Network Traffic Measurements and Analysis

Alessandro E. C. Redondi

DEIB - Politecnico di Milano

Fall, 2017
Introduction
Contact details

- Alessandro E. C. Redondi - Ph.D.
- Email: alessandroenrico.redondi@polimi.it
- Tel: 02 2399 3403
- Office: DEIB building, 3rd floor
- Research interests:
  - Network data analysis
  - Internet of Things
  - Localization systems
Course summary

Part 1: Background
- Introduction on network measurements
- Brief recap on machine learning

Part 2: Applications
- Traffic classification
- Anomaly detection
- Localization in wireless networks
- User behaviour estimation, device classification
Introduction
Network measurements
Active Measurements
Passive Measurements
Data visualization

Course Website

Network Traffic Measurement and Analysis

PhD Course

This course will focus on fundamental aspects of large-scale network traffic measurements and analysis techniques targeting two main challenges: (i) how to measure a network and (ii) how to extract knowledge from network traffic measurements. This course will start off by overviewing the most popular network measurement techniques at different layers of the communication stack: ranging from passive probes at the physical layer for wireless networks up to the Simple Network Management Protocol (SNMP) at the application layer.

Building upon these tools, the course will then focus on several use cases, overviewing for each one the main building blocks of the measurement system and the specific data processing algorithms that can be used to extract knowledge from the acquired data:

- Anomaly detection and traffic identification
- Load and performance prediction in mobile radio networks
- Device profiling and classification in campus-wide WiFi networks
- User localization and behavior estimation through WiFi traffic measurements

Depending on the specific application scenario, supervised and unsupervised machine learning approaches will be studied and applied. The course will include “hands-on” lectures to implement network measurement and analysis systems.

Schedule

- Tue 7/11 - 14:30 - 17:30, Room PT1
Practical activities using Matlab (Octave) and/or Python to apply the techniques we’ll see on realistic data sets.

Examples:

- 3G/LTE data from cellular base stations
- Wi-Fi traffic from the campus network
- Data from a smart meter sensor network
Evaluation

Presentation of a survey paper regarding a particular topic related to network measurements and analysis

or

Presentation of a practical project
Network measurements
When you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind. (Lord Kelvin)
Network measurements

Why measuring a network?
Why measuring a network?

Without measurements, we would not have objective records or benchmark of how a network behaves.
Network measurements

Why measuring a network?

Without measurements, we would not have objective records or benchmark of how a network behaves.

Measurements show whether changes improve or degrade the network performance, and by how much.
Why measuring a network?

Without measurements, we would not have objective records or benchmark of how a network behaves.

Measurements show whether changes improve or degrade the network performance, and by how much.

Similarly, by measuring the network behaviour and comparing it to the benchmark, we can also identify if something has changed!
Network measurements

Why analysing network measurements?
Network measurements

Why analysing network measurements?

Performance evaluation: how fast is the network? How many users it can support now and in the future?
Network measurements

Why analysing network measurements?

Performance evaluation: how fast is the network? How many users it can support now and in the future?

Anomaly detection: is there something wrong/unusual happening?
Network measurements

Why analysing network measurements?

Performance evaluation: how fast is the network? How many users it can support now and in the future?

Anomaly detection: is there something wrong/unusual happening?

Traffic classification: how is the network used? By whom and for what?
Network measurements

Why analysing network measurements?

Performance evaluation: how fast is the network? How many users it can support now and in the future?

Anomaly detection: is there something wrong/unusual happening?

Traffic classification: how is the network used? By whom and for what?

Answers provide starting points for network control and optimization and for providing advanced services to the users/operators.
Classification of network measurements

- Active VS Passive
- Physical Layer VS Upper Layers
- Packet-level VS Flow-level
- Single point VS Multi-point
- Network VS Application
Active VS Passive Measurements

Active measurements: obtained through *injected traffic*.

- Used for fault/vulnerability detection, network/application performance tests.
- Impose extra traffic on the network (most operators are unwilling to do so)
- Can distort the network behaviour in the process
- Scalability issues (large number of trials)
- Examples: ping, iperf, traceroute
Passive measurements: observation of existing traffic
- Do not inject extra traffic
- Observe network packets and connections (flows) in details
- Massive amount of data (hardware and memory issues)
- Legal issues
- Examples: tcpdump, WireShark, DPI
Physical VS Upper Layers

Physical measurements can be always obtained, regardless of the upper layers protocol used.

