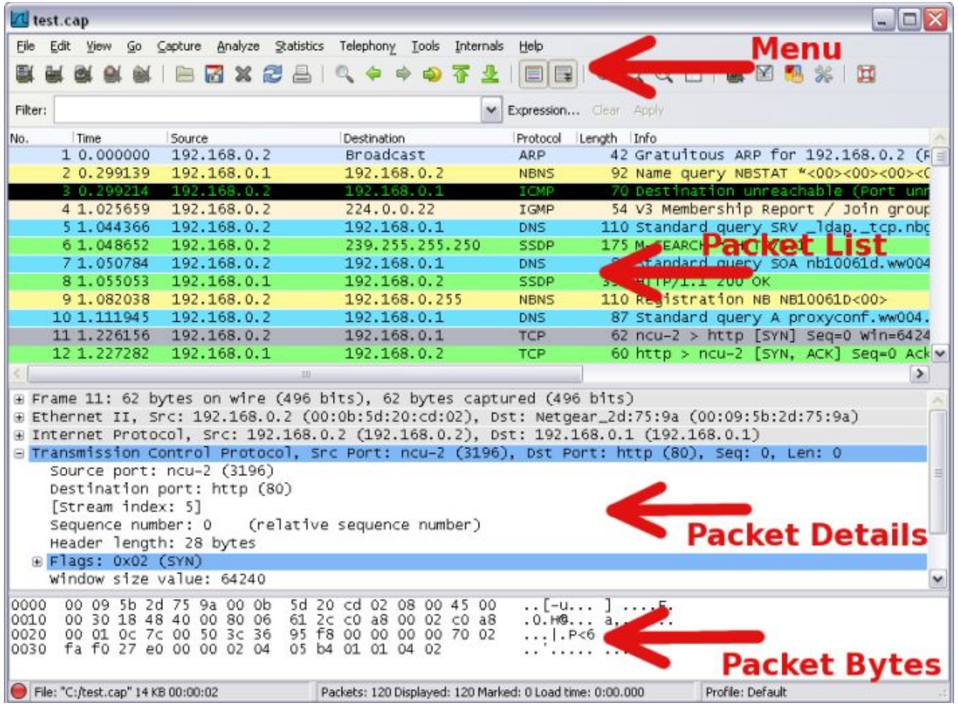
Recitation 12

Internet Technology (Section 01)

Assignment 4: Packet Trace Analysis

- Pcapng - a file format for storing packet traces


test.cap
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Menu
Filter:
Expression... Clear Apply
Packet List
Packet Details
Packet Bytes

```
No.   Time      Source        Destination      Protocol Length Info
    1 0.000000  192.168.0.2   Broadcast        ARP       42 Gratuitous ARP for 192.168.0.2 (R
    2 0.299139  192.168.0.1   192.168.0.2      NBNS      92 Name query NBSTAT *<00><00><00><0
    3 0.299214  192.168.0.2   192.168.0.1      ICMP      70 Destination unreachable (Port unr
    4 1.025659  192.168.0.2   224.0.0.22       IGMP      54 V3 Membership Report / Join group
    5 1.044366  192.168.0.2   192.168.0.1      DNS      110 Standard query SRV _ldap._tcp.nbc
    6 1.048652  192.168.0.2   239.255.255.250  SSDP     175 M-SEARCH
    7 1.050784  192.168.0.2   192.168.0.1      DNS          Standard query SOA nb10061d.ww004
    8 1.055053  192.168.0.1   192.168.0.2      SSDP         HTTP/1.1 200 OK
    9 1.082038  192.168.0.2   192.168.0.255    NBNS     110 Registration NB NB10061D<00>
   10 1.111945  192.168.0.2   192.168.0.1      DNS       87 Standard query A proxyconf.ww004.
   11 1.226156  192.168.0.2   192.168.0.1      TCP       62 ncu-2 > http [SYN] Seq=0 Win=6424
   12 1.227282  192.168.0.1   192.168.0.2      TCP       60 http > ncu-2 [SYN, ACK] Seq=0 Ack
```

```
⊞ Frame 11: 62 bytes on wire (496 bits), 62 bytes captured (496 bits)
⊞ Ethernet II, Src: 192.168.0.2 (00:0b:5d:20:cd:02), Dst: Netgear_2d:75:9a (00:09:5b:2d:75:9a)
⊞ Internet Protocol, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.1 (192.168.0.1)
⊟ Transmission Control Protocol, Src Port: ncu-2 (3196), Dst Port: http (80), Seq: 0, Len: 0
    Source port: ncu-2 (3196)
    Destination port: http (80)
    [Stream index: 5]
    Sequence number: 0    (relative sequence number)
    Header length: 28 bytes
  ⊞ Flags: 0x02 (SYN)
    Window size value: 64240
```

```
0000  00 09 5b 2d 75 9a 00 0b  5d 20 cd 02 08 00 45 00   ..[-u... ] .....E.
0010  00 30 18 48 40 00 80 06  61 2c c0 a8 00 02 c0 a8   .0.H@... a,.....
0020  00 01 0c 7c 00 50 3c 36  95 f8 00 00 00 00 70 02   ...|.P<6
0030  fa f0 27 e0 00 00 02 04  05 b4 01 01 04 02         ..'.....
```

