Enhanced Network Security by a Dynamic Packet Filter

by
Heikki Julkunen

List of Figures

List of Tables

Introduction

A 'firewall' is a program that is used to control the access to a host on the internet. The term firewall has no exact definition!

In order to be able to specify a computer or host on a network, some addressing scheme must be used. There are five types of addresses in the TCP/IP stack. Table 1,page 25.

Two common firewall types:

- Packet filter firewall. Can be *simple* or *dynamic*. A dynamic packet filter maintains a history of packets.
- Proxying firewall. Requires modifications of existing software either by relinking or using shared dynamic libraries. An example of proxying is SOCKS. Figure 1,page 4.

Networks

To make a firewall effective the physical layout of the network must only allow one path between the hosts. The firewall host must be in the middle.

Commonly used network terms used. Figure 2,page 6.

- Segment. A physical part of the network where the hosts communicate using the same medium.
- Subnet. A logical grouping of hosts. May or may not be the same as a segment. UCCS is subnetted. IP address space: 128.198.x.y
- Router. The packets travel a route from the source to the destination. The packet is routed through the

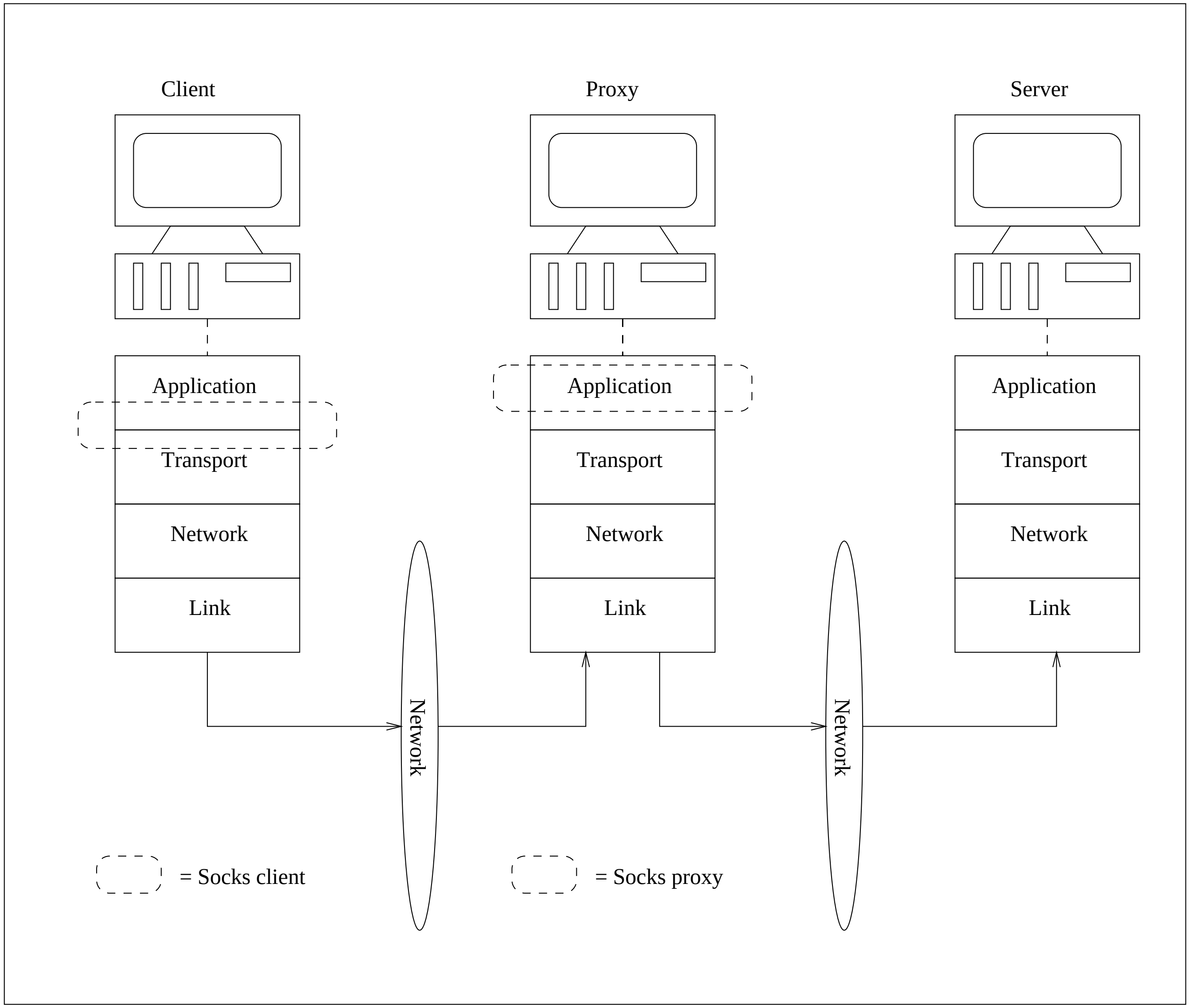

Figure 1: SOCKS at work

router. Most new routers have some kind of 'screening' of packets. They are called screening routers or packet filters. A screening router follows a set of rules which determine what will happen to the packet.

- Gateway. Same as router, but should be used when connecting two different protocol suites like TCP/IP and SNA.
- Repeater or Bridge. Repeat or Bridge packets between two segments. May be done only in hardware.
- Hub. Controls a couple of hosts using twisted pair ethernet. Is a combination of a router and an access mediator in the same box. Uses the 'star' topology.

Firewall architectures

There are three commonly used architectures:

- Dual homed host. Access to the internet is through a computer with two network devices. Not transparent to the user. Figure 3,page 7.
- Screened host. A screening router is setup to only allow packets to and from the Bastion host which provides proxying for the other hosts on the intranet. Figure 4,page 8.
- Perimeter network. An enhanced screened host setup using an additional screening router for the intranet. Figure 5,page 9.

