ORNL/TM-2000/xxx

FLOW CHARACTERIZATION FOR INTRUSION DETECTION

Tom Dunigan
Network Research Group

George Ostrouchov
Statistics and Data Sciences Group

Oak Ridge National Laboratory
P.O. Box 2008, Bldg. 6012
Oak Ridge, TN 37831-6367

{thd,ost}@ornl.gov

Date Published: November 27, 2000

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831-6285
managed by
UT-Battelle, LLC
for the
U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

Contents

Abstract

Network intruders often use non-standard ports or standard ports in non-standard ways to bypass detection. This report describes techniques and software for collecting, analyzing, and classifying Internet packet flows to assist in intrusion detection. Flows are characterized by packet size, interarrival times, direction, and inter-packet correlations without looking at packet contents. Statistical signatures for known flows are used to classify unknown flows.

1 Introduction

Most intrusion detection systems (IDS) are based on recognizing known attack signatures and/or anomalous activity. Network-based IDSs look for attack signatures on standard service ports (DNS, IMAP, POP, SNMP, SYSLOG) or monitor interactive activity on standard interactive service ports (TELNET, RLOGIN, RSH, FTP), or look for activity on ports used for known backdoors (NETbus, BackOrifice). In our experience over the last few years, these detection methods are quite effective. However, we have had several intrusions where the attacker has used non-standard ports and avoided detection.

An attacker may have gained access to an internal system by capturing an account password at another site. The attacker can then access an internal system and install backdoors on non-standard ports for later access. This is also an insider threat. Even the currently popular PC-based backdoors (Netbus and BackOrifice) are "port agile" – the attacker can choose the port to use for the backdoor. These backdoors can later be used to provide interactive access, chat channels, pass-throughs to other hosts, or file transfers.

The attacker also may use standard services in non-standard ways. Firewalls may pass HTTP, mail, DNS, or ICMP traffic, and IDS systems often ignore these services when they originate from the inside. The attacker can tunnel his own services through these standard ports, for example, transferring files in what looks like DNS packets, or providing interactive service through ICMP echo packets.

The objective of this research is to identify network flows by their statistical signature and hopefully identify intruder flows. This research is divided into three broad areas:

- network traffic capture, data reduction/visualization/storage
- statistical analysis of flow characteristics
- learning and decision systems for classifying flows

Network traffic data is captured and some portion of each packet is saved for later post-processing. Information retained for each packet includes time of arrival (to the microsecond), source address, source port, destination address, destination port, packet length, and TCP flags. This reduces a packet of hundreds of bytes to twenty or thirty bytes.

Using the collected network data, statistical signatures of known flows are developed using various features of the flow. Features include packet lengths inbound versus outbound, inter-packet arrival times, session duration, number of packets inbound and outbound, or other temporal characteristics. Flow categories include interactive sessions, file transfer, chat sessions, audio/video flows, web flows, DNS flows, remote console flows (PCAnywhere, Timbuktu), email, ICMP, IP tunnels.

Another part of this research developed a decision system to determine the effectiveness of various flows features in classifying a flow. A flow-classifier was developed to classify an unknown flow using the statistical signatures of known flows.

In the following section, other work similar to ours is discussed. Section 3 describes how we capture network flows. Section 4 describes various ways to characterize network flows, and section 5 describes techniques for classifying unknown flows.

2 Related work

Most of the research on Internet traffic consider aggregate behavior (link utilization, packet loss, packet/data volume, network service/port usage, time-of-day, packet size, interarrival times, flow direction) [1] [17] [24]. Since web/http traffic accounts for most of the Internet traffic, much work has been done on analyzing web traffic [14] [11]. Mena [13] analyzes Internet audio flows and derives metrics for identifying audio flows. He finds that audio flows have distinct packet length distributions and temporal signatures. He shows that packet interdeparture times have a distinct regularity across all audio flows. Cleveland [3] looks at the statistical properties of Internet traffic and the difficulties of handling the complex and very large data bases that result from collecting packet headers. Cleveland utilized the S statistical package for analyzing the data.

Frank [7] used data collected by NSM [8] from the Internet to classify network flows. The features of each flow included:

- flow duration
- packets from source
- packets from destination
- bytes from source
- bytes from destination

- intrusion warning

The intrusion warning field was from content analysis of the flow by NSM and represented how likely the flow was to be an intrusion. The classification error rate of a decision tree was used to evaluate which features were most effective in classifying a flow. Feature selection is used to reduce the amount of information needed to classify a flow and to improve the error rate of the classification. Frank used three different search algorithms to evaluate features for a collection of known flows. With as few as three features, the classifier error rate was just under 2% (All error rates actually reported are a factor 100 smaller, which is not possible with the data set size used. This misprint, superfluous % signs, was confirmed by the author to us in an e-mail.) The three features found by all the search algorithms were flow duration, destination packets, and source bytes. Frank's data included login, email, and remote shell services. Frank used one set of data of about 16,000 flows. Considering the small error rates and that this contained three classes of connections, it is not clear if the training and testing data sets were independent by further apriori division of this data into two sets. Doak [5] has a more general evaluation of search algorithms for feature selection.

Cannady [2] describes a neural network trained to detect network intrusions from packet header data. Cannady uses IP protocol, source and destination addresses, source and destination ports (or ICMP type and code fields), and packet length to characterize each packet. In training the neural network an additional attack element was included to indicate if the packet was from an attack.

Porras [18] looks for statistical anomalies in network traffic that might indicate intrusions. Current traffic is compared against a data base of historical traffic characteristics. The data base includes metrics for traffic intensity, typical port usage, typical active hosts. They also check the content of some network packets against a list of attack signatures.

Paxson [15] describes both content-based and heuristic techniques for identifying backdoor services established by an intruder. Paxson monitors packet content of arbitrary flows to detect signatures of the various interactive services (rlogin, telnet, ssh). He also uses a heuristic to identify interactive traffic $\gamma = \frac{S-G-1}{N}$ where S is the number of small packets, G the number of gaps between small packets, and N the total number of packets He also has a metric for the interarrival of small packets, noting that non-interactive traffic tends to send packets back to back. He uses Internet traces to test his classifiers.

Paxson [16] also describes a timing-based system for detecting stepping stones – a host compromised to forward traffic from one host to another. He notes that interactive sessions can be characterized by active and inactive periods. He detects forwarders by correlating the start of active periods between flows.

3 Packet capture

The Internet Protocol (IP) is used to carry data over the Internet and IP-based intranets. Every data packet carries at least an IP header, and often additional protocol headers in addition to the user data (Figure 1).

IP header	TCP/UDP header	application header	user data

Fig. 1. IP packet layout.

Each IP packet, including fragments, carries the IP header in the first 20 bytes (or more if options are present). Figure 2 shows the layout of the IP header. For our analysis, we are interested only in the 32-bit destination and source addresses, length, and the protocol field. The 8-bit protocol field specifies the type of payload following the IP header, for example, UDP, TCP, or ICMP. For more details on the function of each field refer to Stevens [23], Comer [4], or RFC791 [20].

<table>
<tr><td>version/lth</td><td>TOS</td><td colspan="2">length</td></tr>
<tr><td colspan="2">ID</td><td>flags</td><td>offset</td></tr>
<tr><td>TTL</td><td>proto</td><td colspan="2">checksum</td></tr>
<tr><td colspan="4">source address</td></tr>
<tr><td colspan="4">destination address</td></tr>
<tr><td colspan="4">options (if any)</td></tr>
</table>

Fig. 2. IP header.

The primary transport protocols are UDP, TCP, and ICMP. UDP is a connectionless, lossy datagram protocol used for audio/video streams, time synchronization (NTP), and host-name resolution (DNS) on the Internet. For our analysis we are only interested in the source and destination ports of the 8-byte UDP header (Figure 3).

TCP is a reliable protocol used for most services on the Internet (file transfer, telnet, web/http, email). Our analysis needs the source and destination port from the 20-byte TCP

source port	destination port
length	checksum

Fig. 3. UDP packet header.

header (Figure 4). We also save the flags byte since it can indicate the start and finish of a flow.

<table>
<tr><td colspan="2">source port</td><td>destination port</td></tr>
<tr><td colspan="3">sequence number</td></tr>
<tr><td colspan="3">acknowledgement number</td></tr>
<tr><td>hdrlth</td><td>flags</td><td>window</td></tr>
<tr><td colspan="2">checksum</td><td>urgent pointer</td></tr>
</table>

Fig. 4. TCP header.

The Internet Control Message Protocol (ICMP) is used by a host or router to report unusual conditions or to control packet flows [19]. Most ICMP messages are generated at the kernel level of an OS or by a router. The format of the ICMP payload varies depending on ICMP type, but the first word consists of an 8-bit type field, 8-bit code field, and a 16-bit checksum (Figure 5). For our analysis, we are interested in only the type and code fields.

type	code	checksum

Fig. 5. First word of ICMP header.

There is software available (like tcpdump) that can capture all of the packet headers and record the data to disk. However, to reduce disk usage and speed packet capture, we developed our packet-capture software that only saves a portion of each packet to disk. The software is based on *libpcap* [12], a library available on most UNIX systems. The software runs in promiscuous mode and reads every IP packet on the network interface saving data from the IP header and transport header of each packet. The software records the following information (24 bytes) from each packet:

- timestamp (8 bytes, microseconds)
- source IP address (4 bytes)
- destination IP address (4 bytes)
- IP length (2 bytes)

- IP protocol (1 byte)
- TCP flags (1 byte)
- UDP/TCP source port (2 bytes)
- UDP/TCP destination port (2 bytes)

The C structure (Figure 6) provides another view. The timestamp is provided by *libpcap* and it is in the byte-order of the capturing machine. The remaining data elements are in network-byte order. *Libpcap* is portable across most UNIX platforms thus making our data collector portable as well.

```
struct FlowRec {
        unsigned int secs;
        unsigned int usecs;
        unsigned int srcip;
        unsigned int dstip;
        unsigned short lth;
        unsigned char proto,flags;
        unsigned short sport,dport;
}
```

Fig. 6. C structure data for each packet.

For this research, we collected IP packets from the external FDDI ring at ORNL. For a typical 24-hour period, the ring would carry over 80 gigabytes of data carried in 150 million packets. Our 24-byte packet summary file then is nearly 4 gigabytes for 24 hours. The raw data file has packets from various network connections (flows) intermixed, so we developed additional software to split the data into flows based on source address/port and destination address/port for each IP protocol (TCP, UDP, ICMP, or other).

4 Characteristics of Internet traffic flows

We developed software to split the packet summary data into individual flows. Although TCP has a flags field which can indicate the beginning and end of a flow (and we have that field in our packet data), we chose to define the end of a flow as 10 minutes of idle time. (We did some experiments varying the flow cutoff from 5 to 20 minutes and found 10 minutes to be an effective limit.) Our initial studies concentrated on flows for telnet, rlogin, ssh, rsh, ftp, smtp (email), irc (chat), ICMP. We also had some "mystery" flows from machines that had been compromised, and ICMP flows that were being used for remote shell services. We discarded one-way flows and flows with fewer than 10 packets as part of this initial study. Our data differs from earlier studies in that we have time-stamps for each packet, so we can study inter-packet arrival times.

