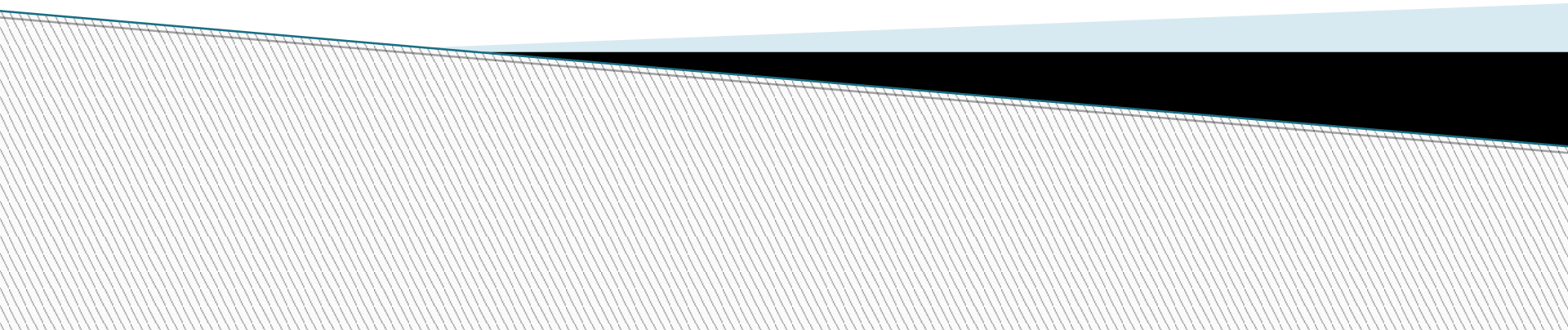
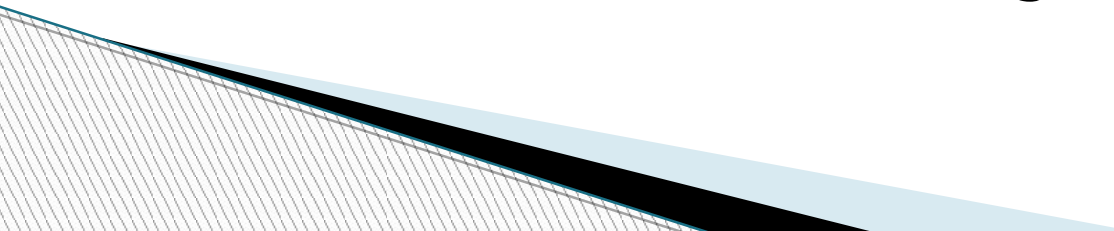


Introduction to passive network traffic monitoring

CS459 ~ Internet Measurements
Spring 2016



Active measurements

- ▶ active probe tools send packets into network; measure response
 - network, transport, application layer probes
 - ▶ can measure many things
 - delay/loss
 - topology/routing behavior
 - bandwidth/throughput
 - ▶ Internet Control Message Protocol (ICMP)
- 

Passive monitoring

- ▶ Passive monitoring is less of an experiment and more of an observational study. Instead of injecting artificial traffic into a network, passive monitoring entails monitoring traffic that is already on the network.
- ▶ Passive network monitoring can collect large volumes of data and from that we can derive a wide range of information.
 - For example, TCP headers contain information that can be used to derive network topology, identify services and operating systems running on networked devices, and detect potentially malicious probes.

Passive measurements

Capture packet data as it passes by

- packet capture applications on hosts use packet capture filters (tcpdump)
 - ▢ requires access to the wire
 - ▢ promiscuous mode network ports to see other traffic
- flow-level, packet-level data on routers
 - ▢ SNMP MIBs
 - ▢ Cisco NetFlow

Notable tools for passive monitoring (1/2)

- ▶ Tcpcap & libpcap

<http://www.tcpdump.org/>

- ▶ Wireshark

<http://www.wireshark.org/>

- ▶ tshark

<http://www.wireshark.org/docs/man-pages/tshark.html>

- ▶ ssldump

(an SSLv3/TLS network protocol analyzer-decrypts HTTPS)

<http://www.rtfm.com/ssldump/>



Notable tools for passive monitoring (2/2)