Mostly used in cellular/wireless networks (LTE, Wi-Fi, Bluetooth, Sensor Networks...). Evaluated on a frame-basis.

- Received signal strength (RSS)
- Channel quality indicator (CQI)
- Modulation and coding scheme (MCS)
- Bit error rate (BER)
- Throughput (THR)

Upper layers measurements generally used in IP-based networks.
Packet-level VS Flow-level

Packet-level: microscopic scale, every single packet that travels across the measurement point is captured

Collected information:
- Source and Destination IP addresses
- Source and Destination ports
- packet size, timestamp
- protocol information
- application data

Storing packet traces is costly! Need for high-end network interfaces, high CPU power and massive data storage capabilities!

De facto standard: libpcap/WinPcap library (tcpdump, Wireshark)
Packet-level VS Flow-level

Flow-level: macroscopic scale, packets are aggregated into flows

Flow: set of packets that share origin and destination address and ports, observed within a pre-defined time frame.

Collected information:

- Source and Destination IP addresses
- Source and Destination ports
- Type of transport protocol
- Flow size, duration, bitrate
- Number of flows per unit time

De facto standard: Cisco NetFlow
Higher level monitoring

Generally, higher-level aggregate information are always available at the measurement points

Collected information:
- Counters of bytes travelling through the measurement points
- Number of users (IP addresses / MAC addresses)
- Number of bytes per user (without flow/packet information)
- Channel quality (in wireless networks)
Single-point VS Multi-point

Depending on the analysis task, traffic can be captured at different monitoring points.

In general it is much simpler to observe traffic at the ingress/egress links of the target network (e.g., customs and border protection).
Sometimes, one is interested in the performance of a specific application rather than the overall network.

Example: monitor the performance of HTTP downloads through different providers.

In these cases it is easier to get measurements through specific application tests rather than synthesise them from the packet-level or flow-level data.
Data aggregation

Passive monitoring (both packet- and flow-level) suffers from a lack of scalability (too much data!)

Data reduction is needed (preferably in an online fashion)

Three methods:
- Aggregation (e.g., total traffic passing through an interface)
- Filtering (e.g., selecting data from a particular source / protocol / application)
- Sampling (random/pseudorandom selection of data)

Aggregation and filtering require to know the traffic features of interest in advance. Sampling does not.
**Sampling**

Uniform sampling:

- **Systematic** (retain only $M \geq 1$ packets every $N$, or every $T$ seconds)
- **Random** (select $M \geq 1$ packets every $N$, with $N \sim \mathcal{U}(a, b)$)
- **Stratified** (first assigns packets to strata according to an attribute, then sample from strata)

Non-uniform sampling:

- Select a packet with probability $p$ that depends on the packet content

Sampling is a lossy process and has impact on the quality of the analysis (apply it carefully!)
Sampling error

Example: estimate the average size of $N$ flows, sampling only a subset $M << N$

The distribution of IP flow size is known to be heavy-tailed, therefore the inclusion/exclusion of some samples can lead to large changes in the estimate!


Active Measurements
Active measurements are primarily used for testing the correct behaviour of a network. Most common tools:

- Simple Network Monitoring Protocol (SNMP)
- Packet INternet Groper (ping)
- traceroute
- iperf
SNMP: Simple Network Monitoring Protocol

- Application protocol to get information from network devices
- A *manager* retrieves data from *agents* running on the network devices.
- Data is described in a Management Information Bases (MIB) through OID (Object Identifiers)
SNMP messages

- GET messages: transmitted from the manager to ask for information from an agent
- SET messages: used to change an agent device configuration
- RESPONSE message: transmitted from agents to reply to a GET/SET message
- TRAP message: proactively sent by agents to a manager when an unexpected event occurs
- INFORM message: similar to TRAP, with ACK.

All messages are transported using UDP (port 161 for GET/SET and RESPONSE, 162 for TRAP and INFORM messages)
An Object IDentifier (OID) corresponds to a node in the OID tree.