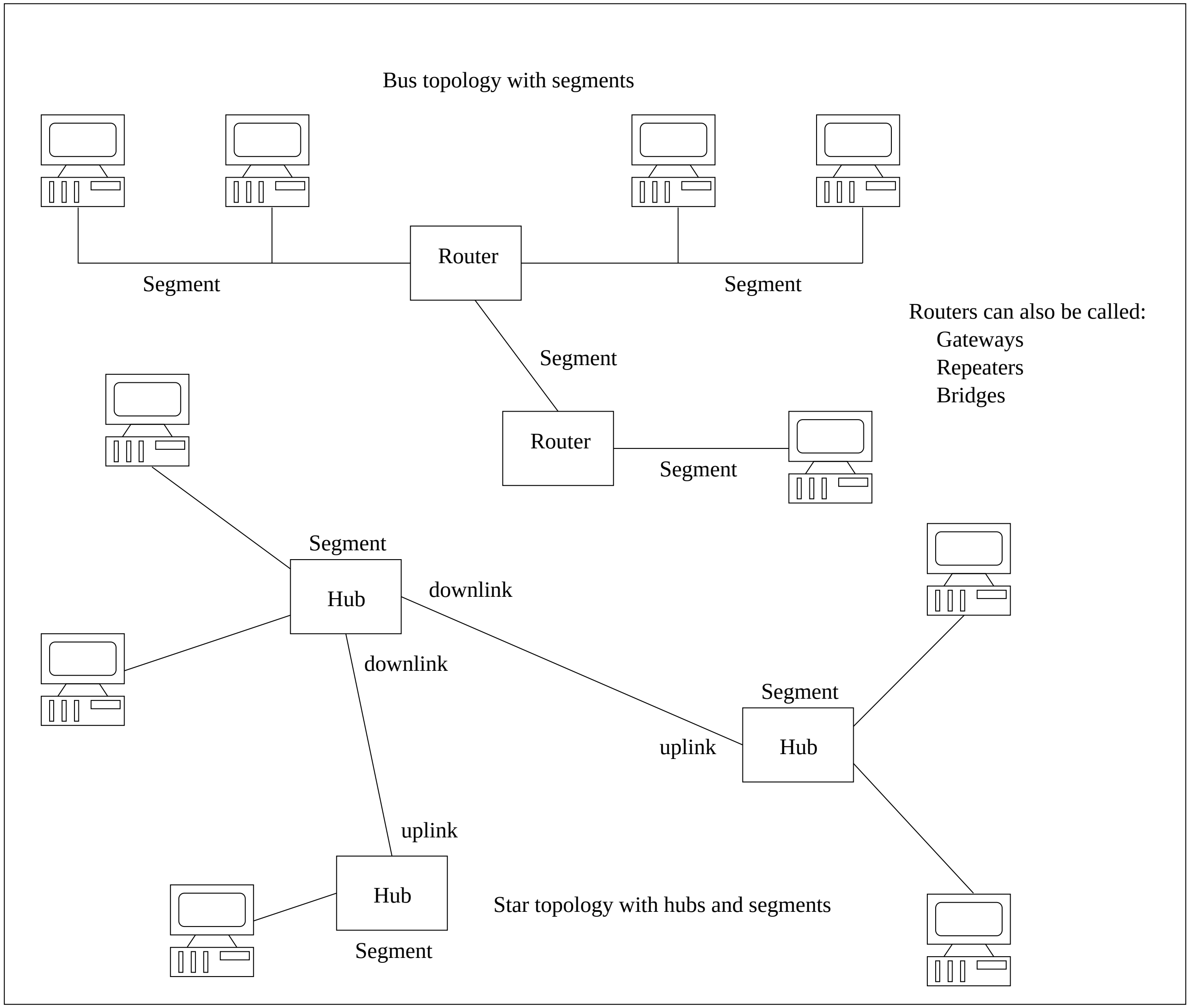

Figure 2: Network topologies

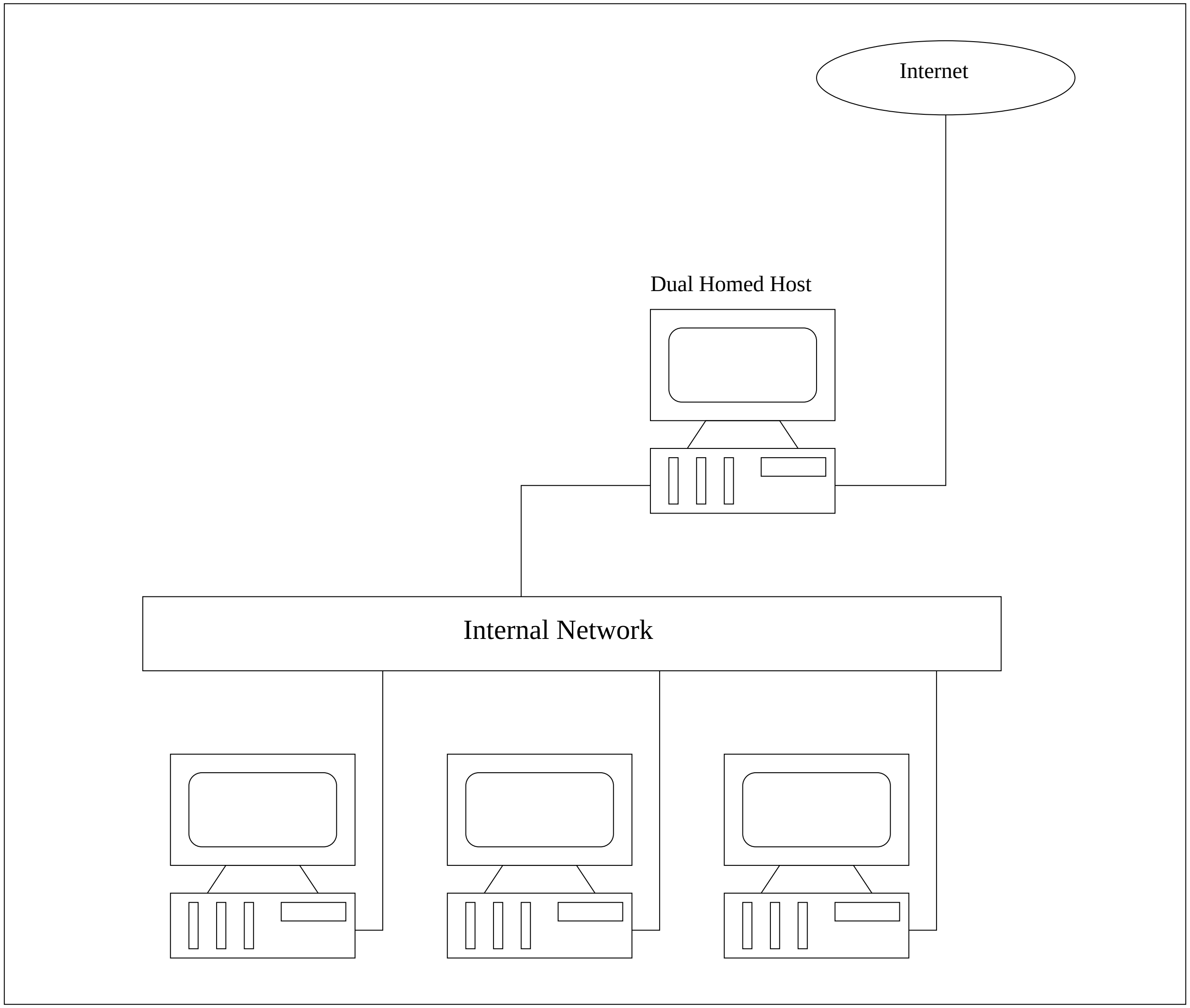

Figure 3: Dual Homed Host

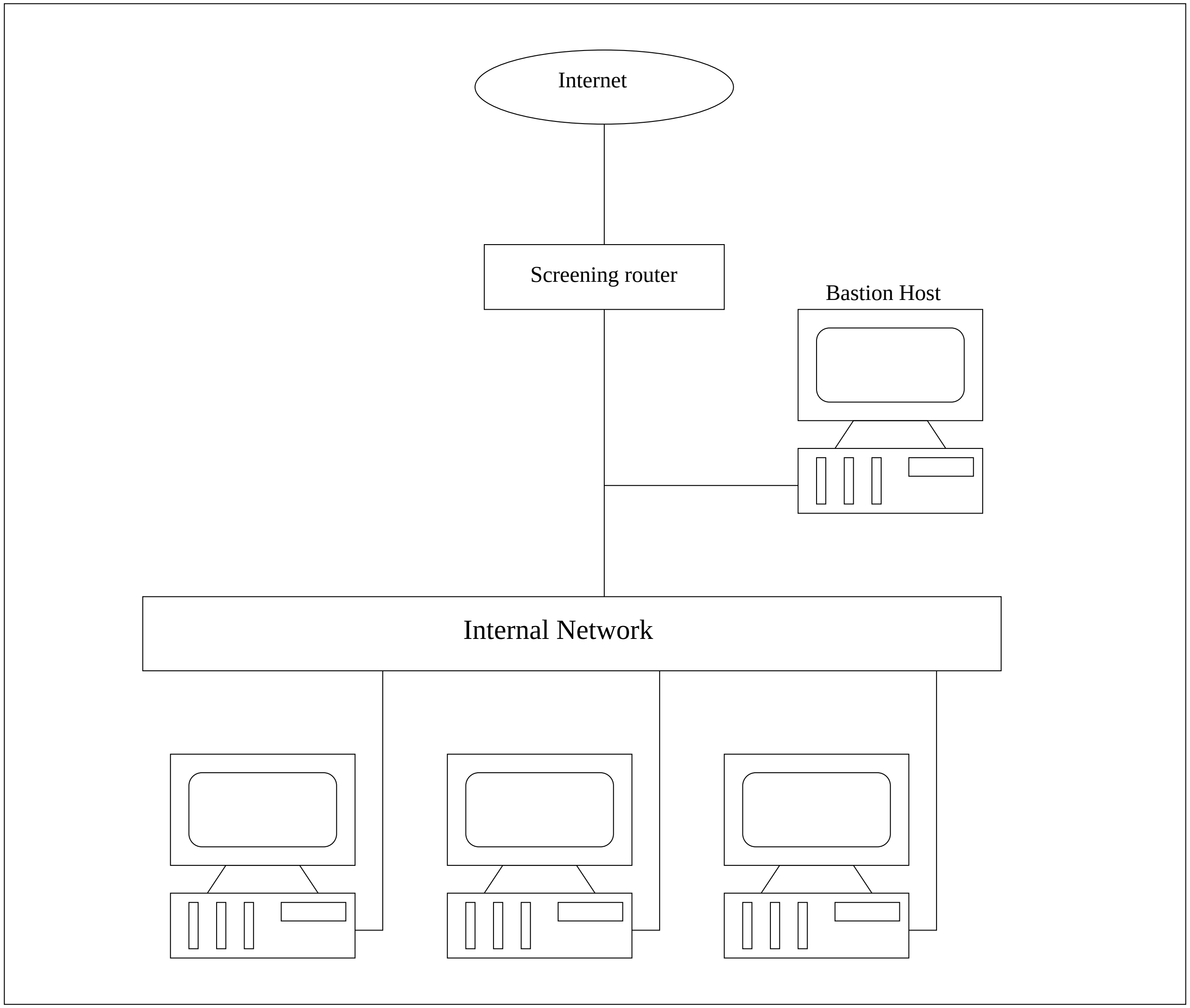

Figure 4: Screened Host

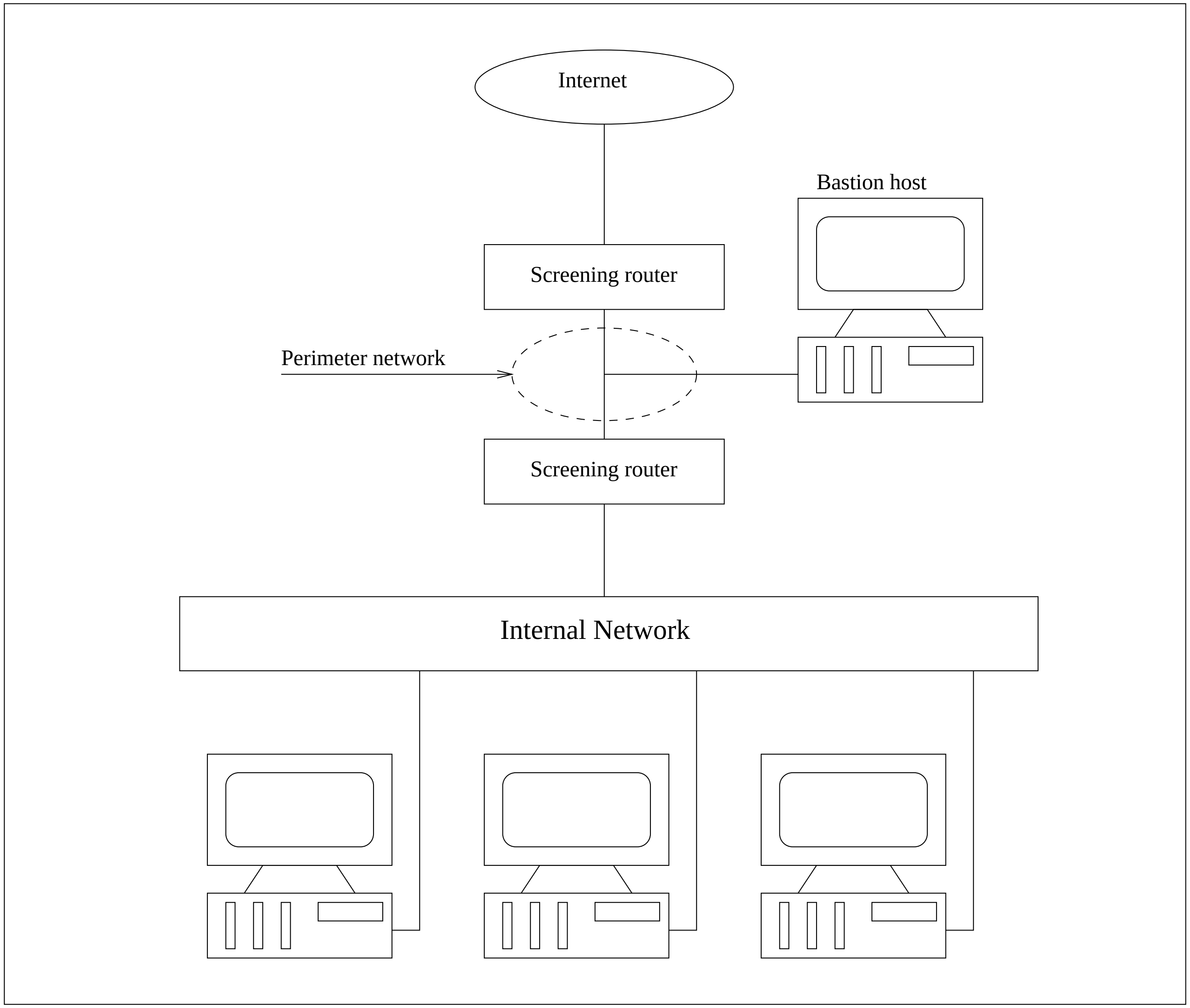

Figure 5: Perimeter network

Break-in techniques

Some of the most common break-in techniques are the following:

- Password cracking by guessing passwords in the encrypted password file.
- Exploiting bugs in existing software like the mail delivery system. Even if a login is not possible, make the mail delivery program to mail the password file to the intruder.
- IP spoofing. Use a trusted hosts IP address to get access.

A dynamic packet filter will make the system more difficult to penetrate. There is no protection against IP spoofing. A DPF does not either protect against 'fixed' data in packets.

Protocols

A firewall should work with the most commonly used network application protocols. Some of these are:

- Hyper Text Transfer Protocol (HTTP) is the basis for the world wide web which has had an incredible growth rate since January 1994.
- Network News Transfer Protocol (NNTP) is used to read the UseNet mailing groups.
- File Transfer Protocol (FTP) is used to transfer files.
- Telnet is used to remotely logon to hosts.

- Simple Mail Transfer Protocol (SMTP) is used for delivery of email.
- Domain Name System (DNS) is used for host to IP name resolution.

Linux

The dynamic packet filter was developed using Linux.

- The Linux kernel was created by Linus Torvalds, Finland.
- Linux runs currently on these processors:
  - i386
  - Sparc
  - Alpha
  - m68k
- The kernel source is included.
- Linux has all the common free tools for Unix.

The Linux kernel supports packet filters in the kernel. The PFs provided are the following:

- Incoming PF. All incoming packets are checked against the rules in the incoming PF chain.
- Forwarding PF. All packets that need to be forwarded are checked against these rules.
- Outgoing PF. All outgoing packets are checked against these rules.

- Accounting PF. All packets are checked against these rules. Increases a counter for each rule that matches a packet. Used for statistics.

Administration of the Linux packet filters

An example of administration using the *ipfwadm* program:
ipfwadm -I -a reject -P all -S badhost -D 0/0

Operation of the DPF

The idea behind a dynamic packet filter is to limit incoming traffic. If traffic is allowed, the hosts on the intranet should initiate the connections to hosts on the internet.

Setup

A straightforward setup is used to be able to confirm the function of the DPF. Figure 6,page 14.

Outgoing packets

Sniff for packets the DPF host sends and add outgoing packets in 'reverse' on Incoming PF. The packet has to pass the Outgoing PF first. Figure 7,page 15.

Forwarding packets

Sniff for packets that are to be forwarded between two or more network devices. The packet has to pass the Incoming PF first.

Figure 8,page 16.

