established by `xmond` to the X server (*skippy* in this case) indicated in the `-server` option. The X application was run on *skimo* and `cubby:1` was specified as the display for the application. This established a TCP socket connection between the application and `xmond`. `xmond` further displayed the application on *skippy*. All the messages between *skimo* and *skippy* were exchanged via *cubby* and were recorded by `xmond`.

2.3 Measured X Applications

The network traffic measurements for the experiment were collected using `tcpdump` for eight X applications: `twm, xfig, xterm, xedit, gnuplot, xdvi, xv,` and `videopix`. This subset of X applications represents typical X activity on a local area network. Furthermore, these applications have diverse network usage characteristics and hence form a suitable sample set for studying the performance of the X Window System on an Ethernet. The trace collection procedure for each application was replicated three times. The trace duration was 100 seconds for each application.

`twm` (Tab Window Manager) is a window manager for the X Window System. It provides titlebars, shaped windows, several forms of icon management, user-defined macro functions, click-to-type and pointer-driven keyboard focus, and user-specified key and pointer button bindings. The measurements on `twm` were performed on a full-zoomed window of *skimo* filled with text. Iconification and deiconification operations were successively performed on the window.

`xfig` (Facility for Interactive Generation of figures) is a menu-driven tool that allows the user to draw and manipulate objects interactively in an X window. The measurements on `xfig` were recorded by loading a 5 Kbyte Fig file and performing different operations on the graphical objects. The operations included scaling, movement, deletion, and rotation of objects. Various primitive objects were also drawn during the measurements.

`xterm` is a terminal emulator for the X Window System. The traces for `xterm` were collected by printing a 1.2 Mbyte text file on the `xterm` window using VT102 emulation.

`xedit` is a simple text editor for X11. For recording the measurements, operations were performed on a 1.2 Mbyte text file. The operations included vertical scrolling through the file and displaying successive pages of text.

`gnuplot` is an interactive function plotting program that uses the X Window System for displaying plots. Traces for `gnuplot` were collected by plotting a succession of different functions on *skimo* and displaying the functions on *skippy*.

`xdvi` is used to preview DVI files (such as those produced by TeX) under the X Window System. A 0.1 Mbyte DVI file that contained text, figures, and graphs was viewed for the measurements. Certain figures in the file were zoomed (using the "magnifying glass" subwindow).

`xv` is an X application that interactively displays images in various formats. The measurements on `xv` were recorded by displaying a 0.3 Mbyte GIF (Graphics Interchange Format) image on *skippy*. The operations performed on the image were resizing, rotation, and reloading.

`videopix` is a program that displays NTSC-standard video under X11. `videopix` was run on *skimo* and the video images were displayed on *skippy*. For the experiment, coloured images of full size (640x480 pixels, with 8 bit colour information per pixel) were displayed.

Apart from the above traces, `xmon` was also used in a separate measurement to record X protocol messages for an `xfig` session. The purpose of this measurement was to examine the requests and events generated by the X protocol and to record the corresponding packets and data bytes transferred over the network. A 5 Kbyte Fig file (the same as the Fig file used for the `xfig` measurements) was loaded and displayed on *skippy* and various operations were performed on the graphical objects. The trace duration for this measurement was 55 seconds. Statistics on requests and events were collected by `xmon`. The messages were simultaneously recorded by `tcpdump` on *cubby*. The trace collected by `tcpdump` provided statistics on the packets and data bytes transferred between *skippy* and *skimo*.

2.4 Performance Metrics

The following performance metrics were derived from the data provided by the `tcpdump` packet traces:

1. *Bidirectional packet transfer rate*: This metric measures the total number of packets transferred between the X client and the X server in each one second interval of the trace. This metric provides a measure of the total bidirectional traffic between the client and the server.

2. *Unidirectional packet transfer rate*: This metric records the number of packets sent in each direction, i.e., from the X server to the X client and from the X client to the X server, per one second interval of the trace. This metric provides an indication of the volume of traffic in each direction between the client and the server.

3. *Bidirectional data transfer rate*: The total number of data bytes (in the TCP packets) transferred between the X client and the X server (bidirectional traffic) per one second interval are recorded by this metric.

4. *Unidirectional data transfer rate*: This metric records the number of data bytes sent from the X server to the X client (unidirectional traffic) per one second interval, and vice versa.

5. *Percent network utilization*: The percentage network utilization for every one second interval of the trace is recorded by this measure.

   The network utilization is calculated as the ratio of the total number of bits transmitted per second to the maximum number of bits (10 Mbits for Ethernet) that could be sent over the network per second. Each log entry in `tcpdump` records the number of data bytes sent at the transport level. The total bytes sent over the Ethernet were then computed by adding the overhead for the TCP/IP protocols and the Ethernet frame overhead. The header overheads of the TCP and IP protocols are 20 bytes for each protocol (excluding the variable length optional part of the headers). Ethernet frames have 26 bytes of header and trailer overhead. The maximum length of the data field for an Ethernet frame is 1500 bytes. This implies that a maximum of 1460 data bytes (1500-20-20=1460 bytes) at the transport level can be accommodated in a single Ethernet frame. Furthermore, if the data portion of an Ethernet frame is less than the minimum 46 bytes, a pad field is used to fill out the remaining bytes. An interframe gap of 9.6 microseconds is also required over the Ethernet [9]. On a 10 Mb/s Ethernet, this corresponds to 12 bytes. Hence for computing the network utilization, 12 extra bytes were added to each frame transmitted over the network.

6. *Packet size distribution*: The distribution of the TCP packet sizes (data bytes) for all the packets collected in a trace is displayed by this metric.

7. *Packet interarrival times*: This performance metric measures the interarrival times of packets over the network. The interarrival times were computed by subtracting the timestamps of successive packets from each other. Since the timestamp reported by `tcpdump` for each packet represents the arrival time of the last bit of the packet, the interarrival

time measured the time between the arrivals of the last bit of successive packets. The frequency distribution of the interarrival times is reported by this metric.

3 Measurement Results and Analysis

Figures 2 through 9 show the results of the network traffic measurements for the X applications. All data are plotted at intervals of one second.

Figure 2 presents the measurements for `twm`. As stated earlier, the measurements on `twm` were recorded by iconifying/deiconifying a full-zoomed window filled with text. The average bidirectional transfer rates measured for `twm` are 12.8 packets/second and 5.4 Kbytes/second. The peak transfer rates are 17 packets/second and 6.2 Kbytes/second. The average network utilization for `twm` is 0.5%, with a peak of 0.6%. Thus `twm` causes very little packet traffic on the network.

The number of packets exchanged between the client and the server are almost identical for `twm` (55.3% from the client to the server and 44.7% in the other direction). However, the data transfer is primarily unidirectional, from the client to the server (96.9% of the transferred data bytes are directed from the client to the server).