- ▶ Microsoft Network Monitor

<http://blogs.technet.com/netmon>

- ▶ tcpflow

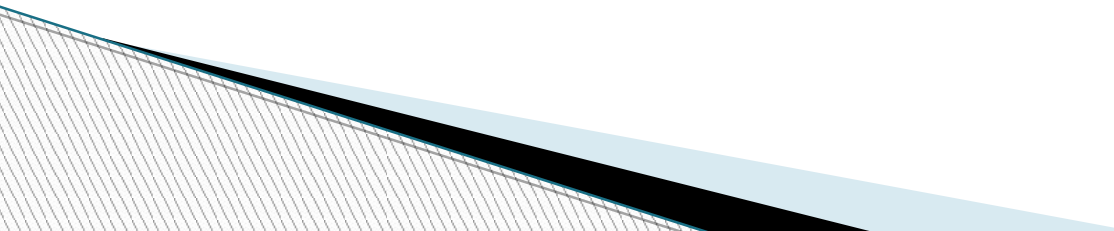
<https://github.com/simsong/tcpflow>

- ▶ dSniff

<http://www.monkey.org/~dugsong/dsniff/>

- ▶ tcptrace

<http://www.tcptrace.org/>



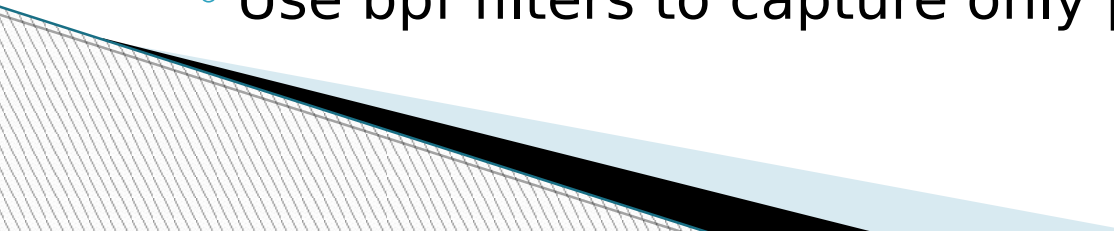
tcpdump

Tcpdump is the premier network analysis tool for information security professionals.

- ▶ Syntax:

```
tcpdump [options] [filter expression]
```

- ▶ Filters:

- We are often not interested in all packets flowing through the network
 - Use bpf filters to capture only packets of interest to us
- 

Berkeley Packet Filter or BPF

BPF allows a user-space program to attach a filter onto any socket and allow or disallow certain types of data to come through the socket.

The human readable filter is converted to a bytecode send to kernel -> **efficient!**

See more at: <http://www.tcpdump.org/papers/bpf-usenix93.pdf>



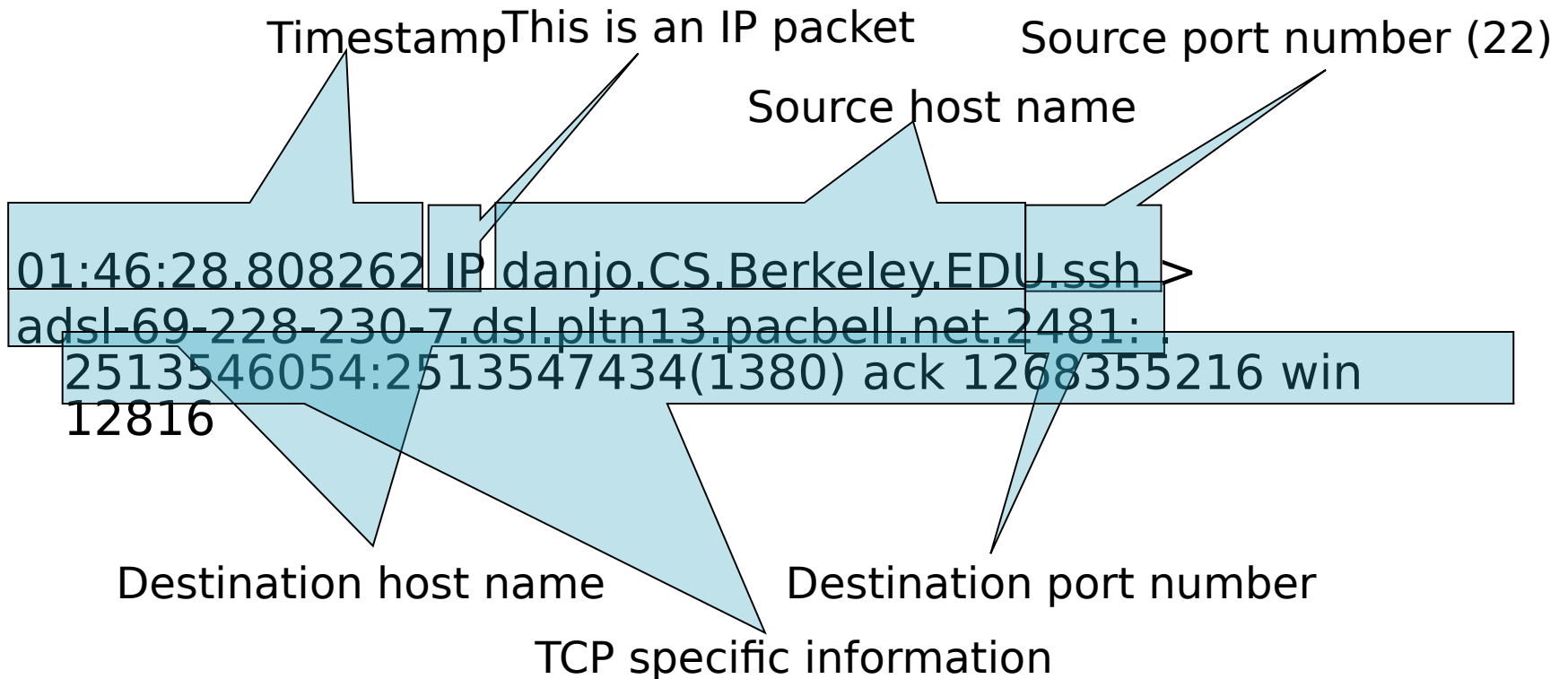
Demo (1/2)