The root of the tree defines the standard used (0:ITU-T, 1:ISO, 2:joint)

‘Dot’ notation used to progress into the tree (e.g. 1.3.6.1.2.1.1)

For SNMP, all OID start with 1.3.6.1.2

`snptranslate` translates a numerical OID into a ‘readable’ information

`snmpwalk` retrieve a subtree of OID using multiple GET messages
SNMP Examples

Ask system information to a DEIB network printer (131.175.126.65, 3rd floor). OID: 1.3.6.1.2.1.1.1

```
MacBook-Pro-di-Alessandro-2:~ alessandro$ snmpwalk -c public 131.175.126.65 1.3.6.1.2.1.1.1
SNMPv2-MIB::sysDescr.0 = STRING: Xerox WorkCentre 5875 v1; SS 073.190.075.34540, NC 073.195.34540, UI 073.195.34540, ME 096.074.000, CC 073.195.34540, DF 110.011.021, FI 003.064.000, FA 003.012.005, CCOS 073.005.34540, NCOS 073.005.34540, SC 002.098.000, SU 073.195.34540
```

Ask number of network interfaces. OID: 1.3.6.1.2.1.2.1

```
MacBook-Pro-di-Alessandro-2:~ alessandro$ snmpwalk -c public 131.175.126.65 1.3.6.1.2.1.2.1
IF-MIB::ifNumber.0 = INTEGER: 2
```

Ask description of first interface: OID: 1.3.6.1.2.1.2.2.1.2

```
MacBook-Pro-di-Alessandro-2:~ alessandro$ snmpwalk -c public 131.175.126.65 1.3.6.1.2.1.2.2.1
IF-MIB::ifDescr.1 = STRING: Xerox Embedded Ethernet Controller, 10/100/1000 Mbps, v1.0, RJ45, 100 Mbps full duplex
```
SNMP Examples

Ask for total bytes received on the first interface: OID:
1.3.6.1.2.1.2.2.1.10.1

```
MacBook-Pro-di-Alessandro-2:~ alessandro$ snmpwalk -c public 131.175.126.65 1.3.6.1.2.1.2.2.1.10.1
IF-MIB::ifInOctets.1 = Counter32: 510936250
```

Ask for total bytes transmitted on the first interface: OID:
1.3.6.1.2.1.2.2.1.16.1

```
MacBook-Pro-di-Alessandro-2:~ alessandro$ snmpwalk -c public 131.175.126.65 1.3.6.1.2.1.2.2.1.16.1
IF-MIB::ifOutOctets.1 = Counter32: 74844833
```
SNMP versions

- V1: initial implementation, poor security. It uses a community string (-c) transmitted in clear text as authentication method.
- V2: two versions (V2 and V2c). V2 included GetBulkRequests messages and improvements in security but abandoned (overly complex). Version V2c comprises V2 without the complex security model and was the de-facto standard until 2004.
- V3: Adds cryptographic security and many other improvements.

In practice, SNMP implementations often support multiple versions.
**PING**

*ping*: Packet INternet Groper

- Written in 1983 as a tool to troubleshoot problems in IP networks
- Based on the Internet Control Message Protocol (ICMP)
- Measures the round-trip time (RTT) between the source and a given destination

![Diagram of ICMP ECHO REQUEST and ECHO REPLY]

```
MacBook-Pro-di-Alessandro-2:~ alessandro$ ping -c 10 10.79.1.12
PING 10.79.1.12 (10.79.1.12): 56 data bytes
64 bytes from 10.79.1.12: icmp_seq=0 ttl=64 time=669 ms
64 bytes from 10.79.1.12: icmp_seq=1 ttl=64 time=669 ms
64 bytes from 10.79.1.12: icmp_seq=2 ttl=64 time=669 ms
64 bytes from 10.79.1.12: icmp_seq=3 ttl=64 time=669 ms
64 bytes from 10.79.1.12: icmp_seq=4 ttl=64 time=669 ms
64 bytes from 10.79.1.12: icmp_seq=5 ttl=64 time=669 ms
64 bytes from 10.79.1.12: icmp_seq=6 ttl=64 time=669 ms
64 bytes from 10.79.1.12: icmp_seq=7 ttl=64 time=669 ms
64 bytes from 10.79.1.12: icmp_seq=8 ttl=64 time=669 ms
64 bytes from 10.79.1.12: icmp_seq=9 ttl=64 time=669 ms

--- 10.79.1.12 ping statistics ---
10 packets transmitted, 10 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.550/0.644/0.699/0.071 ms
```
RTT

The RTT depends on packet size $L$, link capacity $C$, link length $d$ and propagation speed $d/v$.

\[ \text{RTT} = 2\left(\frac{L}{C} + \frac{d}{v}\right) + \nu, \]  

where $\nu$ is a noise term.

On fiber links, $\nu \sim 2 \times 10^8 \text{ m/s} = 200 \text{ Km/ms}$.

Therefore one may expect an RTT of 1 ms every 100 Km.

