Scamper

http://www.wand.net.nz/scamper/

Matthew Luckie
mjl@wand.net.nz

Introduction

- It is coming up towards the end of a year’s contract between the University of Waikato and WIDE that funded the development of scamper
  - 1 April 2004 – 31 March 2005

- This talk describes the core areas of scamper’s progress over the past year

Introduction

- Expected Results (Contracted)
- Other inputs
- Core Areas of Work / Results
- Conclusions
- Collaboration Items
- Future Work

Expected Results (Contracted)

- Development of an open-source topology probe tool including implementations of
  - The skitter compatible output format
  - PMTUD functionality
  - Performance optimisation
  - Scamper-library functions to read the existing skitter arts files
  - Updated **`sdcollect`** and **`sdserver`** using the new scamper library

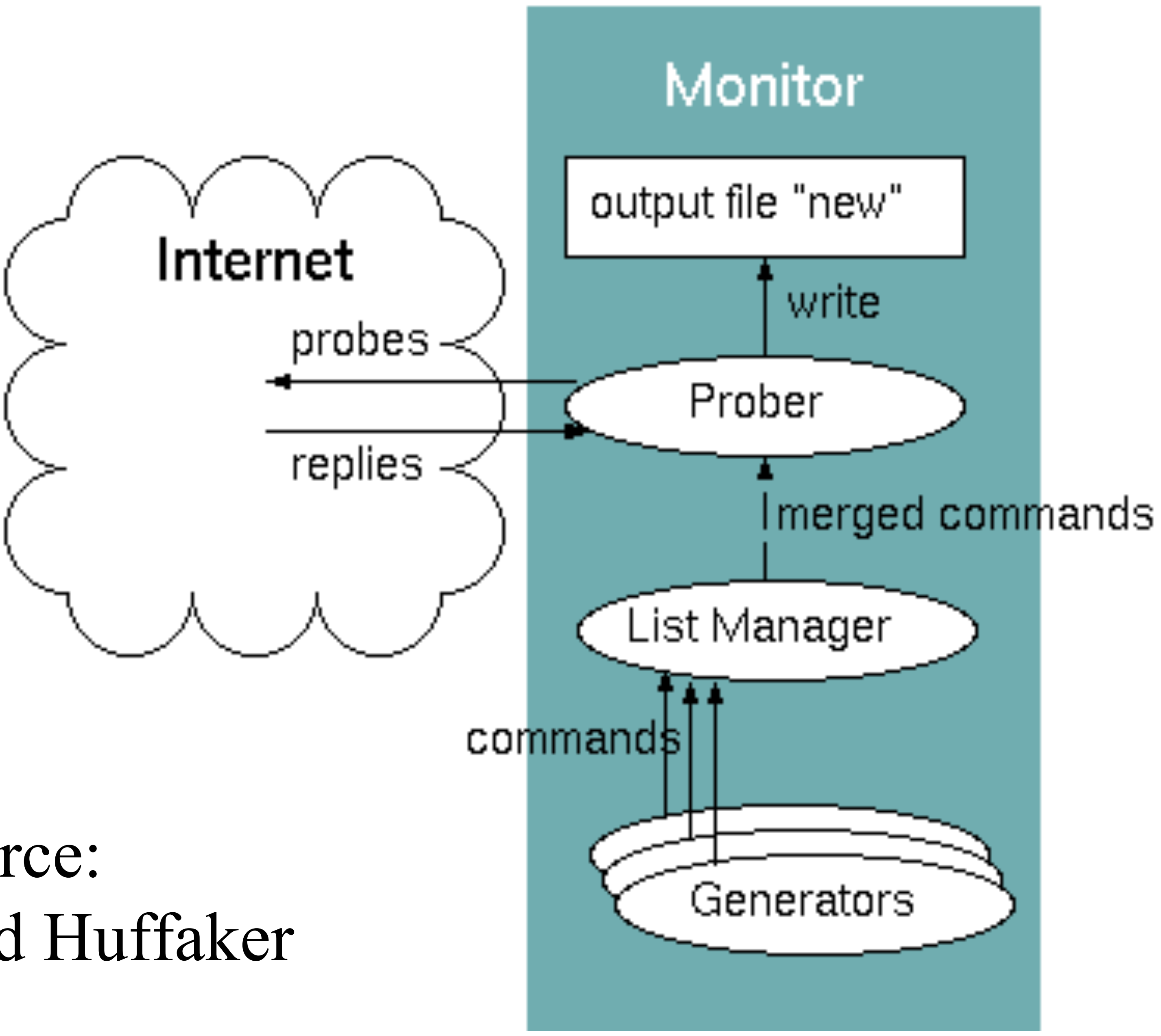
Expected Results (Contracted)