DPF timeout values

The rule appended by the DPF will timeout after a predefined interval. The interval can be changed. The intervals are:

- TCP. 15 minutes, except when both the client and server has sent the FIN flag. In this case the timeout is 2 minutes.
- UDP. 5 minutes.
- ICMP. 2 minutes.

Test of DPF

The hosts must be carefully setup to use Gandalf as router. The initial setup can be tested by rejecting forwarding of packets.

Figure 6,page 14.

Telnet, FTP and NFS was tried with the DPF in operation to verify its function. The DPF should append rules in the according PF chains when a packet is sent or forwarded.

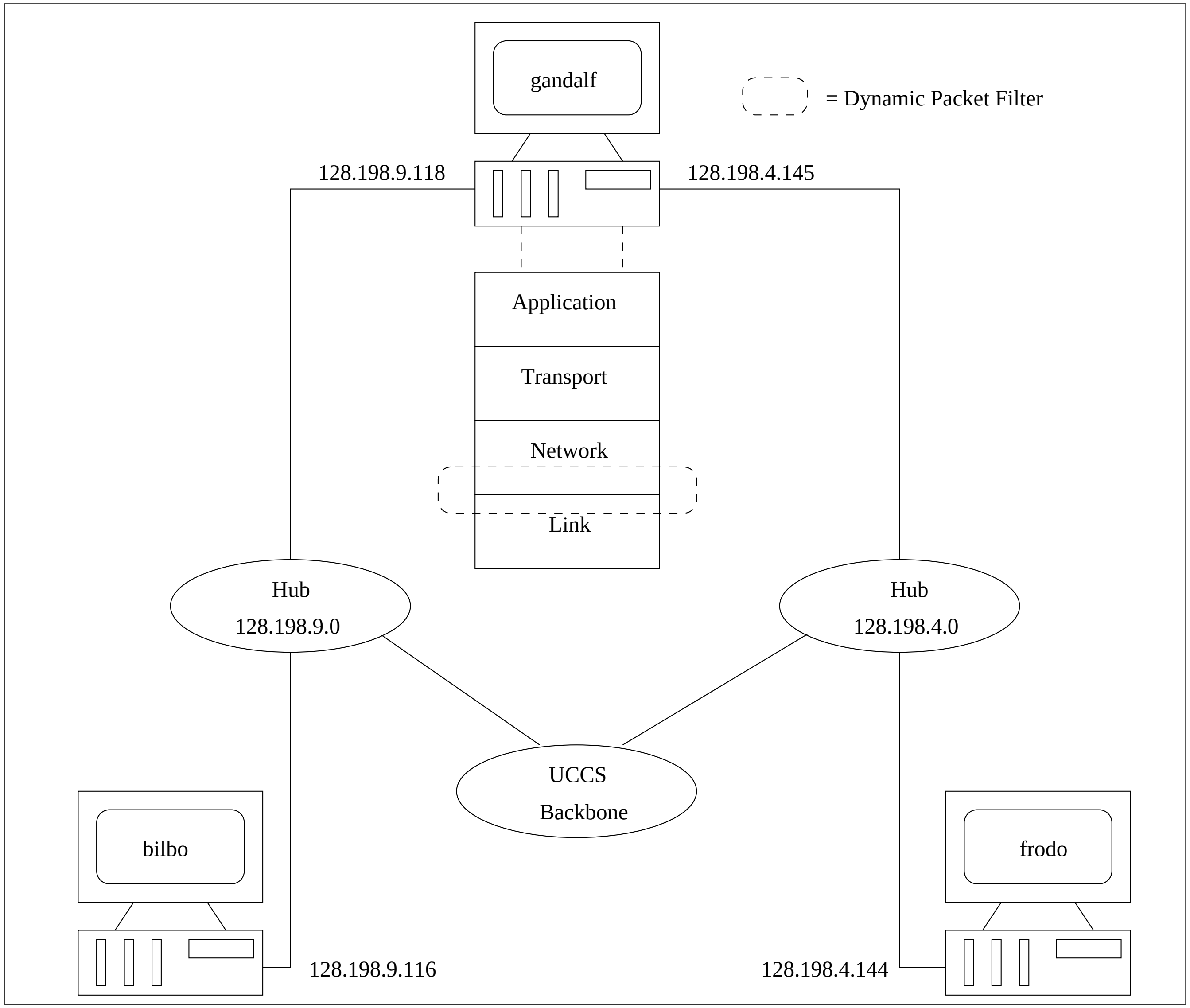

Figure 6: Setup used in tests

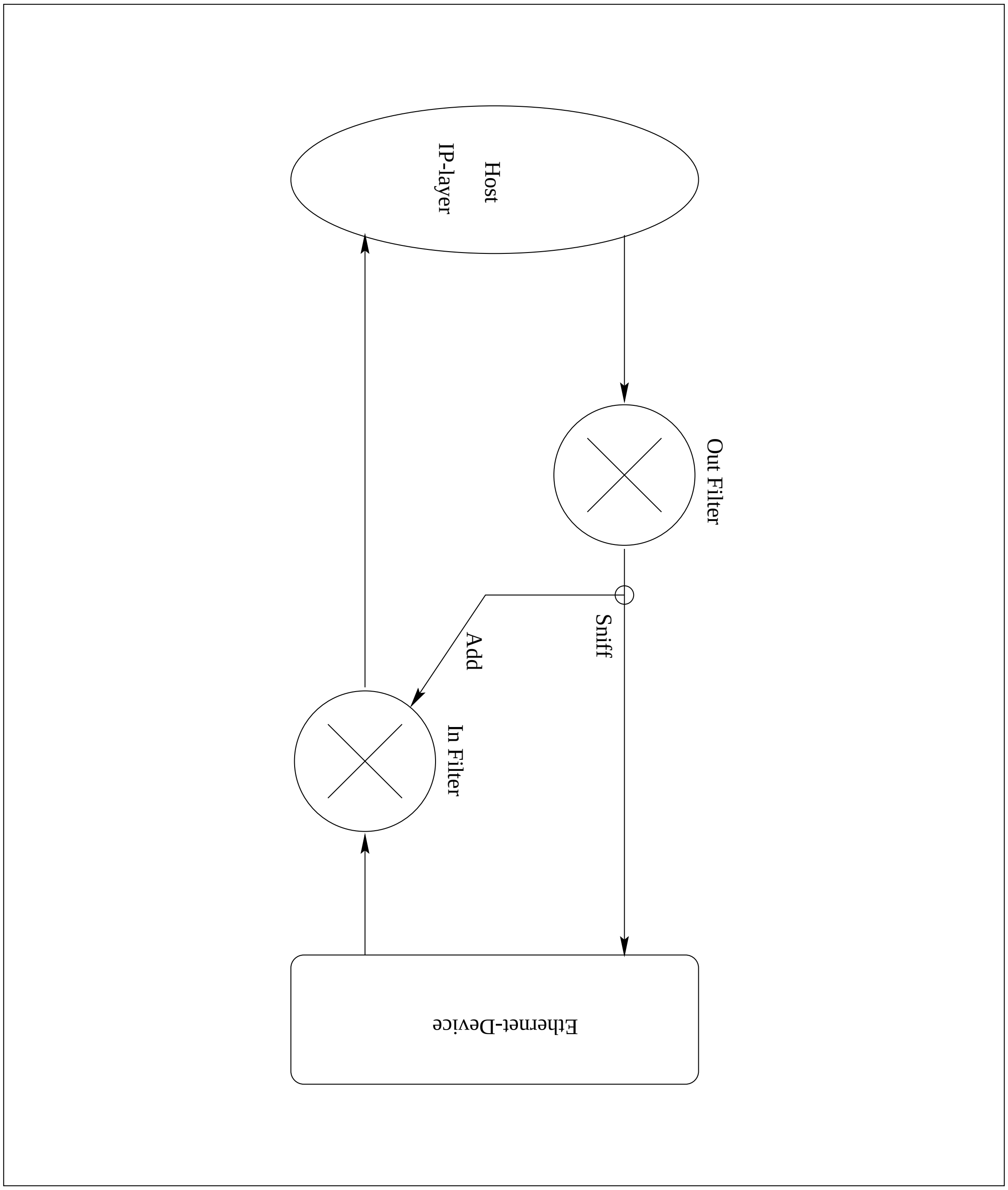

Figure 7: DPF processing of outgoing packets

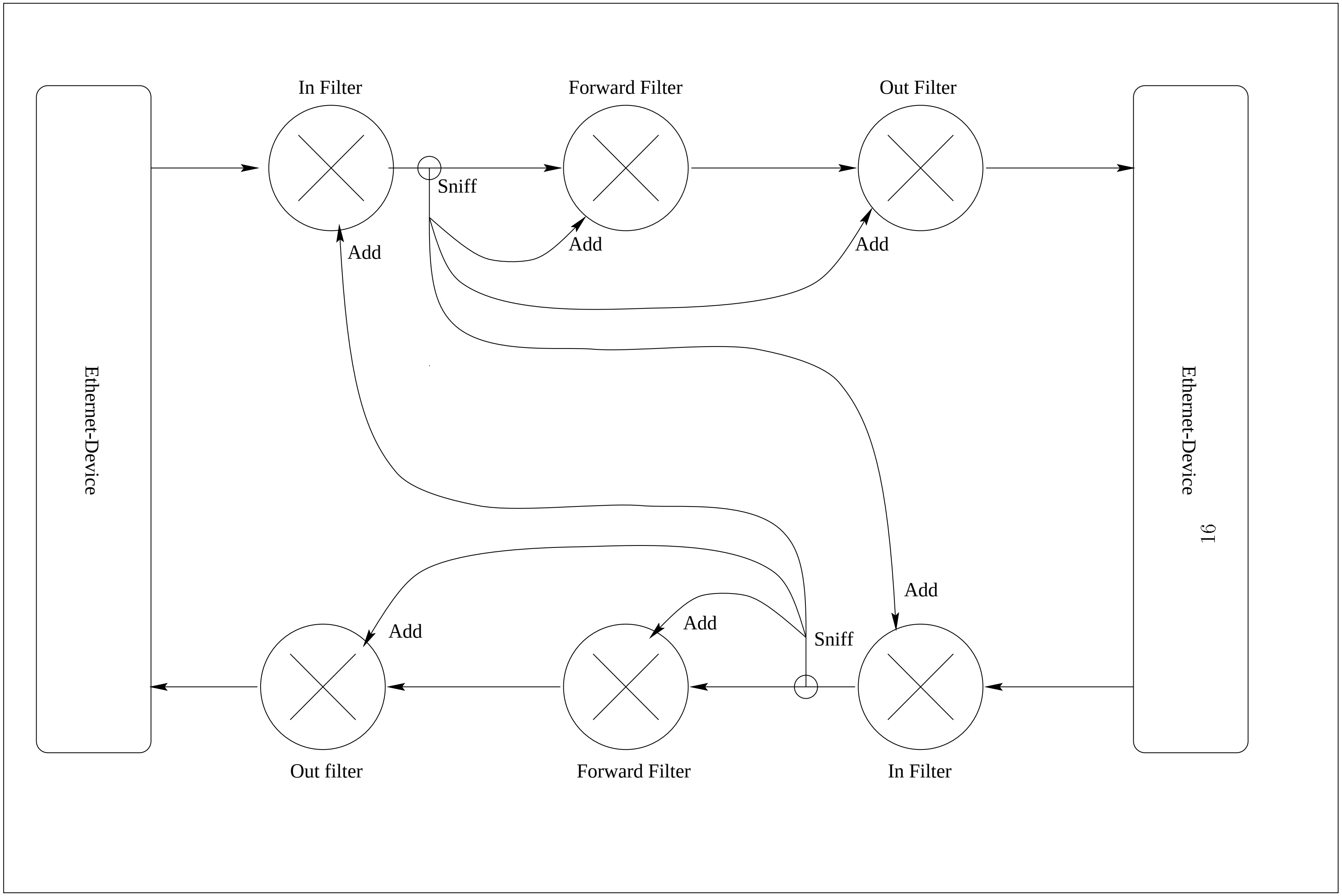

Figure 8: DPF processing of forwarded packets

FTP

The FTP application is ill suited for running behind the DPF as it is the server that initiates the network connection. There are two solutions:

1. Let the DPF scan for the PORT command sent to the server. Figure 9,page 19.
2. Compile a newer version of the FTP client which initiates the network connection. The server must support the PASV command.