The packet size distribution for `twm` is trimodal. 20.7% of the packets are of size 1460 bytes (maximally sized), 65.6% are acknowledgement packets (containing zero data bytes) and other small sized packets, and the rest (13.7%) are fragmented packets (typically 550-950 bytes).

Figure 3 shows the measurements for `xfig`. The large initial spike in the data transfer plot is caused by the loading of the Fig file. Subsequently, when operations are performed on the graphical objects, it is observed that the data transfer rate drops and maintains a low value. `xfig` requires an average transfer rate of 88.3 packets/second and 6.2 Kbytes/second. The average network utilization for `xfig` is 1.1%, with the peak utilization recorded as 6.3% of the network bandwidth (when `xfig` is first invoked). Thus the packet transfer rate for `xfig` is higher than for `twm`, although the data transfer rate and the network utilization are not too different. The packet and the data traffic between the client and the server are almost the same in each direction (packets transferred to the server form 42.3% of the total packets and bytes transferred to the server form 56.2% of the total data bytes). This is in contrast to `twm` where most of the data traffic was directed from the client to the server. The packet size distribution for `xfig` is unimodal, containing mostly small packets.

Figure 4 shows the network traffic generated by `xterm`. The measurements for `xterm` were collected by displaying a text file. A large spike is observed in the initial seconds when the display is initiated. `xterm` requires a transfer rate averaging 19.8 packets/second and 12.4 Kbytes/second. The average utilization of `xterm` is 1.1% and the peak utilization is 3.3%. Load is directed from the client to the server (98.9% of the bytes are transferred to the server). The packet size distribution for `xterm` is bimodal. Acknowledgement packets and other small sized packets form 49.3% of the packets. Maximally sized packets (1460 bytes) and other large sized packets form 41.2% of the packets transferred. The rest of the packets represent a low percentage of the total packets (less than 10%).

The X Window traffic generated by `xedit` is shown in Figure 5. `xedit` produces on average 24.6 packets/second and 12.9 Kbytes/second. `xedit` has an average utilization of 1.2% with the peak utilization being 2.0% of the network bandwidth. It is thus observed that `xedit`, like the previous applications, does not use significant network resources. Equal number of packets are sent between the client and the server, although 97.5% of the bytes are transmitted by the client. The packet size distribution for `xedit` is trimodal.

The network traffic measurements for `gnuplot` are presented in Figure 6. A series of different functions were plotted for the measurements. The large spikes in the plots are associated with some of the functions that transfer large amounts of data. `gnuplot` requires a transfer rate averaging 42.9 packets/second and 28.8 Kbytes/second. The average utilization of `gnuplot` is 2.6%, with the peak utilization recorded as 6.0%. 66.1% of the total packets are transferred from the client to the server, whereas the respective bytes transferred are 99.8%. The packet sizes are trimodally distributed: 33.5% of the packets are acknowledgement packets (zero bytes), 29.2% of the packets have a size of 584 bytes, and 32.2% of the packets are maximally sized (1460 bytes). The rest of the packets are insignificant.

Figure 7 presents the measurements for the `xdvi` application. The average transfer rates for `xdvi` are 79.8 packets/second and 53.1 Kbytes/second.

The average network utilization for `xdvi` is 4.8%, with a peak of 9.9%. The packets transferred from the client to the server represent 76.8% of the packets. Also, 99.7% of the bytes are transmitted by the client. The packet size distribution for `xdvi` is trimodal. 36.5% of the packets are acknowledgement packets and other small sized packets, 33.4% are maximally sized (1460 bytes) and other large sized packets, and the rest (30.1%) are fragmented packets (typically 550-750 bytes).

The X Window traffic generated by `xv` is shown in Figure 8. The plots show that the traffic produced by `xv` is very bursty. Image operations (resizing, rotation, and loading) cause lots of network activity. `xv` produces on average 170.1 packets/second and 82.1 Kbytes/second. `xv` has an average utilization of 7.3%. The peak utilization is, however, very high (27%). Load is directed from the client to the server (98.6% of the bytes are sent by the client). The packet sizes are trimodally distributed.

The traffic measurements for `videopix` are shown in Figure 9. The plots show that `videopix` causes significant network activity [1]. `videopix` produces an average transfer rate of 306.7 packets/second and 221.6 Kbytes/second. The average utilization for `videopix` is 19.7%, with a peak of 27.4% (which is the highest of all applications). It is also observed from the figure that `videopix` consistently maintains a high utilization of the network (in contrast to the other applications). The high average utilization implies that only a few concurrent sessions of `videopix` (3 to 4) can chew up the entire bandwidth of a 10 Mb/s Ethernet. The bulk of the bytes (99.9%) are transferred from the client to the server. The packet size distribution for `videopix` shows that the packet sizes are primarily trimodal, with a few other small peaks.

Finally, Figure 10 displays the packet interarrival time characteristics for all the X applications studied. As expected, applications that generate high packet arrival rates and high network utilizations have very low packet interarrival times. For example, 95% of the `videopix` packets have interarrival times $\leq$ 3 ms. Other applications with low packet interarrival times are `xv` and `xdvi`. On the other hand, X applications like `twm`, `xfig`, `xterm`, `xedit`, and `gnuplot` have low packet arrival rates and low network utiliza-

[1] `videopix` transmits images consecutively, where a single image transmission time exceeds one second. However, all data has been plotted at intervals of one second. Due to this reason, the graphs for `videopix` display a lot of jagged behaviour.

tions. These applications generate relatively higher packet interarrival times. 90% of the `twm` packets, for example, have interarrival times $\leq$ 400 ms.

Table 1 and Table 2 summarize the measurement results. The results show that certain applications, like `xv` and `videopix`, can consume a significant portion of the network bandwidth. The average network utilization of these applications is quite high (especially `videopix`) and their peak utilizations are close to 30% of the total bandwidth.

In another measurement, `xmon` was used to monitor the messages for an `xfig` session. The results of the measurement are shown in Table 3. The table shows that the most frequent requests for `xfig` are `CopyArea` and `ClearArea`, and the most frequent events are `NoExposure` and `Expose`. It is observed that a large number of requests and events are generated by the X protocol. Also, the number of requests exceed the number of events for the recorded `xfig` session. Table 4 shows the transferred packets and data bytes measured by `tcpdump` for the `xfig` session. Load is almost uniform between the client and the server. The measurements also show that the packets transferred between the client and the server are fewer than the corresponding requests and events generated. Thus an effort is made by X11 to compact messages into packets so as to minimize system overhead and improve resource allocation [8].

