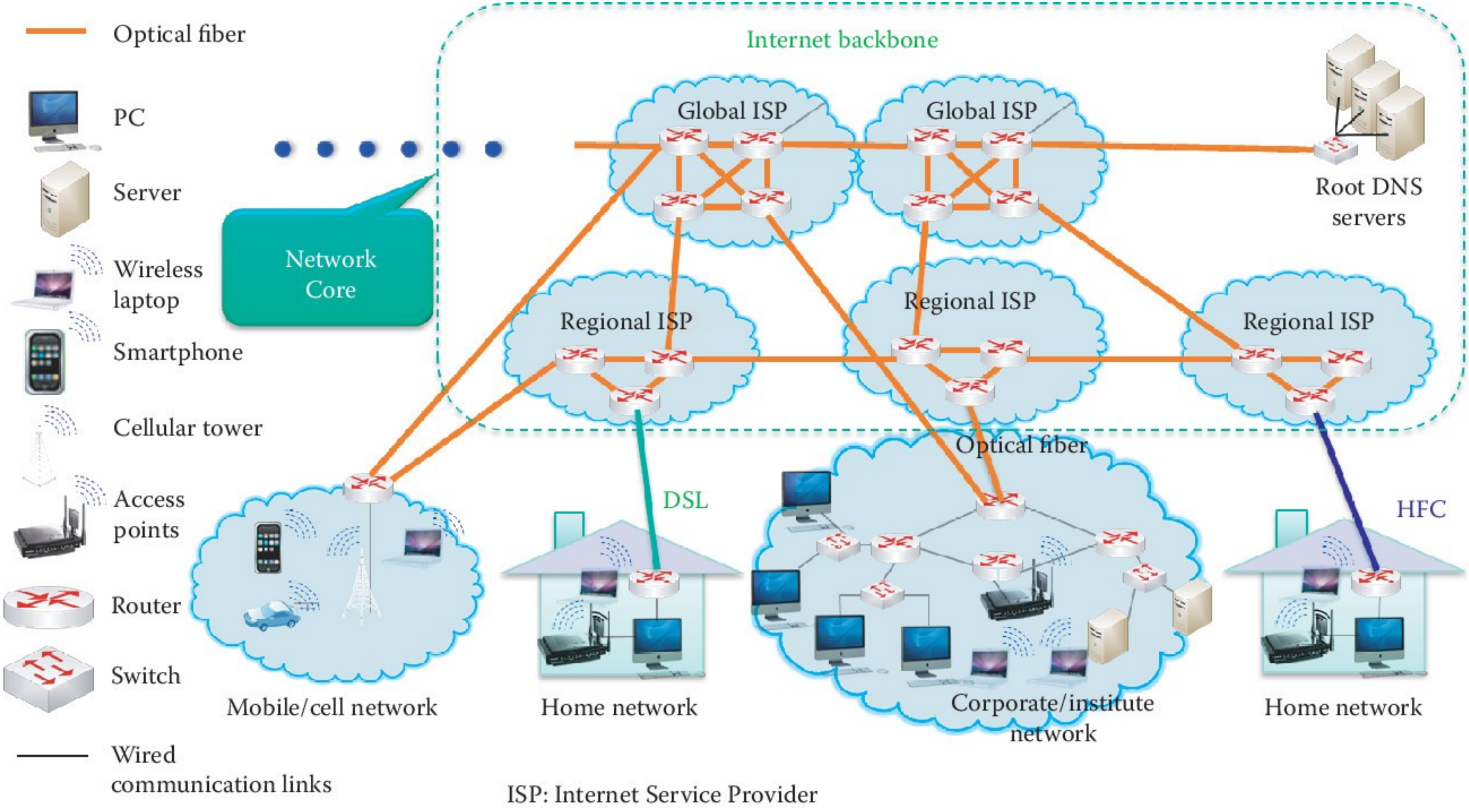
CyberX – Mind4Future
Network traffic dissection

Angelo Spognardi
spognardi@di.uniroma1.it
Dipartimento di Informatica
Sapienza Università di Roma

Internet architecture



Protocols

- Specify rules about the desired service
  - Procedure Rules
    - Types and sequences of messages exchanged
      - Syntax and semantics
    - Actions to take with respect to messages and events
  - Message Format: format, size and coding of messages.
  - Timing: the time to wait between any event.
    - Access to medium
    - Flow control
    - Timeouts

