DCML-CPS - Module 2

Monitoring

Tommaso Zoppi
University of Trento – Povo (IT)
tommaso.zoppi@unifi.it
tommaso.zoppi@unitn.it

UNIVERSITÀ DEGLI STUDI FIRENZE
DIMAI
DIPARTIMENTO DI MATEMATICA E INFORMATICA "ULISSE DINI"

Course Map

1. Basics and Metrology
2. Monitoring

Monitoring

Testing

3. Fault Injection
4. Robustness Testing

5. Data Analysis
6. Supervised ML
7. Unsupervised ML
8. Meta-Learning
9. Error/Intrusion Detection

Anomaly Detection

Tools & Libs

Deep Learning

Basics of Monitoring

Monitoring - Aim

Aim: to constantly monitor the system running in its final environment, verifying that the observed behavior and performance meet well-defined rules (requirements)

UNIVERSITÀ DEGLI STUDI FIRENZE
DIMAI
DIPARTIMENTO DI MATEMATICA E INFORMATICA "ULISSE DINI"

Monitoring and Verification

- Monitoring is usually paired by a **verification** activity to analyze gathered data
  - it can be done at **run-time**
    - Example: attempts to restore from a run-time fault, and diagnosis and maintenance activities directly on the runtime system

  - or **off-line**, examining the results afterwards

Target System

- The system to which we apply the monitoring, control and verification activities is called the target system

- The monitored hardware component or software application are called target component or target application

Monitoring - Purpose

- It can be used to take decisions about the management of a system, with the final objective of controlling its behavior

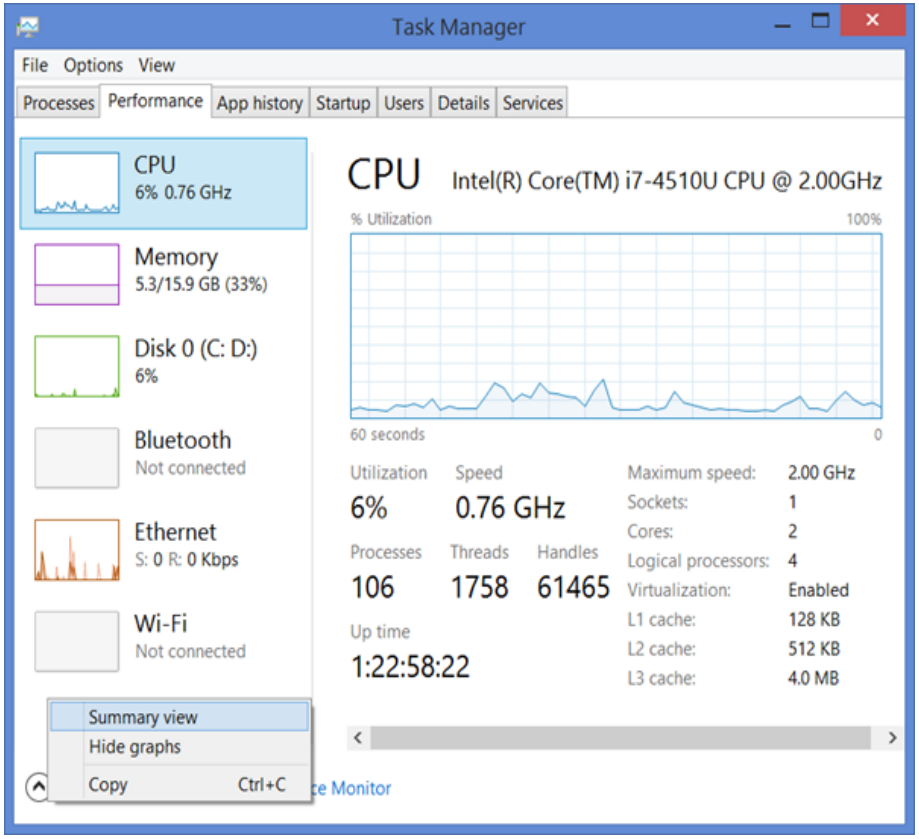
- It can also be used for debugging and to evaluate the **performance** or **quality of service** (QoS) of a system
  - The online monitoring has been recognized as a viable approach to evaluate system attributes of dependability, security, performance, correctness…

Components of Monitors

- For the monitoring of computer systems we have:
  - One or more targets, or rather **elements to be monitored** (e.g., computers, operating systems, application packages)
  - Signals, or **measurements**, collected while observing the behavior of these elements
  - A **monitoring node** (logically centralized) that can extract and process (automatically or not) the collected data, obtaining the information of interest

Example 1a – Windows' Task Manager

- Which, surprisingly, was not primarily meant to force-close apps.



Example 1a – UNIX TOP Command

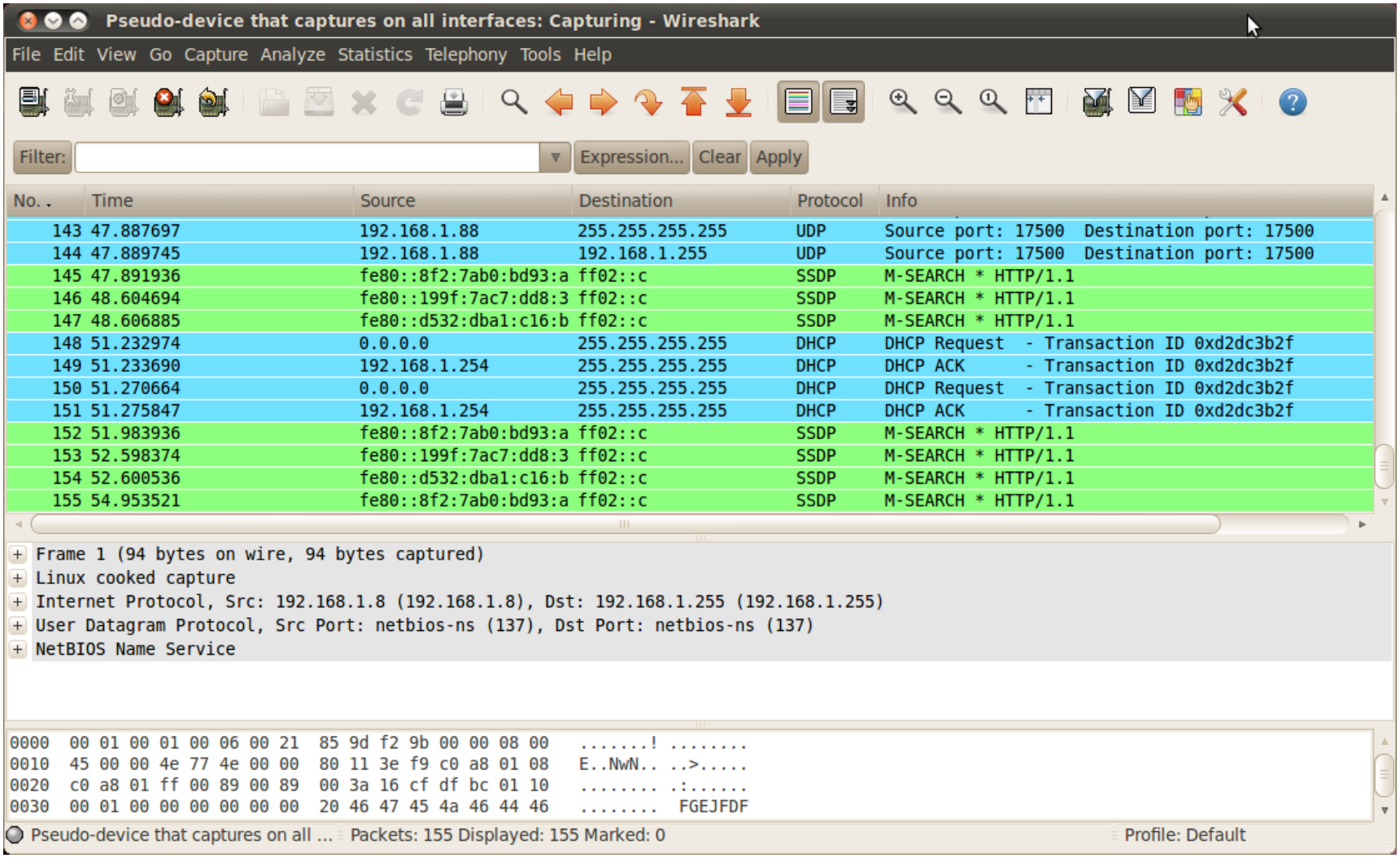
- Top command
- Must-have command in linux/unix environments