Visual examination of raw data is an important first step in any data analysis exercise. When the data sets are very large, as is the case with network flows, visualization tools that present the data in a manageable way need to be developed. We used S-Plus[21] to plot collections of individual flows. Presenting a number of individual flow plots together on a page provides a useful facility for studying flows. We agree with Cleveland and Sun [3], who emphasize that Internet traffic data must be explored in its full complexity and relying on summaries is inadequate.

Figures 7 to 13 present individual flow plots in groups of 80 flows. We present the first 80 flows with at least 10 packets within each flow type from a two hour collection of IP packets. Rlogin and irc have fewer flows within this period. The following flow types are presented:

Figure 7	file transfer (port 20)
Figure 8	remote shell (port 21)
Figure 9	secure shell (port 22)
Figure 10	telnet (port 23)
Figure 11	email (port 25)
Figure 12	rlogin (port 513)
Figure 14	icmp
Figure 13	irc (port 6667)

Within individual flow plots, packets are distinguished by four colors indicating the side of the connection and whether they follow a packet from the same or other side of the connection. Each packet in an individual flow is plotted as a point and connected with a line to the previous packet point. The horizontal axis is time between packets (log10 of seconds) and the vertical axis is packet size (log10 of size).

These flow plots capture nearly all of the heuristics that are often used for classifying flows. Interactive traffic (presumably with a human on one end) has longer inter-packet

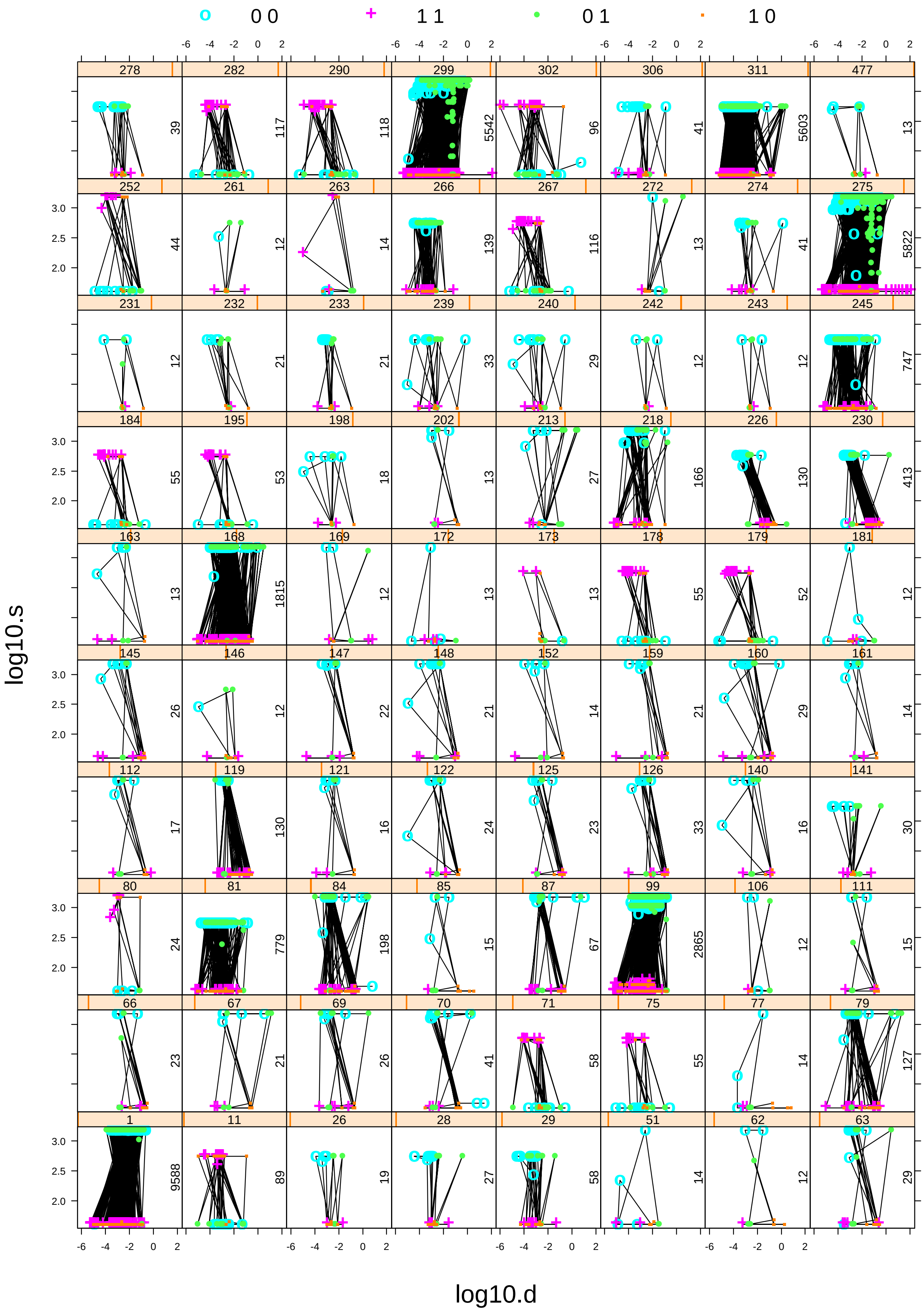

Fig. 7. Log10 inter-packet time versus log10 packet size for individual FTP flows (port 20). Lines indicate packet chronology and color indicates direction of current and preceding packet.

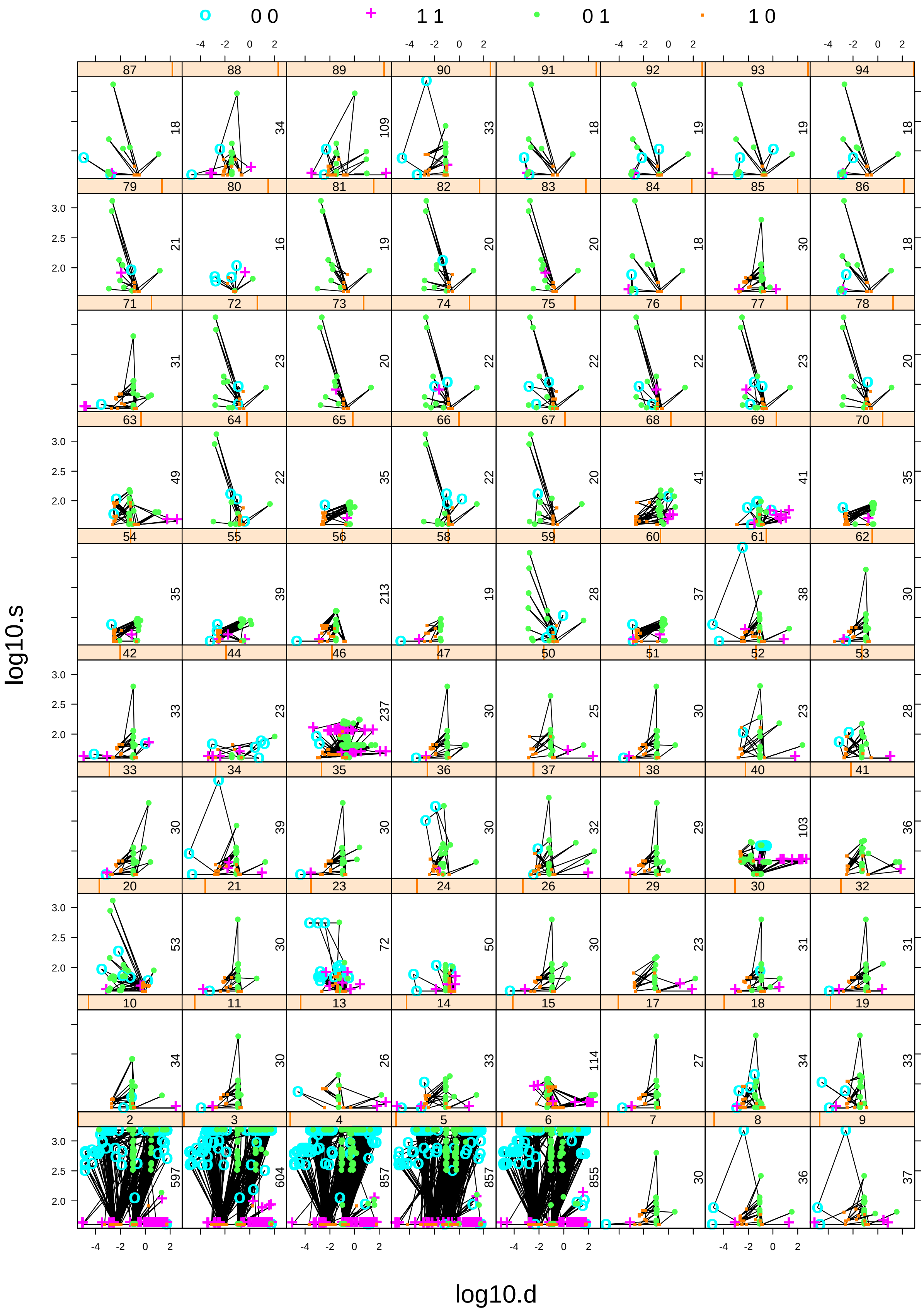

Fig. 8. Log10 inter-packet time versus log10 packet size for individual RSH flows (port 21). Lines indicate packet chronology and color indicates direction of current and preceding packet.