4 Conclusion

Experimental results show that the average packet transfer and data transfer rates between clients and servers are not very high for most of the X applications. Packets are transmitted in both directions, though the bulk of the bytes are sent from clients to servers. Measurements also show that the average network utilization for the majority of the X applications is not high. However, it is observed that the traffic is bursty and that the network load associated with some bursts is significant. Some clients in the X Window System generate enough traffic to represent a significant load on the network. Applications like `xv` and `videopix` were observed to consume close to 30% of the total communication resources of an Ethernet.

The observed traffic characteristics must be considered when designing and configuring workstations and applications for local area networks. Too many workstations running X11 may overload the network and increase the contention for the shared channel. This might lead to performance degradation in terms of response times of the X applications.

References

[1] R. Caceres, P. B. Danzig, S. Jamin and D. J. Mitzel, "Characteristics of Wide-Area TCP/IP Conversations", *Proceedings of ACM SIGCOMM 91*, pp. 101-112, September 1991.

[2] D. E. Comer, *Internetworking with TCP/IP: Principles, Protocols, and Architecture*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1988.

[3] R. Gusella, "A Measurement Study of Diskless Workstation Traffic on an Ethernet", *IEEE Transactions on Communications*, Vol. 38, No. 9, pp. 1557-1568, September 1990.

[4] V. Jacobson, C. Leres and S. McCanne, `tcpdump`, SunOS 4.1 Man Page, Lawrence Berkeley Laboratory, University of California, Berkeley, CA.

[5] G. McFarlane, `xmon`, X11 Man Page, OTC, Australia.

[6] D. Moon, *Chaosnet*, AI Memo 628, Artificial Intelligence Laboratory, MIT, Cambridge, MA, June 1981.

[7] J. Postel, "Transmission Control Protocol", Internet Network Working Group, RFC 793, USC Information Sciences Institute, September 1981.

[8] R. W. Scheifler and J. Gettys, "The X Window System", *ACM Transactions on Graphics*, Vol. 5, No. 2, pp. 79-109, April 1986.

[9] W. Stallings, *Handbook of Computer-Communications Standards, Local Network Standards*, Macmillan Publishing Company, New York, NY, 1987.

[10] A. S. Tanenbaum, *Computer Networks*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1988.

[11] S. Wecker, "DNA: The Digital Network Architecture", *IEEE Transactions on Communications*, Vol. 28, No. 4, pp. 510-526, April 1980.

Table 1: Bidirectional traffic characteristics for X Window applications

X Window application	Packets per sec	Bytes per sec	Utilization per sec	Packet sizes	Interarrival times
twm	12.8	5.4 K	0.5%	Trimodal	90% $\leq$ 400 ms
xfig	88.3	6.2 K	1.1%	Unimodal	90% $\leq$ 33 ms
xterm	19.8	12.4 K	1.1%	Bimodal	90% $\leq$ 170 ms
xedit	24.6	12.9 K	1.2%	Trimodal	90% $\leq$ 130 ms
gnuplot	42.9	28.8 K	2.6%	Trimodal	90% $\leq$ 26 ms
xdvi	79.8	53.1 K	4.8%	Trimodal	90% $\leq$ 7 ms
xv	170.1	82.1 K	7.3%	Trimodal	95% $\leq$ 4 ms
videopix	306.7	221.6 K	19.7%	Trimodal	95% $\leq$ 3 ms

Table 2: Unidirectional packet and data transfer for X Window applications

X Window application	Packets transferred		Bytes transferred	
	ClientToServer	ServerToClient	ClientToServer	ServerToClient
twm	55.3%	44.7%	96.9%	3.1%
xfig	42.3%	57.7%	56.2%	43.8%
xterm	54.5%	45.5%	98.9%	1.1%
xedit	51.2%	48.8%	97.5%	2.5%
gnuplot	66.1%	33.9%	99.8%	0.2%
xdvi	76.8%	23.2%	99.7%	0.3%
xv	60.5%	39.5%	98.6%	1.4%
videopix	65.6%	34.4%	99.9%	0.1%

Table 3: Messages recorded by `xmon` for `xfig` session

Requests generated		Events generated	
Name	Count	Name	Count
CopyArea	1041	NoExposure	1041
ClearArea	996	Expose	607
ChangeWindowAttributes	602	MotionNotify	574
ImageText8	550	LeaveNotify	71
CreateGC	304	EnterNotify	70
PolyText8	287	ButtonPress	40
QueryPointer	249	ButtonRelease	40
CreatePixmap	244	MapNotify	6
PutImage	188	ConfigureNotify	4
FreeGC	185	FocusIn	3
MapWindow	168	FocusOut	2
CreateWindow	163	ReparentNotify	2
PolySegment	137		
PolyRectangle	120		
ChangeGC	107		
CopyGC	88		
PolyLine	72		
ConfigureWindow	57		
UnmapWindow	56		
PolyFillRectangle	44		
FreePixmap	42		
GetGeometry	42		
PolyArc	32		
ChangeProperty	28		
ListFonts	27		
SetClipRectangles	22		
MapSubwindows	21		
CreateGlyphCursor	19		
RecolorCursor	19		
AllocNamedColor	8		
OpenFont	7		
QueryFont	6		
InternAtom	5		
GetProperty	3		
QueryColors	1		
TranslateCoordinates	1		
TOTAL	5941	TOTAL	2460

Table 4: Packet and data transfer for `xfig` session

Packets transferred			Bytes transferred		
ClientToServer	ServerToClient	TOTAL	ClientToServer	ServerToClient	TOTAL
1007	1005	2012	197105	130553	327658