Linux Commands for Beginners (Old Version)_ 10 - The top Command.mp4

https://www.youtube.com/watch?v=M4kM0JleVDs

UNIVERSITÀ DEGLI STUDI FIRENZE
DIMAI
DIPARTIMENTO DI MATEMATICA E INFORMATICA "ULISSE DINI"

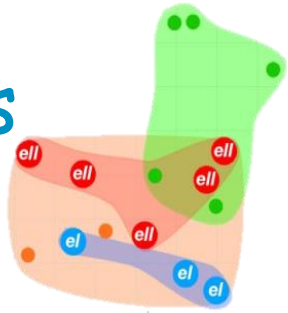
Example 2 – Network monitor (Wireshark - *https://www.wireshark.org/*)



Example 3 – Metro (San Paolo)

Technical Problems

- Indeed, we typically need to face and solve different **technical problems**, in relation to:
  - **Identification of the events or measures** of interest
  - **Labeling** data, with sufficient information to gather useful measurements
  - **Transmission** of the data to the node / system where they will be processed and used (if different from target system)
  - Filtering and **classification** of the events wrt the measurements of interest



Black Box System / Component

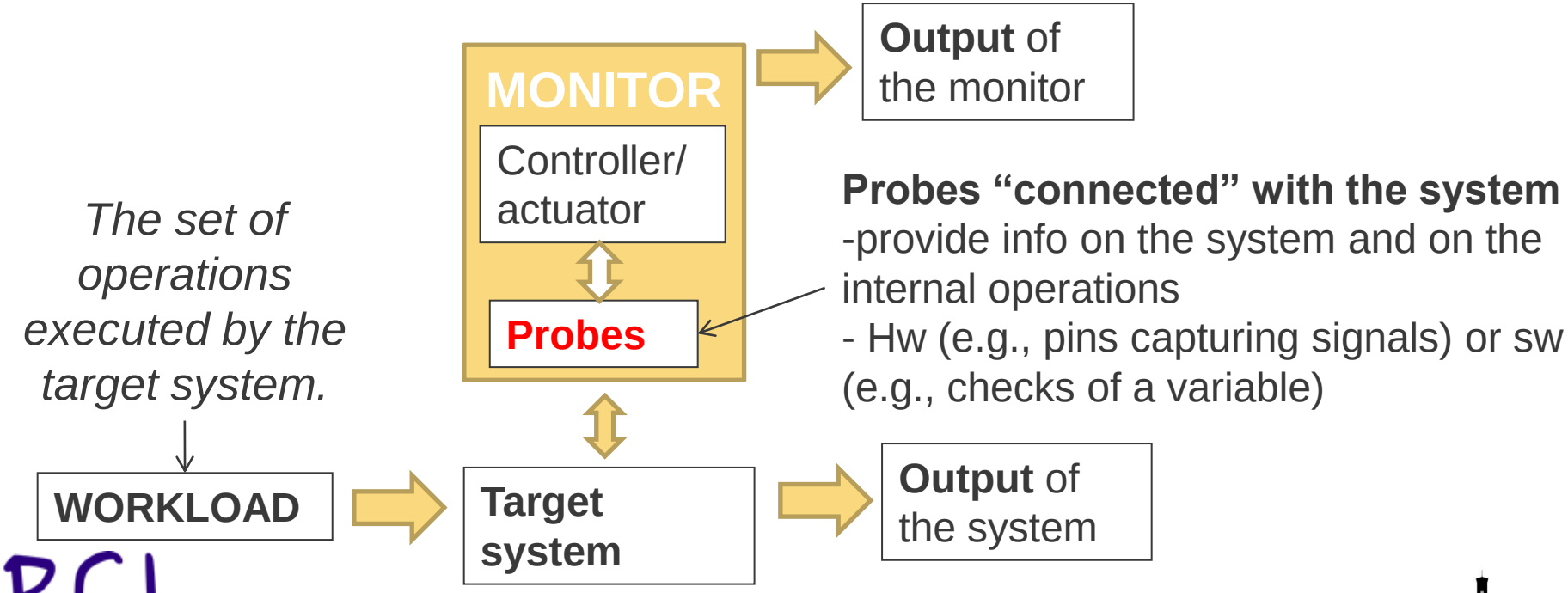
- Systems can be viewed depending on the knowledge that we have of their internals
  - When the details of the systems are known, or freely accessible (e.g., source code is available, allowing customizing software), it is possible to observe its internals (e.g., amount of memory used)
  - When the system is developed by a third-party, or when its details are unknown, we can only submit inputs and wait for outputs (e.g., Google's search)
    - In this case, the system is considered as **black box**

UNIVERSITÀ DEGLI STUDI FIRENZE
DIMAI
DIPARTIMENTO DI MATEMATICA E INFORMATICA "ULISSE DINI"

Black vs White-Box

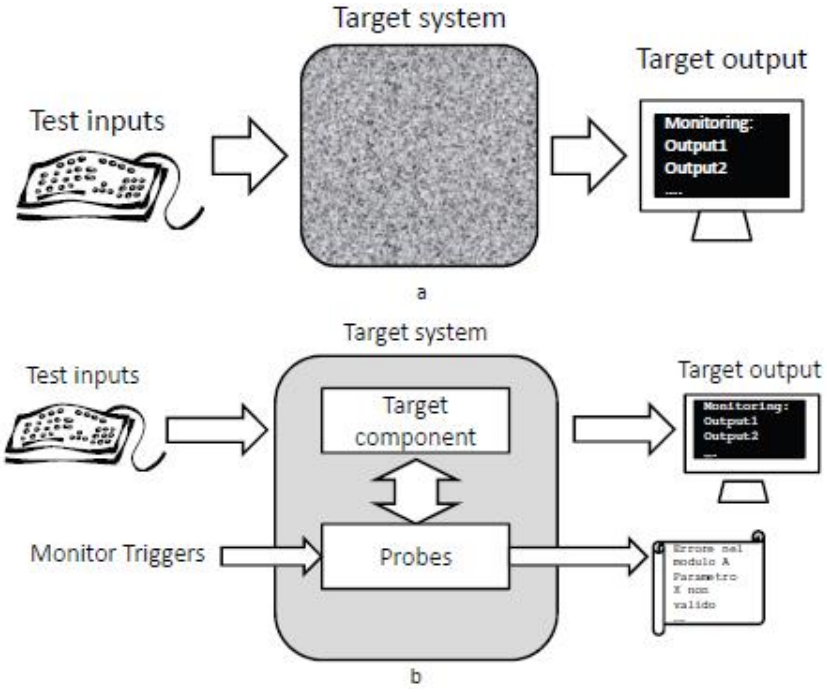
a) Target System as a **black box**

b) Target system instrumented with a monitor



Means for monitoring: Probes

- **Probes** gather data from a target system.
  - the probes can be positioned either inside or outside the system;
  - their purpose is to provide information on system operation