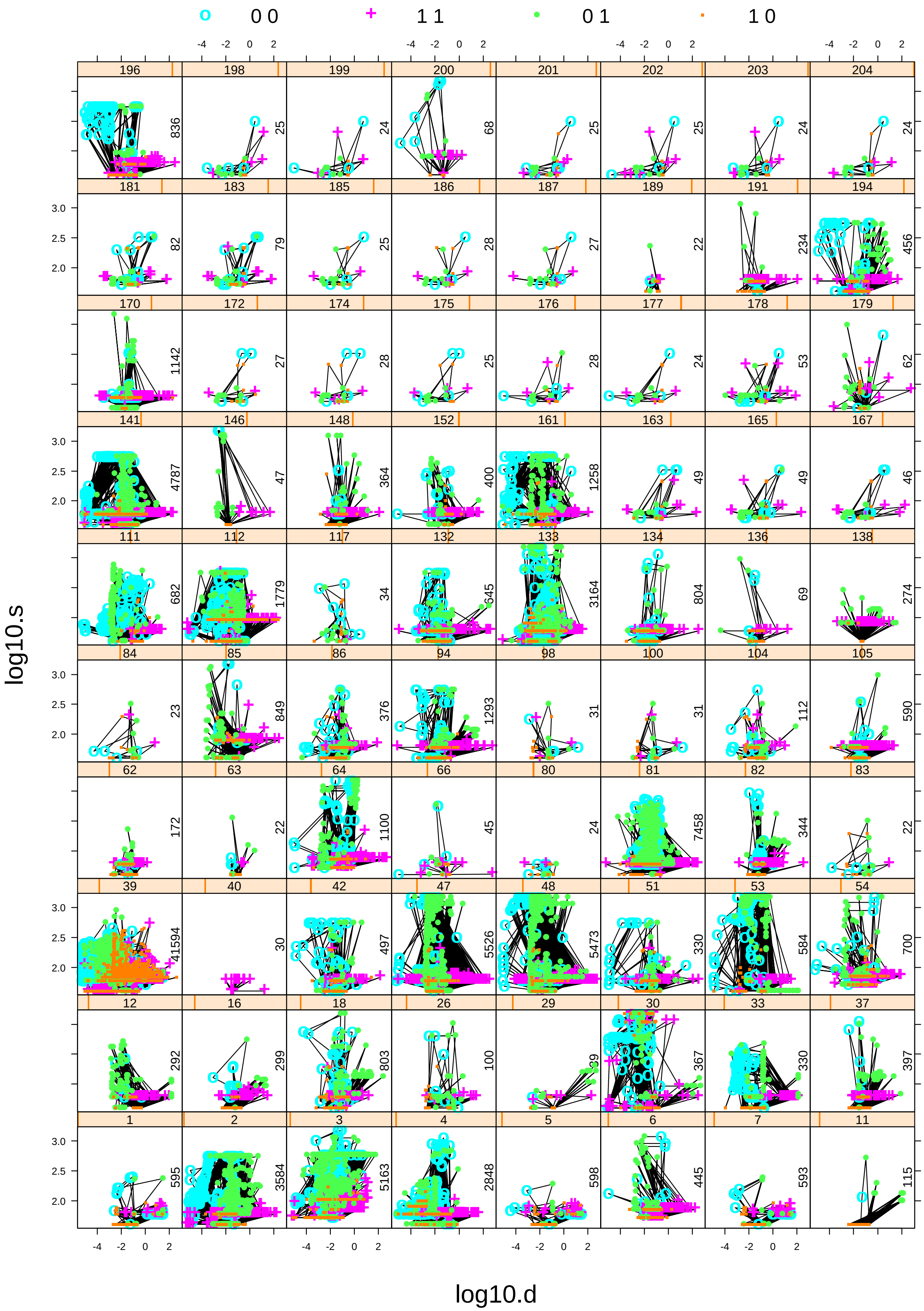

Fig. 9. Log10 inter-packet time versus log10 packet size for individual SSH flows (port 22). Lines indicate packet chronology and color indicates direction of current and preceding packet.

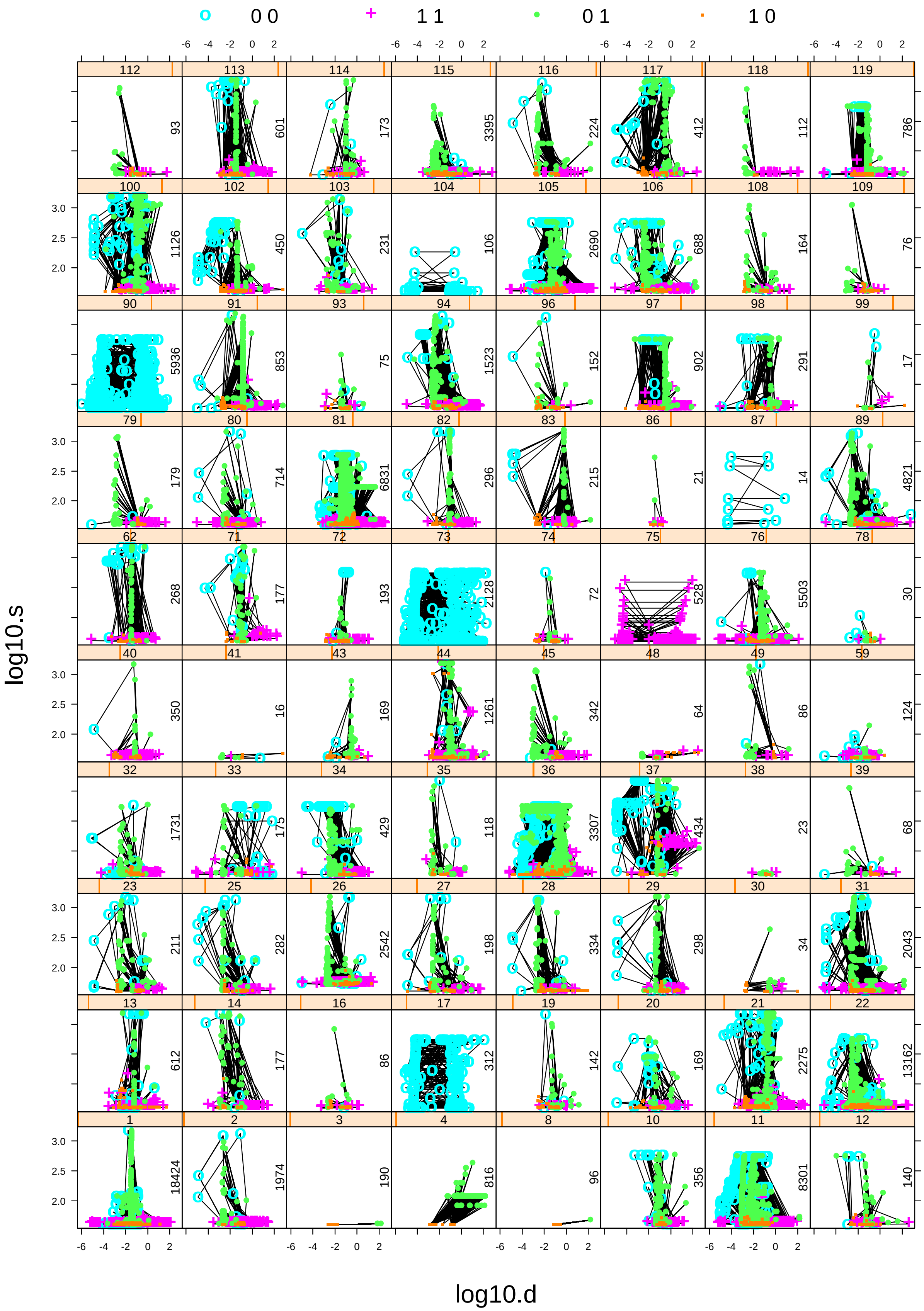

Fig. 10. Log10 inter-packet time versus log10 packet size for individual telnet flows (port 23). Lines indicate packet chronology and color indicates direction of current and preceding packet.

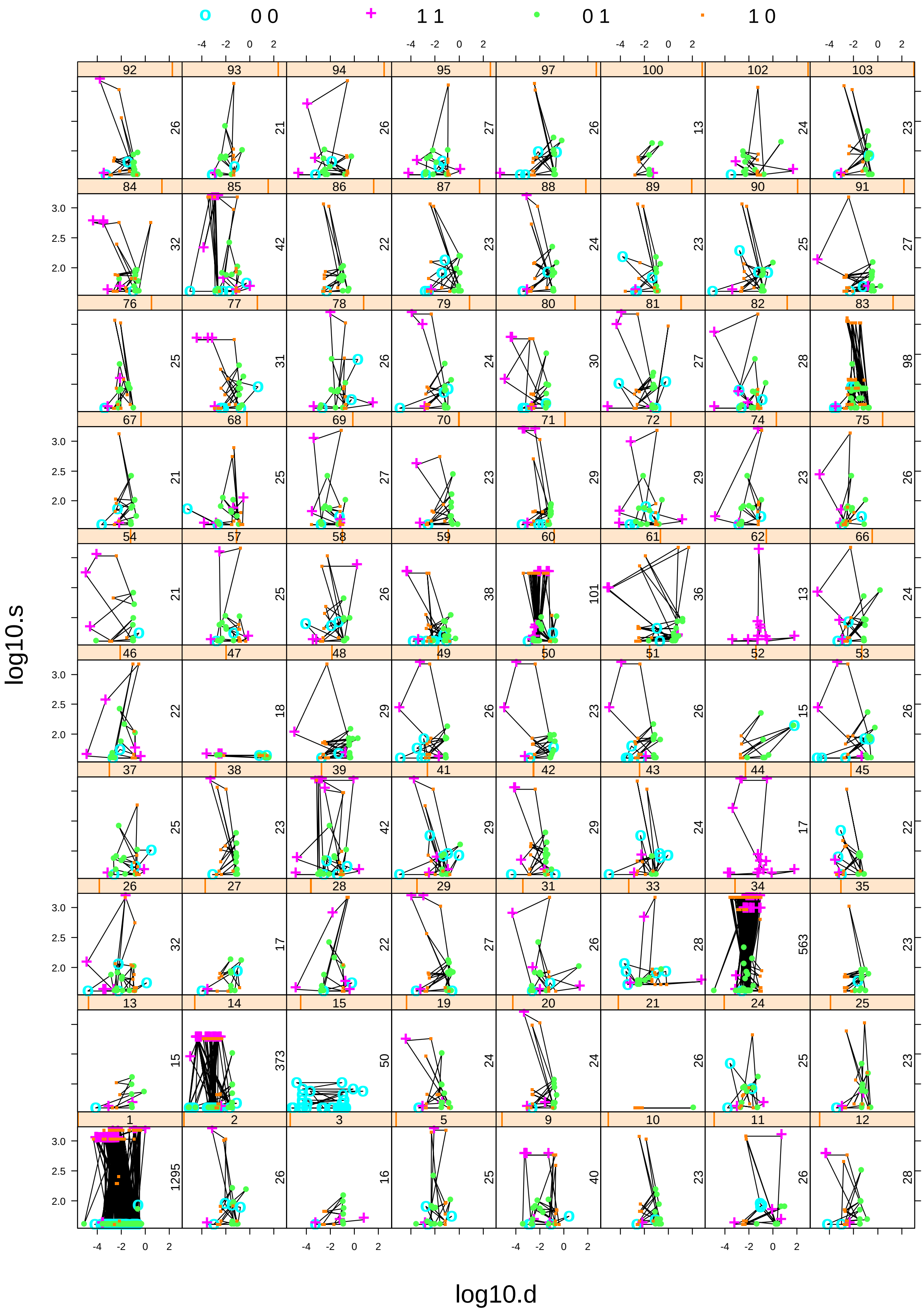

Fig. 11. Log10 inter-packet time versus log10 packet size for individual email flows (port 25). Lines indicate packet chronology and color indicates direction of current and preceding packet.