Protocol specification examples

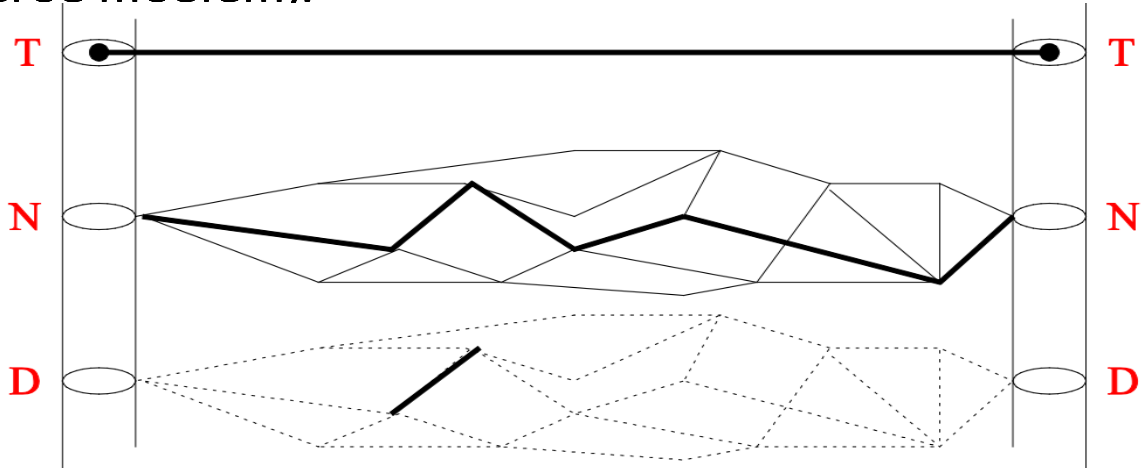
- Modularization → Many protocols for each layer
  - Hides implementation details
  - Layers can change without disturbing other layers
    - Development (one company can tackle one module)
    - Maintenance
    - Updating the system
- Packet switching
  - Best effort delivery
  - Better for resource sharing
- Network congestion and flow control

Layering Concepts

- The communication between the hosts in the network is organized in tasks, each assigned to a **layer**
- Each layer:
  - offers a service (a host of facilities) to the "Users" in the layer above
  - exploits the services offered the layer below
- The task of a level involves the exchange of messages that follow a set of rules defined by a **protocol**.
- Example:
  - Layer (N - 1) provides an insecure service in which data can overheard by unauthorized persons.
  - Protocol of level N specifies that messages sent via (N - 1)-service are encrypted with symmetric encryption.
  - Layer N offers a secure, confidential service.

Layer ideal representation

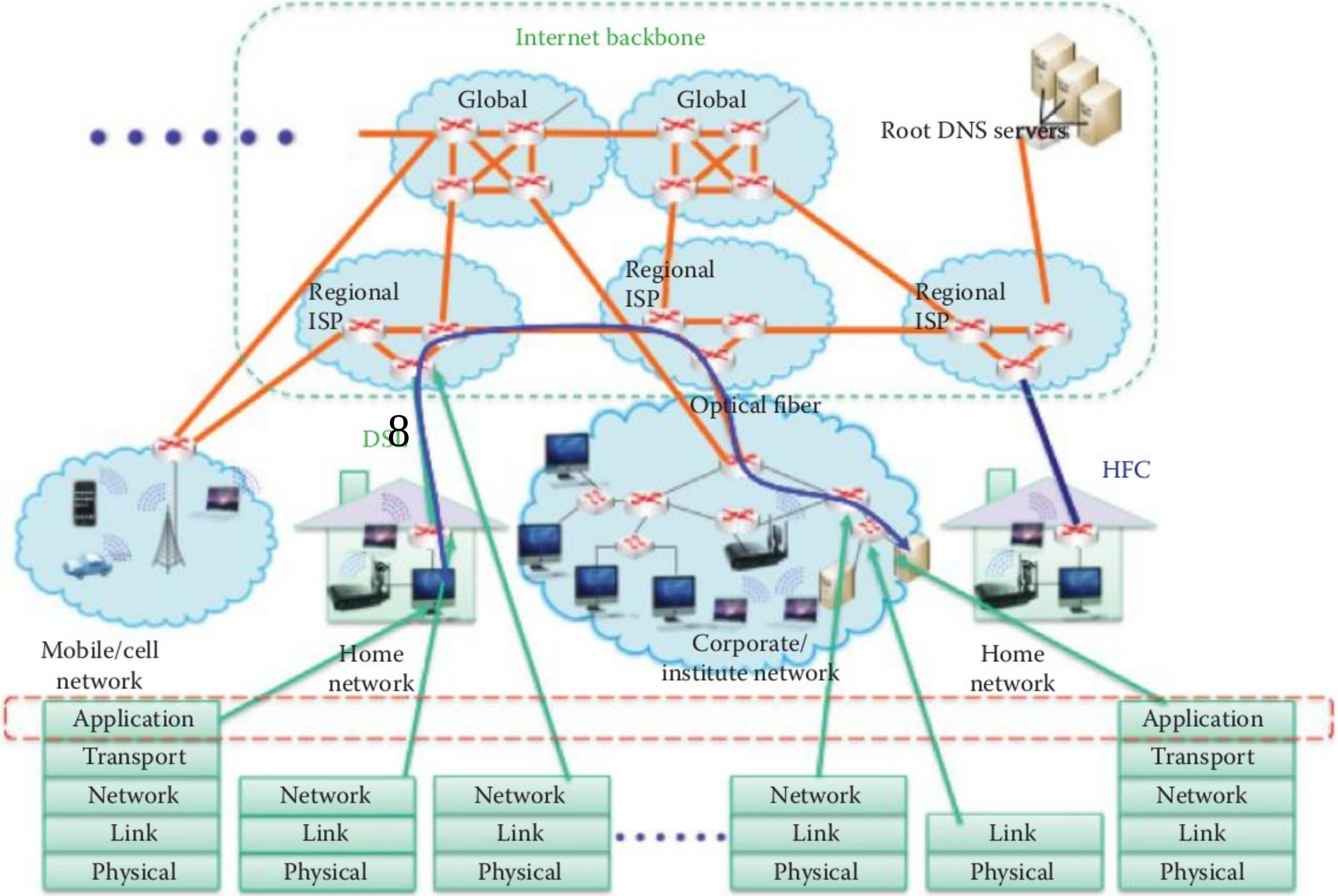
- **Transport**: the illusion of direct end-to-end connection between processes in arbitrary systems.
- **Network**: transferring data between arbitrary nodes.
- **Data Link**: transferring data between directly connected systems (via direct cable or shared medium).