A round trip around the world at the equator via a fiber route would take $\sim 400$ ms.
Traceroute

*traceroute* can be used to analyse a route from a source to a given destination.

How does it work?

Send an ICMP Echo request/UDP packet to the destination with TTL = 1. The first router will decrement the TTL and reply to the source with an ICMP TTL exceeded message. Now we know the address of the first router on the route. Send another ICMP Echo request with TTL = 2. Now we know the address of second router. Continue up to a pre-defined TTL (max number of hops). Issue: no guarantee that every packet will follow the same path (routing decisions, load balancers...).
traceroute can be used to analyse a route from a source to a given destination.

How does it work?

- Send an ICMP Echo request/UDP packet to the destination with TTL = 1. The first router will decrement the TTL and reply to the source with an ICMP TTL exceeded message. Now we now the address of the first router on the route.
Traceroute

`traceroute` can be used to analyse a route from a source to a given destination.

**How does it work?**

- Send an ICMP Echo request/UDP packet to the destination with TTL = 1. The first router will decrement the TTL and reply to the source with an ICMP TTL exceeded message. Now we now the address of the first router on the route.
- Send another ICMP Echo request with TTL = 2. Now we now the address of second router.
**Traceroute**

*traceroute* can be used to analyse a route from a source to a given destination.

**How does it work?**

- Send an ICMP Echo request/UDP packet to the destination with TTL = 1. The first router will decrement the TTL and reply to the source with an ICMP TTL exceeded message. Now we now the address of the first router on the route.
- Send another ICMP Echo request with TTL = 2. Now we now the address of second router.
- Continue up to a pre-defined TTL (max number of hops).

**Issue:** no guarantee that every packet will follow the same path (routing decisions, load balancers...)
Iperf

Iperf is another widely used tool for active network performance measurement:

- Client/Server paradigm (must be active on both ends)
- Measures UDP/TCP/SCTP/ bandwidth, packet loss, etc...
- Cross-platform (Win, Linux, Mac, Android, iOS)
Iperf example

```
|antlab@precision:~$ iperf -s

Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)

[ 4] local 10.79.1.12 port 5001 connected with 10.48.44.118 port 56305
[ ID] Interval     Transfer     Bandwidth
[ 4]   0.0-10.3 sec  15.0 MBytes  12.2 Mbits/sec
```

```
MacBook-Pro-di-Alessandro-2:iperf-2.0.5-macos-x86_64 alessandro$ ./iperf -c 10.79.1.12

Client connecting to 10.79.1.12, TCP port 5001
TCP window size: 128 KByte (default)

[ 4] local 10.48.44.118 port 56305 connected with 10.79.1.12 port 5001
[ ID] Interval     Transfer     Bandwidth
[ 4]   0.0-10.2 sec  15.0 MBytes  12.4 Mbits/sec
MacBook-Pro-di-Alessandro-2:iperf-2.0.5-macos-x86_64 alessandro$
```
Passive Measurements
Passive Measurements

Measurements of existing traffic carried out by *observing* network packets (without injection of new traffic).

Most techniques capture and analyze only packet headers. On-going discussion and legal issues when inspecting packet payloads (forbidden in many countries due to privacy laws).

Main limitation: dealing with massive amount of data!

Golden standard tools: *libpcap, tcpdump*
tcpdump prints out a description of the contents of packets on a network interface that match a boolean expression.

Packets can be saved to a .pcap file for later analysis. This is the starting point of several network traffic analysis experiments.

Basic usage: tcpdump [-i interface], capture all traffic on a particular network interface (Ethernet, Wi-Fi)
tcpdump filters

*tcpdump* expression allow to filter out and select particular types of packets

Some examples:

- `tcpdump host 1.2.3.4` (shows traffic incoming from and outgoing to 1.2.3.4)
- `tcpdump src/dst 1.2.3.4` (shows traffic from/to 1.2.3.4)
- `tcpdump src/dst port 1234` (shows only traffic to/from port 1234)
- `tcpdump udp` (shows only udp packets. works also for icmp and udp)