- Large scale IPv6 topology measurement using scamper, and analysis of the obtained data

Other Inputs

- Brad Huffaker et al (CAIDA)
  - Probing of the network should be as unintrusive as possible.
  - scamper should be able to interleave and concurrently probe different lists of destinations
  - The destination lists can overlap, but at any moment of time there should be no more than one instance of a given IP address in the currently probed set of IPs.
  - Scamper should probe lists in cycles

scamper Data Collect



Source:
Brad Huffaker

Other Inputs

- Mark Crovella via kc:
  - Support "some measurement technique" – more than just traceroute
  - The ability to connect to 3rd party scamper processes and use them for measurement
- Young Hyun (CAIDA)
  - Allow more than one method of traceroute probing (more than UDP to high numbered ports)

Other Inputs

- David Moore (CAIDA)
  - Use BPF to get transmit timestamps from datalink
- Andre Broido (CAIDA)
  - Send probes with arbitrary content

Core areas of work

- File format / data API
- Process control
- Path MTU Discovery
- Privilege Separation
- Datalink-provided Transmit Timestamps
- Addition of more traceroute probe methods
- Addition of arbitrary measurement tasks
- Portability

File format / data API

- Arts (++) is fairly convoluted for traceroute storage and access requirements, and doesn’t speak IPv6
- Design a new file format and API to store traceroute data that is extensible, but that is not needlessly complex

File format

```
scamper_file_t *scamper_file_open(char *fn, char
  mode, char *type);

void scamper_file_close(scamper_file_t *sf);

scamper_trace_t
  *scamper_file_read_trace(scamper_file_t *sf);

int scamper_file_write_trace(scamper_file_t *sf,
  scamper_trace_t *trace);
```

Trace Format

```
typedef struct scamper_trace
{
  scamper_list_t  *list;
  scamper_cycle_t *cycle;

  scamper_addr_t  *src;
  scamper_addr_t  *dst;

  struct timeval   start;
```

Trace Format

```
scamper_hop_t  **hops;
uint8_t          hop_count;

uint8_t          stop_reason;
uint8_t          stop_data;

scamper_pmtu_t  *pmtu;
```

Trace Format

```
  /* trace parameters */
  uint8_t            type;
  uint8_t            flags;
  uint8_t            attempts;
  uint8_t            hoplimit;
  uint16_t           size;
  uint16_t           sport;
  uint16_t           dport;
} scamper_trace_t;
```

Hop Format

```
typedef struct scamper_hop
{
  scamper_addr_t     *addr;
  uint8_t             flags;
  uint8_t             probe_id;
  uint8_t             probe_ttl;
  uint16_t            probe_size;
  uint16_t            reply_size;
  int16_t             reply_ttl;
```

Hop Format

```
  uint8_t                 icmp_type;
  uint8_t                 icmp_code;

  struct timeval          rtt;

  scamper_tlv_t          *tlvs;
  struct scamper_hop     *next;

} scamper_hop_t;
```

Process Control

- Scamper began as a command line tool that made its way through an address list doing traceroute to each address
  - Once it has started, you have to wait until it finishes
  - Can’t change output files midway through a run

Process Control

- Scamper's approach to process control is a localhost socket
  - Goal to eventually have some authentication code to enable remote control and monitoring of scamper processes
  - But also need to define how data might be returned over a control socket

Process Control

- get [attempts | dport | hoplimit | holdtime | pps | sport | timeout | version]
- set [attempts | holdtime | hoplimit | pps | timeout]
- help
- exit

Process Control

- shutdown [done | flush | now | cancel]
- source [add | cycle | delete | list]
- outfile [open | close | list | swap]
- traceroute [source <name>] addr

Process Control

- Source add

```
[name <name>]
[descr <descr>]
[id <id>]
[file <name>]
[priority <priority>]
[adhoc <on|off>]
[outfile <name>]
[cycle <on|off>]
[autoreload
<on|off>]
```