Encapsulation/decapsulation

- The data to be transferred from the application layer to application layer over a network.
- Each layer adds some protocol information and provides data to the layer below.
- The physical layer (bottom) sends data over the physical medium to the destination.
- The physical layer in the destination sends the data up the "stack".
- Each protocol in the destination reads the appropriate protocol information and forwards the data to the layer above.

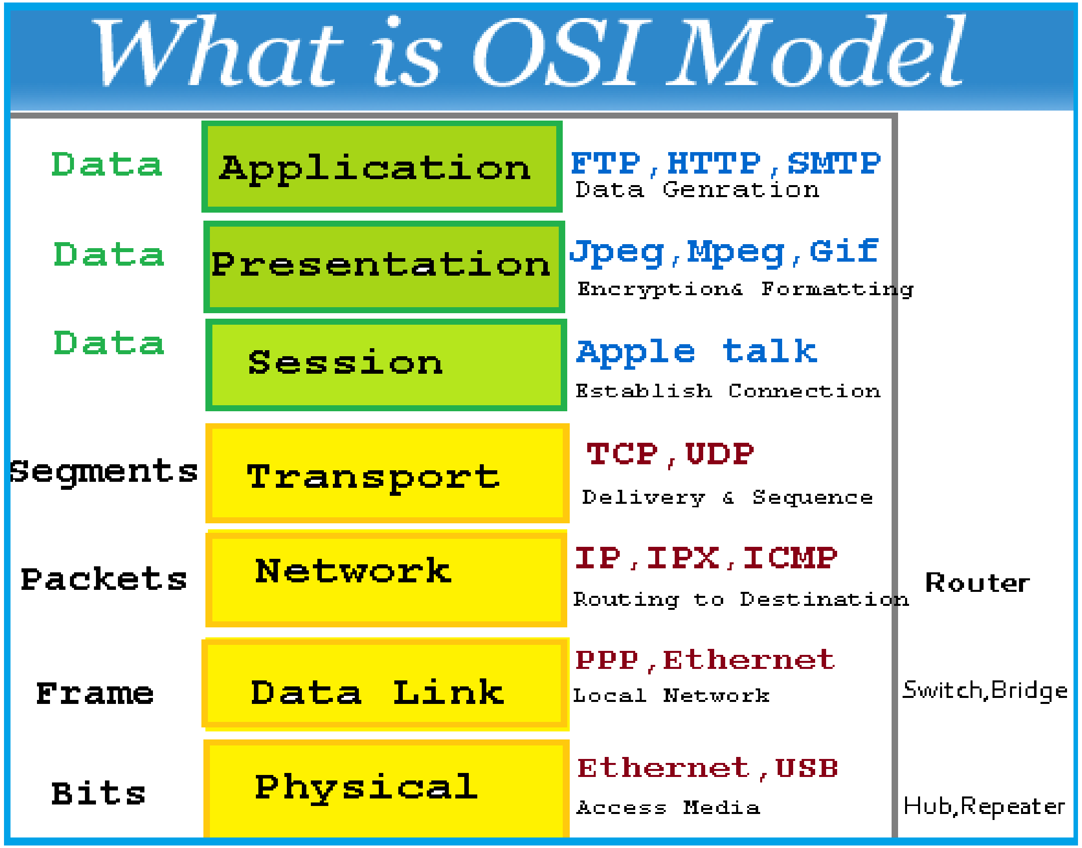
Client-server communication example

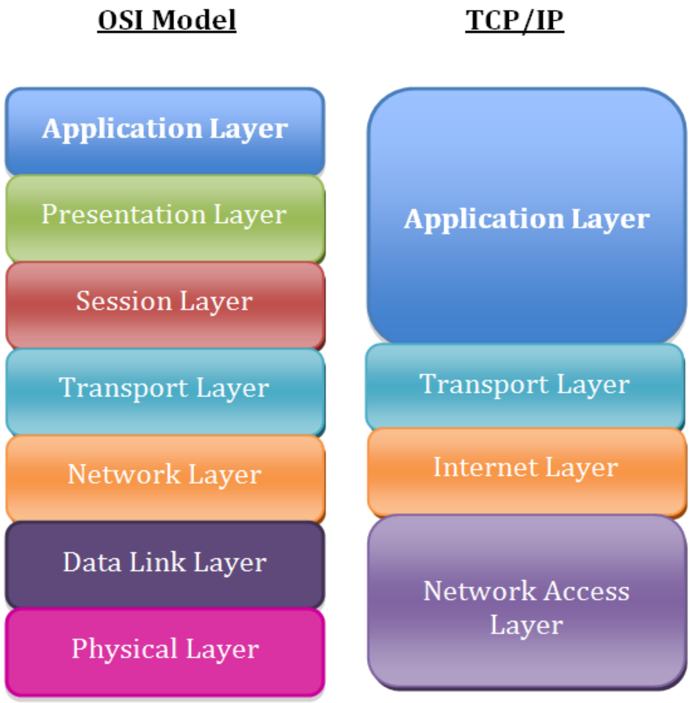


2 layered architectures

- ISO/OSI model: based on a reference model with 7 layer.
- TCP/IP model: created by the IETF, based on a reference model with 4 layers.