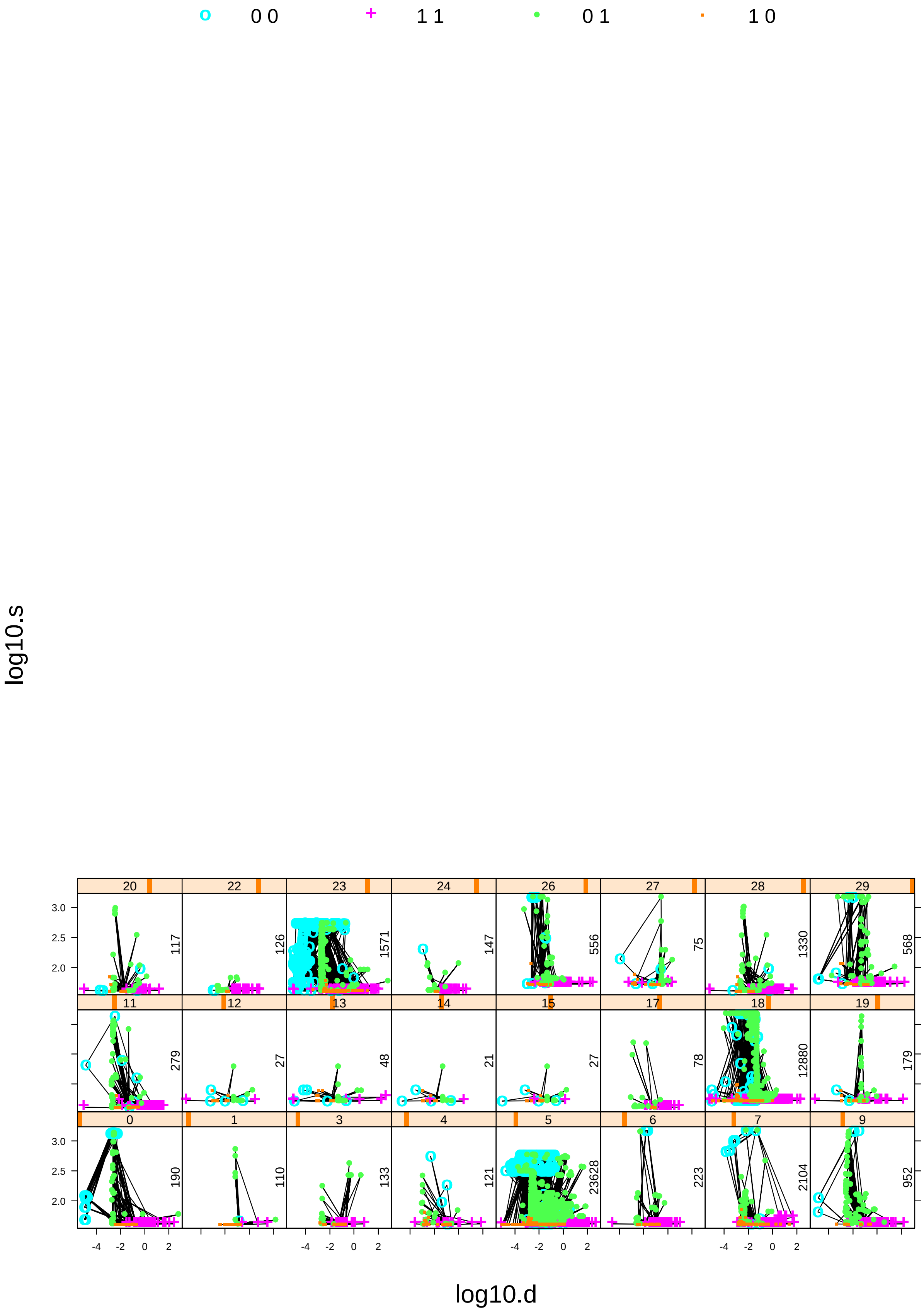

Fig. 12. Log10 inter-packet time versus log10 packet size for individual rlogin flows (port 513). Lines indicate packet chronology and color indicates direction of current and preceding packet.

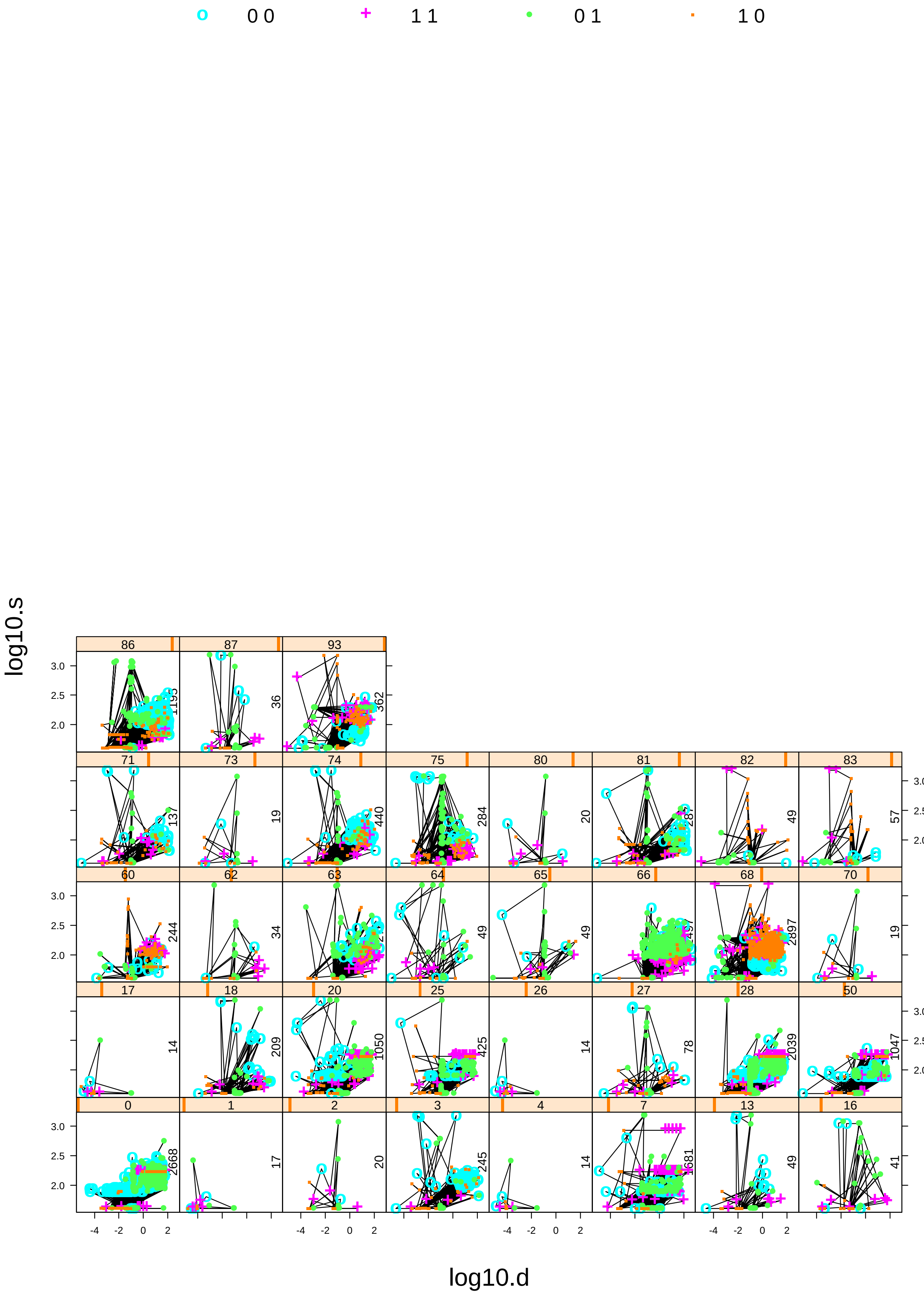

Fig. 13. Log10 inter-packet time versus log10 packet size for individual irc flows (port 6667). Lines indicate packet chronology and color indicates direction of current and preceding packet.

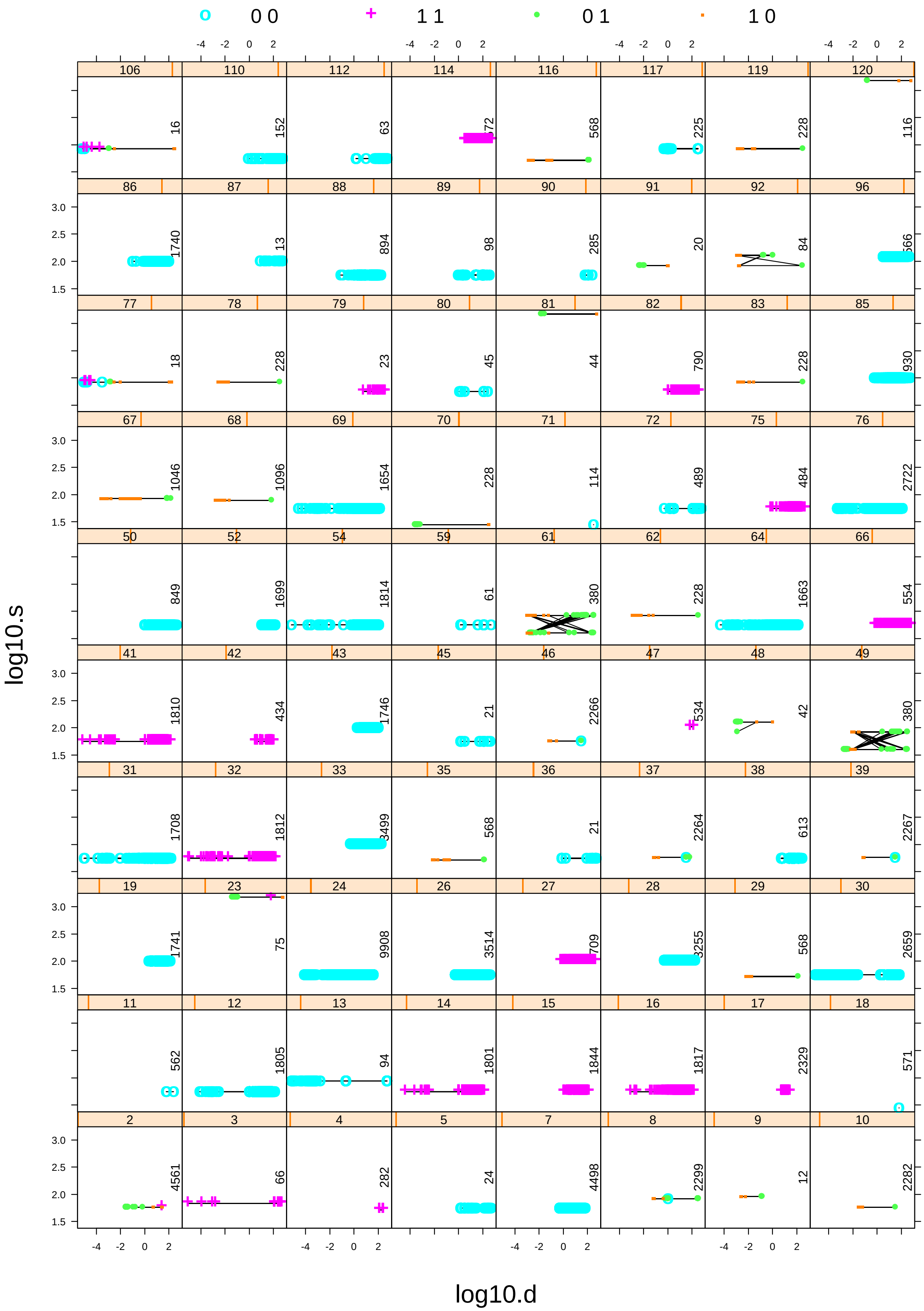

Fig. 14. Log10 inter-packet time versus log10 packet size for individual icmp flows. Lines indicate packet chronology and color indicates direction of current and preceding packet.

delays than bulk-transfer services like email or ftp. We can also determine which end is the client and which is the server for interactive flows. Bulk transfers (ftp) tend to have big packets originating from just one side of the flow and often have back to back packets. One can also detect machine-controlled (fixed interval packets) interaction over interactive (telnet/rlogin) ports.