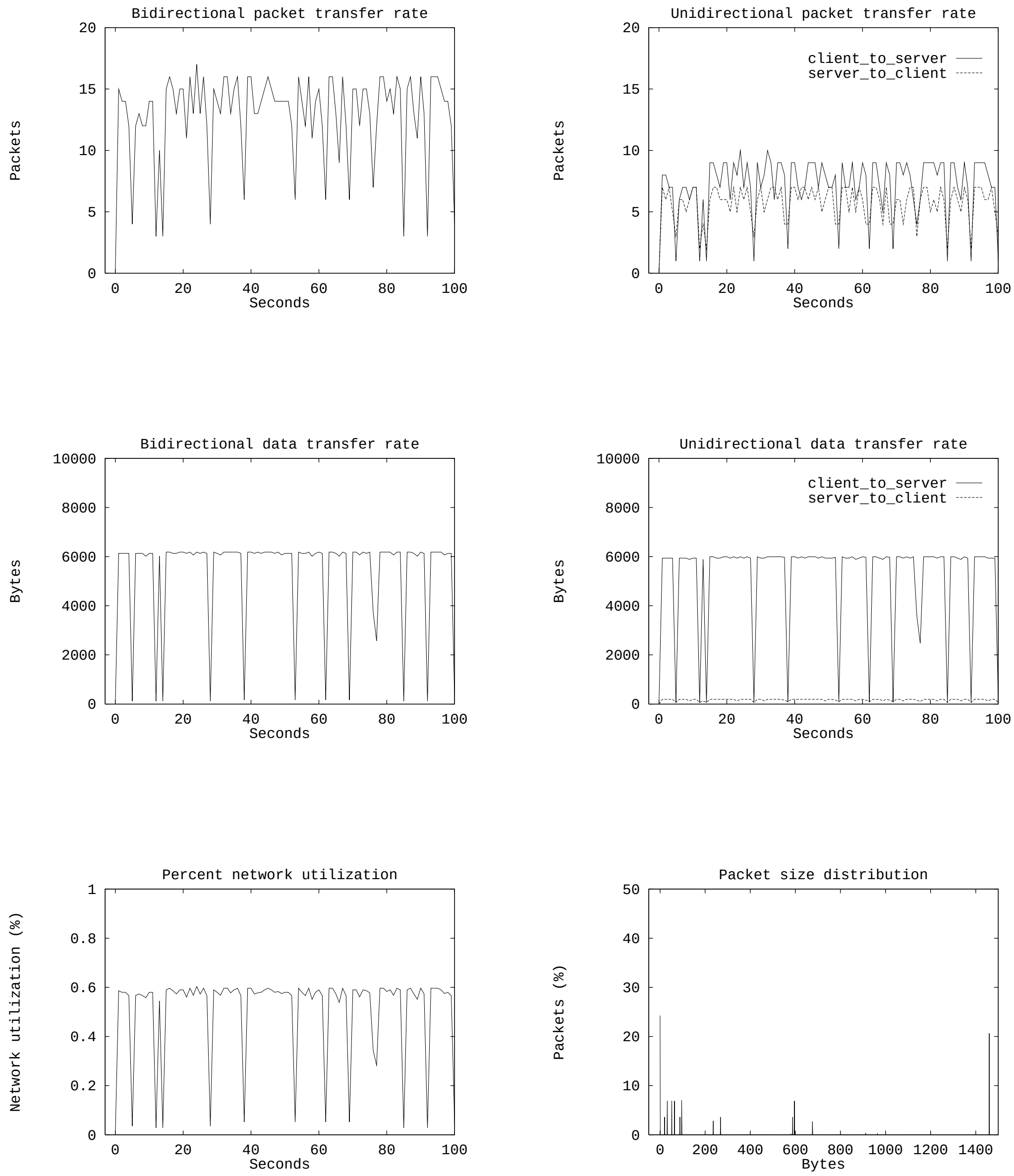

Figure 2: X Window traffic generated by `twm`

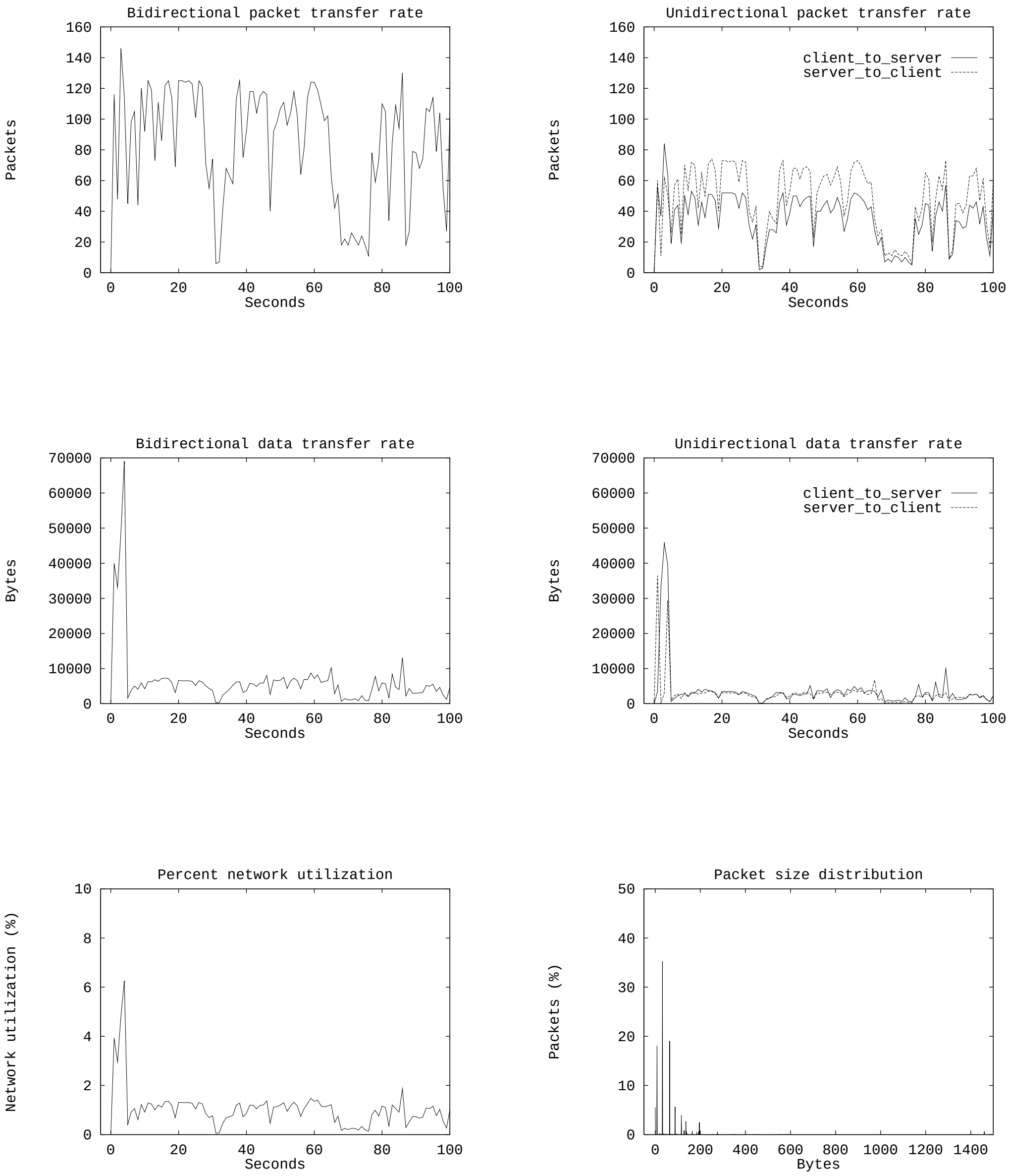

Figure 3: X Window traffic generated by `xfig`