FTP tcpdump trace

The *tcpdump* program has been extensively used to trace what happens on the ethernet. The example shows the server trying to connect to the client. It works since the DPF scanned for the PORT command.

```
  12:13:08.677104 odorf.uccs.edu.ftp-data > bilbo.uccs.edu.40001:
S 3245975086:3245975086(0) win 512 <mss 1460> [tos 0x8]
12:13:08.677104 bilbo.uccs.edu.40001 > odorf.uccs.edu.ftp-data:
S 256500895:256500895(0) ack 3245975087 win 15360 <mss 1460>
12:13:08.677104 odorf.uccs.edu.ftp-data > bilbo.uccs.edu.40001:
.  ack 1 win 15360 [tos 0x8]
12:13:08.677104 odorf.uccs.edu.ftp > bilbo.uccs.edu.1031:
P 204:257(53) ack 70 win 15360 (DF) [tos 0x10]
12:13:08.697105 bilbo.uccs.edu.1031 > odorf.uccs.edu.ftp:
.  ack 257 win 15360 [tos 0x10]
12:13:08.727105 odorf.uccs.edu.ftp-data > bilbo.uccs.edu.40001:
P 1:1025(1024) ack 1 win 15360 (DF) [tos 0x8]
12:13:08.727105 odorf.uccs.edu.ftp > bilbo.uccs.edu.1031:
P 257:281(24) ack 70 win 15360 (DF) [tos 0x10]
12:13:08.747105 bilbo.uccs.edu.1031 > odorf.uccs.edu.ftp:
.  ack 281 win 15360 [tos 0x10]
```

```
12:13:08.747105 bilbo.uccs.edu.40001 > odorf.uccs.edu.ftp-data:
.  ack 1025 win 15360 [tos 0x8]
12:13:08.747105 odorf.uccs.edu.ftp-data > bilbo.uccs.edu.40001:
P 1025:1265(240) ack 1 win 15360 (DF) [tos 0x8]
12:13:08.747105 odorf.uccs.edu.ftp-data > bilbo.uccs.edu.40001:
F 1265:1265(0) ack 1 win 15360 [tos 0x8]
12:13:08.747105 bilbo.uccs.edu.40001 > odorf.uccs.edu.ftp-data:
.  ack 1266 win 15360 [tos 0x8]
12:13:08.757105 bilbo.uccs.edu.40001 > odorf.uccs.edu.ftp-data:
F 1:1(0) ack 1266 win 15360 [tos 0x8]
12:13:08.757105 odorf.uccs.edu.ftp-data > bilbo.uccs.edu.40001:
.  ack 2 win 15360 [tos 0x8]
```

Performance

The DPF adds overhead to the packet processing so it is interesting to see how much. The benchmarks were done using the *netperf* program. A bulk transfer and a request response measurement was performed. As Bilbo and Frodo use a 100Mbps and a 10Mbps ethernet device it would be interesting to see the difference.

Three types of kernels where benchmarked:

1. Without the packet filters mentioned earlier.

2. With the packet filters.

3. With the DPF functionality.

As each step adds overhead there should be a performance loss.

Bulk transfer

The bulk transfer performance test sends packets to

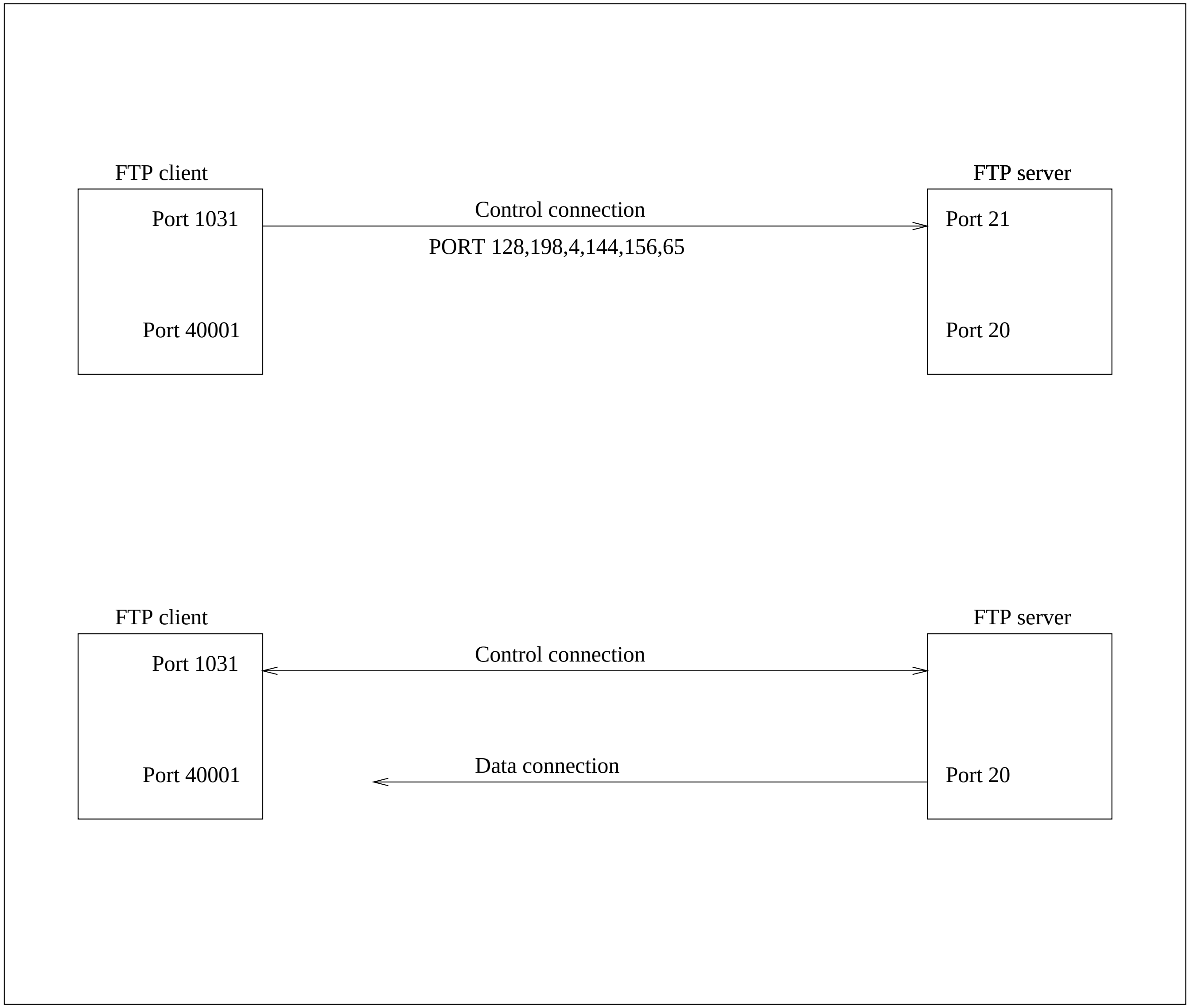

Figure 9: FTP port command. 156*256+65=40001

the DPF host as fast as possible. The DPF host running the netperf server counts the number of packets and returns this number to the netperf client. Table 3,page 25. Figure 10,page 21.

Request Response

The RR test sends like the bulk test a packet to the DPF host, but this time the netperf client wants a response. When the DPF host replies, the DPF will add an entry to the linked list. The next time a request is sent to the DPF host the entry is found quickly. Figure 11,page 22.

Conclusions and improvements

- The DPF will correctly used enhance the security for an intranet in one step. No software has to be replaced.
- Linux is well suited for development, but it would need some more documentation.
- The performance of the DPF could be improved by using two linked lists, one for outgoing packets and one for incoming ones.