The flow visualization has proved useful in classifying unknown flows from compromised machines. Figure 15 illustrates two unknown flows from a compromised machine. The two flow images are nearly identical in structure but reversed in color. Later forensic analysis of the compromised machine found that the intruder had installed an Internet-chat relay. These two images are an example of a relay (Paxson's stepping stone [15]). The inbound traffic from one remote host is forwarded by the compromised host as outbound traffic to another remote host (hence the color reversal).

5 Classifying network flows

The SPlus images of flows provide a visual summary of flow behavior that can be used to study flow traffic in great detail. However, for practical monitoring we need an automated way of classifying the flows, and then a way to compare an unknown flow to a knowledge-base of known flows. For this research, we experimented with both heuristic and statistical methods for classifying flows. From our observations and visualizations of various network flows, we experimented with various heuristics for classifying flows. We developed various indexes for a flow based on packet length (average length inbound versus outbound, or maximum/median packet length), packet runs or bursts (number of back-to-back packets inbound versus outbound), inter-packet times (distribution of on/off times, which side of flow would restart after an idle period). These heuristics can be effective but they do not provide a systematic way of selecting characteristics for classification.

Most of our effort was directed at using statistical techniques to classify flows, trying to mimic possible visual differentiation of our flow plots. Our approach at quantizing the flow plots was to "bin" the image. We can logically superimpose a grid over the flow plot and count the data points in the plot that fall in each bin. A few packet sizes are binned individually because of their high frequency and special use. This is motivated by the appearance of high frequencies at a few specific packet sizes in many flow plots. See Figure 16 for an enlarged

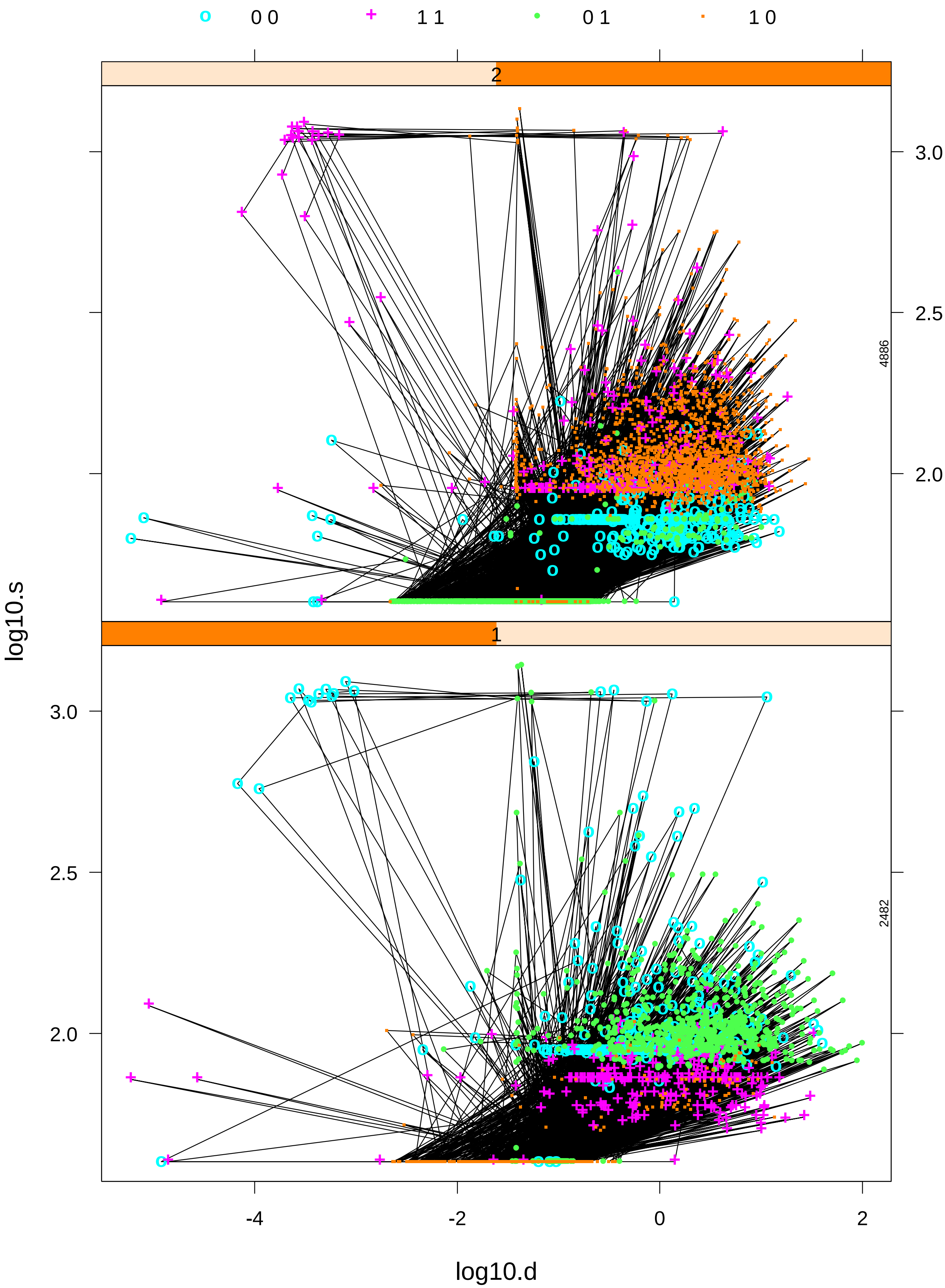

Fig. 15. Two flows from a compromised machine (number of packets shown on right margin of each flow plot).

example of a rlogin flow and an irc flow showing concentrated traffic at distinct packet sizes. The packets that are not binned individually are binned in intervals. Figure 17 is an example of a grid that results in 288 (4×72) bins. In this example, special packet sizes 40, 41, 576, and 1500 and four log10(size) bins provide 8 bins for size across nine bins for delay. This gives a total of 72 exclusive bins. When repeated for the four types of packets (00, 01, 10, and 11, giving the current packet side and last packet side) this gives 288 bins. Thus in this example, each flow is characterized with a vector of 288 counts. We further divide the counts of each flow by the total count for that flow. The result is a packet distribution on 288 points for a single flow that is independent of the number of packets in the flow. This distribution is a characterization of a single flow as a set of 288 characteristics, all on the [0,1] interval. These characteristics are mostly a reduction of the information contained in a flow plot, but they do not suffer from over-plotting loss as do some longer flow plots.

Conceptually, we can use a set of flows from a single flow type, to estimate a 288-dimensional density. This density then provides the probability that an unknown flow belongs to that flow type. With estimated densities for a number of flow types, an unknown flow would be classified to the flow type that provides the highest probability.

In order to visualize the density shapes and suggest a density estimation technique, we used two dimension reduction techniques. One, a linear projection technique called principal components (see [10], for example) and the other a nonlinear projection technique of metric multi-dimensional scaling (see [9], for example).

Principal components analysis (PCA) is a technique that sequentially finds orthogonal directions of highest variation in the original high dimensional space. The underlying problem is an eigenvalue computation on a covariance matrix of the packet flow distributions. Associated with the largest eigenvalues are the eigenvectors that capture most of the variation in the original variables (288 in the preceding binning example). This is also useful for visualization of the separation that the original variables achieve among the flow types. Although high variation is not the same criterion as flow type separation, separation is often obtained. This is because variation within a flow-type is usually less than variation between flow-types. An example giving pairwise plots of the first three principal components is in Figure 18. In computing the principal components, the more frequent flow types (port 25 and icmp) were limited to 500 (by sub-sampling), while the least frequent rlogin had only 24 flows. This was done to reduce the domination of the more frequent flows.