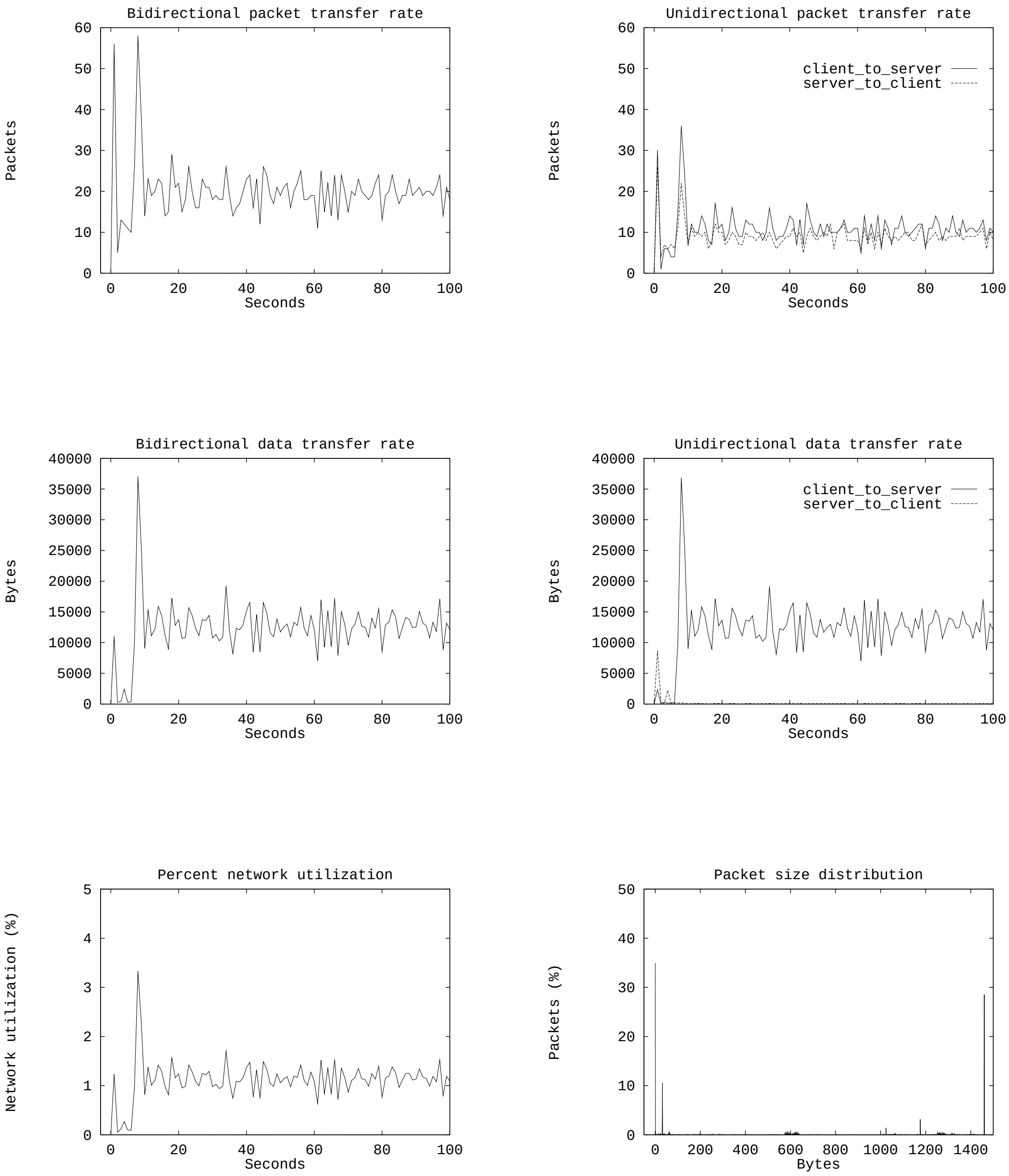

Figure 4: X Window traffic generated by `xterm`

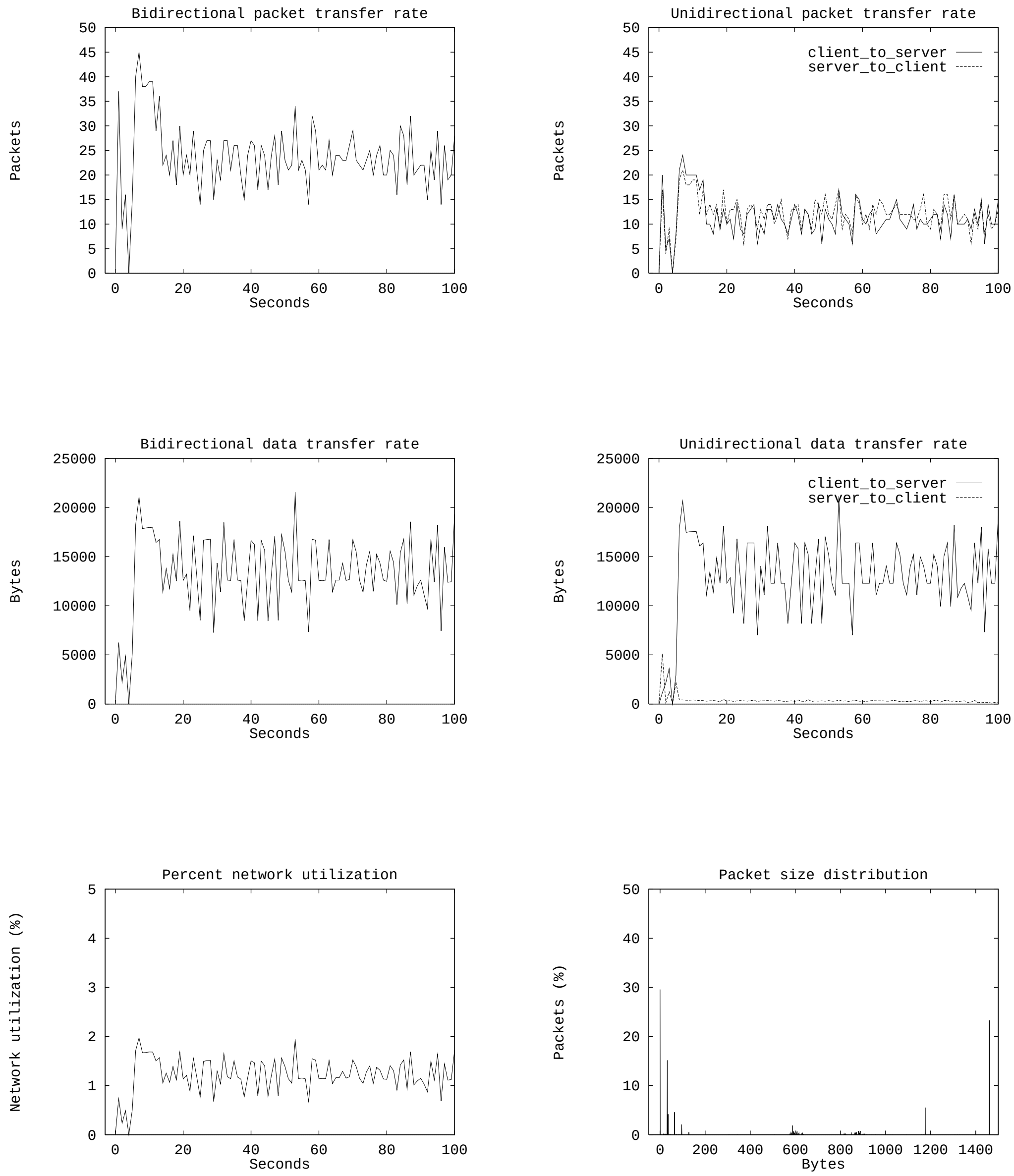