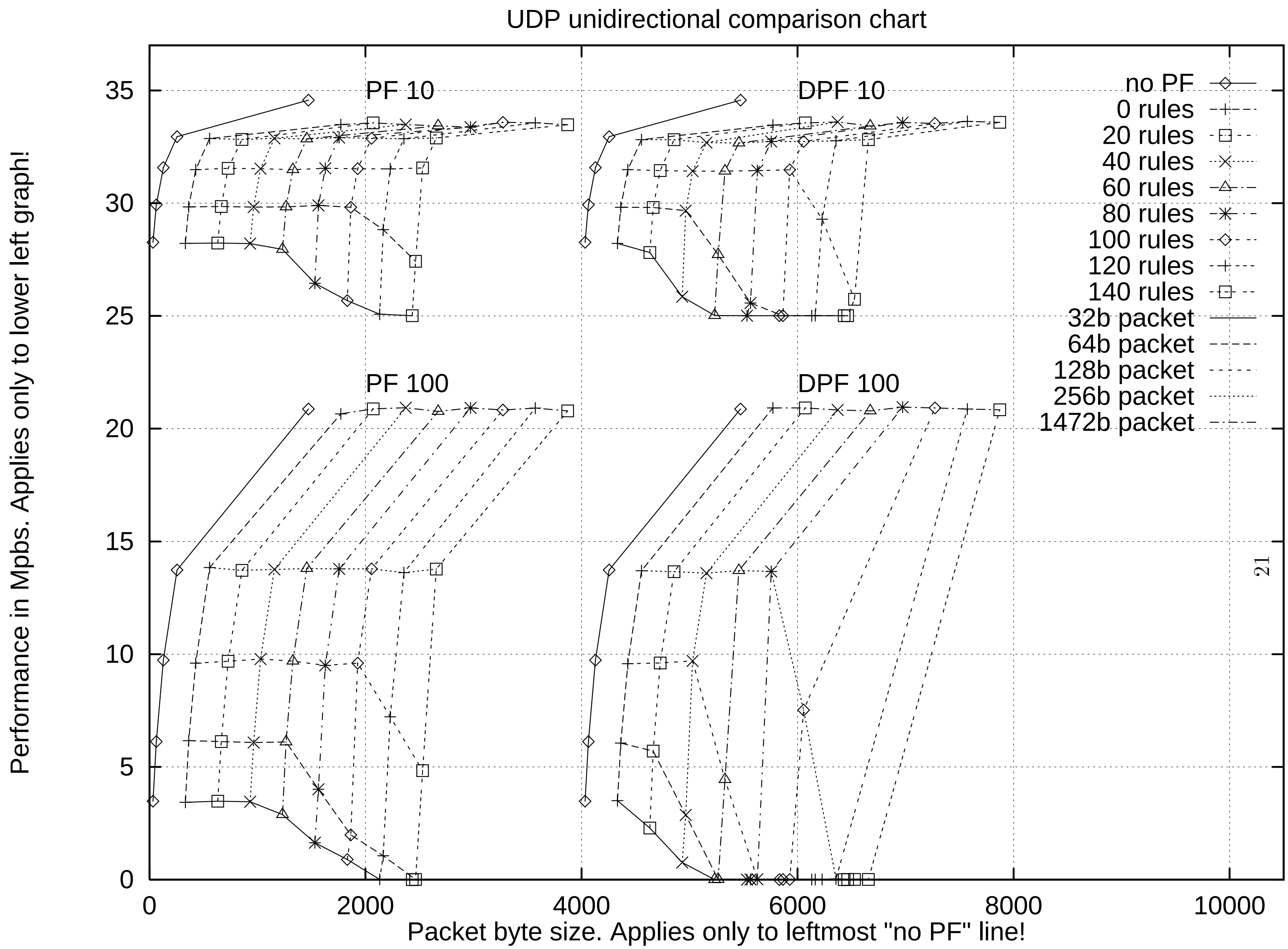

Figure 10: Bulk transfer comparison

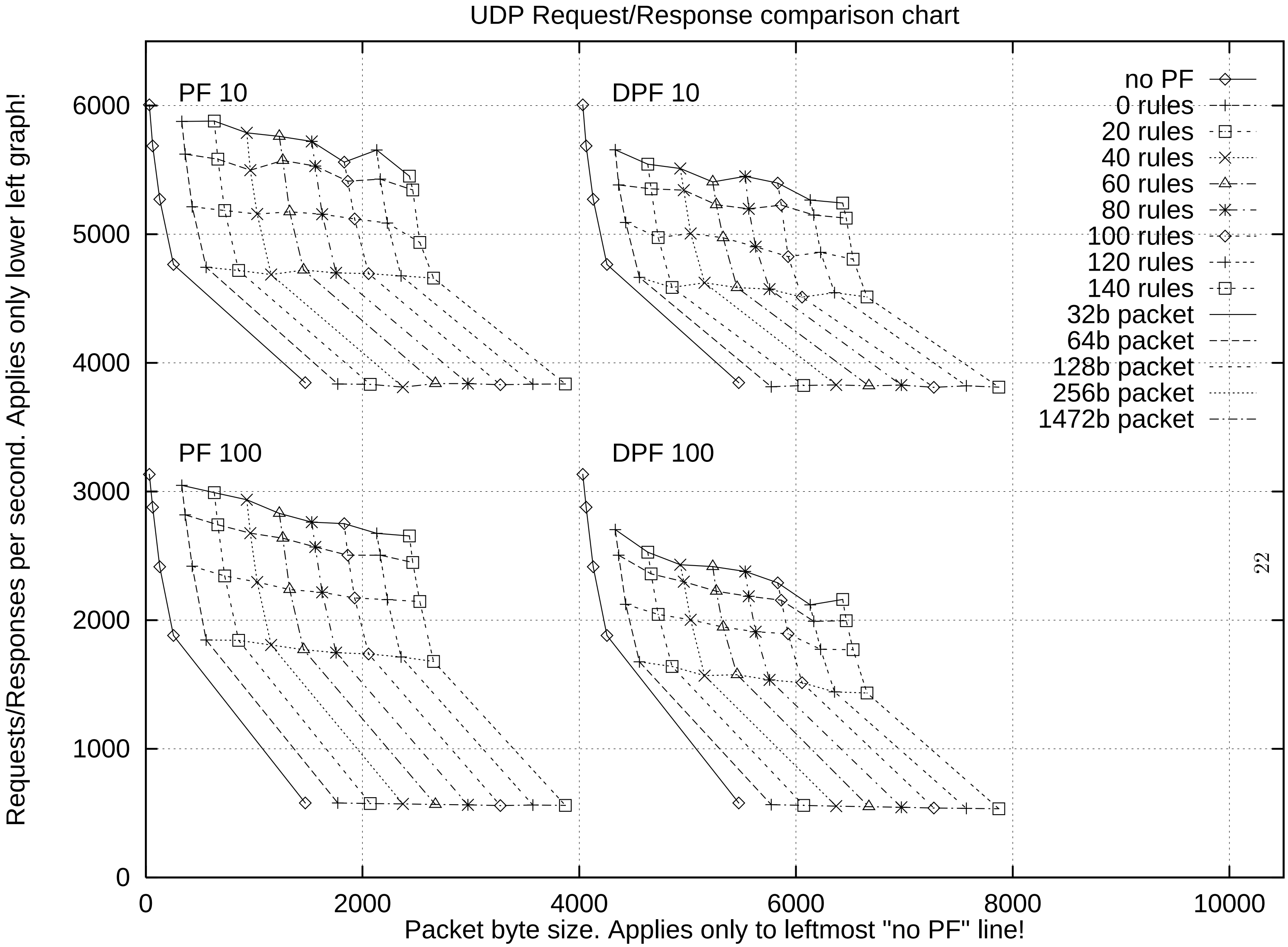

Figure 11: Request Response comparison

Extra

Material which might be used to answer questions, or if ample of time is left at the end of the presentation.

Hashing

Hashing is done to speed up matching of packets. The search for a packet in a linked list starts at the index calculated below.

```
struct ip_dyn *ip_dyn_get(struct iphdr *iph)
{
  __u16 icmp_ports[] = {0, 0};
  __u16 *portptr = icmp_ports;
  int protocol;
  __u32 s_addr, d_addr;
  __u16 s_port, d_port;
  struct ip_dyn *ret;

  protocol = iph->protocol;
  if (protocol != IPPROTO_ICMP)
          portptr = (__u16 *)&(((char *)iph)[iph->ihl*4]);

  s_addr = iph->saddr;
  s_port = portptr[0];
  d_addr = iph->daddr;
  d_port = portptr[1];

  /* Try first to find the 5-tuple as it is */
  ret = ip_dyn_get_2(protocol, s_addr, s_port, d_addr, d_port);
  if (ret != NULL) {
          ret->flags &= ~IP_DYN_F_REVERSED;
  }

  /* Now try to find the reversed ditto */
  else {
```

```
        ret = ip_dyn_get_2(protocol, d_addr, d_port, s_addr, s_port);
        if (ret != NULL)
                ret->flags |= IP_DYN_F_REVERSED;
  }
  return(ret);
}
```

To get the the hashed index, the ip_dyn_get_2() seen above will in turn call the following:

```
static __inline__
unsigned ip_dyn_hash_key(unsigned proto, __u32 addr, __u16 port)
{
  return (proto^ntohl(addr)^ntohs(port)) &
         (IP_DYN_TAB_SIZE-1);
}
```

Example: The netperf server is resident at port 12865 on Gandalf. 12865 is 0x3241 in hex. Gandalfs IP address is 128.198.9.118, which is 0x80c60976 in hex. The netperf uses UDP for the the benchmark.

NFS

NFS was one of the applications used when verifying the function of the DPF. Figure 12,page 27.

The following steps are done when mounting a NFS filesystem. Figure 13,page 28.

1. The *mountd* daemon registers itself on the server with the portmapper program which is started when the system boots.

2. The *mount* program is executed on the client to mount a remote server.

Class	*Range*	*Net Id bits*	*Host Id bits*
A	0.0.0.0 to 127.255.255.255	7	24
B	128.0.0.0 to 191.255.255.255	14	16
C	192.0.0.0 to 223.255.255.255	21	8
D	224.0.0.0 to 239.255.255.255	28	N/A
E	240.0.0.0 to 247.255.255.255	N/A	N/A

Table 1: Ranges of IP classes

Date	*HTTP*	*NNTP*	*FTP(data)*	*Telnet*	*SMTP*	*DNS*	*#Packets x10^9*
1994 Jan.	1.5%	8.8%	21.4%	15.4%	7.4%	5.8%	55
1994 Apr.	2.8%	9.0%	20.0%	13.2%	8.4%	5.0%	71
1994 Jul.	4.5%	10.6%	19.8%	13.9%	7.5%	5.3%	74
1994 Oct.	7.0%	9.8%	19.7%	12.6%	8.1%	5.3%	100
1995 Jan.	13.1%	10.0%	18.8%	10.4%	7.4%	5.4%	87
1995 Apr.	21.4%	8.1%	14.0%	7.5%	6.4%	5.4%	59

Table 2: Packet count for protocols on NSFnet backbone

Variable	*Value*	*Operation*	*Result*
IP	0x7609c680	assign	0x7609c680
Port	0x4132	assign	0x4132
Proto	0x00000006	assign	0x00000006
SIZE	0xff	assign	0xff
Proto	0x00000006	xor	0x00000006
ntohl(IP)	0x80c60976	xor	0x80c60970
ntohs(Port)	0x00003241	xor	0x80c60b31
SIZE	0x000000ff	and	0x00000031
Hash		assign	0x00000031

Table 3: Hash calculation

3. The portmapper returns the port of the mount daemon.
4. The mount program sends a request to the server that it wants to mount some parts of the server's file-system.
5. The mount daemon sends back a file-handle to the file-system in question.
6. The file-handle is passed to the NFS portion of the client kernel.

Detailed graphs of all the tests

The graphs are the following:

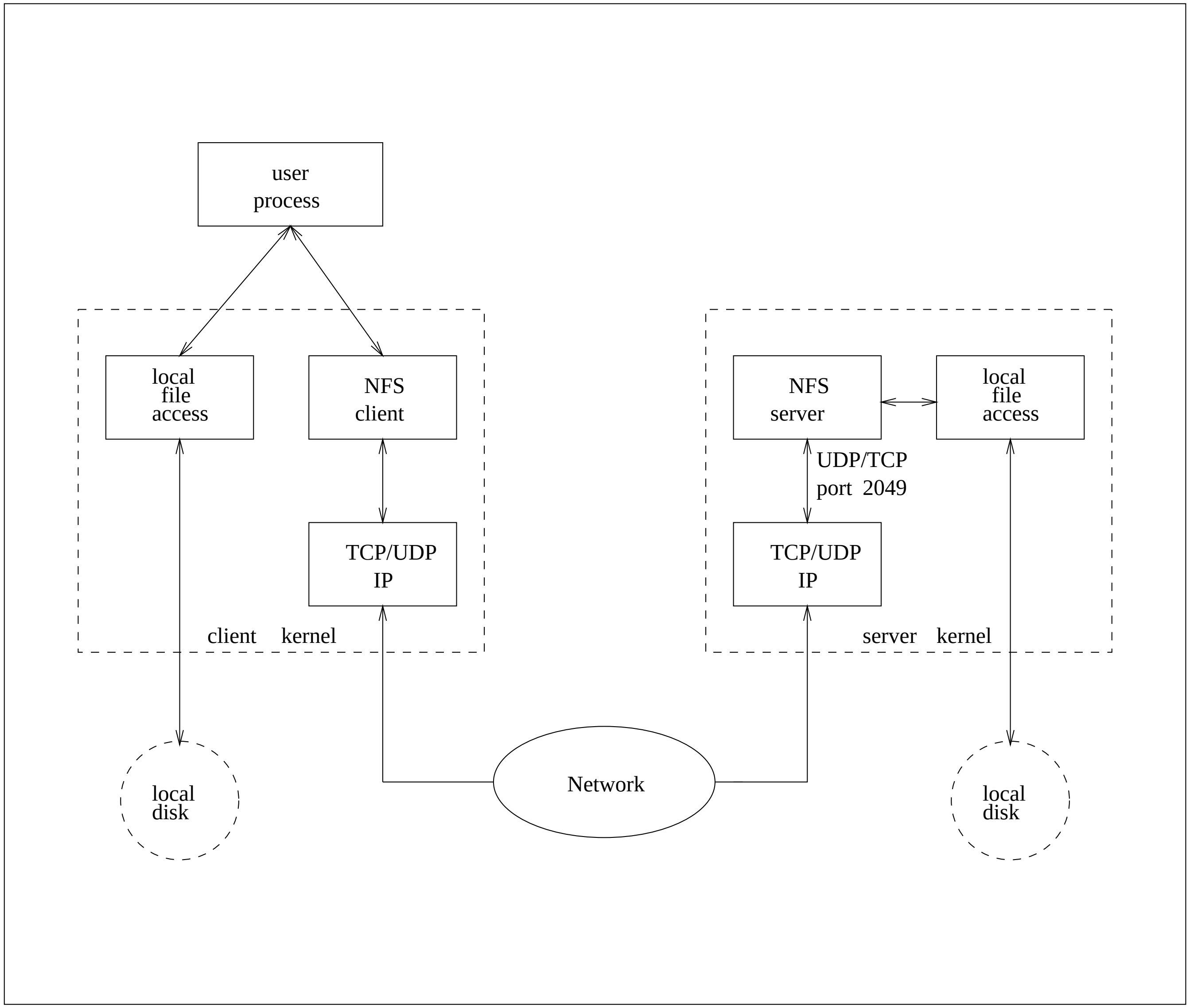

Figure 12: NFS client and server

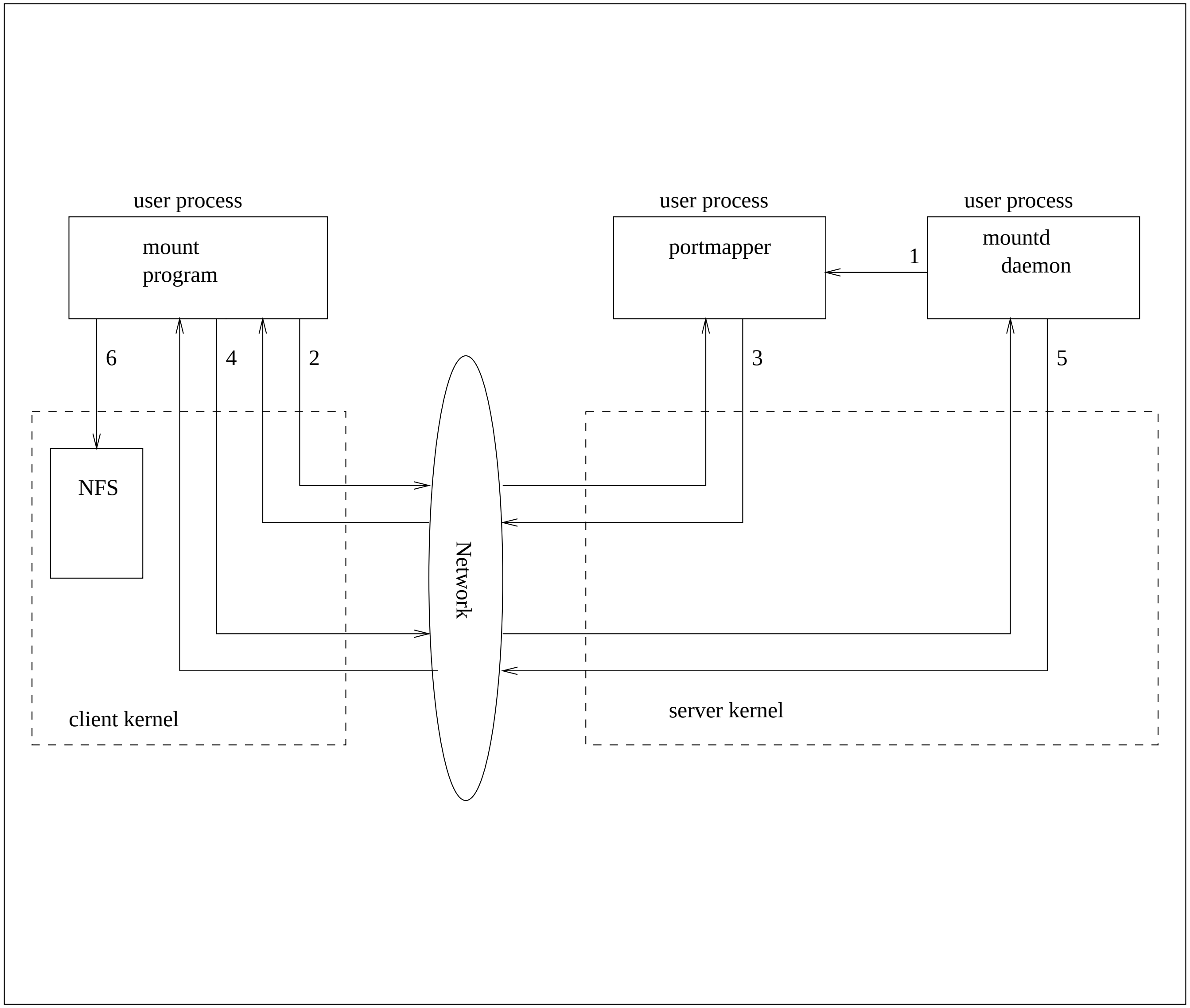

Figure 13: Steps taken to mount a remote file-system

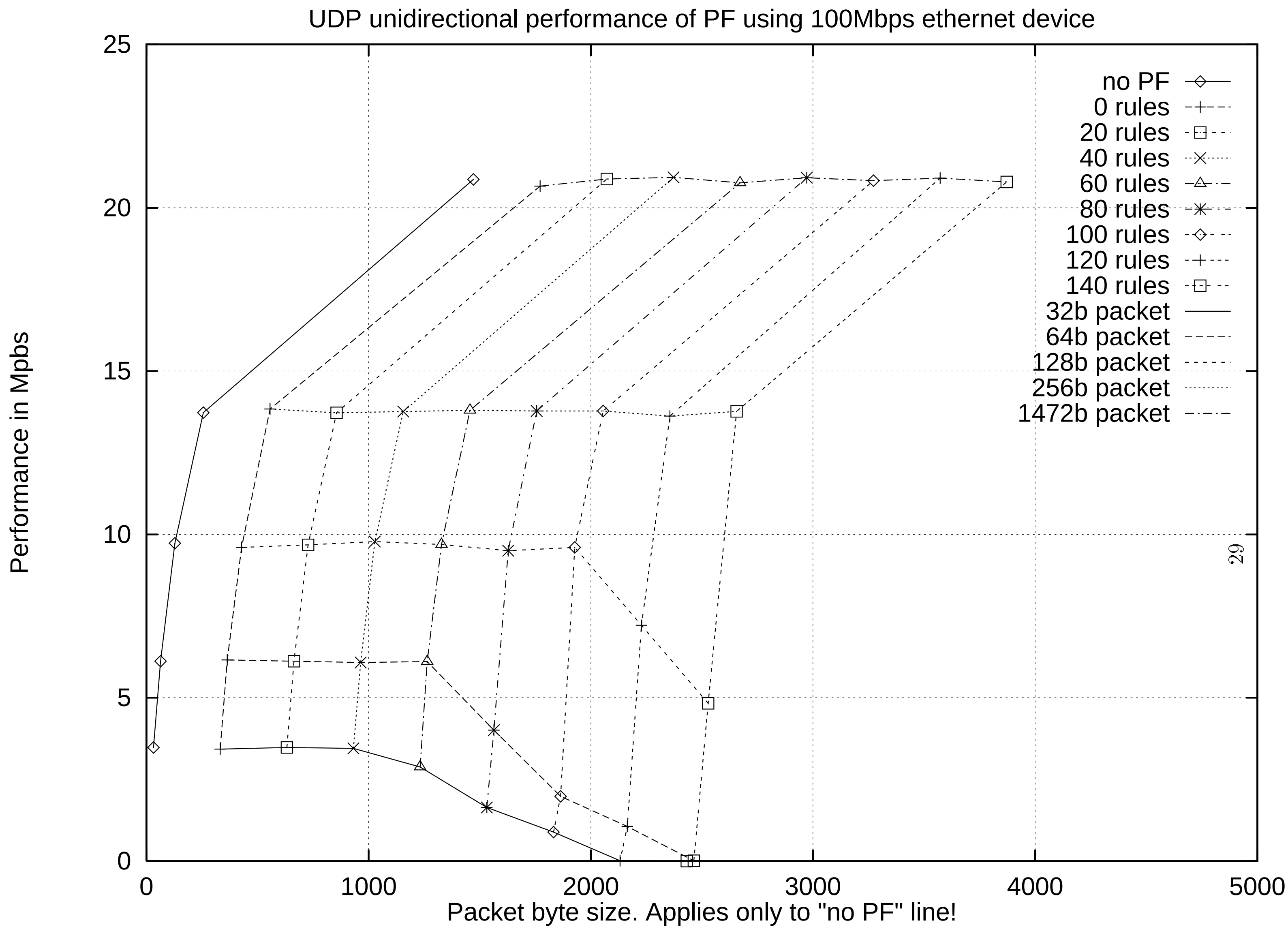

Figure 14: 100Mbps PF results.

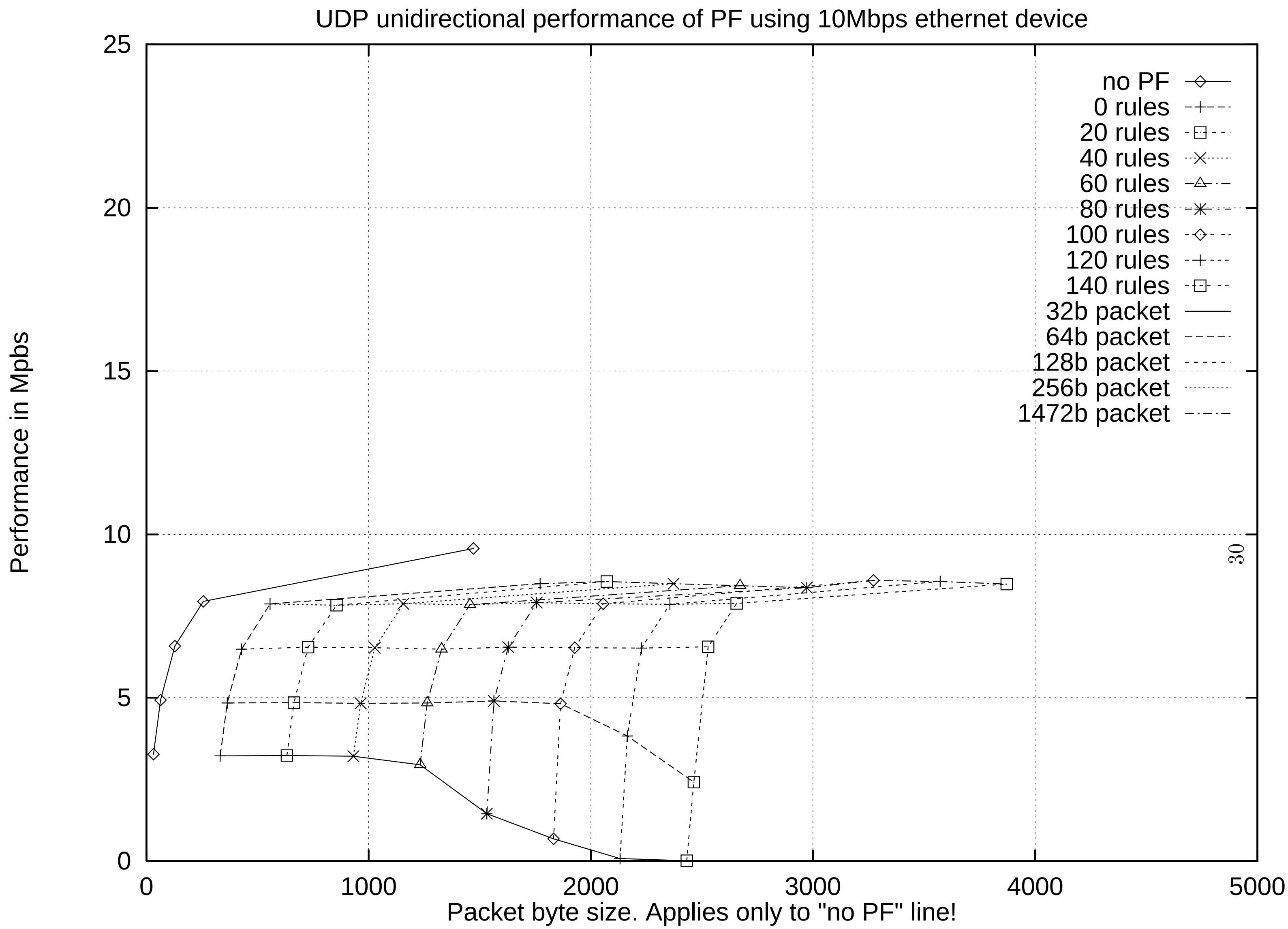

Figure 15: 10Mbps PF results.