Probing: Basic rules

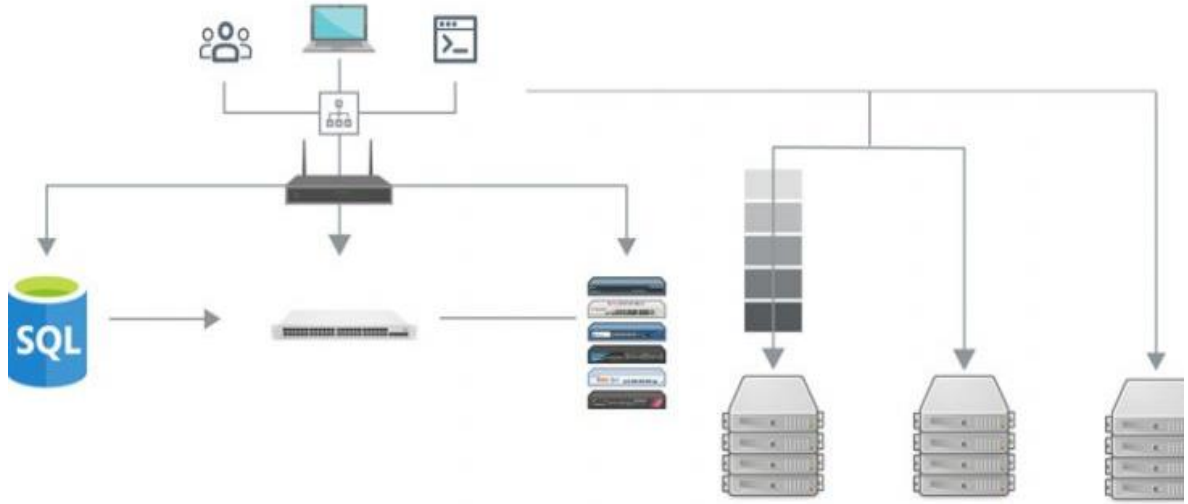
- **Selection**: the probes must be able to observe an adequate number of meaningful information to meet the objectives of the monitoring activity
  - Beware of the data deluge!

- **Intrusiveness**: The behavior of the target system must not be affected by the probes

HW / SW / Hybrid Monitoring

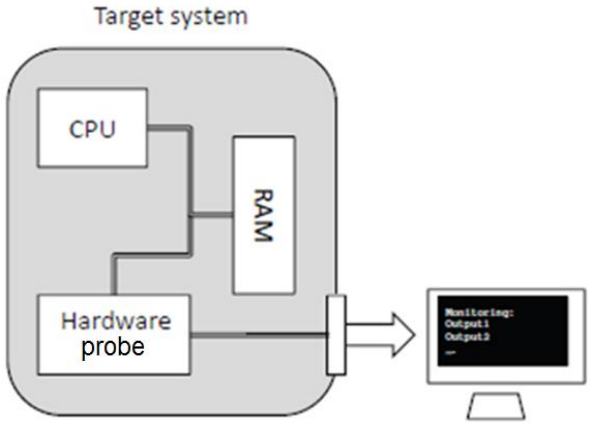
Probing: Basic rules

- Depending on where we decide to put probes, monitoring can be defined as:
  - **Hardware** Monitoring
  - **Software** Monitoring
  - **Hybrid** Monitoring



Hardware monitoring - 1

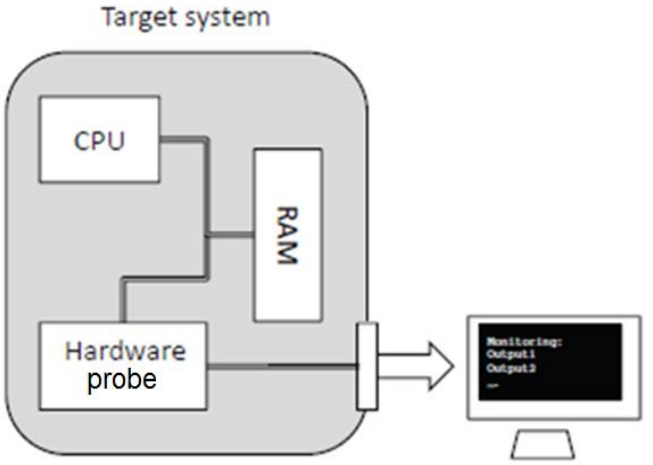
- Uses a **dedicated hardware circuit** connected to target system
  - "reads" the information to monitor and send it to external hardware



- Pros ?
- Cons?

Hardware monitoring - 2

- Uses a **dedicated hardware circuit**
  - "reads" information and sends it to external HW
  - Pros:
    - **greatly reduces the intrusiveness** of the monitoring system, which does not execute any additional code



  - Cons:
    - Systems are becoming more and more complex, making it **difficult or impossible to install hardware probes**
    - This further reduced the visibility of the execution information from components external to the target system

HW Monitoring: Example

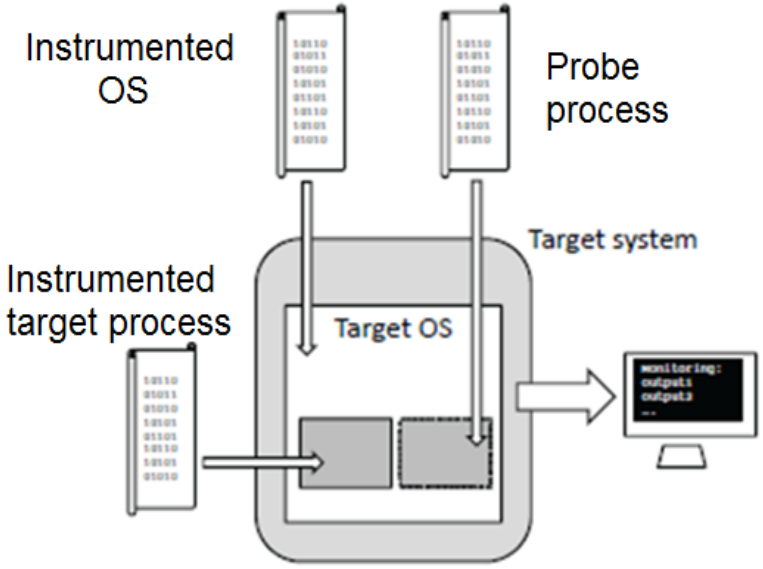
- SMART monitoring for memory drives ->

https://www.youtube.com/watch?v=mfkLik1RGJ8

SMART.mp4

Software monitoring - 1

- Three types of "probe" code:
  - Case 1: inserted in the code of the **target process** or application
  - Case 2: inserted **in the operating system**
  - Case 3: a separate **probe process**



Software monitoring - 2

- Pros?

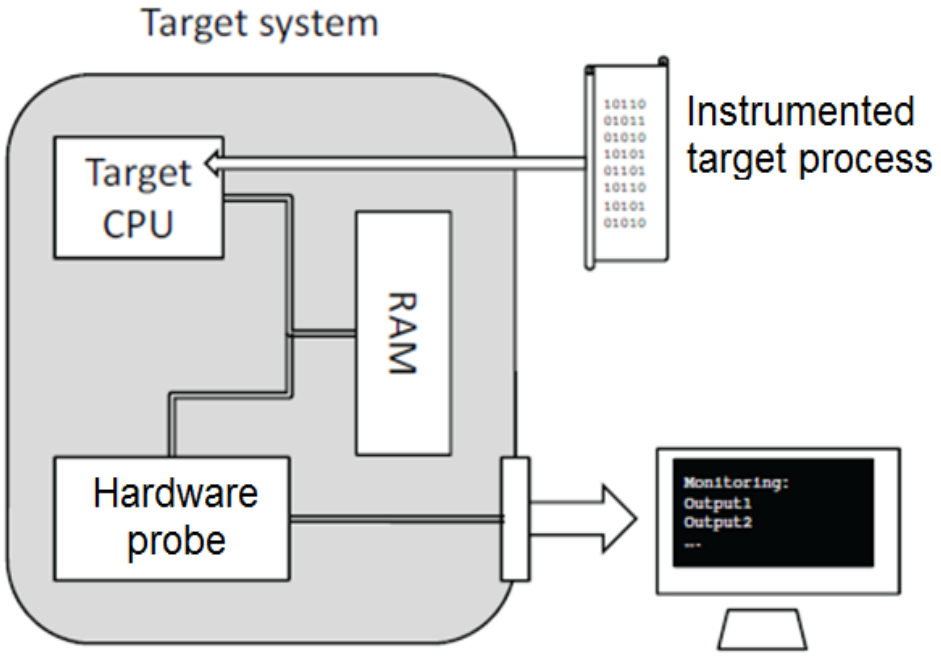
- Cons?