Path MTU Discovery

- Conducted after traceroute phase so MTU changes can be signaled in the traceroute output
- Original goal was to help find and characterise IPv6-in-IPv4 tunnels
  - Tunnels restrict the MTU available, so infer tunnels with PMTUD
- Now a fairly useful operational tool for debugging PMTUD faults on the forward path

Path MTU Discovery

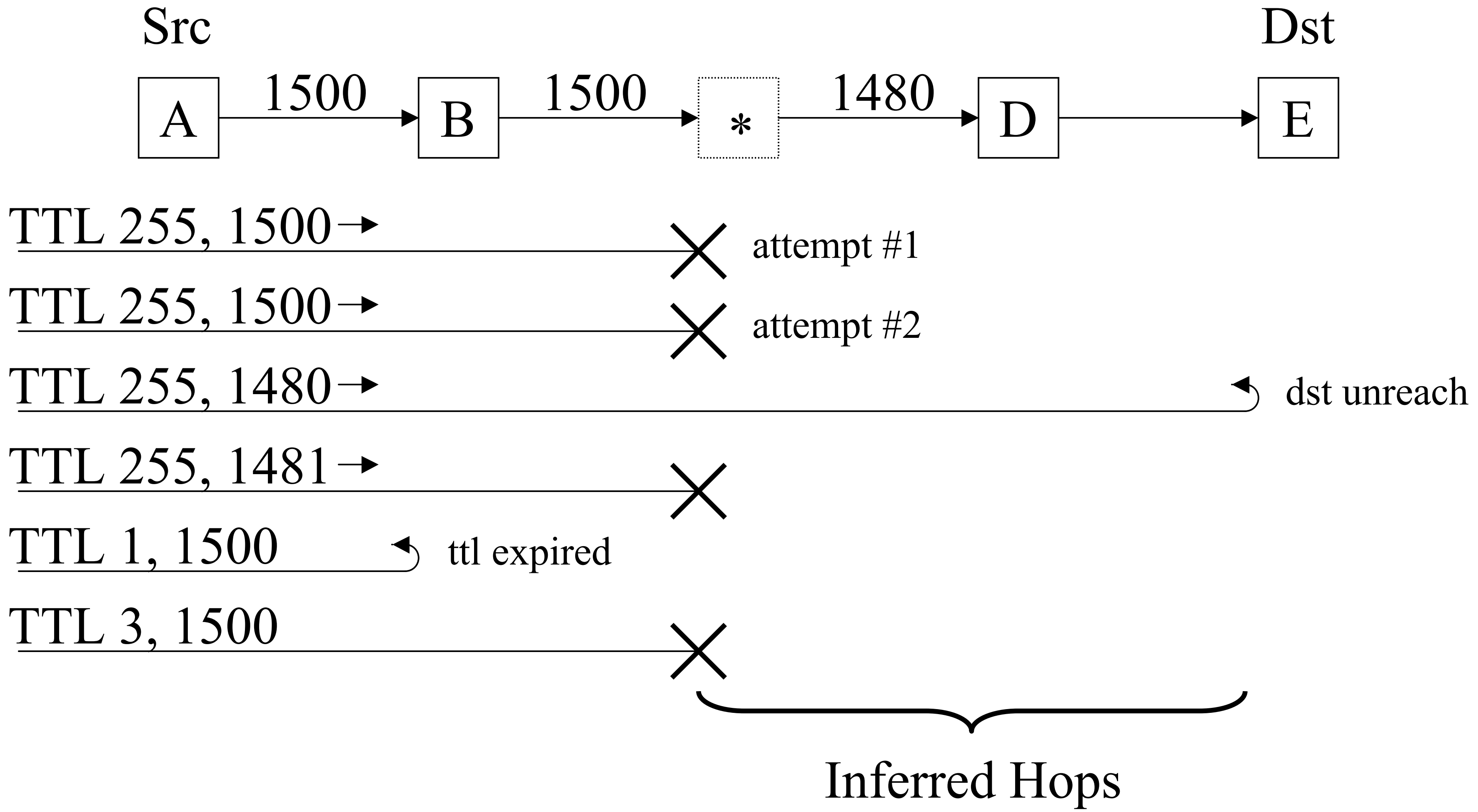
- If scamper cannot successfully complete PMTUD to a destination it knows should respond
  - it tries to infer the largest packet that can get through
  - and then does a TTL search to infer the series of hops to further investigate
- Scamper comes with a table of known MTUs to aid in finding the largest packet able to be sent