Many other options are available. Logical operators (and, or, not) can be also used to combine different filters.
tcpdump example

```bash
sudo tcpdump -i eth0 -n icmp
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
11:23:54.916296 IP 10.79.1.193 > 10.79.1.12: ICMP echo request, id 44549, seq 0, length 64
11:23:54.916319 IP 10.79.1.12 > 10.79.1.193: ICMP echo reply, id 44549, seq 0, length 64
11:23:55.919765 IP 10.79.1.193 > 10.79.1.12: ICMP echo request, id 44549, seq 1, length 64
11:23:55.919786 IP 10.79.1.12 > 10.79.1.193: ICMP echo reply, id 44549, seq 1, length 64
11:23:56.920277 IP 10.79.1.193 > 10.79.1.12: ICMP echo request, id 44549, seq 2, length 64
11:23:56.920299 IP 10.79.1.12 > 10.79.1.193: ICMP echo reply, id 44549, seq 2, length 64
11:23:56.920700 IP 10.79.1.12 > 10.79.1.193: ICMP echo request, id 44549, seq 3, length 64
11:23:57.920721 IP 10.79.1.193 > 10.79.1.12: ICMP echo reply, id 44549, seq 3, length 64
11:23:58.920978 IP 10.79.1.193 > 10.79.1.12: ICMP echo request, id 44549, seq 4, length 64
11:23:58.921000 IP 10.79.1.12 > 10.79.1.193: ICMP echo reply, id 44549, seq 4, length 64
11:23:59.921630 IP 10.79.1.193 > 10.79.1.12: ICMP echo request, id 44549, seq 5, length 64
11:23:59.921882 IP 10.79.1.12 > 10.79.1.193: ICMP echo reply, id 44549, seq 5, length 64
11:24:00.928540 IP 10.79.1.12 > 10.79.1.193: ICMP echo request, id 44549, seq 6, length 64
11:24:00.928561 IP 10.79.1.193 > 10.79.1.12: ICMP echo reply, id 44549, seq 6, length 64
11:24:01.933764 IP 10.79.1.193 > 10.79.1.12: ICMP echo request, id 44549, seq 7, length 64
11:24:01.933786 IP 10.79.1.12 > 10.79.1.193: ICMP echo reply, id 44549, seq 7, length 64
```

```bash
64 bytes from 10.79.1.12: icmp_seq=7 ttl=64 time=0.684 ms
64 bytes from 10.79.1.12: icmp_seq=8 ttl=64 time=0.674 ms
64 bytes from 10.79.1.12: icmp_seq=9 ttl=64 time=0.556 ms
^C
10.79.1.12 ping statistics --
10 packets transmitted, 10 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.556/0.639/0.700/0.052 ms
MacBook-Pro-di-Alessandro-2:~ alessandro
MacBook-Pro-di-Alessandro-2:~ alessandro ping 10.79.1.12
PING 10.79.1.12 (10.79.1.12): 56 data bytes
64 bytes from 10.79.1.12: icmp_seq=0 ttl=64 time=0.648 ms
64 bytes from 10.79.1.12: icmp_seq=1 ttl=64 time=0.688 ms
64 bytes from 10.79.1.12: icmp_seq=2 ttl=64 time=0.607 ms
64 bytes from 10.79.1.12: icmp_seq=3 ttl=64 time=0.686 ms
64 bytes from 10.79.1.12: icmp_seq=4 ttl=64 time=0.651 ms
64 bytes from 10.79.1.12: icmp_seq=5 ttl=64 time=0.656 ms
64 bytes from 10.79.1.12: icmp_seq=6 ttl=64 time=0.570 ms
64 bytes from 10.79.1.12: icmp_seq=7 ttl=64 time=0.677 ms
^C
10.79.1.12 ping statistics --
8 packets transmitted, 8 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.570/0.647/0.688/0.038 ms
MacBook-Pro-di-Alessandro-2:~ alessandro
```
Other tools for passive measurements

Other tools based on the *libpcap* engine are available: *dumpcap*, *tshark*.

*WireShark* in particular provides a nice graphical interface and a packet dissector for many protocols.

The output of these tools is generally a `.pcap` file. One tool that can be used to parse and process such file in an easy way is Python *pyshark* library.
Introduction
Network measurements
Active Measurements
Passive Measurements
Data visualization

Wireshark example

Alessandro E. C. Redondi
Network Traffic Measurements and Analysis
Data visualization
Active and passive measurements generally produce lots of data.

The most important thing to do when first exploring such data is to visualize it through graphs.