Software monitoring - 2

- Pros
  - Software probes, especially in complex systems, can have **access to more information** with respect to hardware probes
- Cons
  - However, executing additional code can be **intrusive**
  - A (partial) solution is to insert in the target system some permanent probes, which become part of the system itself

Hybrid Monitoring (hw/sw)

- Uses together **hardware** and **software** probes, exploiting the advantages of both the methodologies.
  - It uses HW probes where possible, **minimizing intrusiveness**



Known Issues in System Monitoring

Monitoring Problems

- Sample questions:
  - Is it easy to monitor a distributed system?
  - Is it always allowed to put probes into systems?
  - Where do we analyze data?
  - What about decision-making?

Monitoring Problems - 1

- Direct and indirect observations
  - Many events can not be observed directly,
- Complete and incomplete observations
  - Any observation of a system reveals only a part of the system itself
- Monitoring and intrusiveness
  - The intrusiveness of the measuring instrument can impact the measurements in a non-negligible way
  - Typical relationship between flexibility and generality of
    - the monitoring instrument, implementation cost, intrusiveness

Monitoring Problems - 2

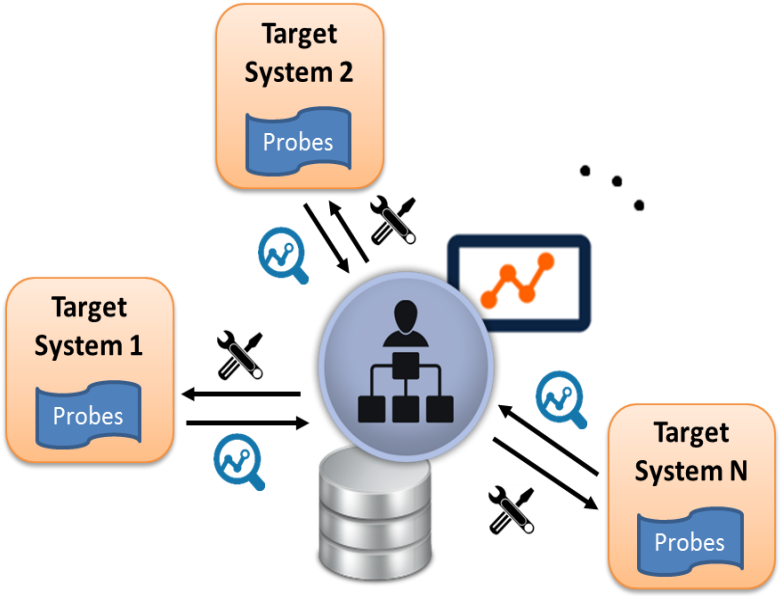
- Presentation of information
  - It is often necessary to manipulate the received information to overcome problems such as:
    - the observed events appear in a form that the observer (user) is not able to use or comprehend easily
    - high occurrence rate of the events
    - high amount of events
    - in a distributed system, the events of interest may occur in different parts of the system (collection and sorting of information)

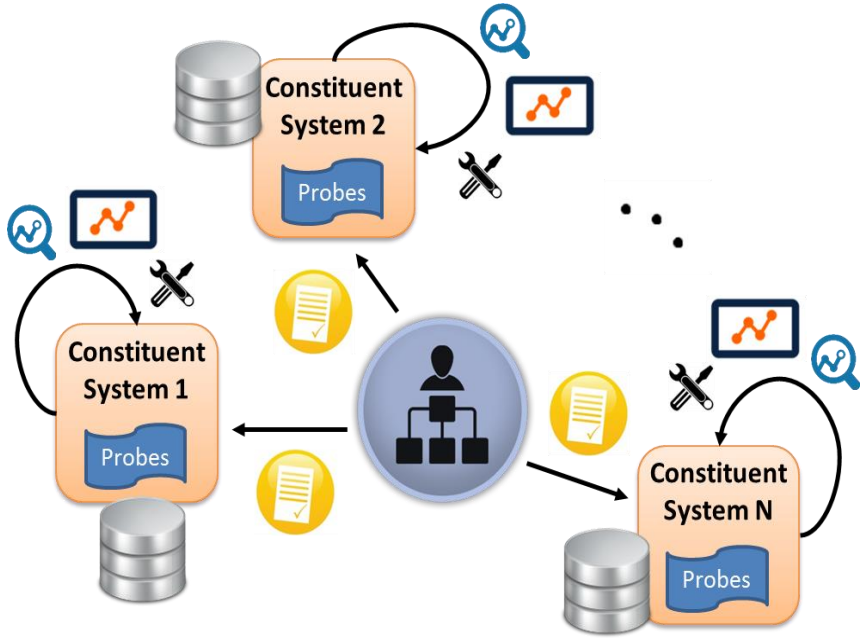
Monitoring Distributed Systems

- Lack of a central control point
  - Need to use different processes controlling different parts of the system
- Absence of a central observation point
  - Need to build global views by the local events observed in the various nodes of the distributed system
- Lack of a central source for monitoring information
  - Strategies are needed for the collection and assembly of observations, to merge the observations from various sources
- Lack of a central point for taking decisions
  - The decision making process in a distributed system can itself be distributed

Possible Monitoring Strategies

- Election of a Leader with Responsibility





Monitoring Coordinator

Monitored Data

Maintenance Actions

Data Storage

- Distributing Responsibilities

UNIVERSITÀ DEGLI STUDI FIRENZE
DIMAI
DIPARTIMENTO DI MATEMATICA E INFORMATICA "ULISSE DINI"

Focusing on Distributed Systems

- Incomplete observability
  - Incomplete information about events that occur in various parts of the system
- Non-determinism
  - Two executions of the same program can produce different sequences of observed events (in a different order)
- Interference in the observations
  - (strongly related to the intrusiveness problem)
- Objects, encapsulation and security
  - In an object-oriented system, the notion of monitoring is opposed to the concept of encapsulation

Examples of Monitors

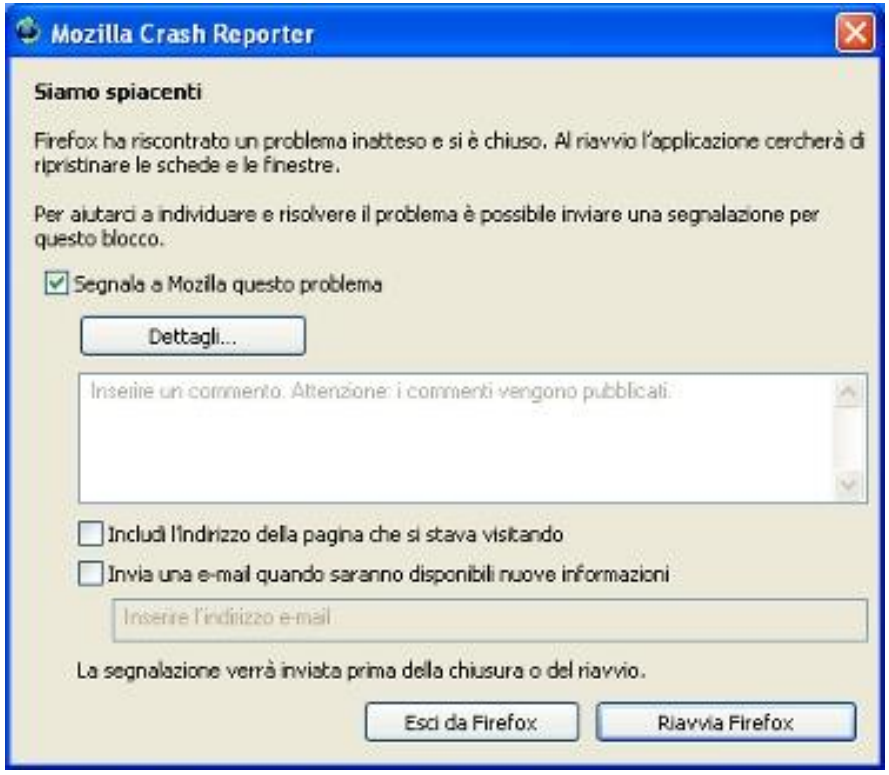
UNIVERSITÀ DEGLI STUDI FIRENZE
DIMAI
DIPARTIMENTO DI MATEMATICA E INFORMATICA "ULISSE DINI"