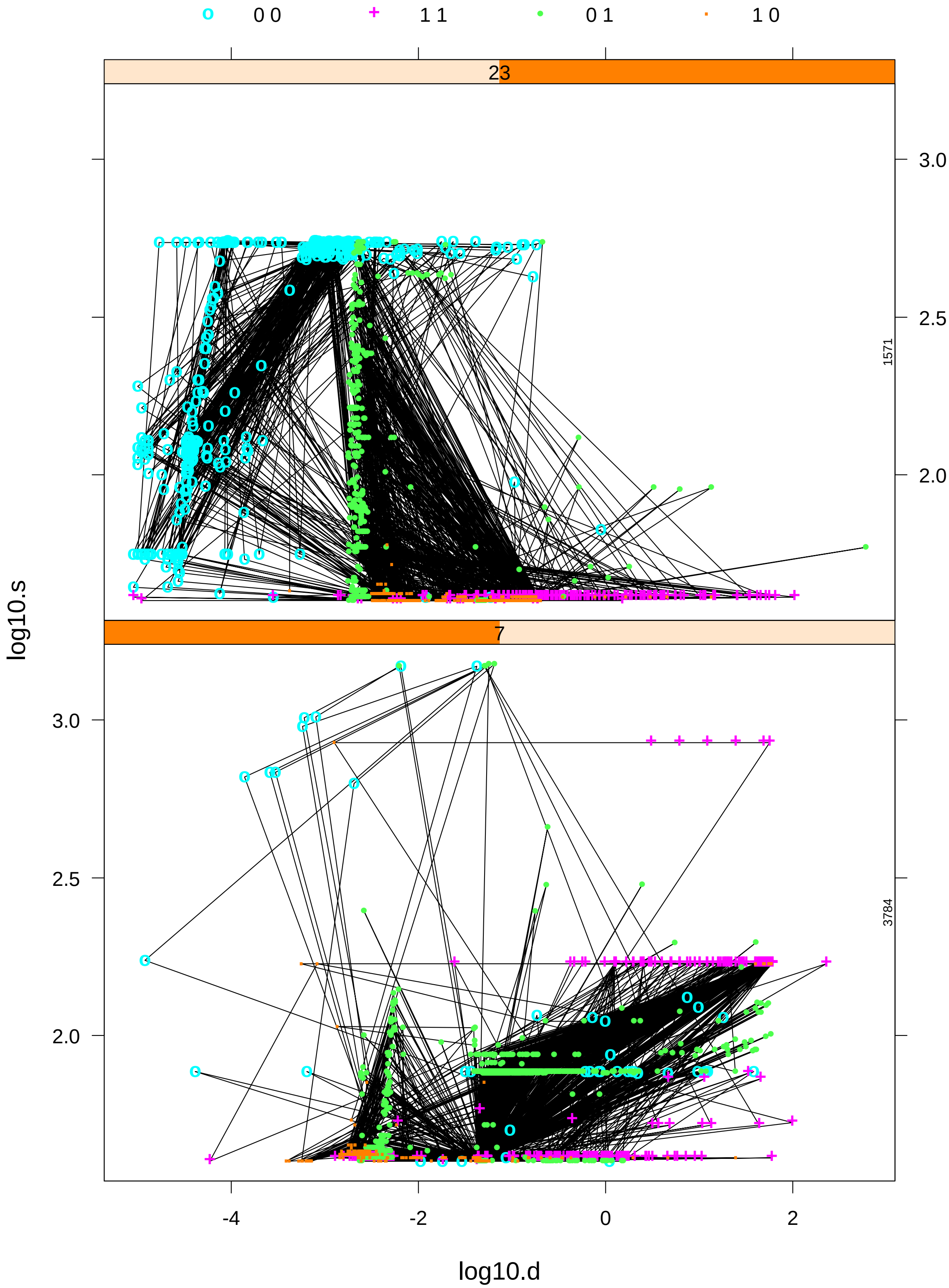

Fig. 16. Detailed flow plot for an rlogin flow (7 bottom) and an irc flow (23 top).

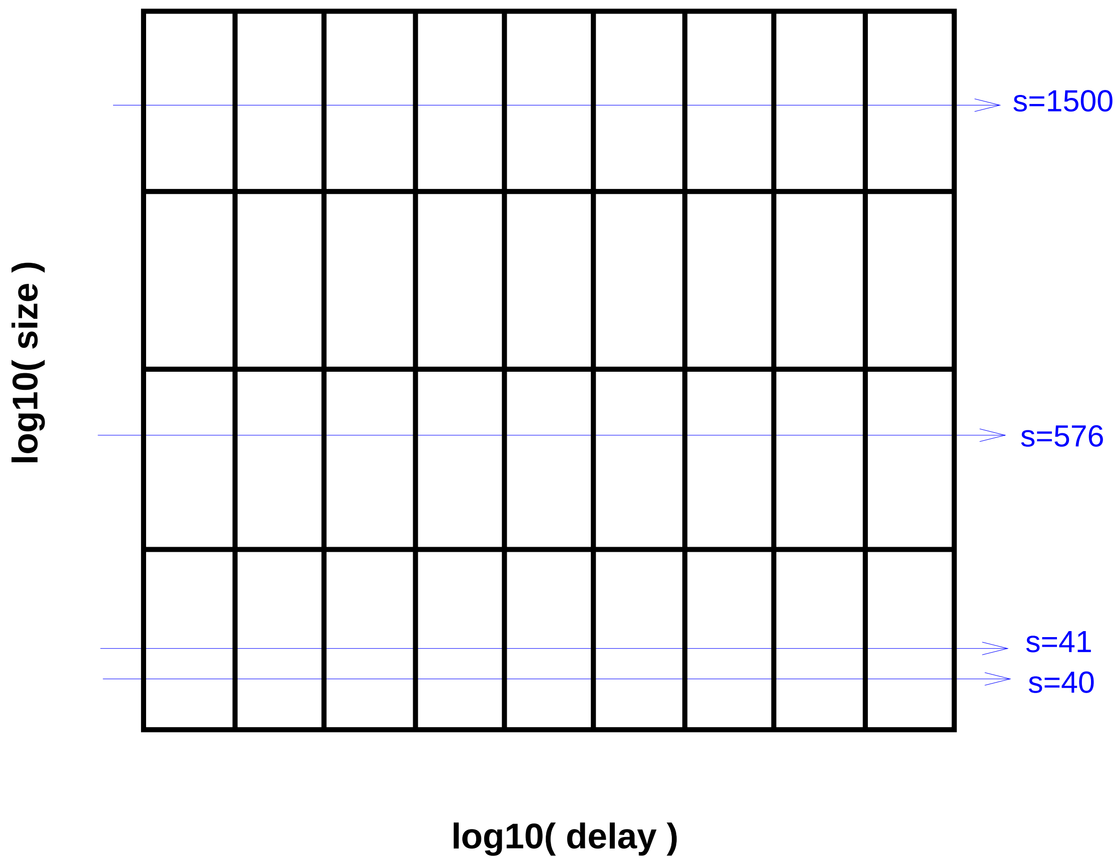

Fig. 17. Binning a flow plot: intervals define a set of bins; intersections of special packet sizes with interval bins define additional bins.

Metric multidimensional scaling (MDS) is a non-linear technique that begins with a distance matrix from the high dimensional representation and attempts to preserve those distances in a lower dimensional space. The solution involves an eigenvalue problem on the distance matrix. Pairwise plots of a four-dimensional MDS representation applied to a group of network services is in Figure 19.

Particularly the first two dimensions of the MDS projection in Figure 19 show a separation of flows that have a human on one end (ports 21, 22, 23, and 513) from those that have a machine at both ends (ports 20 and 25). The two "unknown" flows (marked as port 0) from a compromised machine are clearly on the human side.

Repeating the multidimensional scaling on the machine flows only (Figure 20) shows that 20 (file transfer) clusters into two distinct groups: one easily separated from 25 and the other more difficult. Although the number of packets per flow is not used in the analysis, the easily separated group is a set of massive file transfers that are one to two orders of magnitude

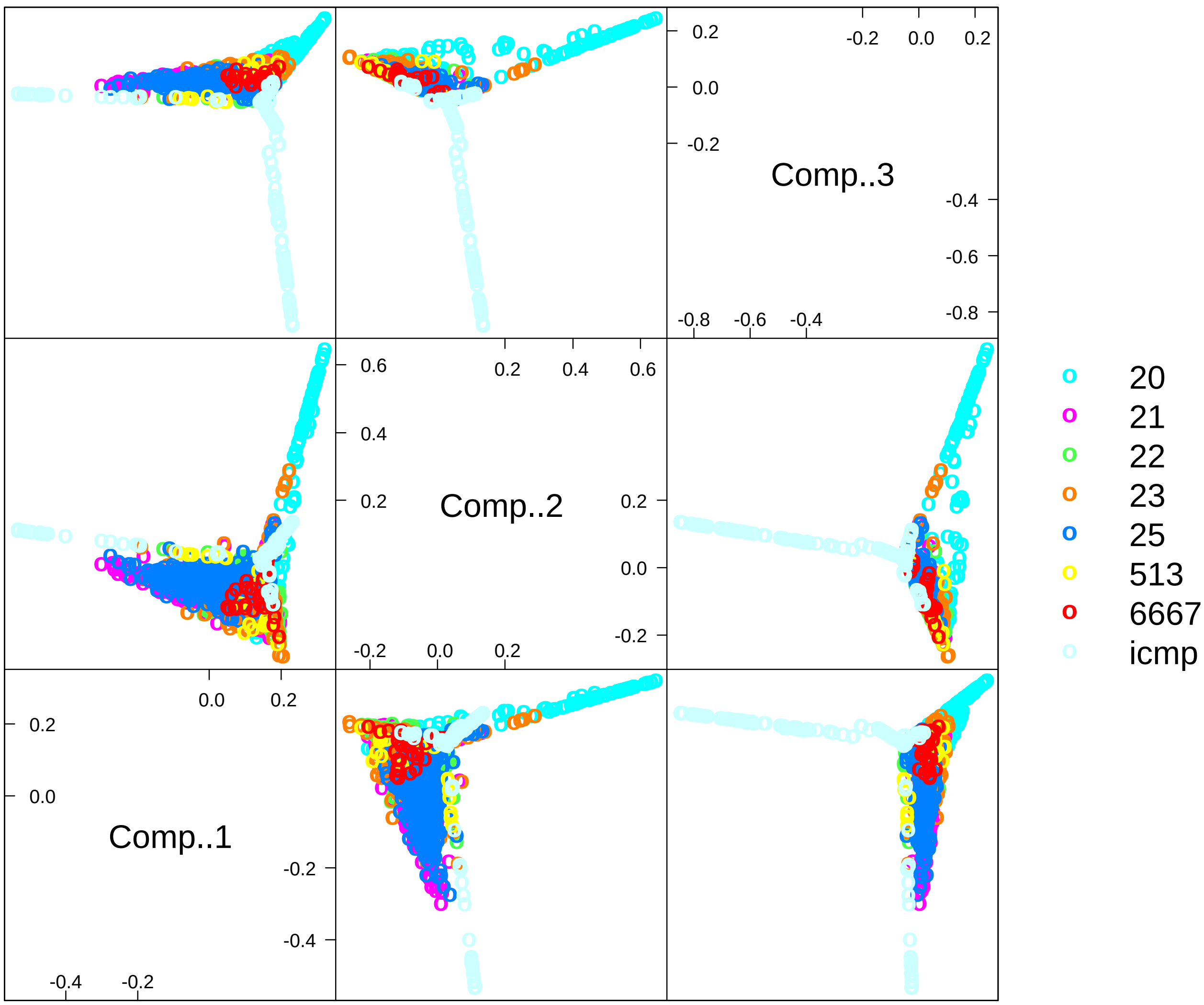

Fig. 18. Pairwise plots of the first three principal components for a 2 hour collection of flows.