Figure 5: X Window traffic generated by `xedit`

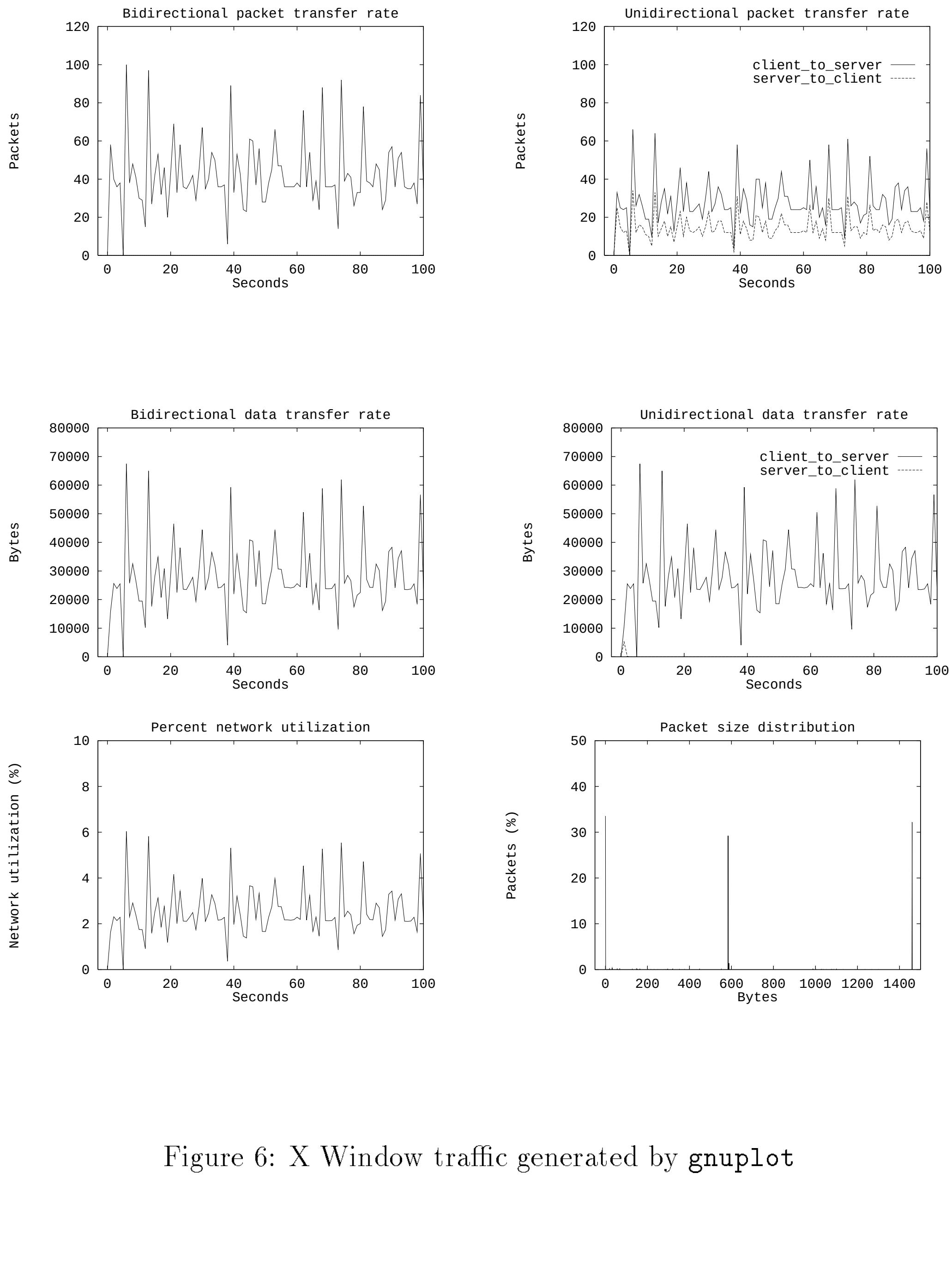

Figure 6: X Window traffic generated by `gnuplot`

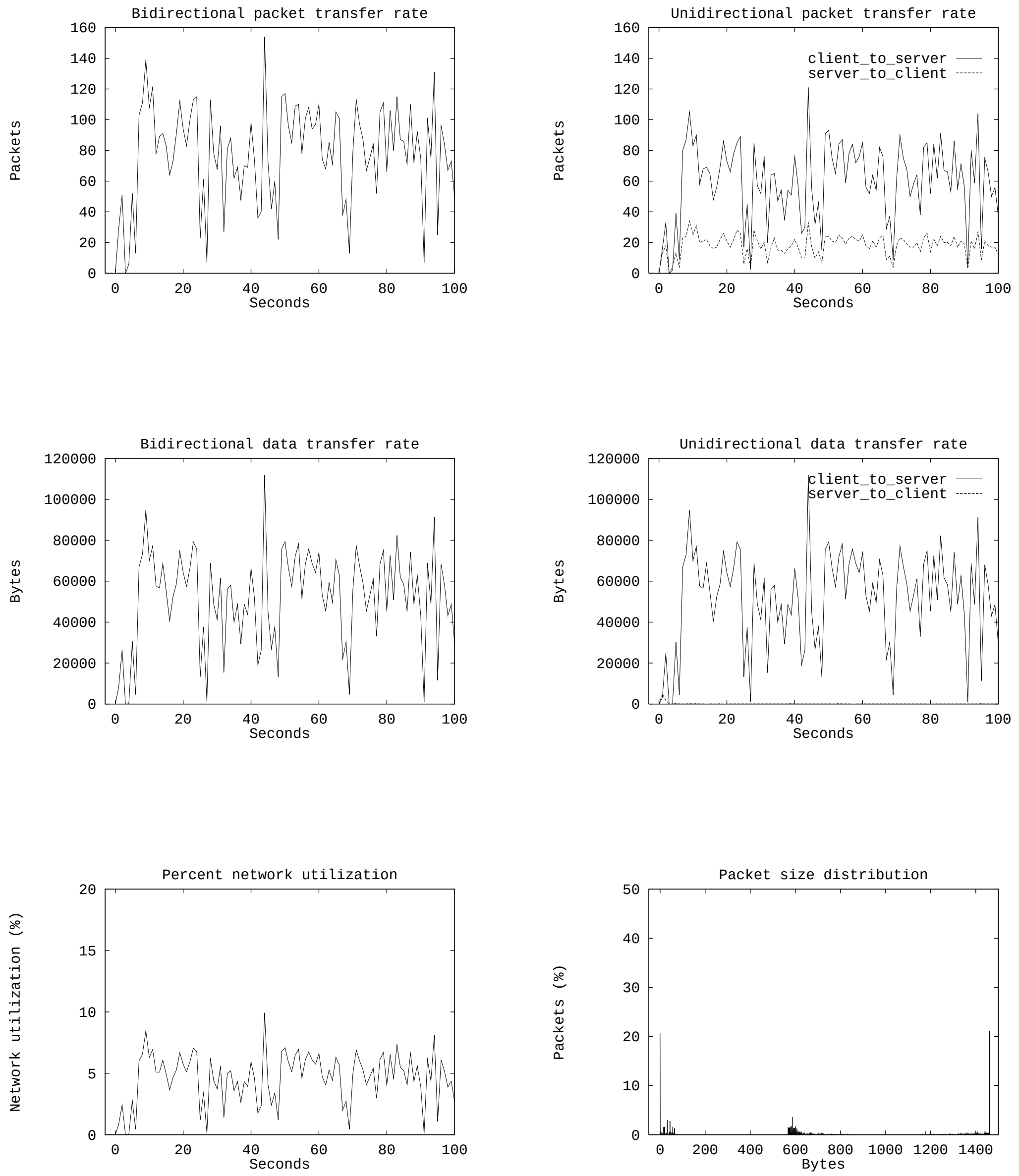