1. Capture only udp packets
 - tcpdump "udp"
2. Capture only tcp packets
 - tcpdump "tcp"

Demo (2/2)

1. Capture only UDP packets with source or destination port 53 (DNS requests and replies)
 - `tcpdump "udp port 53"`
2. Capture only UDP packets with destination port 53 (DNS requests)
 - `tcpdump "udp dst port 53"`
3. Capture only UDP packets with source port 53 (DNS replies)
 - `tcpdump "udp src port 53"`

tcpdump output example



- Different output formats for different packet types

Useful tcpdump commands (1/2)

- ▶ *See the list of interfaces on which tcpdump can listen:*
`tcpdump -D`
- ▶ *Listen on interface eth0:*
`tcpdump -i eth0`
- ▶ *Be verbose while capturing packets:*
`tcpdump -v`
- ▶ *Limit the capture to 100 packets:*
`tcpdump -c 100`
- ▶ *Record the packet capture to a file called capture.cap:*
`tcpdump -w capture.cap`
- ▶ *Display the packets of a file called capture.cap:*
`tcpdump -r capture.cap`
- ▶ *Display IP addresses and port numbers instead of domain and service names when capturing packets:*
`tcpdump -n`
- ▶ *Capture any packets where the destination host is 192.168.1.1. Display IP addresses and port numbers:*
`tcpdump -n dst host 192.168.1.1`
- ▶ *Capture any packets where the source host is 192.168.1.1. Display IP addresses and port numbers:*
`tcpdump -n src host 192.168.1.1`
- ▶ *Capture any packets where the source or destination host is 192.168.1.1:*
`tcpdump host 192.168.1.1`

Useful tcpdump commands (2/2)

- ▶ *Capture any packets where the destination network is 192.168.1.0/24:*
`tcpdump dst net 192.168.1.0/24`
- ▶ *Capture any packets where the source or destination network is 192.168.1.0/24:*
`tcpdump net 192.168.1.0/24`
- ▶ *Capture any packets where the destination port is 23:*
`tcpdump dst port 23`
- ▶ *Capture any packets where the destination port is between 1 and 1023 inclusive:*
`tcpdump dst portrange 1-1023`
- ▶ *Capture only TCP packets where the destination port is between 1 and 1023 inclusive:*
`tcpdump tcp dst portrange 1-1023`
- ▶ *Capture only UDP packets where the destination port is between 1 and 1023 inclusive:*
`tcpdump udp dst portrange 1-1023`
- ▶ *Capture any packets with destination IP 192.168.1.1 and destination port 23:*
`tcpdump "dst host 192.168.1.1 and dst port 23"`
- ▶ *Capture any packets with destination IP 192.168.1.1 and destination port 80 or 443:*
`tcpdump "dst host 192.168.1.1 and (dst port 80 or dst port 443)"`
- ▶ *Capture any ICMP packets:*
`tcpdump -v icmp`