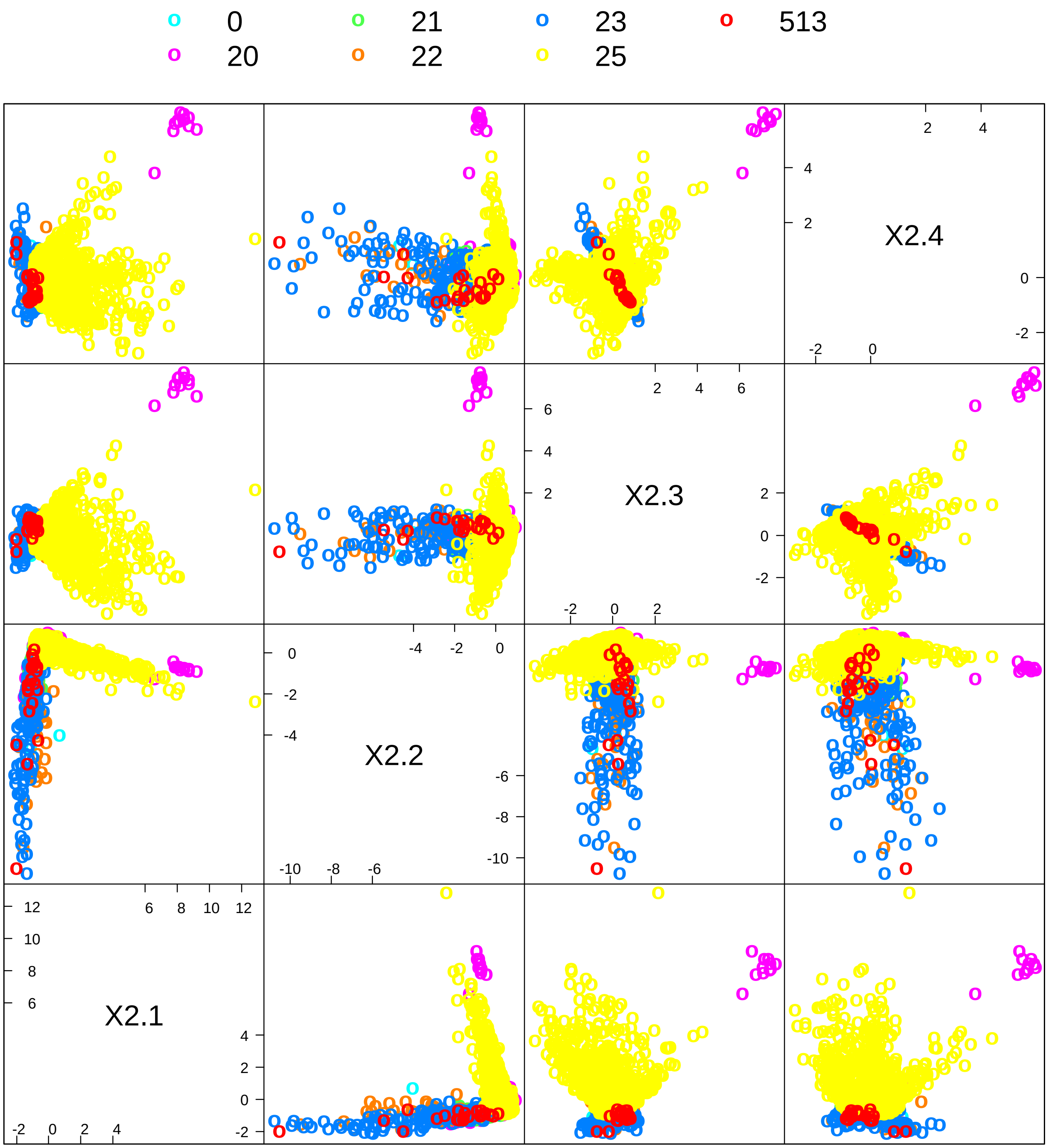

Fig. 19. Pairwise plots of a four-dimensional MDS representation of a group of network services.

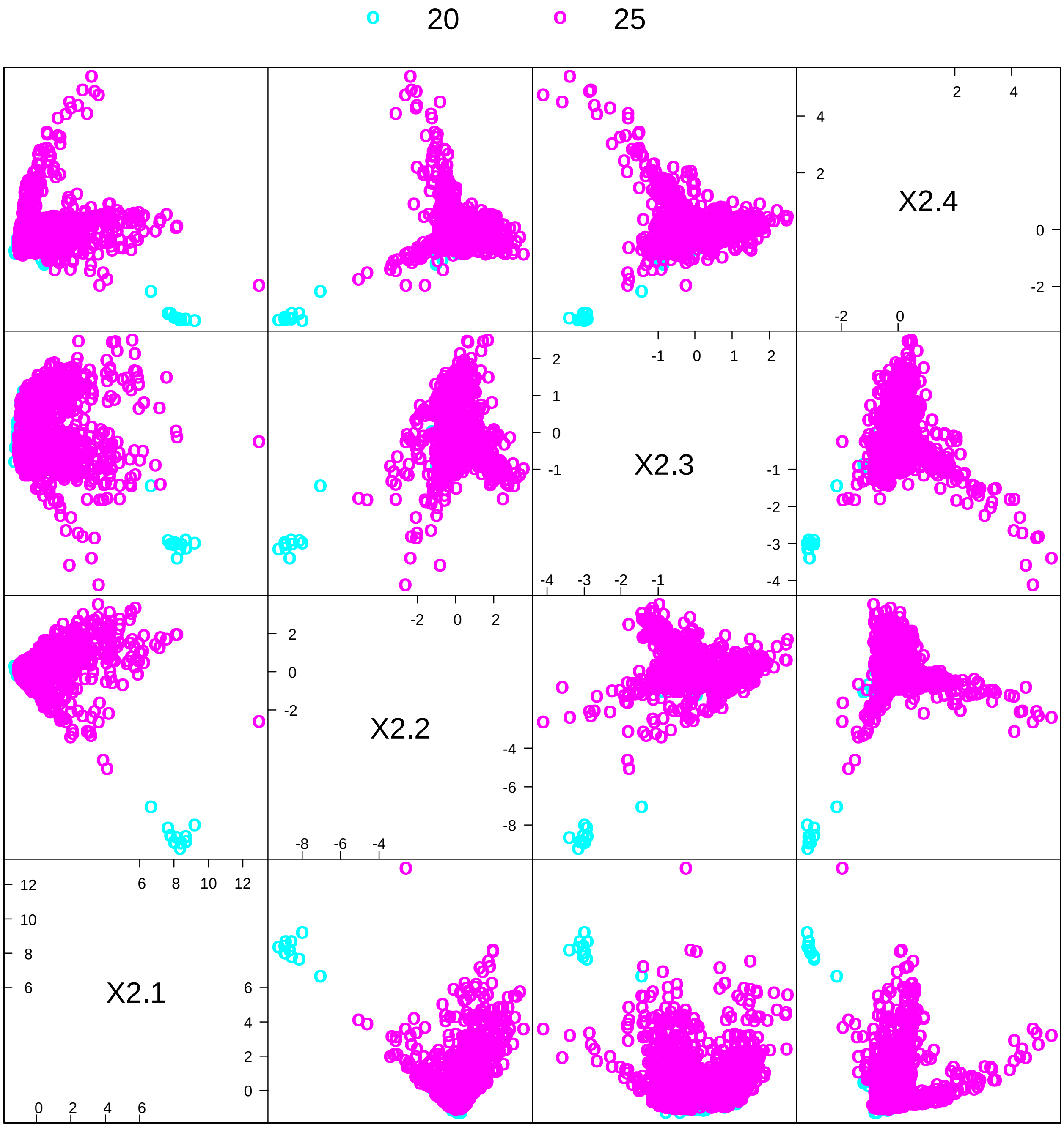

Fig. 20. Pairwise plots of a four-dimensional MDS representation of two machine flows.

larger than the rest. Their packet size and inter-packet delay profile must be different from the rest.

Applying multidimensional scaling to the flows with a human on one end (Figure 21) shows some separation but also considerable overlap between the flows. Flow 23 (telnet) also presents two distinct clusters. The two "unknown" flows (marked as port 0) are most likely classified as port 22 (secure shell). This is reasonable as the flows were an Internet-chat relay, that carries an overhead machine traffic not unlike secure shell.

The visualization of low-dimensional representations of flows is useful for understanding of various flow characteristics and can be a source of more in-depth study of flow patterns and network service relationships. Our goal was to examine the spatial distributions of individual network services. These distributions are irregularly shaped, as can be seen in the MDS and PCA plots. The reasons underlying this irregularity are probably that each service can likely be further subdivided into subgroups. The two file transfer groups are a clear example. This irregularity suggests that nonparametric density estimation is appropriate.

Nonparametric density estimation works well up to about three or four dimensions [22]. For this reason, we use dimension reduction to three extracted features before density estimation in the reduced space. We use PCA for this dimension reduction because features are extracted with a linear transformation, resulting in fast evaluation of new flows. MDS requires inter-flow distances to every flow in the training set in the original high-dimensional space.

Binning and kernel density estimation are possible nonparametric density estimation candidates [22]. We choose binning because our goal is fast density evaluation for unknown flows. Evaluation of kernel density estimates requires visiting every data point (every flow) that was used to construct the estimate. Binned density estimates only require the evaluation of an index to access the appropriate bin.

The learning process for flow classification is as follows:

Learning Process
Compute flow distributions on selected binning parameters.
Compute PCA transformation from full collection of flow distributions.
Estimate density for each flow type from three PCA.

Selecting flow binning parameters is a critical step because these are the features which will discriminate between network services. Based on a tabulation of packet size frequencies for