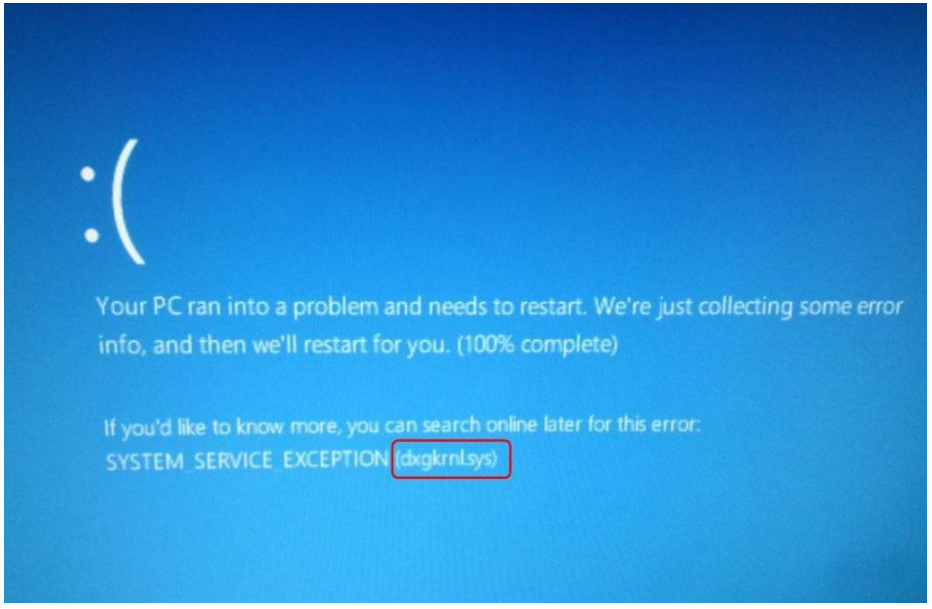
Automatic Failure Reporting for SW

- The commercial open-source software is increasingly using the automatic failure reporting solutions
  - Allows you to find bugs that occur more frequently
    - In Windows XP, when issued, most of the system's failures were due to a very narrow set of bugs
  - Primarily, it allows detection of *crash failures*
    - It mainly serves for facilitate bug fixing, although potentially can be extended to study the reliability of the system
- It can bring benefits to
  - Manufacturers (improves the quality assurance process)
  - Customers (better products thanks to the

UNIVERSITÀ DEGLI STUDI FIRENZE
DIMAI
DIPARTIMENTO DI MATEMATICA E INFORMATICA "ULISSE DINI"

Automatic Failure Reporting for SW



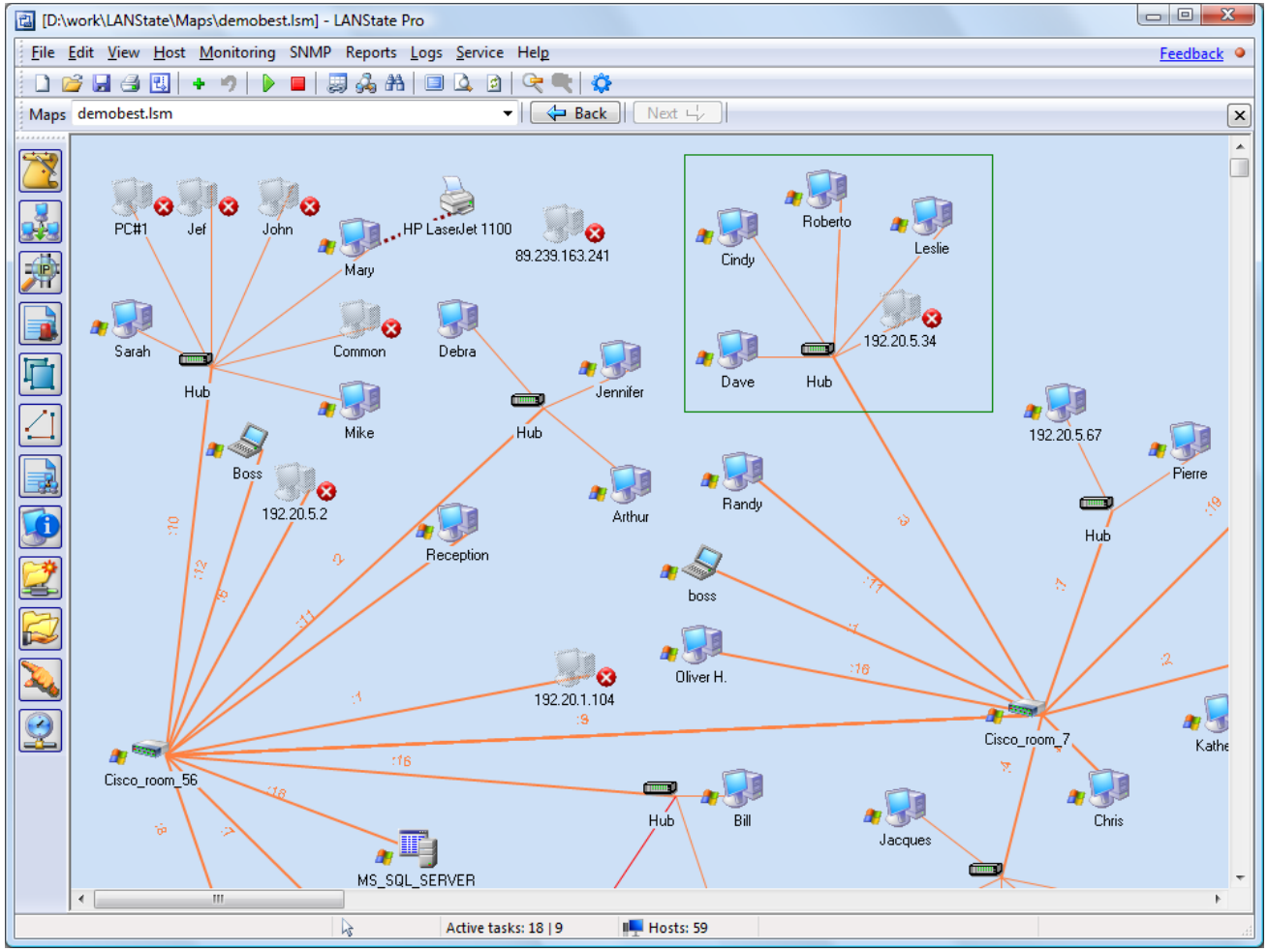


Automatic Failure Reporting for SW

- Two examples:
  - Mozilla Automatic bug tracking
    - Talkback (Quality Feedback Agent - QFA) in the period 2000-2004 has allowed to identify and resolve nearly 800 bugs
    - More than 20000 users have provided information
  - Windows Error Reporting (now removed) in Windows Vista extracted data about a crash of an application (reporting for kernel failures is managed by other tools of Windows Vista) by sending it to Microsoft for analysis.
    - The specific problems of an application were communicated through Windows Error Reporting, under the user's explicit consent

UNIVERSITÀ DEGLI STUDI FIRENZE
DIMAI
DIPARTIMENTO DI MATEMATICA E INFORMATICA "ULISSE DINI"

Network and QoS Monitoring



Network and QoS Monitoring

- Network and QoS monitoring are part of the network management functionalities. They include:
  - Solutions to identify problems caused by failures or network overload and faulty devices
  - Continuous measurements of QoS attributes to allow corrections (corrective management) or to produce measures, e.g.,
    - # packets loss,
    - throughput,
    - round-trip delay
  - Solutions to alert network managers of the presence of viruses or malware, suspicious user activity, etc.