Data visualization quickly and easily reveals basic features of the data, including patterns and unusual observations.
Visualization example

RTT measurements (PING) from DEIB to google.com, passing or not passing through a VPN

Simplest visualization: time series

What can we say about the use of a VPN?
Another visualization example

Weekly downlink/uplink usage of a PoliMI Wi-Fi access point:

What can we say about it?
Guidelines for plotting

- Label the axes clearly
- Label the tics on the axes
- Use legend to label individual curves
- Select appropriate font size
- Select range properly
- Do not present too many items in a single chart
- When comparing plots, use same scale for the axes
- When using colors, make sure it's ok in bw
Summary Statistics

Often, it is useful to represent data using *summary statistics*, rather than the complete set of observations. Such description is performed through:

- A measure of location or central tendency (mean, median, mode)
- A measure of spread / dispersion (standard deviation, variance, range)
measures of location

- mean: average, sensitive to outliers

\[ \bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i \]  

- median: middle value (50th-percentile), adds robustness
- mode: value with highest frequency observed
Introduction
Network measurements
Active Measurements
Passive Measurements
Data visualization

Measures of dispersion

- range: difference between the max and min observed values
- variance:

\[
\sigma^2 = \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})^2 = \frac{1}{N} \sum_{i=1}^{N} x_i^2 - \bar{x}^2
\]  

- standard deviation: \( \sigma \)
  - most common measure of statistical dispersion
  - can be directly compared with the mean (same unit)
  - for a quantity with normal (Gaussian) distribution:
    - 68% of the measurements fall into \( \bar{x} \pm \sigma \)
    - 95% fall into \( \bar{x} \pm 2\sigma \)
    - 99.7% fall into \( \bar{x} \pm 3\sigma \)
The summary table is a visualization that summarizes the statistical information about data in table form.

<table>
<thead>
<tr>
<th></th>
<th>Min</th>
<th>Max</th>
<th>Mean</th>
<th>Median</th>
<th>Std Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTT without VPN</td>
<td>1.6160</td>
<td>3.7240</td>
<td>1.8157</td>
<td>1.8075</td>
<td>0.1832</td>
</tr>
<tr>
<td>RTT with VPN</td>
<td>2.2620</td>
<td>6.7580</td>
<td>2.6456</td>
<td>2.6105</td>
<td>0.3136</td>
</tr>
</tbody>
</table>
Visualizing distribution

Instead of plotting the raw data as a time series, one can plot its distribution (histogram):

- split the data into equal-sized bins by value
- count the frequency of each bin
- normalize the count by the total number of observations (so that the histogram sums up to 1)
Why distribution?

When visualizing a distribution, it is easier to identify:

- center (location) of the data
- spread (scale) of the data
- skewness of the data
- presence of outliers
- presence of multiple modes in the data

Limitations of distributions/histograms

- needs appropriate bin size
- too small: each bin doesn’t have enough samples (empty bins)
- too large: only few regions available
- in general, needs large number of samples
Cumulative Density Function

Distribution / density function is the probability of observing value $x$ for variable $X$

$$f(x) = P[X = x]$$ (4)

Cumulative density function: probability that $X$ is less or equal than $x$

$$F(x) = P[X \leq x]$$ (5)

It’s a better visualization tool than pdf when the number of samples is low
Introduction
Network measurements
Active Measurements
Passive Measurements
Data visualization

CDF

Empirical CDF

RTT [ms]
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
F(RTT)

without VPN
with VPN

Alessandro E. C. Redondi
Network Traffic Measurements and Analysis
Boxplots

A nice graphical representation of an observation can be obtained with boxplots, where you can show:

- median value
- 25th and 75th percentile
- max and min
- outliers

![Boxplot Diagram]

<table>
<thead>
<tr>
<th>RTT [ms]</th>
<th>without VPN</th>
<th>with VPN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Scatter Plots

What can we say about this data?

![Scatter Plots]

Alessandro E. C. Redondi
Network Traffic Measurements and Analysis
Often, it is useful to plot two quantities against each other.
A convenient representation is the *scatterplot matrix* which can be used to identify relationships between couples of variables.
To do list

- Install Matlab (free license for Polimi students) or Octave
- Install Python with scikit-learn, matplotlib and pyshark
- Try the Matlab visualization examples available on the course website
Assignment

- Perform RTT measurements to different IPs in the world.
- A list of Major IP blocks assigned by country is available here http://www.nirsoft.net/countryip/
- For each IP tested, keep track of the distance between yourself and the location of the IP. Several tools are available to retrieve the approximative location of an IP (e.g., https://www.iplocation.net/)
- Try to automate the process, e.g., write a Matlab/Python script to do that (you can call system programs from both environments)!
- Goal: produce a file with many (RTT, distance) couples. Plot it on a scatter plot