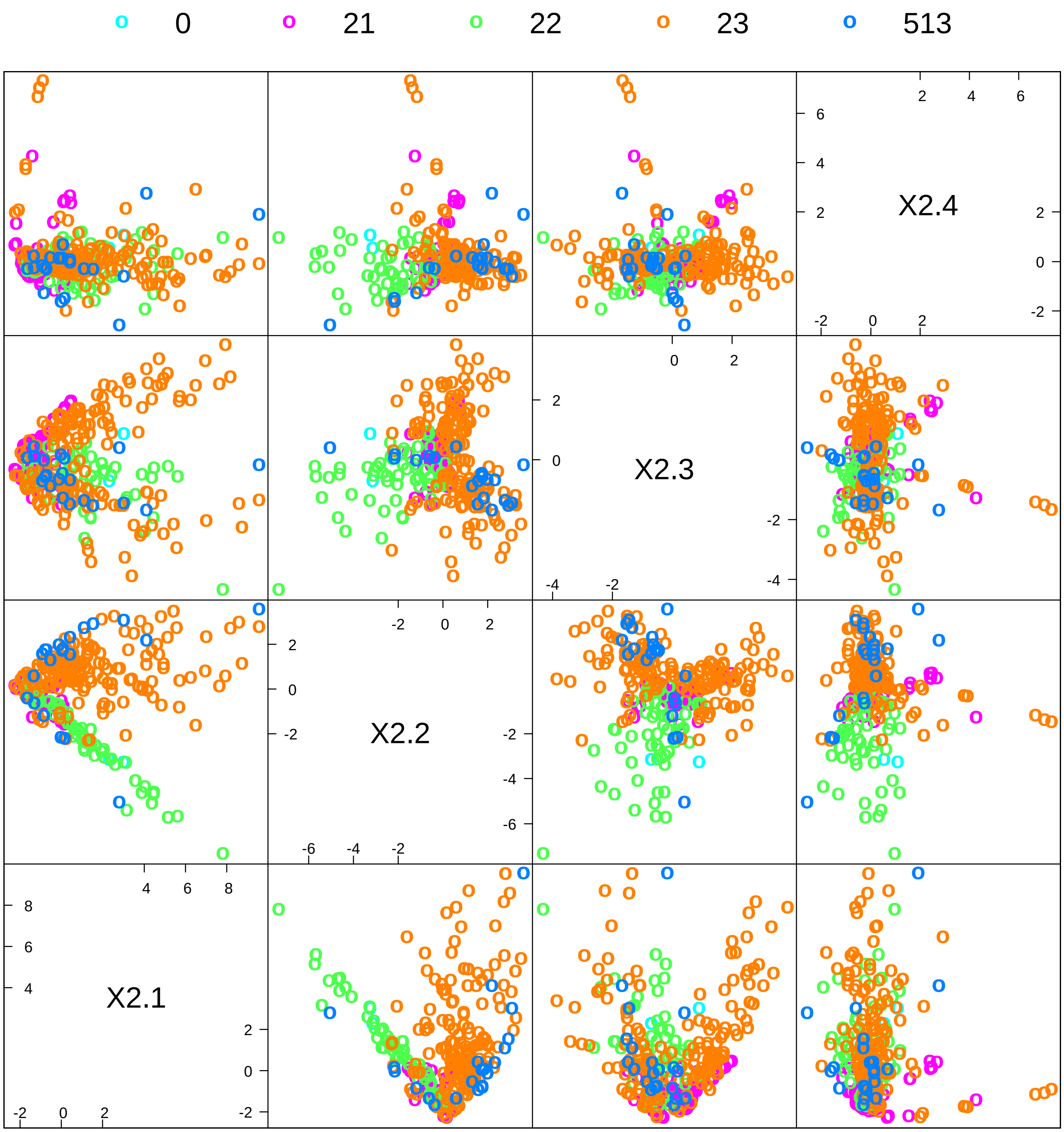

Fig. 21. Pairwise plots of a four-dimensional MDS representation of flows with a human on one end.

eight network services (ftp, rsh, ssh, telnet, e-mail, rlogin, icmp, and irc) over a two hour period we selected the following packet sizes as an initial binning set: 40, 41, 42, 52, 56, 60, 64, 68, 70, 76, 84, 576, 1064, and 1500. The remainder were grouped into two intervals 100 bytes and below, and above 100 bytes. Inter-packet delays (in seconds) were binned into four groups: under 10^{-3}, 10^{-3} to under $10^{-0.1}$, $10^{-0.1}$ to under $10^{1.5}$, and $10^{1.5}$ and over. These binning parameters are not optimal and are selected from aggregate information on each network service. An optimization of these parameters should be based on individual flows rather than an aggregate, because it is the flows that we ultimately classify. We discuss such optimization elsewhere.

Currently we have implemented a three-dimensional binned density estimation algorithm for the *Estimate* step of the learning process. A small amount of density smoothing (or regression toward the mean) is useful to prevent overfitting and needs to be added for effective classification of unknown flows.

With an estimated density for each network service we can rapidly classify flows with the following algorithm:

Classification of a flow
Compute flow distribution on the binning parameters.
Apply PCA transformation to flow distribution.
Evaluate flow probability for each network service density.
Select flow type with highest probability.

The *Compute* step is O(n), where n is the number of packets in the flow. In relation to the *Compute* step, the remaining steps are fast O(1) computations. Overall, this is feasible in real time.

Based on a two hour collection of traffic, we took all two-sided flows with over 10 packets and selected those that belonged to one of eight network services. This provided 16,949 flows. Using the learning process and classification of the same flows, we obtained the table below. Because e-mail and to some extent icmp dominated the data set, we limited these two services to 500 flows in the PCA computation by sub-sampling. We used the first three principal components. Classification performance on this data is shown in Table 1. The results show that it is possible to effectively separate network services on the basis of packet size, direction, and inter-packet delay information only, without content analysis. Preliminary results for classifying a second data set with email flows resulted in a .18 error rate, up from the .07 rate in Table 1 as expected. In order to build an effective classification

Table 1. Classification counts and error rate

	ftp	rsh	ssh	telnet	e-mail	rlogin	icmp	irc	total	error rate
ftp	316	0	2	4	0	0	0	0	322	.0186
rsh	0	436	2	7	4	0	0	0	449	.0229
ssh	0	0	85	0	0	0	0	0	85	.0000
telnet	3	1	1	176	0	0	1	0	182	.0330
e-mail	25	711	5	241	13293	1	99	1	14376	.0753
rlogin	0	0	0	0	0	24	0	0	24	.0000
icmp	0	0	0	2	0	0	1474	0	1476	.0014
irc	0	0	0	0	0	0	0	35	35	.0000

algorithm for new flows, this work needs to continue with optimizing binning parameters, density smoothing, and more data for characterizing less frequent services.

We emphasize that the parameters (packet size and delay bin boundaries) should be optimized with respect to classification performance. The following is a sketch of an optimization algorithm.

Flow Binning Parameter Optimization
While improvement seen in last k iterations
  Replace bins with low *coefficient contribution*
  Bin flows into the resulting bins
  Compute the first p principal components of the binned flows
  For all combinations of 3 components
    Estimate density for each flow type
    Classify all flows and compute maximum error rate
  Set coefficient contribution with best set of PCA components

While the maximum error rate provides a criterion for the optimization, contributions of individual bins enter through their coefficient magnitudes in the best set of 3 principal components. An alternate contribution criterion might be the coefficients of the best multiple linear discriminator function (another eigenvalue problem). The optimization requires substantial computation but it does not need to be repeated unless there is a change in the protocol of a monitored service. Once parameters are optimized, the learning process computes network service density estimates and flows can then be classified in real time. It is also likely that values of the optimized parameters will have an interpretation that enhances our understanding of network service performance and protocols.

Because of the substantial computational requirements of this optimization, a faster implementation (in C) than we currently have in S-Plus is necessary. Preliminary C versions of the *Bin*, *Compute*, and *Estimate* steps have been recently completed by a graduate student.

6 Conclusions

We have shown that it is possible to categorize Internet traffic flows without content analysis. The various Internet services have distinctive statistical signatures and it is possible to identify certain classes of service even when they are running on non-standard ports. Our classifier can detect interactive sessions running over services that are normally used for bulk transfer or network control. Relay-traffic from hosts that intruders have compromised and setup as forwarders can be identified as well. The classifier works off-line and its effectiveness is sensitive to the amount of packets in a flow.

Further work is needed to provide an effective real time classifier. First, binning parameters must be optimized as suggested in Section 5. Second, density smoothing should be added to network service density estimators. And finally, more data needs to be collected for training classification of relatively infrequent network services.

It should be noted that the clever intruder can defeat statistical flow classifiers, though it requires more effort, time, and more sophisticated intrusion software. If only a one-way conversation is required into the attacked site, for example, to initiate a passive daemon, the intruder could change source addresses, protocol, and port numbers for each packet sent to the attacked site, foiling any flow categorization. Packets could be padded to fixed or varying lengths to defeat categorization by packet size. Also the packet rate could be quite low, say, one per hour, further defeating flow categorization. Packet-rate characterizations could also be masked by sending at fixed intervals. Time-based signatures, in general, are difficult because of delays that can be introduced by congestion and queuing along the packet's route through the Internet. Information could also be passed inside legitimate packets [6]. For two-way conversations, defeating detection is a little harder, but the intruder could still use the afore-mentioned techniques for inbound traffic, and the returning traffic could vary port and protocol but would probably need the same destination address. (Though the intruder could have multiple destination addresses to accept "replies.")

Since flow characterization can be fooled, it must be combined with other intrusion detection schemes (content analysis, host traffic profiles, host system logs, host file integrity checks) and defense mechanisms such as filtering routers, firewalls, proper authentication.

We believe that the flow classifier could be made more effective with some additional analysis and research. Future work can adapt the software to work in real-time, identifying "interesting" flows, alerting security personnel, initiating content recording of the flow, or blocking the flow.

References

[1] R. Caceres, P. Danzig, S. Jamin, and D. Mitzel. Characteristics of Wide-Area TCP/IP Converstations. *ACM SIGCOMM*, September 1991.

[2] J. Cannady. Artificial Neural Networks for Misuse Detection. *NISSC*, October 1998.

[3] W. Cleveland and D. Sun. Internet Traffic Data. *Journal of the American Statistical Association*, pages 979–985, September 2000.

[4] Douglas Comer. *"Internetworking with TCP/IP"* . Prentic Hall, 1988.

[5] J. Doak. An Evaluation of Search Algorithms for Feature Selection. *UCD/LANL*, January 1994.

[6] T. H. Dunigan. Internet steganography. Technical report, Oak Ridge National Laboratory, Oak Ridge, TN, 1998. ORNL/TM-limited distribution.

[7] J. Frank. Artifial Intelligence and Intrusion Detection: Current and Future Directions. *Proceedings of the 13th National Computer Security Conference*, 1994.

[8] L. Heberlein, G. Dias, K. Levitt, B.Mukherjee, J. Wood, and D. Wolber. A Network Security Monitor. *IEEE Symposium on Research in Computer Security and Privacy*, 1990.

[9] R. A. Johnson and D. W. Wichern. *Applied Multivariate Statistical Analysis.* Prentice-Hall, 1982.

[10] I. T. Joliffe. *Principal Component Analysis.* Springer-Verlag, 1986.

[11] B. Mah. An Empirical Model of HTTP Network Traffic. *IEEE Infocom*, pages 592–600, April 1997.

[12] S. McCanne and V. Jacobson. The BSD Packet Filter. *USENIX*, January, 1993.

[13] A. Mena and J. Heidemann. An Empirical Study of Real Audio Traffic. *IEEE Infocom*, March 2000.

[14] J. Mogul. The case for persistent-connection HTTP. *ACM SIGCOMM*, pages 299–313, August 1995.

[15] V. Paxson and Y. Zhang. Detecting Backdoors. *USENIX*, 2000.

[16] V. Paxson and Y. Zhang. Detecting Stepping Stones . *USENIX*, 2000.

[17] P.Danzig, R. Caceres, S. Jamin, D. Mitzel, and D. Estrin. An Empirical Workload Model for Driving Wide-Area TCP/IP Network Simulations. *Journal of Internetworking*, March 1992.

[18] P. Porras and A. Valdes. Live Traffic Analysis of TCP/IP Gateways. *Networks and Distributed Systems Security Symposium*, March 1998.

[19] J. Postel. Internet Control Message Protocol. *RFC 792*, September 1981.

[20] J. Postel. Internet Protocol. *RFC 791*, September 1981.

[21] . *S-PLUS 5 for UNIX User's Guide.* Data Analysis Products Division, MathSoft, Seattle, WA, 1998.

[22] David W. Scott. *Multivariate Density Estimation: theory, practice, and visualization.* John Wiley & Sons, Inc., New York, 1992.

[23] Richard Stevens. *TCP/Illustrated, Volume 1* . Addison Wesley, 1994.

[24] K. Thompson, G. Miller, and R. Wilder. Wide-Area Internet Traffic Patterns and Characteristics. *IEEE Network*, November 1997.