  - The lower TCP/IP layer is often split in 2 layers.
- Common idea: **packet switched network**

Architecture comparison





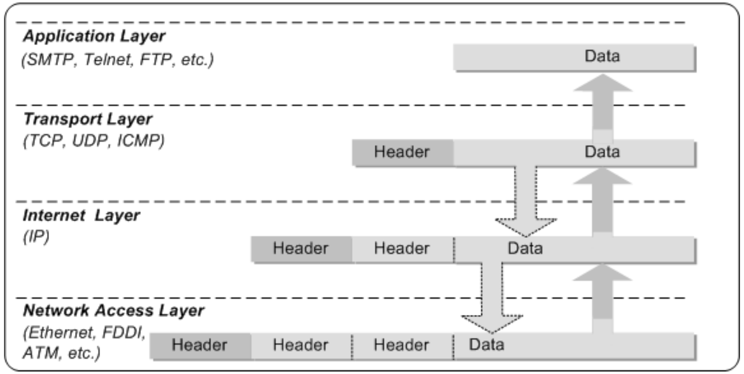
TCP / IP model

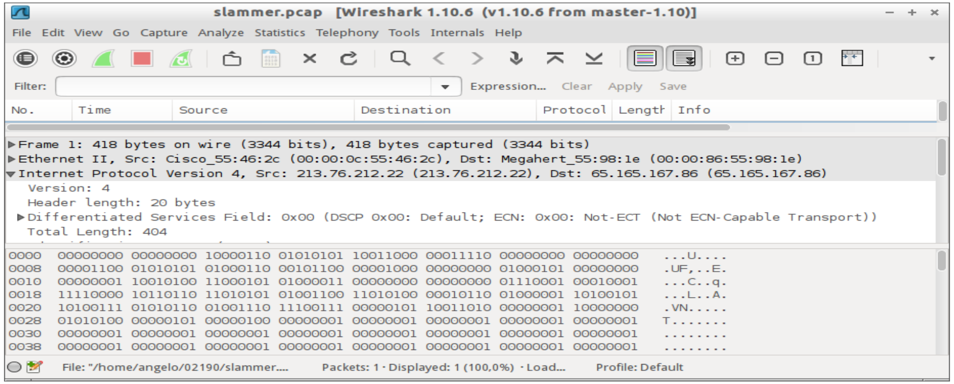
- Application layer: Corresponds to the top three layers of the OSI model.
  - Protocols: SMTP (sending e-mail), HTTP (web), FTP (file transfer), and others
- Transport layer: Equivalent to Layer 4 (Transport) of the OSI model
  - Protocols: TCP, UDP
- Internet: Equivalent to layer 3 (network) of the OSI model.
  - Protocols: IP, ICMP, IPSec
- Datalink: Equivalent to layer 2 (data link) of the OSI model.
  - Protocols: Ethernet, WiFi, ARP, etc.
- Physical layer: Equivalent to Layer 1 (Physical) of the OSI model.
  - NOTE: Datalink + physical layers are known as Network access layer.