File: "C:/test.cap" 14 KB 00:00:02   Packets: 120 Displayed: 120 Marked: 0 Load time: 0:00.000   Profile: Default

Reading pcap files

- Pcapng - a file format for storing packet traces
  - Python Scapy lib can be used to read pcap files

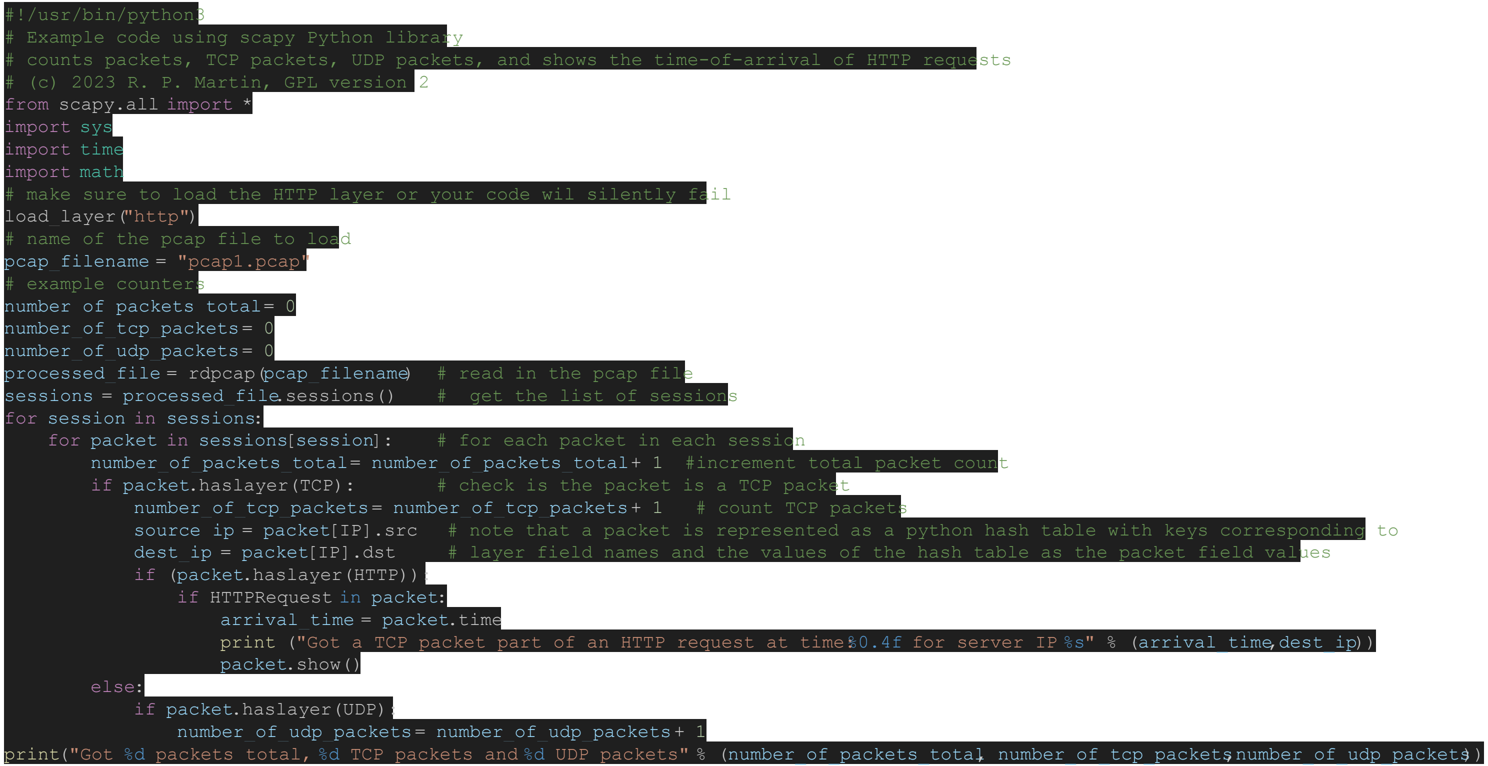
```
from scapy.all import *
pcap = rdpcap("pcap1.pcap")
print(pcap)
```