Figure 7: X Window traffic generated by `xdvi`

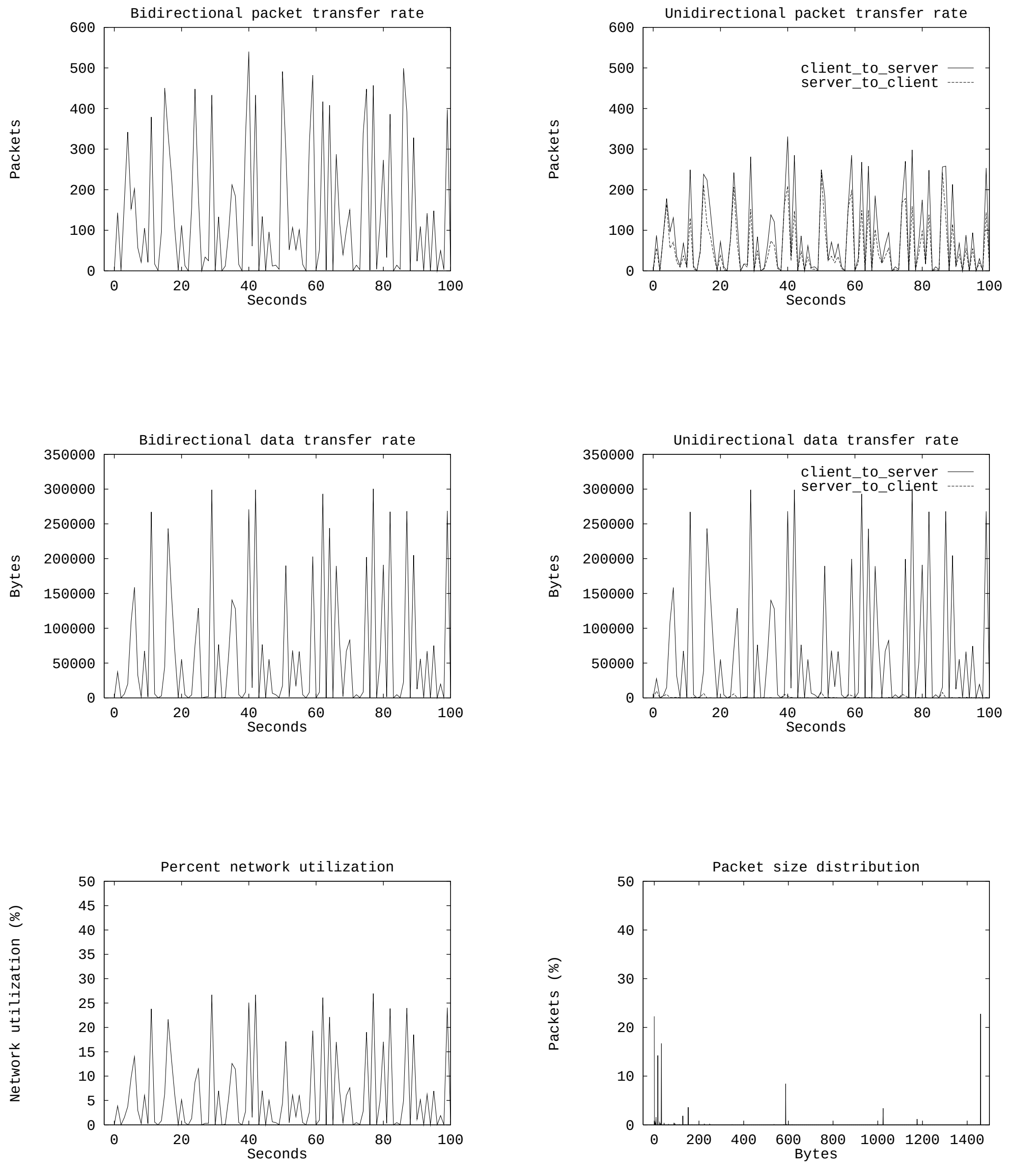

Figure 8: X Window traffic generated by `xv`

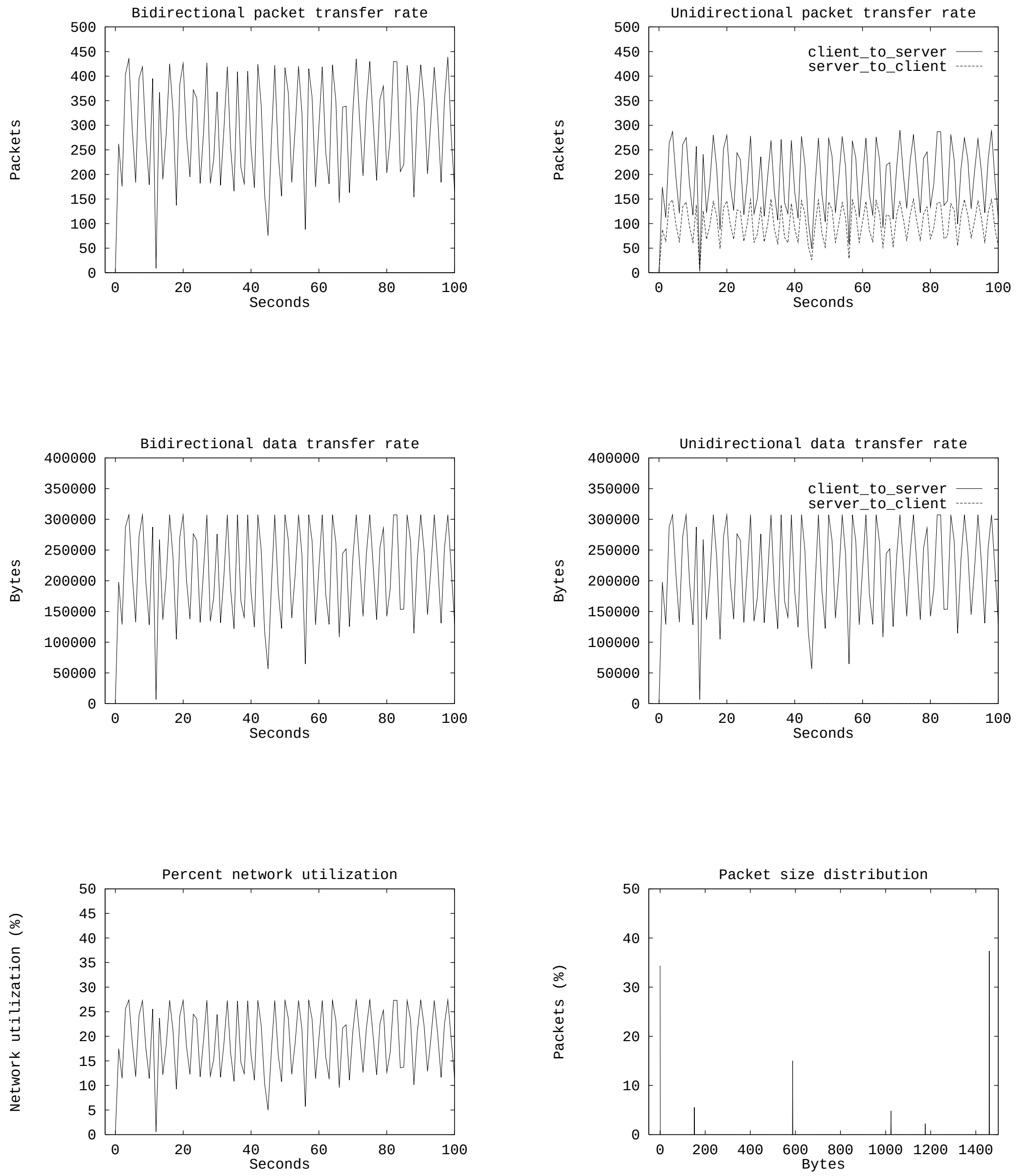