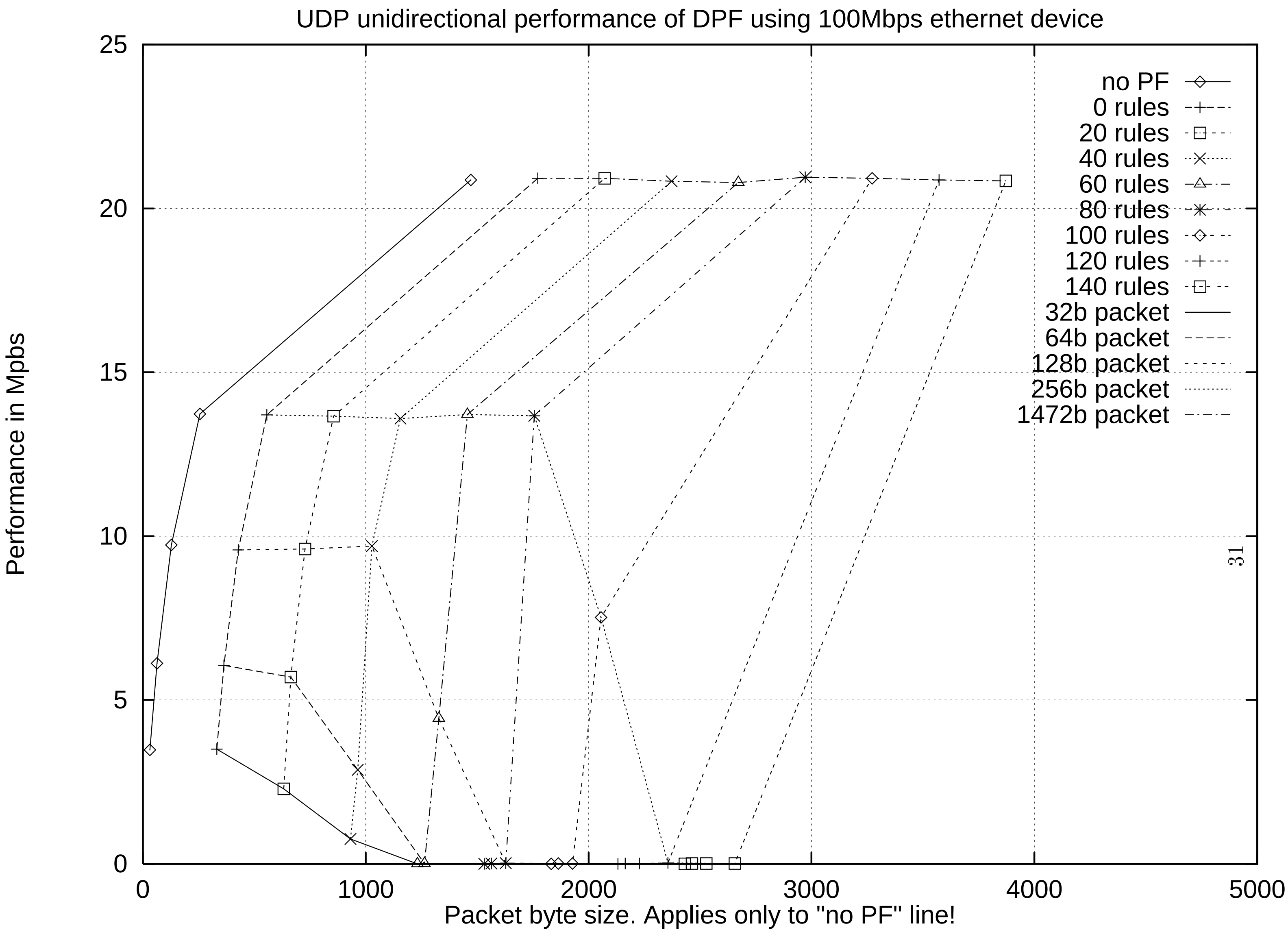

Figure 16: 100Mbps DPF results.

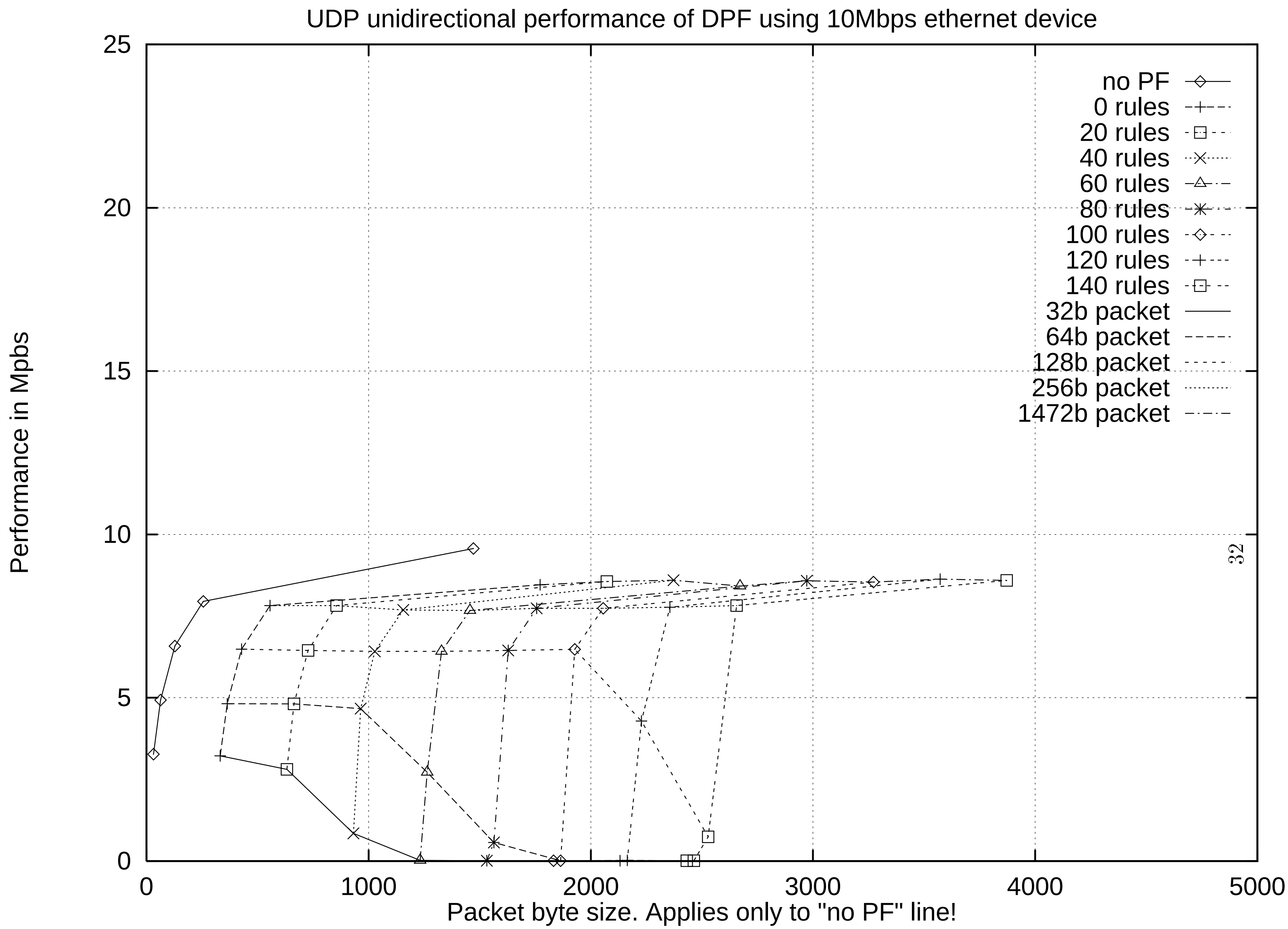

Figure 17: 10Mbps DPF results.

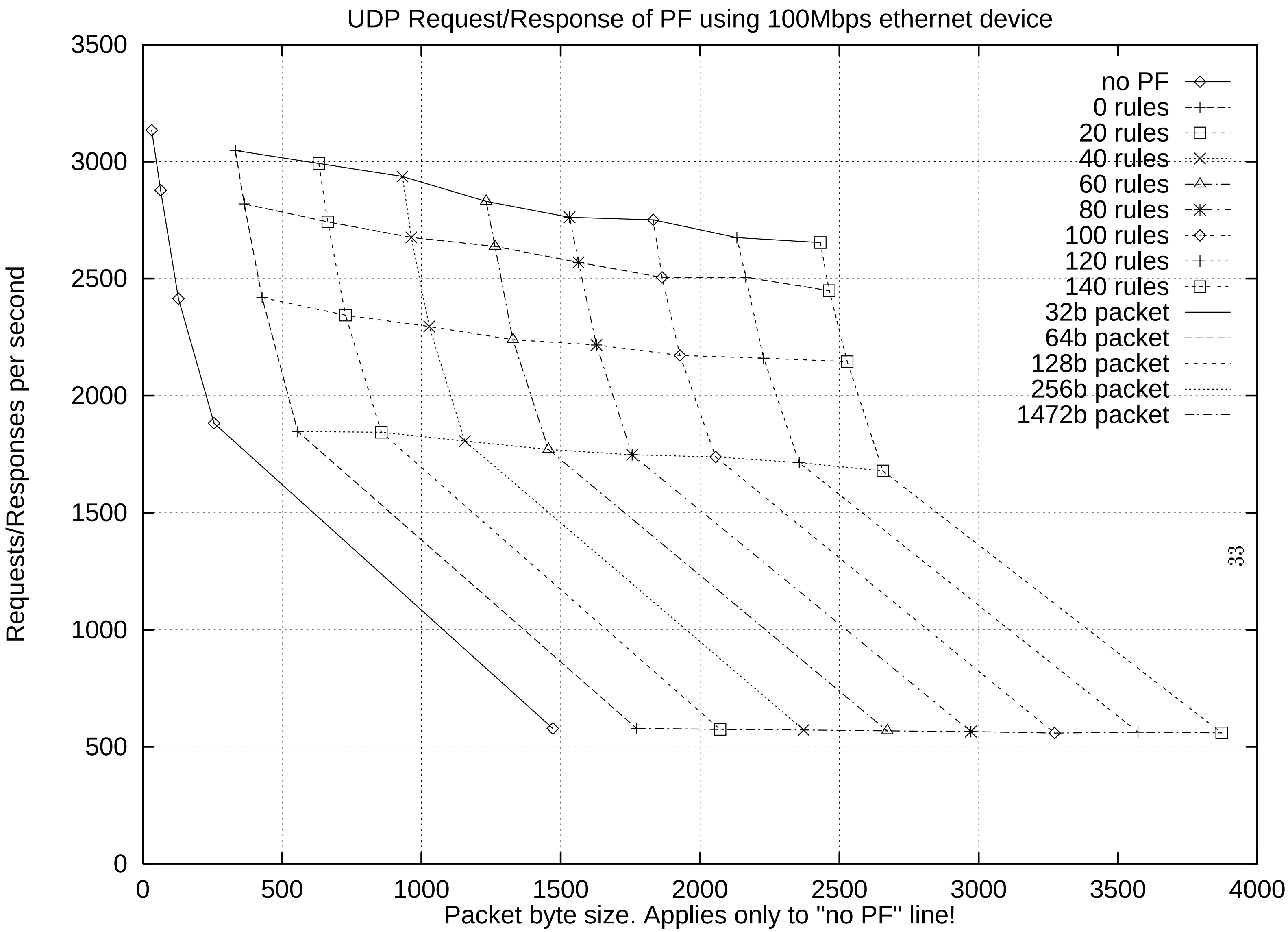

Figure 18: 100Mbps PF RR results.