Addresses in the architectures

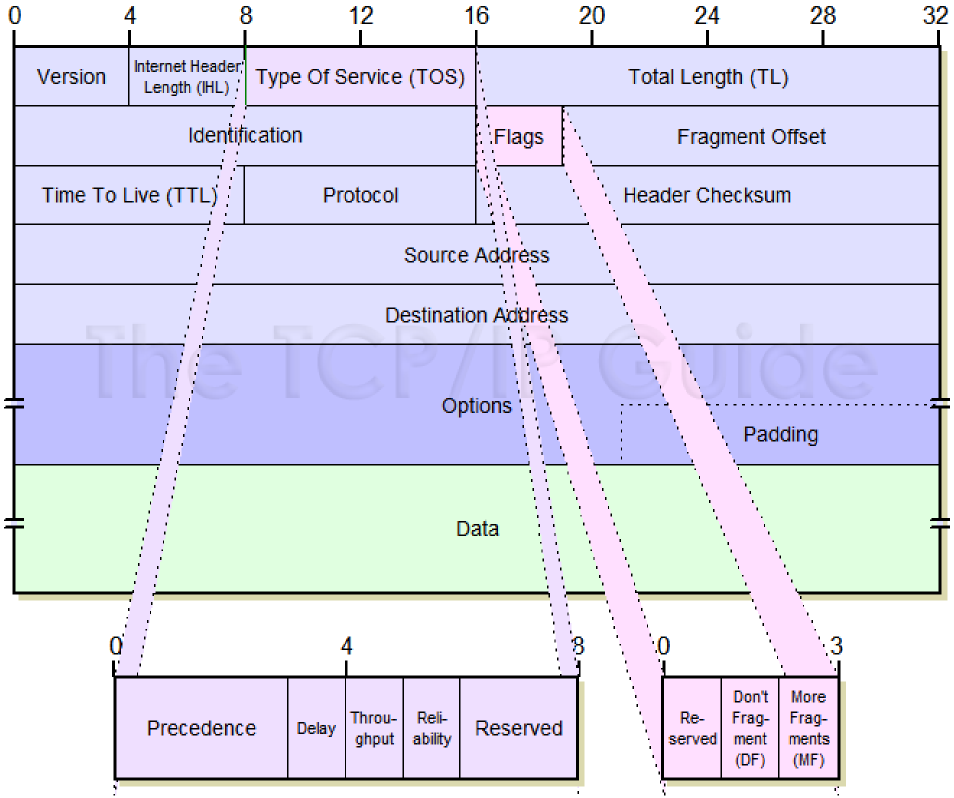
- Each layer has a type of address:
  - Application layer: Internet name, eg. www.*sapienza.it*
  - Transport layer: Port number, in the range [0..65535] that identifies the client or server. For example 80 for HTTP server.
  - Internet layer: IP address that identifies a network card, for example 151.100.17.4
  - Datalink layer: MAC address, also identifies a network cards, for example 49:bd:d2:c7:56:2a