Telemetry

http://www.nasa.gov/centers/kennedy/launchingrockets/comm_telem.html
The Telemetry and Communications Group provides data, voice, video and telemetry for NASA launches around the globe -- even as far away as Alaska and the Marshall Islands. Image credit: NASA

Telemetry of embedded systems

- There is a growing trend to include telemetry capability in systems
  - Allows you to monitor the performance and health of embedded hardware systems that are well supervised remotely
  - Supports maintenance and repair actions
    - aircraft engines, cars, industrial systems ...

Telemetry and Data _ Formula 1 _ Ken Gregory.mp4

https://www.youtube.com/watch?v=1j9r7Ue6XnA&t=2s

Monitoring Networks: Wireshark

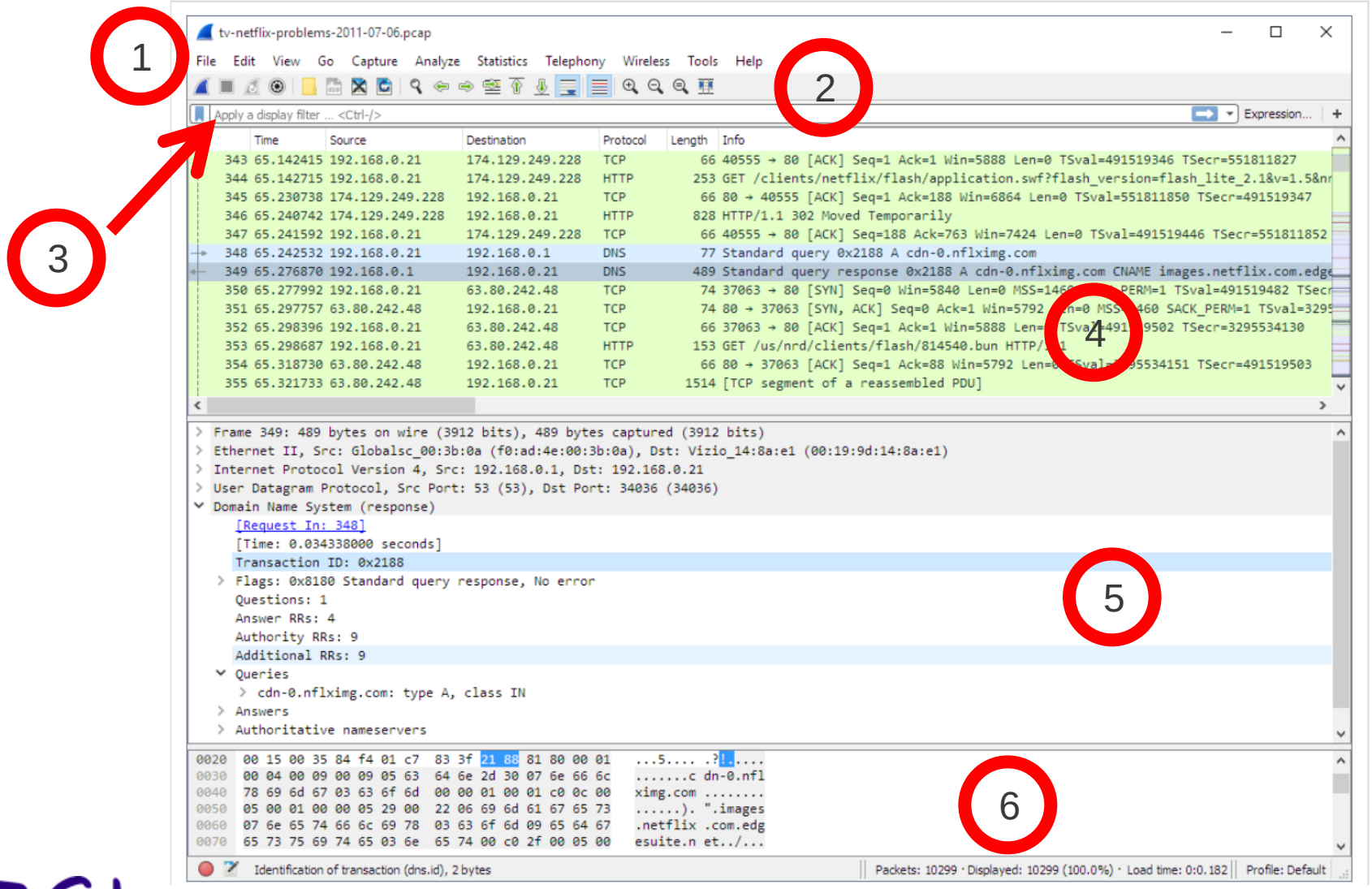
Wireshark in General

- Software Probe
  - In fact, it is open source and downloadable at
    - https://www.wireshark.org/
- Wireshark is the world's foremost and widely-used network protocol analyzer.
  - It lets you see what's happening on your network at a microscopic level and is the de facto (and often de jure) standard across many tools
  - Wireshark development thrives thanks to the volunteer contributions of networking experts since 1998.

Functions at a Glance

- Deep inspection of hundreds of protocols
- Live capture and offline analysis
- Multi-platform: Runs on Windows, Linux, macOS, Solaris, FreeBSD, NetBSD, and many others
- Captured network data can be browsed via a GUI, or via the TTY-mode TShark utility
- Coloring rules can be applied to the packet list for quick, intuitive analysis
- Output can be exported to XML, PostScript®, CSV, or plain text

Main GUI


1
2
3
4
5
6
tv-netflix-problems-2011-07-06.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
Time Source Destination Protocol Length Info
343 65.142415 192.168.0.21 174.129.249.228 TCP 66 40555 → 80 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=491519346 TSecr=551811827
344 65.142715 192.168.0.21 174.129.249.228 HTTP 253 GET /clients/netflix/flash/application.swf?flash_version=flash_lite_2.1&v=1.5&nr
345 65.230738 174.129.249.228 192.168.0.21 TCP 66 80 → 40555 [ACK] Seq=1 Ack=188 Win=6864 Len=0 TSval=551811850 TSecr=491519347
346 65.240742 174.129.249.228 192.168.0.21 HTTP 828 HTTP/1.1 302 Moved Temporarily
347 65.241592 192.168.0.21 174.129.249.228 TCP 66 40555 → 80 [ACK] Seq=188 Ack=763 Win=7424 Len=0 TSval=491519446 TSecr=551811852
348 65.242532 192.168.0.21 192.168.0.1 DNS 77 Standard query 0x2188 A cdn-0.nflximg.com
349 65.276870 192.168.0.1 192.168.0.21 DNS 489 Standard query response 0x2188 A cdn-0.nflximg.com CNAME images.netflix.com.edge
350 65.277992 192.168.0.21 63.80.242.48 TCP 74 37063 → 80 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=491519482 TSecr
351 65.297757 63.80.242.48 192.168.0.21 TCP 74 80 → 37063 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=3295
352 65.298396 192.168.0.21 63.80.242.48 TCP 66 37063 → 80 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=491519502 TSecr=3295534130
353 65.298687 192.168.0.21 63.80.242.48 HTTP 153 GET /us/nrd/clients/flash/814540.bun HTTP/1.1
354 65.318730 63.80.242.48 192.168.0.21 TCP 66 80 → 37063 [ACK] Seq=1 Ack=88 Win=5792 Len=0 TSval=3295534151 TSecr=491519503
355 65.321733 63.80.242.48 192.168.0.21 TCP 1514 [TCP segment of a reassembled PDU]
Frame 349: 489 bytes on wire (3912 bits), 489 bytes captured (3912 bits)
Ethernet II, Src: Globalsc_00:3b:0a (f0:ad:4e:00:3b:0a), Dst: Vizio_14:8a:e1 (00:19:9d:14:8a:e1)
Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.21
User Datagram Protocol, Src Port: 53 (53), Dst Port: 34036 (34036)
Domain Name System (response)
[Request In: 348]
[Time: 0.034338000 seconds]
Transaction ID: 0x2188
Flags: 0x8180 Standard query response, No error
Questions: 1
Answer RRs: 4
Authority RRs: 9
Additional RRs: 9
Queries
cdn-0.nflximg.com: type A, class IN
Answers
Authoritative nameservers
0020 00 15 00 35 84 f4 01 c7 83 3f 21 88 81 80 00 01 ...5.... .?!.....
0030 00 04 00 09 00 09 05 63 64 6e 2d 30 07 6e 66 6c ....c dn-0.nfl
0040 78 69 6d 67 03 63 6f 6d 00 00 01 00 01 c0 0c 00 ximg.com ....
0050 05 00 01 00 00 05 29 00 22 06 69 6d 61 67 65 73 ......). ".images
0060 07 6e 65 74 66 6c 69 78 03 63 6f 6d 09 65 64 67 .netflix .com.edg
0070 65 73 75 69 74 65 03 6e 65 74 00 c0 2f 00 05 00 esuite.n et../...
Identification of transaction (dns.id), 2 bytes
Packets: 10299 · Displayed: 10299 (100.0%) · Load time: 0:0.182
Profile: Default