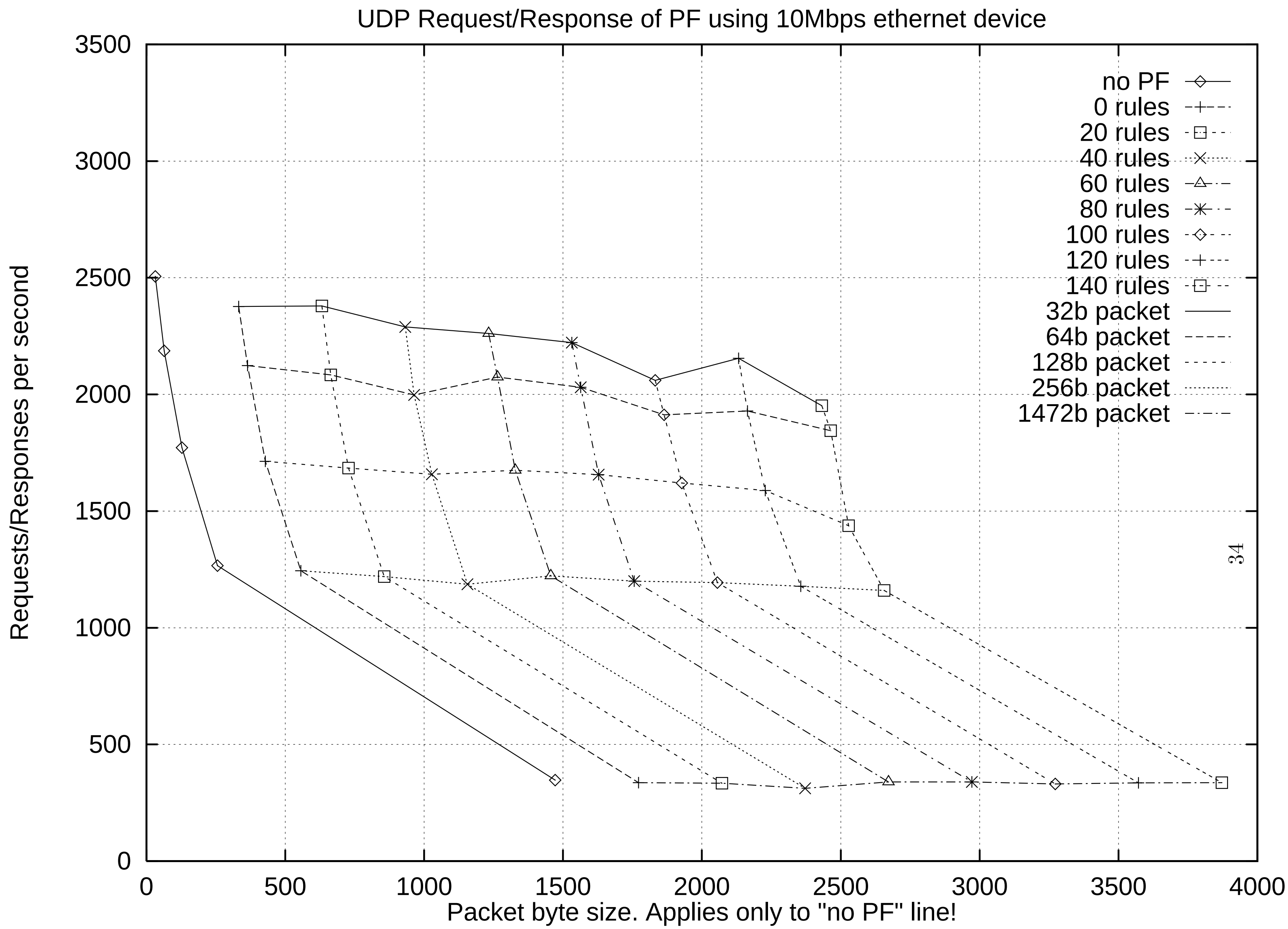

Figure 19: 10Mbps PF RR results.

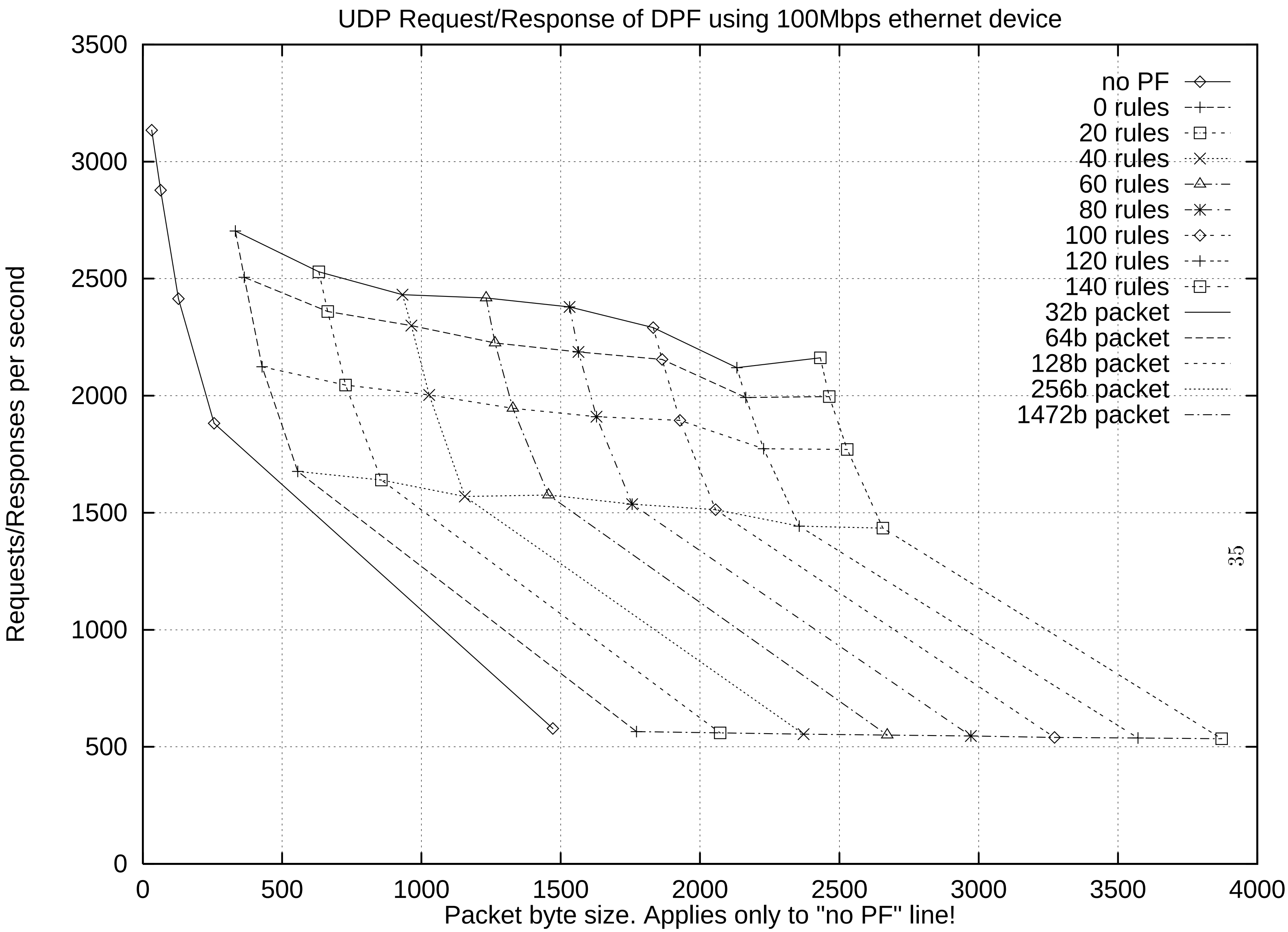

Figure 20: 100Mbps DPF RR results.

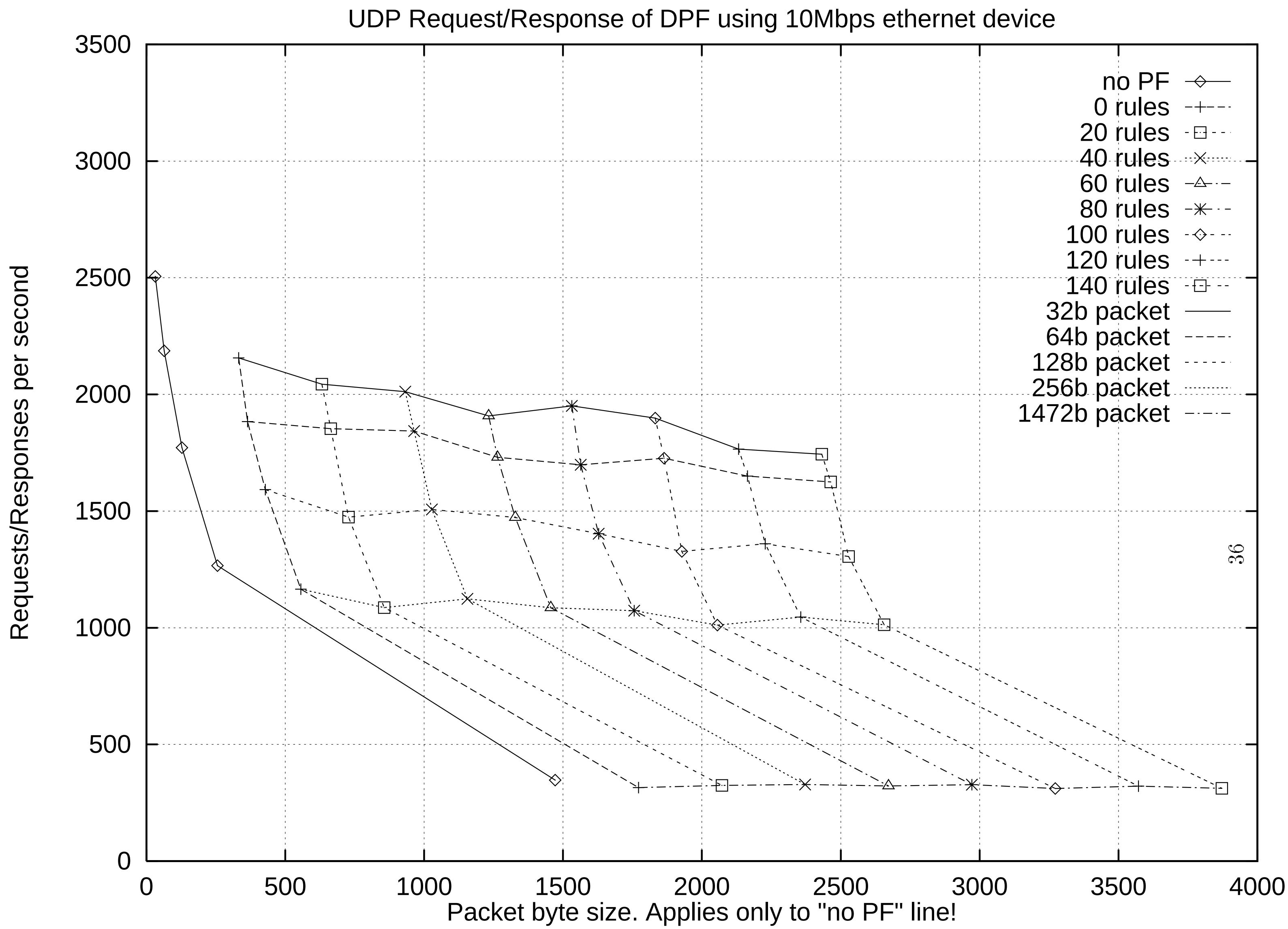

Figure 21: 10Mbps DPF RR results.