Encapsulation in TCP/IP

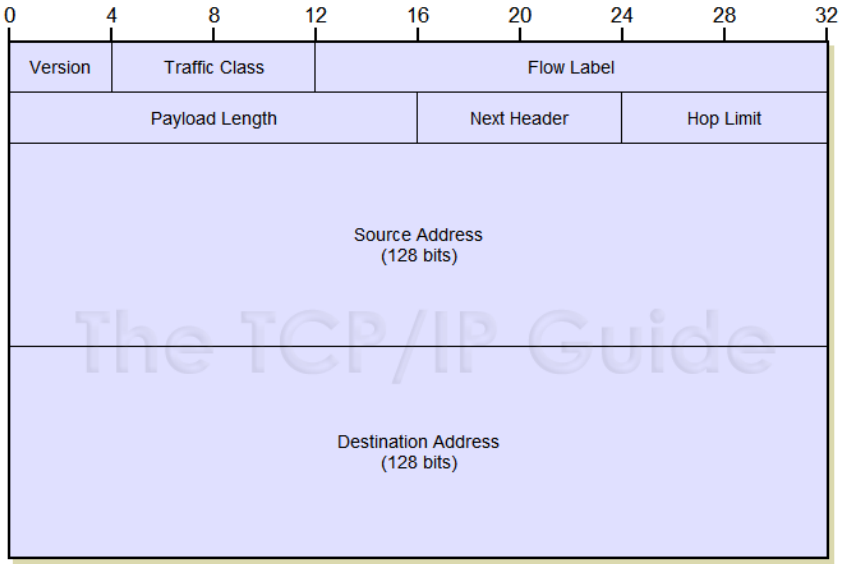




IP packets



http://www.tcpipguide.com/free/t_IPDatagramGeneralFormat.htm

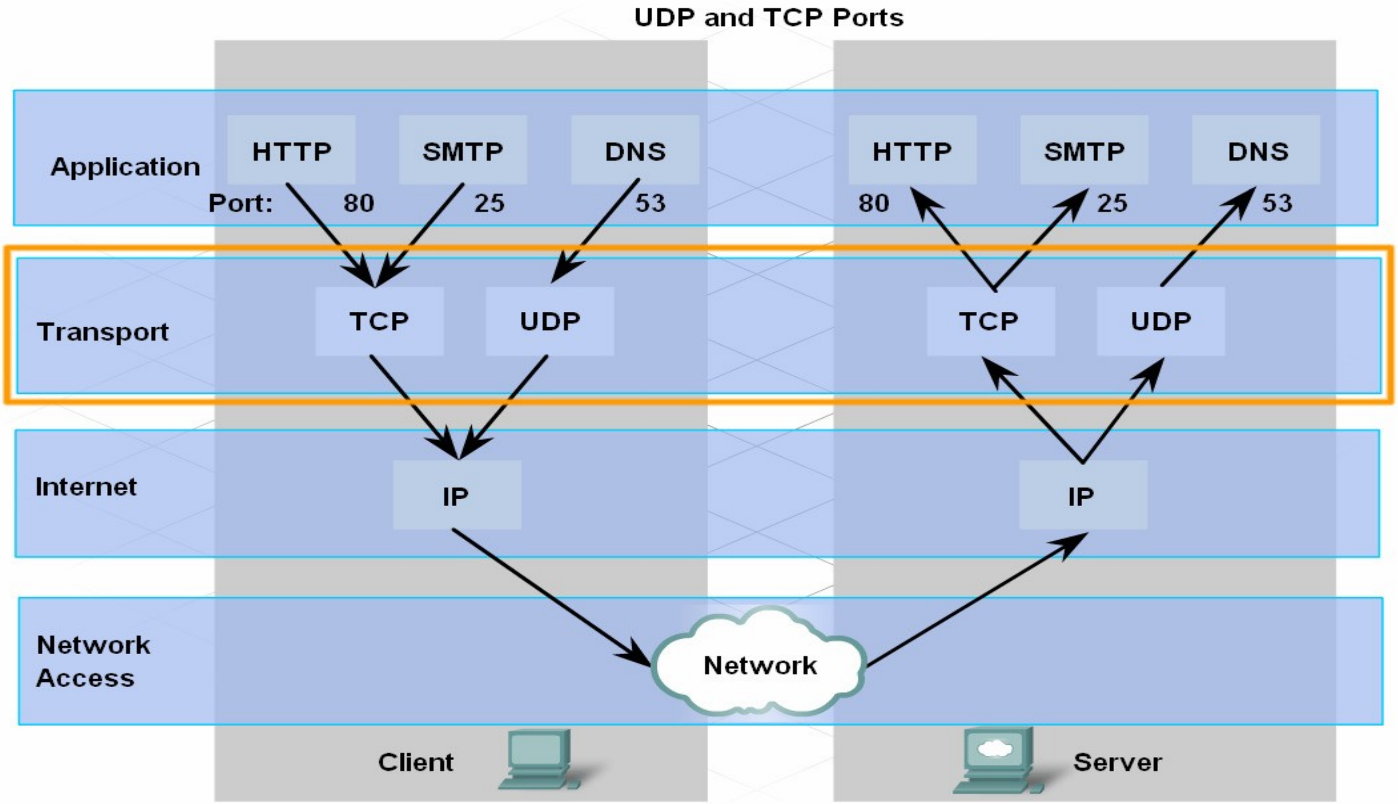


http://www.tcpipguide.com/free/t_IPv6DatagramMainHeaderFormat.htm

Ports

- Range [0..65535]
- Source port: randomly chosen by the OS
- Destination port determines the required service (application)
  - Assigned Ports [0..1023] are said "well-known ports" and used by servers for standard Internet applications:
    - 25: SMTP (sending mail)
    - 80: HTTP (web)
    - 143: IMAP (pick-up of mail)
  - Ports [1024..49151] can be registered with Internet Application Naming Authority (IANA)
  - Ports [49152..65535] ephemeral ports