Main GUI - Details

1. The menu is used to start actions.
2. The main toolbar provides quick access to frequently used items from the menu.
3. The filter toolbar allows users to set display filters to filter which packets are displayed.
4. The packet list pane displays a summary of each packet captured. By clicking on packets in this pane you control what is displayed in the other two panes.
5. The packet details pane displays the packet selected in the packet list pane in more detail.
6. The packet bytes pane displays the data from the packet selected in the packet list pane, and highlights the field selected in the packet details pane.

Examples of Filters (I)

- From
  - https://wiki.wireshark.org/DisplayFilters

Show only SMTP (port 25) and ICMP traffic:

```
tcp.port eq 25 or icmp
```

Show only traffic in the LAN (192.168.x.x), between workstations and servers -- no Internet:

```
ip.src==192.168.0.0/16 and ip.dst==192.168.0.0/16
```

TCP buffer full -- *Source is instructing Destination to stop sending data*

```
tcp.window_size == 0 && tcp.flags.reset != 1
```


Examples of Filters (II)

- From
  - https://wiki.wireshark.org/DisplayFilters

It is also possible to search for characters appearing anywhere in a field or protocol by using the contains operator.

Match packets that contains the 3-byte sequence 0x81, 0x60, 0x03 anywhere in the UDP header or payload:

```
udp contains 81:60:03
```

Match packets where SIP To-header contains the string "a1762" anywhere in the header:

```
sip.To contains "a1762"
```

The matches, or ~, operator makes it possible to search for text in string fields and byte sequences using a regular expression, using Perl regular expression syntax. Note: Wireshark needs to be built with libpcre in order to be able to use the matches operator.

Match HTTP requests where the last characters in the uri are the characters "gl=se":

```
http.request.uri matches "gl=se$"
```

Note: The $ character is a PCRE punctuation character that matches the end of a string, in this case the end of http.request.uri field.

Examples of Filters (III)

- From
  - https://wiki.wireshark.org/DisplayFilters

Some *filter fields* match against multiple *protocol fields*. For example, "ip.addr" matches against both the IP source and destination addresses in the IP header. The same is true for "tcp.port", "udp.port", "eth.addr", and others. It's important to note that

```
ip.addr == 10.43.54.65
```

is equivalent to

```
ip.src == 10.43.54.65 or ip.dst == 10.43.54.65
```

This can be counterintuitive in some cases. Suppose we want to filter out any traffic to or from 10.43.54.65. We might try the following:

```
ip.addr != 10.43.54.65
```

which is equivalent to

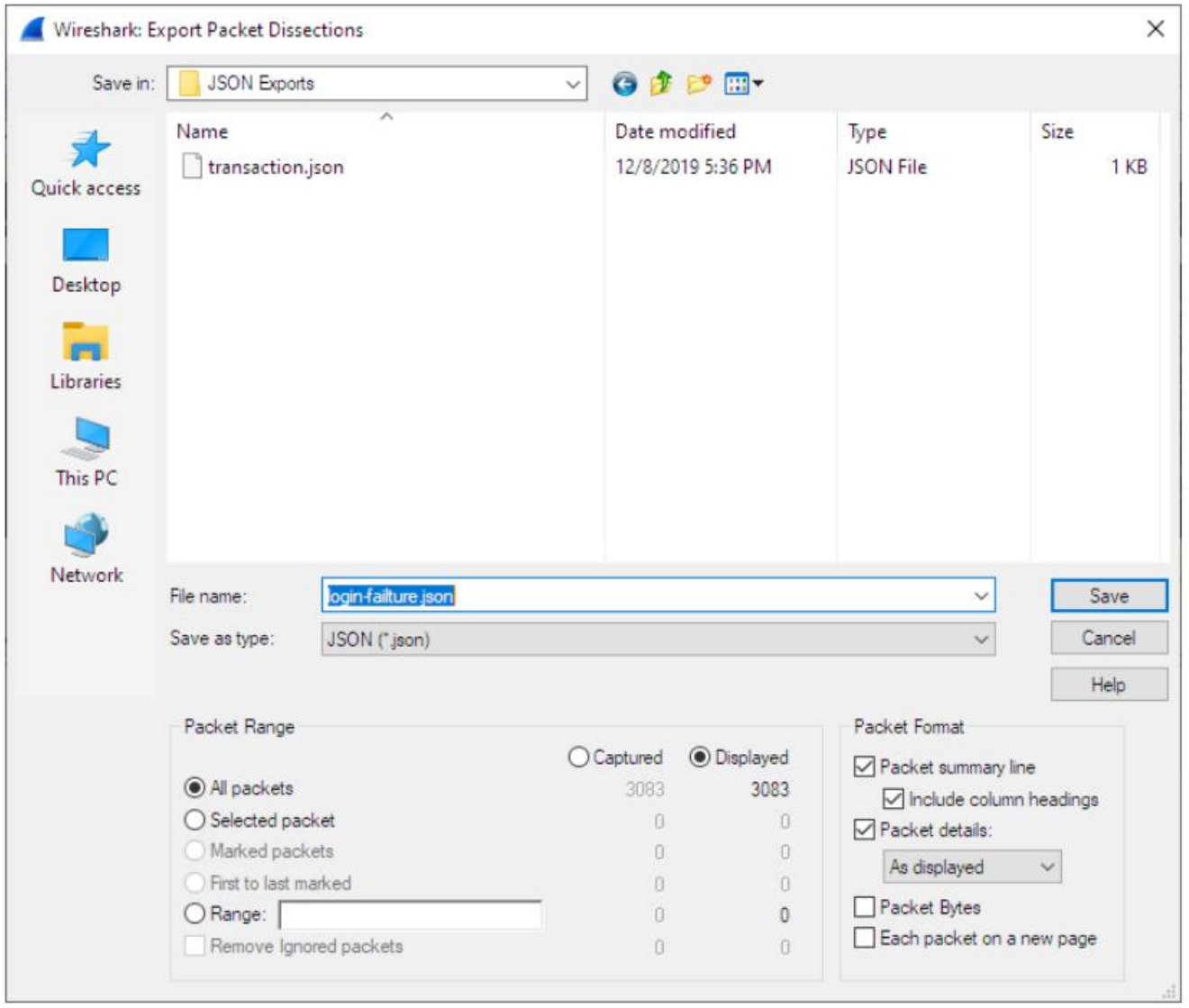
```
ip.src != 10.43.54.65 or ip.dst != 10.43.54.65
```