Path MTU Discovery

- Faults:
  1. Router configured to <u>not</u> send ICMP
  2. Router configured to send ICMP, but does not send fragmentation required
  3. Router configured to send ICMP, but does not send a useful fragmentation required message
     - Next hop MTU of 0
     - Next hop MTU larger than packet sent

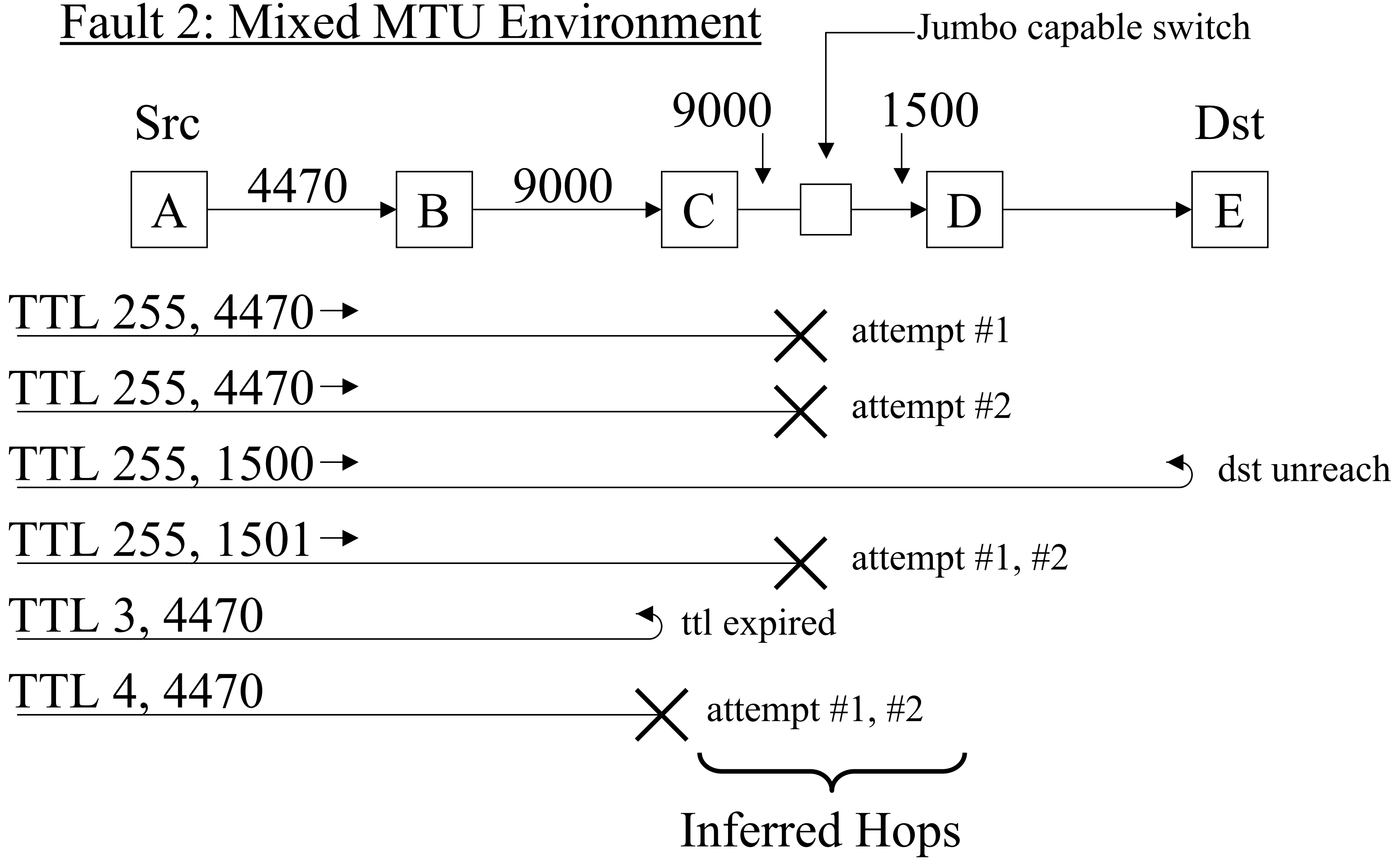
Path MTU Discovery

Fault 1: PMTUD Black Hole



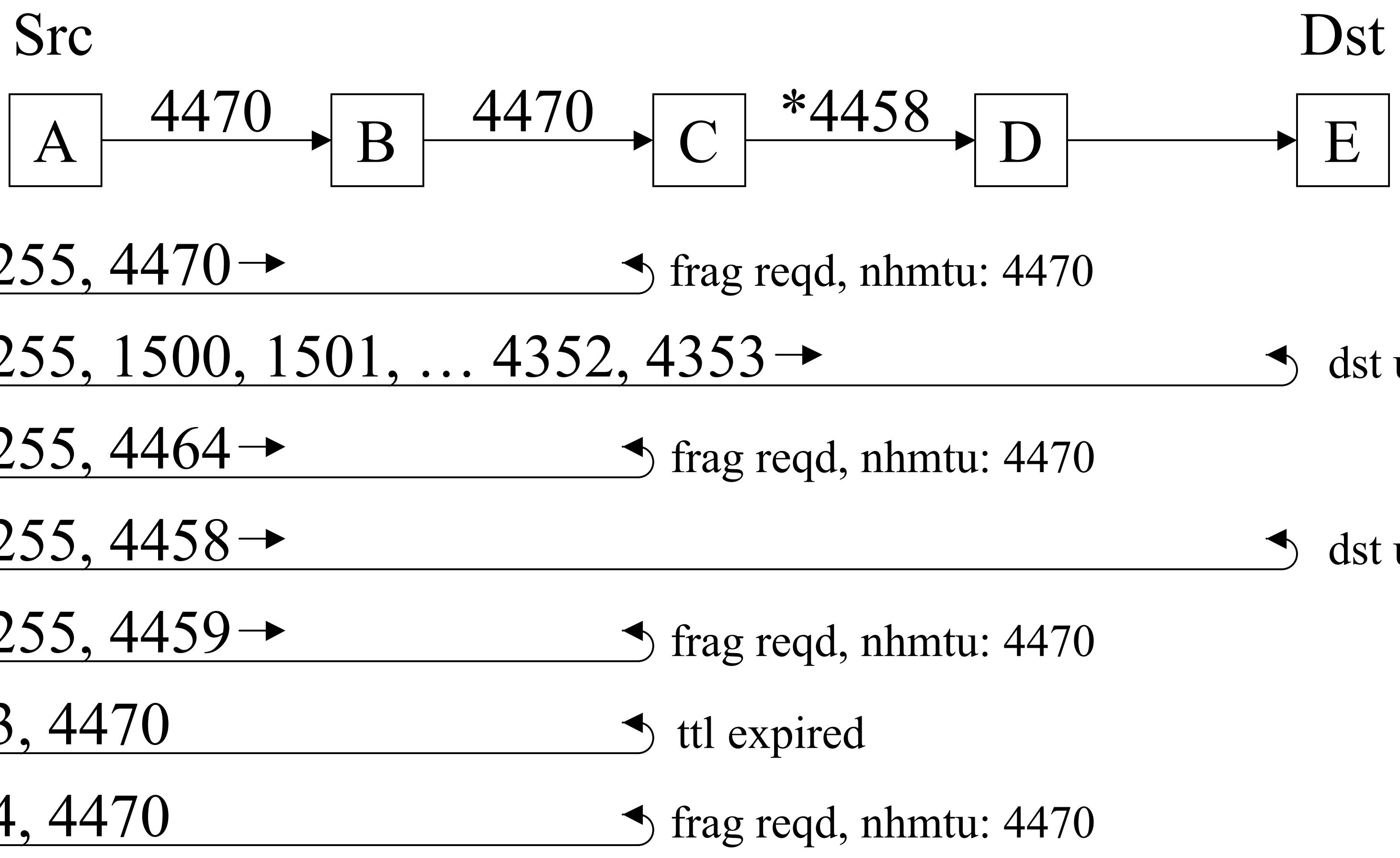
Path MTU Discovery

Fault 2: Mixed MTU Environment



Path MTU Discovery

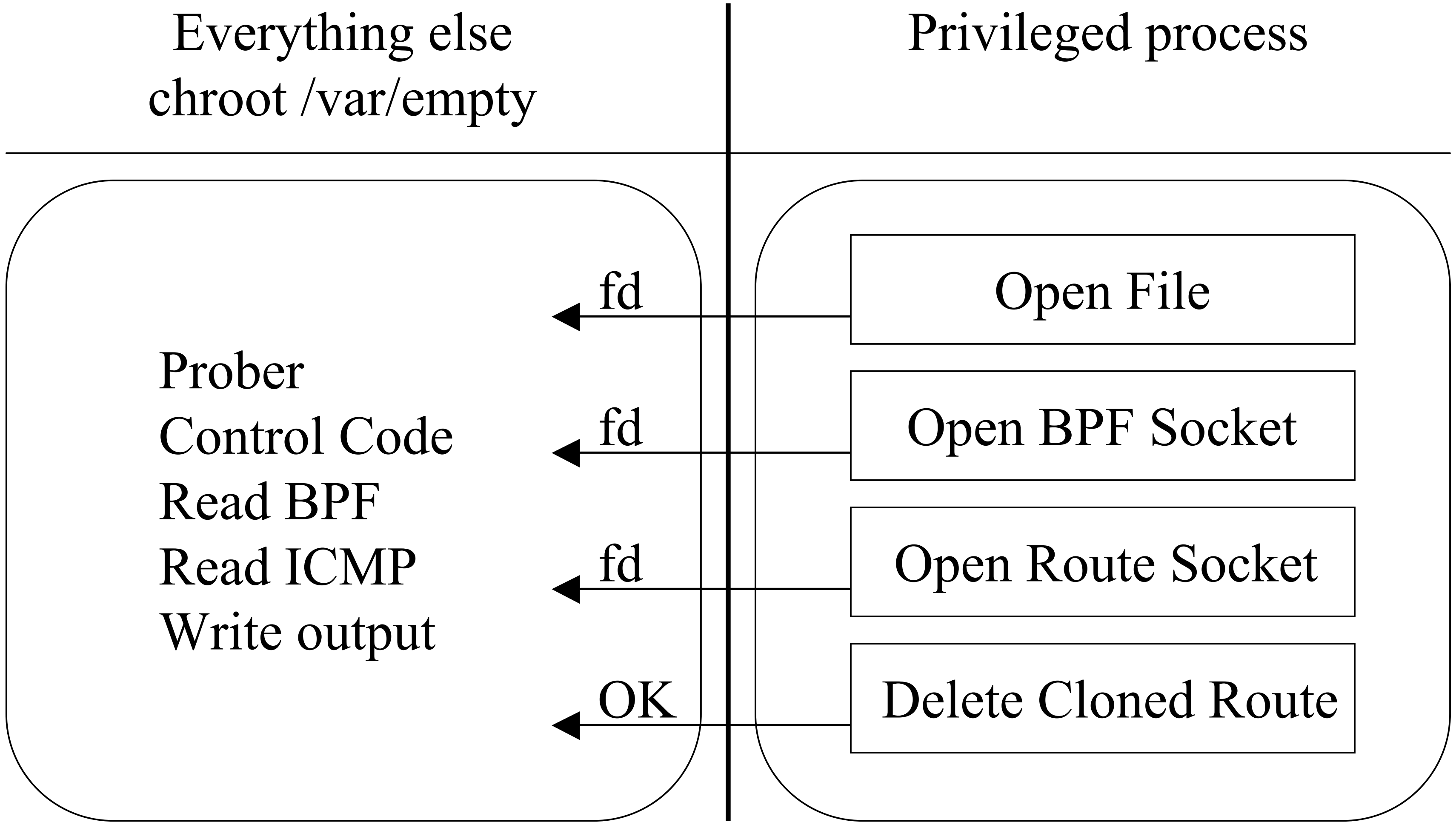
Fault 3: Useless next-hop MTU (nhmtu) returned



Privilege Separation

- Don’t want to deal with scamper being a remote-root attack vector
- scamper does its best to contain any damage in vulnerable code with privilege separation
- Important to do with the source code freely available