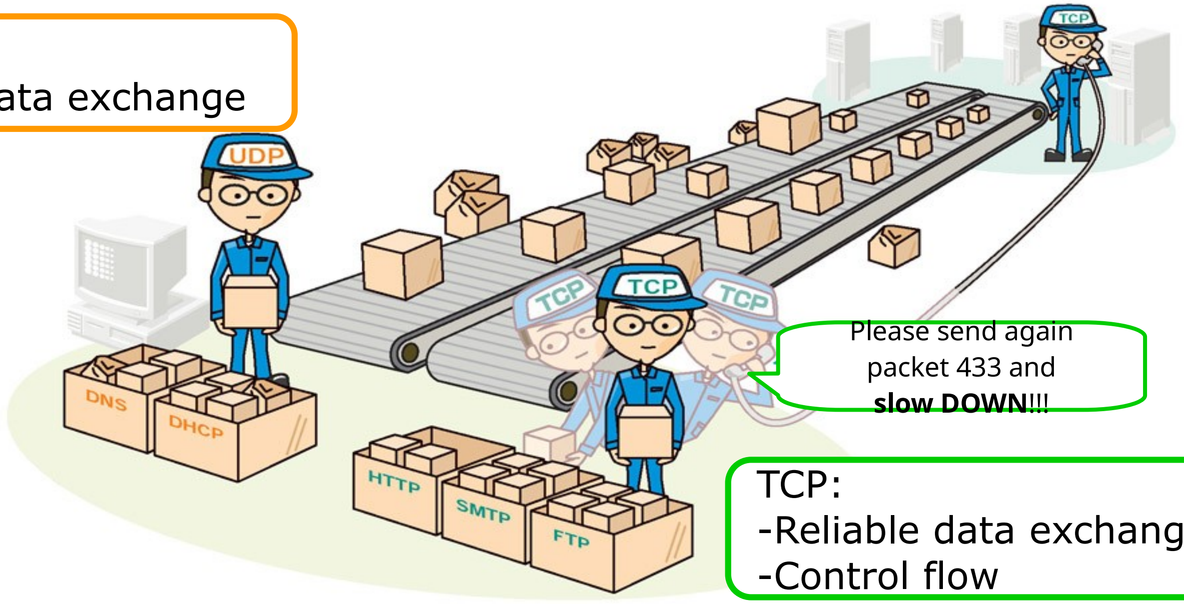
Transport layer: TCP and UDP



TCP vs UDP

- Connection vs Connection-less

UDP:
-No control on data exchange



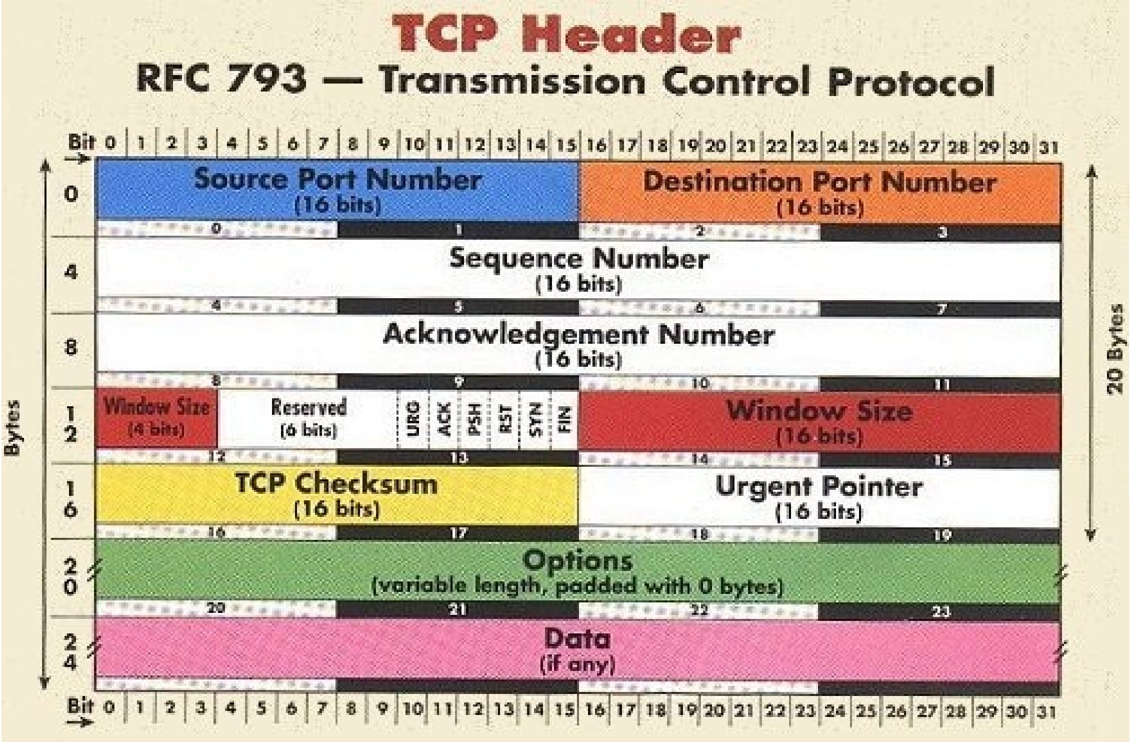
OK, Acknowledge!