See more at: <http://www.tcpdump.org/manpages/tcpdump.1.html>

Online Packet Capturing Example (1/4)

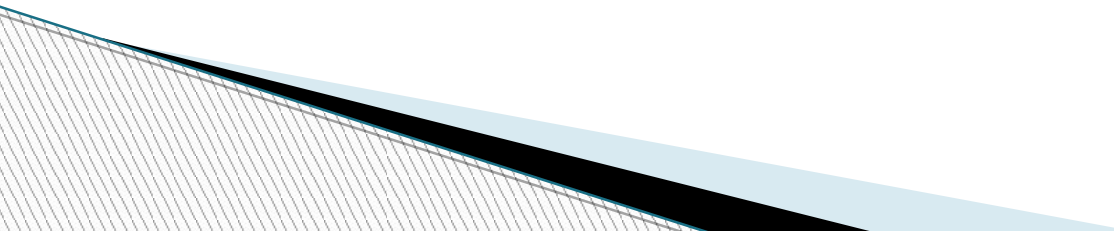
- Start capturing traffic (needs sudo):

login at syko or rodi csd machine

- \$: tcpdump -D (find the appropriate interface e.g. eth0)

- \$: tcpdump -n -i eth0 -w onlineTest.pcap

- Produce traffic:

- Read your email
 - Watch a youtube video for 5 minutes
 - Download torrent
- 

Online Packet Capturing Example (2/4)

Stop packet capture and report:

➤ Average packet size:

- get ip packets:
\$: tcpdump -q -n -r onlineTest.pcap ip > onlineTest.out
- search for 'length' string, get size, sum bytes, print sum/n:
\$: cat onlineTest.out \
| awk -F'length' '{print \$2}' \
| awk -F ':' '{print \$1}' \
| awk '{ sum += \$1; n++ } \
END { if (n > 0) print sum / n; }'

Online Packet Capturing Example (3/4)

➤ Number of packets per destination port:

1. Get the destination port for each packet

```
$: cat onlineTest.out \  
| awk '{print $5}' \  
| awk -F'.' '{print $5}' \  
| awk -F':' '{print $1}' > destPorts
```

2. Remove blank lines if any

```
$: sed '/^$/d' destPorts > destPorts.log
```

3. Sort ports and measure the occurrences of each port

```
$: sort -n destPorts | uniq -c > numbofpackets
```

4. Plot numbofpackets

Online Packet Capturing Example (4/4)

- Plot CDF of packet sizes:

Before you make cdf plot you should prepare your data:

```
$: cat onlineTest.out \  
  | awk -F'length' '{print $2}' \  
  | awk -F ':' '{print $1}' \  
  | sort -n | uniq -c
```

Mawi traces

The **MAWI** Working Group is a joint effort of Japanese network research and academic institutions to study the performance of networks and networking protocols in Japanese wide area networks.

MAWI is an acronym for the **M**easurement and **A**nalysis of **W**ide-area **I**nternet.

See more at: <http://mawi.wide.ad.jp/mawi/>



Offline Packet Capturing Example (1/5)

- ▶ Download a trace (e.g. trace from 7/1/2013):

```
$: wget http://mawi.nezu.wide.ad.jp/mawi/samplepoint-F/2013/201301071400.dump.gz
```

- ▶ Analyze trace with tcpdump:

1. Number of packets in trace:

```
$: tcpdump -n -r 201301071400.dump | wc -l  
-> 100933314 packets
```

Offline Packet Capturing Example (2/5)

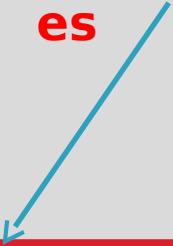
2. CDF of packet sizes, show the 95th and 99th percentile
- ▶ Get the data from trace and measure the occurrences:

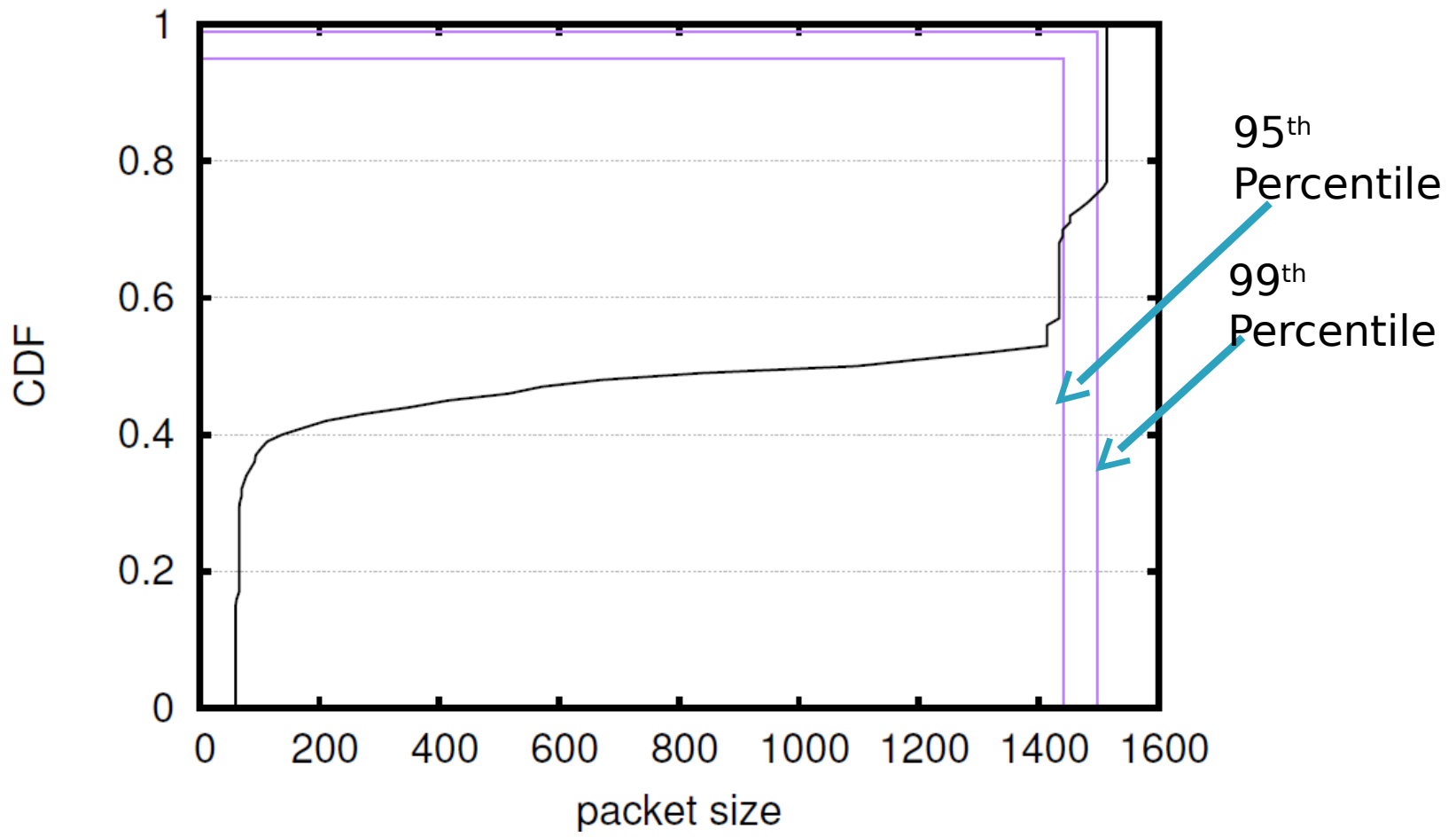
```
cat 201301071400.out \  
| awk -F'length' '{print $2}' \  
| awk -F ':' '{print $1}' \  
| sort -n | uniq -c
```

Plot CDF with gnuplot

```
set terminal postscript enhanced eps defaultplex \ leveldefault
  blacktext\dashed dashlength 2.0 linewidth 3.0 butt\palfuncparam 2000,0.003 \
  "Helvetica" 24
set out "cdf.eps"set grid y lt 0 lw 1 lc rgb "#B8B8B8"
set auto x
set nokey
set xlabel "Packet size"
set ylabel "CDF"
set style data lines
set boxwidth 0.9
perc95=1441
perc99=1499
set arrow from perc95,0 to perc95,0.95 nohead lt 1 lc rgb "purple"
set arrow from graph(0,0),0.95 to perc95,0.95 nohead lt 1 lc rgb "purple"
set arrow from perc99,0 to perc99,0.99 nohead lt 1 lc rgb "purple"
set arrow from graph(0,0),0.99 to perc99,0.99 nohead lt 1 lc rgb "purple"
plot 'datacdf.log'
```

**Draw
percentil
es**





Offline Packet Capturing Example (3/5)