Figure 9: X Window traffic generated by `videopix`

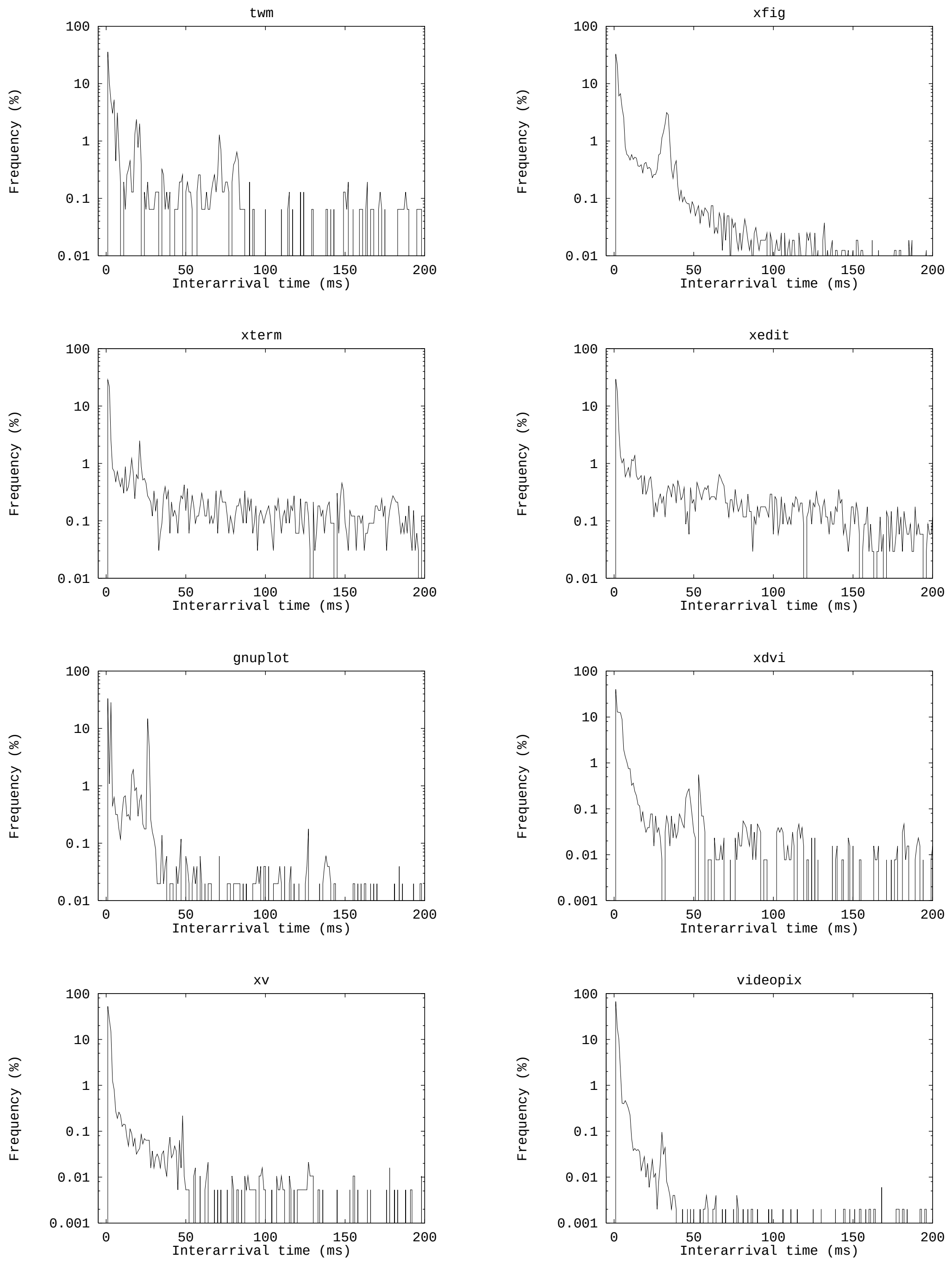

Figure 10: Packet interarrival times for various X applications