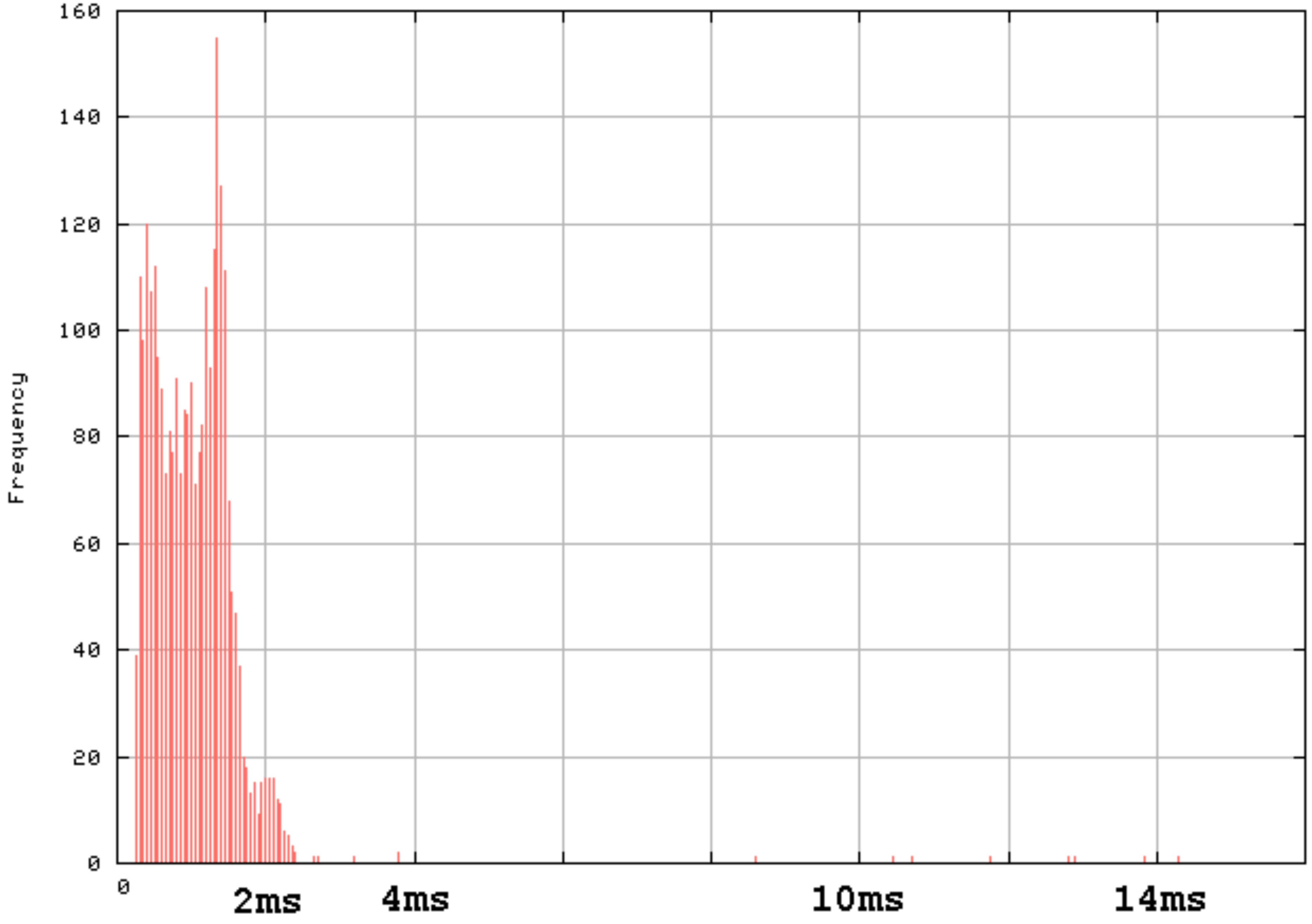
Privilege Separation



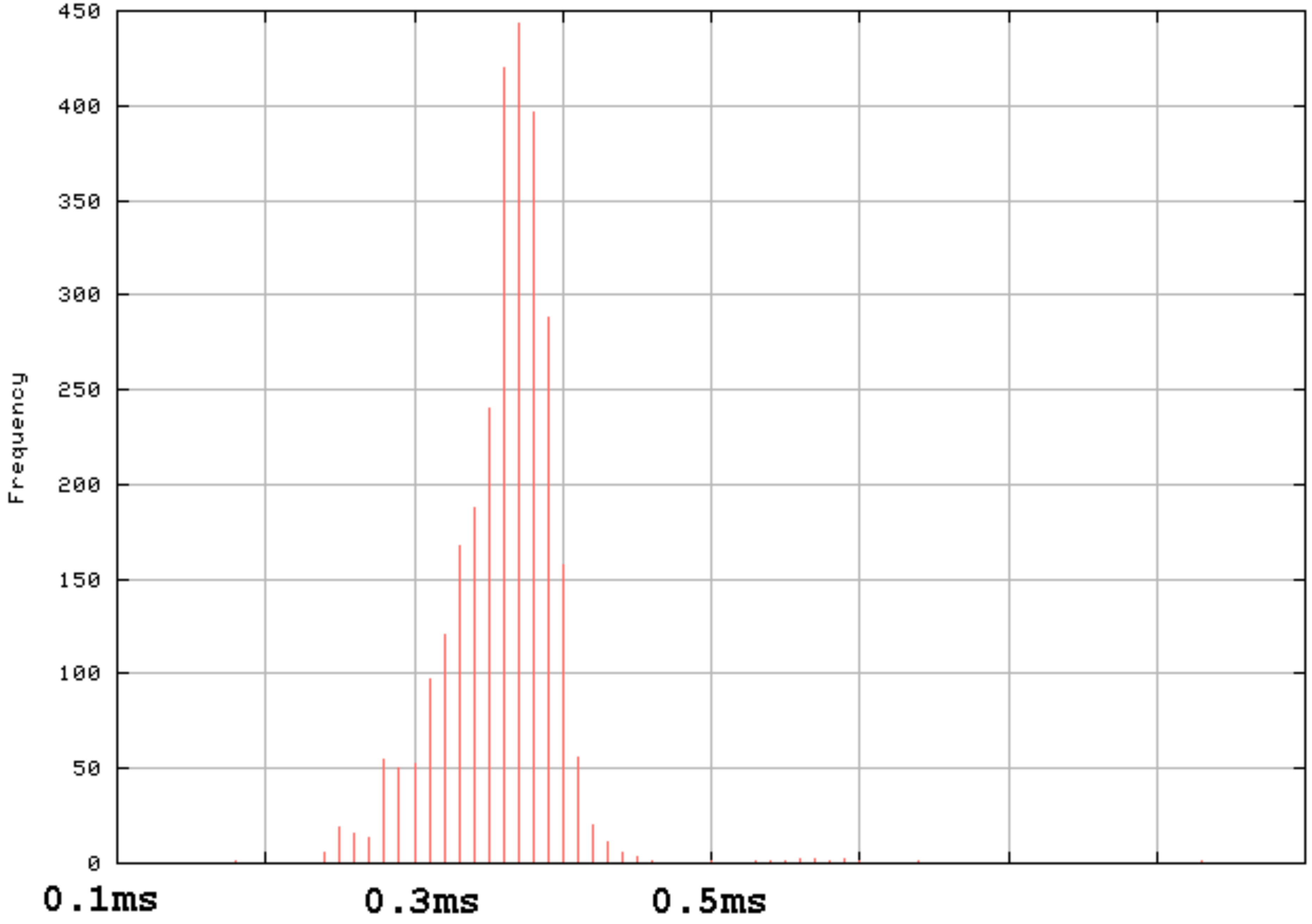
Datalink-provided TX timestamps

- The sockets API provides a method to obtain the time a packet was received by the kernel from a NIC
- But there’s nothing corresponding to when the kernel offloaded a packet to the NIC
- David Moore’s idea: use BPF

BPF - gettimeofday(), PPro 180, FreeBSD 4.10



BPF - gettimeofday(), Athlon 1.2, FreeBSD 4.10



Addition of more traceroute probe methods

- Scamper sends TTL limited probes to high numbered UDP ports by default
- Scamper can also send TTL limited ICMP echo request probes
- Some work has been done to include a TCP traceroute with probes marked by their sequence number, but not completed due to barriers imposed by IPv6 TCP sockets.

Additions of arbitrary measurement tasks

- Scamper’s design makes it fairly simple to add additional measurement tasks
- The only measurement task I’ve added so far is a ping implementation to aid the initial measurement phase of Kenjiro’s dual stack tool set.

Portability

- FreeBSD 4.X, 5.X
- NetBSD 1.6
- OpenBSD 3.4
- MacOS X
- Linux 2.4, 2.6
- Nearly done SunOS 5.8

Conclusions

- Scamper has evolved from a basic command-line driven traceroute-in-parallel tool to …
- … an extensible measurement tool useful for large scale Internet measurement

Collaboration Items

- I would like to pursue the Path MTU Discovery characterisation work I've done towards publication
- Kenjiro has suggested a Freenix publication giving an overview of scamper itself

Future Work

- Autotools
- Non-blocking resolver
  - Can only feed IP addresses to scamper
- Modularise
  - Ability to load new measurement technique modules into scamper at runtime that come with file format logic.
- tcptraceroute6