This translates to "pass all traffic except for traffic with a source IPv4 address of 10.43.54.65 **and** a destination IPv4 address of 10.43.54.65",

Export to File

- Once a Capture has started/ended, it is possible to export results into
  - Plain text as shown in the main window
  - Comma-separated values (CSV)
  - C-compatible byte arrays
  - PSML (summary XML)
  - PDML (detailed XML)
  - JavaScript Object Notation (JSON)
  - Note that CSV exports are not really detailed, whereas JSON exports have far more fields

GUI to Save to File



Monitoring Oss: TOP Command

TOP in General

- We already saw what this command does
- Now we are trying to use it to log a sequence of observation of our system into a file
- You can Google for many bash strings but lets try to recap on what the tool is showing

TOP Example - I

- top

```
top - 10:40:17 up 167 days, 16:45,  4 users,  load average: 1,65, 1,76, 1,74
Tasks: 351 total,   1 running, 350 sleeping,   0 stopped,   0 zombie
%Cpu(s):  2,2 us, 12,9 sy,  0,0 ni, 84,9 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 s
KiB Mem:  24524720 total, 16984604 used,  7540116 free,   851220 buffers
KiB Swap: 29294588 total,  2187408 used, 27107180 free.  2002620 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 9253 mstader+  20   0 10,080g 8,129g 8,075g S 101,4 34,8  29194:07 VirtualBox
 6673 tommaso   20   0  300704 155024  22864 S   0,7  0,6 167:29.79 x2goagent
 1073 root      20   0   69076   6496    976 S   0,3  0,0 474:08.13 x2goclean+
 7369 mstader+  20   0  317804 150232  10104 S   0,3  0,6 196:18.57 x2goagent
17081 tommaso   20   0   94892   4692   3736 S   0,3  0,0   0:00.08 sshd
17670 tommaso   20   0  444548  30864  24116 S   0,3  0,1   0:00.10 gnome-ter+
17709 tommaso   20   0   26128   3164   2484 R   0,3  0,0   0:00.03 top
    1 root      20   0  177432   4292   2256 S   0,0  0,0   2:47.50 systemd
    2 root      20   0       0      0      0 S   0,0  0,0   0:37.22 kthreadd
    3 root      20   0       0      0      0 S   0,0  0,0  10:25.28 ksoftirqd+
    5 root       0 -20       0      0      0 S   0,0  0,0   0:00.00 kworker/0+
    7 root      20   0       0      0      0 S   0,0  0,0 356:51.60 rcu_sched
    8 root      20   0       0      0      0 S   0,0  0,0   0:00.00 rcu_bh
    9 root      rt   0       0      0      0 S   0,0  0,0   0:05.93 migration+
   10 root      rt   0       0      0      0 S   0,0  0,0   0:48.02 watchdog/0
```


TOP Example - II

- top -b -n 1 > top.txt

Apri | top.txt ~/ | Salva

```
top - 10:42:26 up 167 days, 16:47,  4 users,  load average: 1,47, 1,65, 1,70
Tasks: 351 total,   1 running, 350 sleeping,   0 stopped,   0 zombie
%Cpu(s): 11,8 us,  3,1 sy,  0,1 ni, 83,2 id,  1,8 wa,  0,0 hi,  0,0 si,  0,0 st
KiB Mem:  24524720 total, 17018932 used,  7505788 free,   851224 buffers
KiB Swap: 29294588 total,  2187396 used, 27107192 free.  2002680 cached Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 9253 mstader+  20   0 10,113g 8,161g 8,075g S 102,9 34,9  29196:18 VirtualBox
    1 root      20   0  177432   4292   2256 S   0,0  0,0   2:47.50 systemd
    2 root      20   0       0      0      0 S   0,0  0,0   0:37.22 kthreadd
    3 root      20   0       0      0      0 S   0,0  0,0  10:25.29 ksoftirqd+
    5 root       0 -20       0      0      0 S   0,0  0,0   0:00.00 kworker/0+
    7 root      20   0       0      0      0 S   0,0  0,0 356:51.81 rcu_sched
    8 root      20   0       0      0      0 S   0,0  0,0   0:00.00 rcu_bh
    9 root      rt   0       0      0      0 S   0,0  0,0   0:05.93 migration+
   10 root      rt   0       0      0      0 S   0,0  0,0   0:48.02 watchdog/0
   11 root      rt   0       0      0      0 S   0,0  0,0   0:47.34 watchdog/1
   12 root      rt   0       0      0      0 S   0,0  0,0   0:06.39 migration+
   13 root      20   0       0      0      0 S   0,0  0,0   8:33.33 ksoftirqd+
   15 root       0 -20       0      0      0 S   0,0  0,0   0:00.00 kworker/1+
   16 root      rt   0       0      0      0 S   0,0  0,0   0:47.12 watchdog/2
```

Testo semplice | Larg. tab.: 8 | Rg 1, Col 1

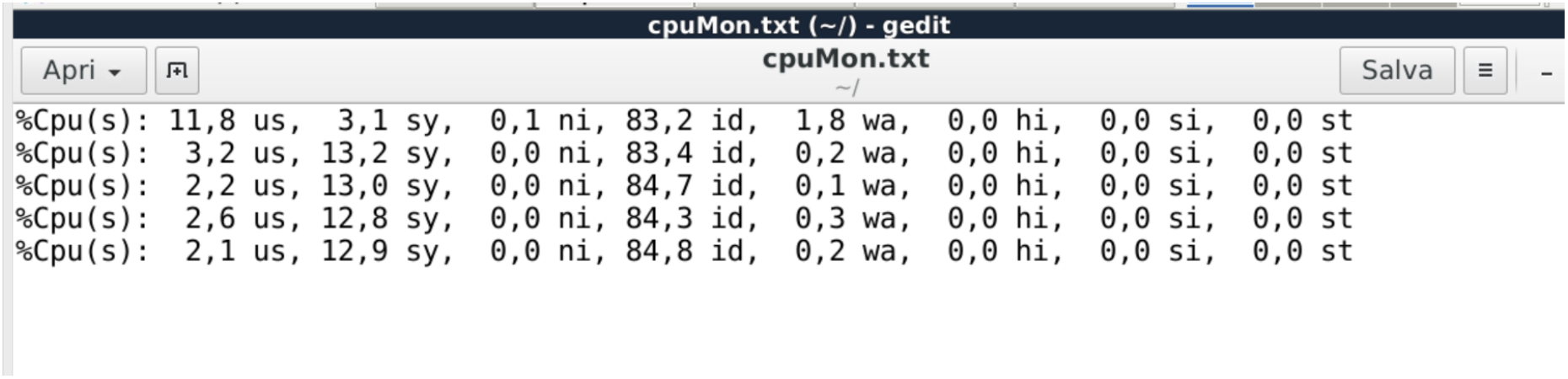
TOP Example - III

- top -b -n 1 | sed 6,1000d
  - To show only general Info

```
tommaso@churrasco:~$ top -b -n 1 | sed 6,1000d
top - 11:07:02 up 167 days, 17:11,  4 users,  load average: 1,41, 1,44, 1,51
Tasks: 353 total,   1 running, 352 sleeping,   0 stopped,   0 zombie
%Cpu(s): 11,8 us,  3,1 sy,  0,1 ni, 83,2 id,  1,8 wa,  0,0 hi,  0,0 si,  0,0
KiB Mem:  24524720 total, 16863908 used,  7660812 free,   851300 buffers
KiB Swap: 29294588 total,  2185808 used, 27108780 free.  2010724 cached Mem
```


TOP Example - IV

- top -b -n 5 | grep '%Cpu*' > cpuMon.txt
  - To monitor CPU across time



  - Similarly, it is possible to list other info across timespans

Lets Try it Together

- Hands-On Experimentation!