Reading pcap files

- Pcapng - a file format for storing packet traces
  - Python Scapy lib can be used to read pcap files
  - Obtain a dictionary of sessions

```
from scapy.all import *
pcap = rdpcap("pcap1.pcap")
sessions = pcap.sessions()
print(sessions)
print(sessions["Other"])
for p in sessions["Other"]:
    print(p)
```

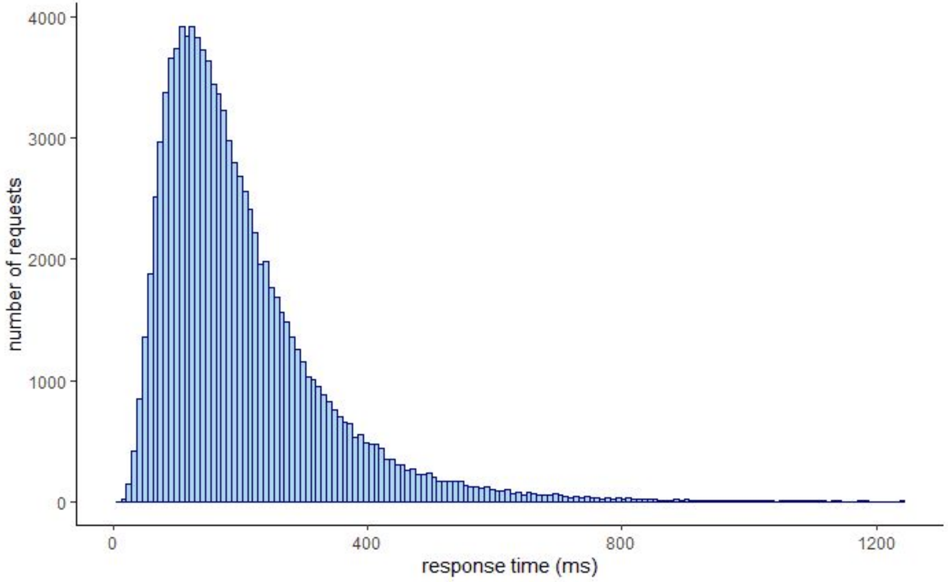
A complete example



```
#!/usr/bin/python3
# Example code using scapy Python library
# counts packets, TCP packets, UDP packets, and shows the time-of-arrival of HTTP requests
# (c) 2023 R. P. Martin, GPL version 2
from scapy.all import *
import sys
import time
import math
# make sure to load the HTTP layer or your code wil silently fail
load_layer("http")
# name of the pcap file to load
pcap_filename = "pcap1.pcap"
# example counters
number_of_packets_total = 0
number_of_tcp_packets = 0
number_of_udp_packets = 0
processed_file = rdpcap(pcap_filename)   # read in the pcap file
sessions = processed_file.sessions()     #  get the list of sessions
for session in sessions:
    for packet in sessions[session]:     # for each packet in each session
        number_of_packets_total = number_of_packets_total + 1  #increment total packet count
        if packet.haslayer(TCP):         # check is the packet is a TCP packet
            number_of_tcp_packets = number_of_tcp_packets + 1   # count TCP packets
            source_ip = packet[IP].src   # note that a packet is represented as a python hash table with keys corresponding to
            dest_ip = packet[IP].dst     # layer field names and the values of the hash table as the packet field values
            if (packet.haslayer(HTTP)):
                if HTTPRequest in packet:
                    arrival_time = packet.time
                    print ("Got a TCP packet part of an HTTP request at time:%0.4f for server IP %s" % (arrival_time,dest_ip))
                    packet.show()
        else:
            if packet.haslayer(UDP):
                number_of_udp_packets = number_of_udp_packets + 1
print("Got %d packets total, %d TCP packets and %d UDP packets" % (number_of_packets_total, number_of_tcp_packets,number_of_udp_packets))
```