3. Traffic percentage per destination port

- ▶ Get number of packets per destination port

```
cat 201301071400.out \  
| awk '{print $5}' \  
| awk -F'.' '{print $5}' \  
| awk -F':' '{print $1}' \  
| sort -n | uniq -c > packets
```

- ▶ Divide each port's number of packets by the total packets that you calculate in (1) e.g.

```
cat packets | awk '{print $1/<totalnum>" "$2}'
```

Offline Packet Capturing Example (4/5)

4. All traffic to/from a given IP address
(let's say 201.57.174.30)
- ▶ Get delta (micro-second resolution) between current and first line on each dump line for a specific host:
\$: tcpdump -q -ttttt -nr 201301071400.dump host 201.57.174.30 > specIP.out

```
00:00:18.569435 IP 201.57.174.30.873 > 208.102.97.136.52151: tcp 1348
00:00:18.569448 IP 201.57.174.30.873 > 208.102.97.136.52151: tcp 1348
00:00:18.569451 IP 201.57.174.30.873 > 208.102.97.136.52151: tcp 1348
00:00:18.569454 IP 201.57.174.30.873 > 208.102.97.136.52151: tcp 1348
00:00:18.569462 IP 201.57.174.30.873 > 208.102.97.136.52151: tcp 1348
00:00:18.569469 IP 201.57.174.30.873 > 208.102.97.136.52151: tcp 1348
00:00:18.569559 IP 201.57.174.30.873 > 208.102.97.136.52151: tcp 1348
00:00:18.574440 IP 208.102.97.136.52151 > 201.57.174.30.873: tcp 0
00:00:18.574443 IP 208.102.97.136.52151 > 201.57.174.30.873: tcp 0
00:00:18.574555 IP 208.102.97.136.52151 > 201.57.174.30.873: tcp 0
00:00:18.574558 IP 208.102.97.136.52151 > 201.57.174.30.873: tcp 0
00:00:18.575306 IP 208.102.97.136.52151 > 201.57.174.30.873: tcp 0
00:00:18.575313 IP 208.102.97.136.52151 > 201.57.174.30.873: tcp 0
00:00:18.575315 IP 208.102.97.136.52151 > 201.57.174.30.873: tcp 0
```

Why is it zero???

Offline Packet Capturing Example (5/5)

5. Get all UDP conversations of the trace:

```
$: tcpdump -n -w udpConvers.pcap -r 201301071400.dump udp
```

```
07:00:01.425777 IP 154.101.17.68.17404 > 60.162.122.122.26928: UDP, length 53
07:00:01.426027 IP 202.37.219.22.53 > 133.231.140.147.1051: 46342-[|domain]
07:00:01.426644 IP 154.101.17.68.17404 > 60.162.122.122.26928: UDP, length 1440
07:00:01.426681 IP 154.101.17.68.17404 > 60.162.122.122.26928: UDP, length 1440
07:00:01.426894 IP 154.101.17.68.17404 > 60.162.122.122.26928: UDP, length 1440
07:00:01.427028 IP 154.101.17.68.17404 > 60.162.122.122.26928: UDP, length 1440
07:00:01.427146 IP 154.101.17.68.17404 > 60.162.122.122.26928: UDP, length 1440
07:00:01.427269 IP 154.101.17.68.17404 > 60.162.122.122.26928: UDP, length 1440
07:00:01.427411 IP 154.101.17.68.17404 > 60.162.122.122.26928: UDP, length 1440
07:00:01.427429 IP 154.101.230.225.8582 > 208.184.175.96.53: 9184[|domain]
07:00:01.427521 IP 154.101.17.68.17404 > 60.162.122.122.26928: UDP, length 1440
07:00:01.427644 IP 154.101.17.68.17404 > 60.162.122.122.26928: UDP, length 1440
07:00:01.427790 IP 154.101.17.68.17404 > 60.162.122.122.26928: UDP, length 1440
07:00:01.427894 IP 154.101.17.68.17404 > 60.162.122.122.26928: UDP, length 1440
07:00:01.427903 IP 154.101.17.68.17404 > 80.193.195.241.9356: UDP, length 53
07:00:01.428149 IP 154.101.17.68.17404 > 20.205.51.32.55284: UDP, length 53
07:00:01.428156 IP 164.114.66.226.1039 > 208.3.199.6.1156: UDP, length 1078
07:00:01.428401 IP 80.85.231.21.1065 > 182.8.68.218.1434: UDP, length 376
```