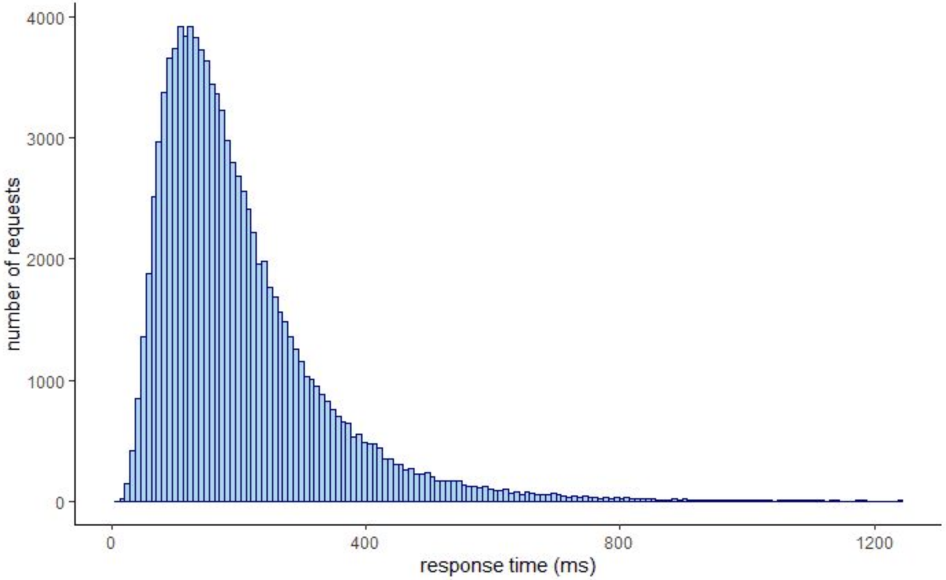
Tail Latency

- High percentile latencies of a distribution of latencies



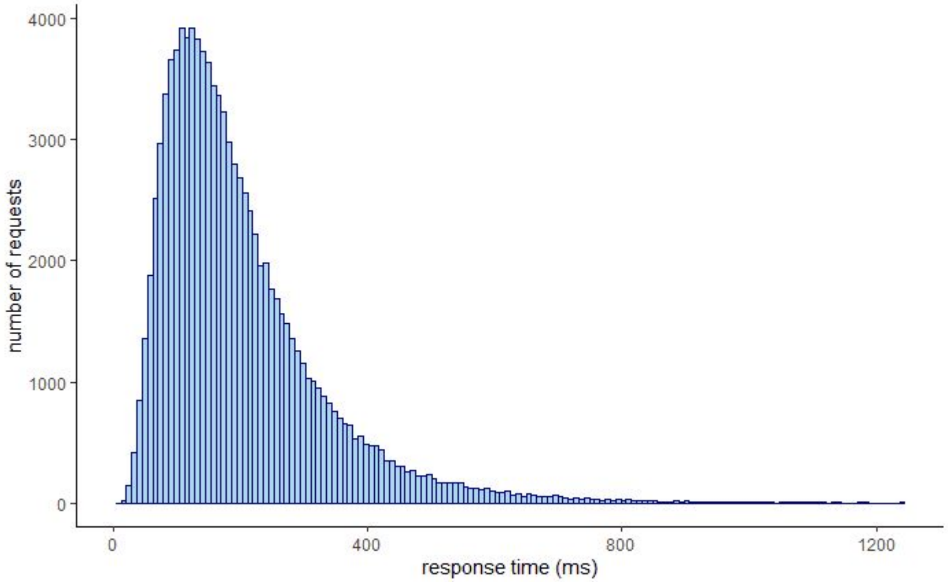
Tail Latency

- High percentile latencies of a distribution of latencies
    - Why is this important?



Tail Latency

- High percentile latencies from a distribution of latencies
  - Why is this important?
- Maybe: The most number of requests come from a very important client, they expect least issues

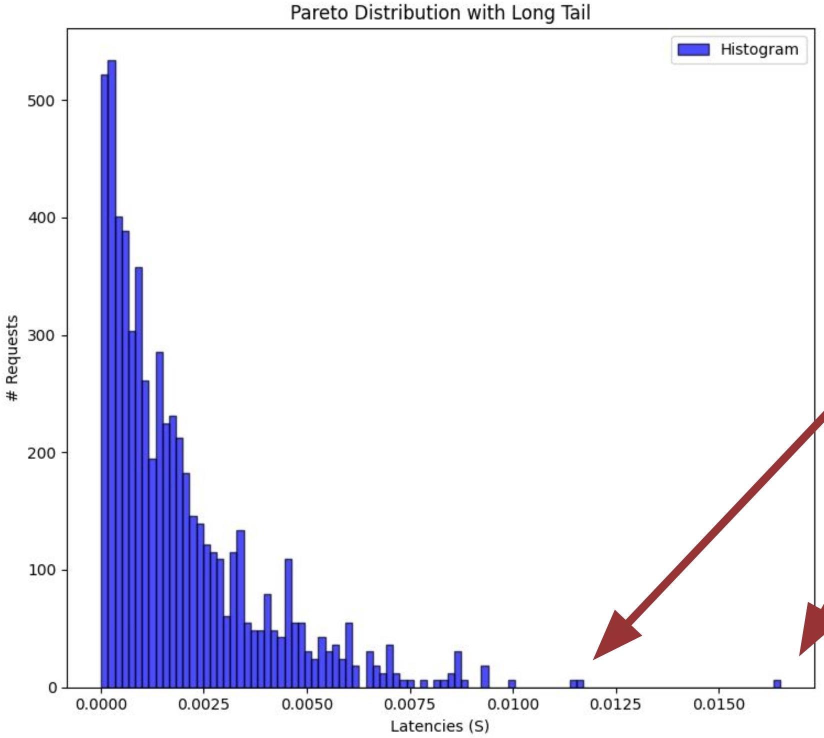


Visualizing Tail Latency

- We can use matplotlib

```
import numpy as np
import matplotlib.pyplot as plt
# Parameters for the Pareto distribution
alpha = 500  # shape parameter (controls tail length)
x_min = 0.0  # minimum value
# Generate random samples from Pareto distribution
np.random.seed(42)  # for reproducibility
pareto_samples = np.random.pareto(alpha, 1000) + x_min
# Plot the histogram of the samples
plt.hist(pareto_samples, bins=100, density=True, alpha=0.7, color='blue', edgecolor='black')
# Set plot labels and title
plt.title('Pareto Distribution with Long Tail')
plt.xlabel('Latencies (S)')
plt.ylabel('# Requests')
plt.legend(['Histogram'])
# Show the plot
plt.show()
```

Visualizing Tail Latency



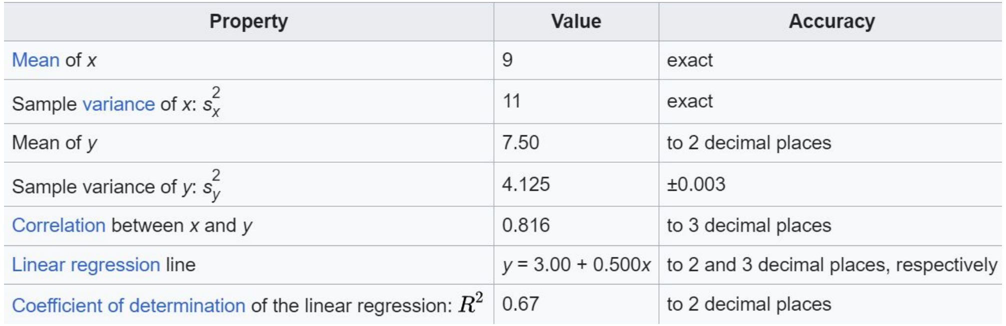
Tail Latencies!

Always Visualize Data!

- Anscombe's quartet
  - 4 datasets with identical summary stats

Always Visualize Data!

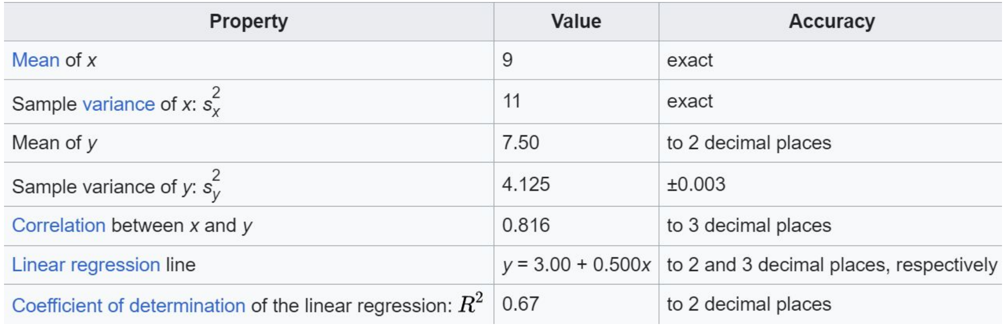
- Anscombe's quartet
  - 4 datasets with identical summary stats



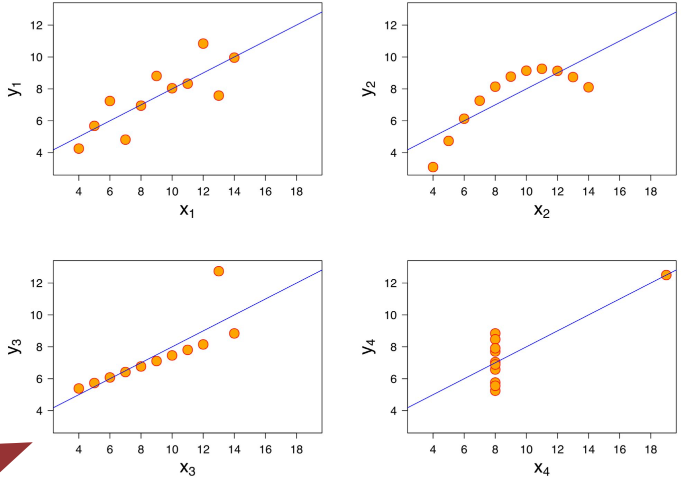
Property	Value	Accuracy
Mean of x	9	exact
Sample variance of x: s_x^2	11	exact
Mean of y	7.50	to 2 decimal places
Sample variance of y: s_y^2	4.125	±0.003
Correlation between x and y	0.816	to 3 decimal places
Linear regression line	$y = 3.00 + 0.500x$	to 2 and 3 decimal places, respectively
Coefficient of determination of the linear regression: R^2	0.67	to 2 decimal places

Always Visualize Data!

- Anscombe's quartet
  - ○ 4 datasets with identical summary stats



Property	Value	Accuracy
Mean of x	9	exact
Sample variance of x: s_x^2	11	exact
Mean of y	7.50	to 2 decimal places
Sample variance of y: s_y^2	4.125	±0.003
Correlation between x and y	0.816	to 3 decimal places
Linear regression line	$y = 3.00 + 0.500x$	to 2 and 3 decimal places, respectively
Coefficient of determination of the linear regression: R^2	0.67	to 2 decimal places



  - ○ Yet when plotted, the data looks very different!