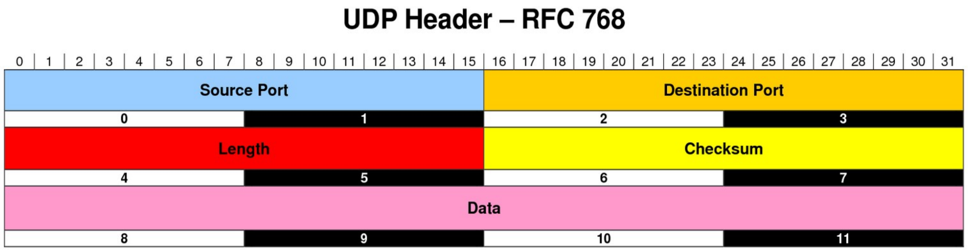
Please send again packet 433 and **slow DOWN**!!!

TCP:
-Reliable data exchange
-Control flow

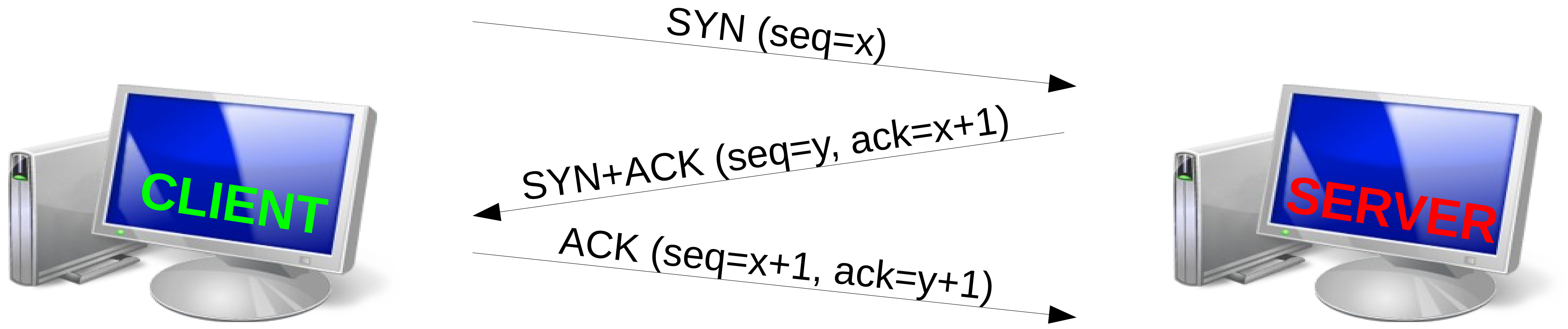
http://itpro.nikkeibp.co.jp/article/lecture/20070305/263897/

TCP header vs UDP header





TCP connection handshake



Services relying on TCP

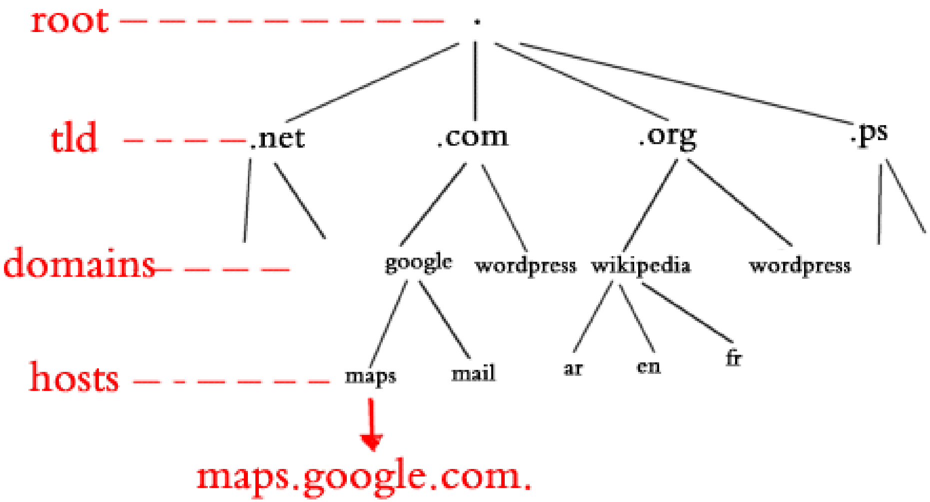
- FTP on port 20 and 21
- SSH on port 22
- Telnet on port 23
- SMTP on port 25
- HTTP on port 80
- IMAP on port 143
- SSL on port 443

Services relying on UDP

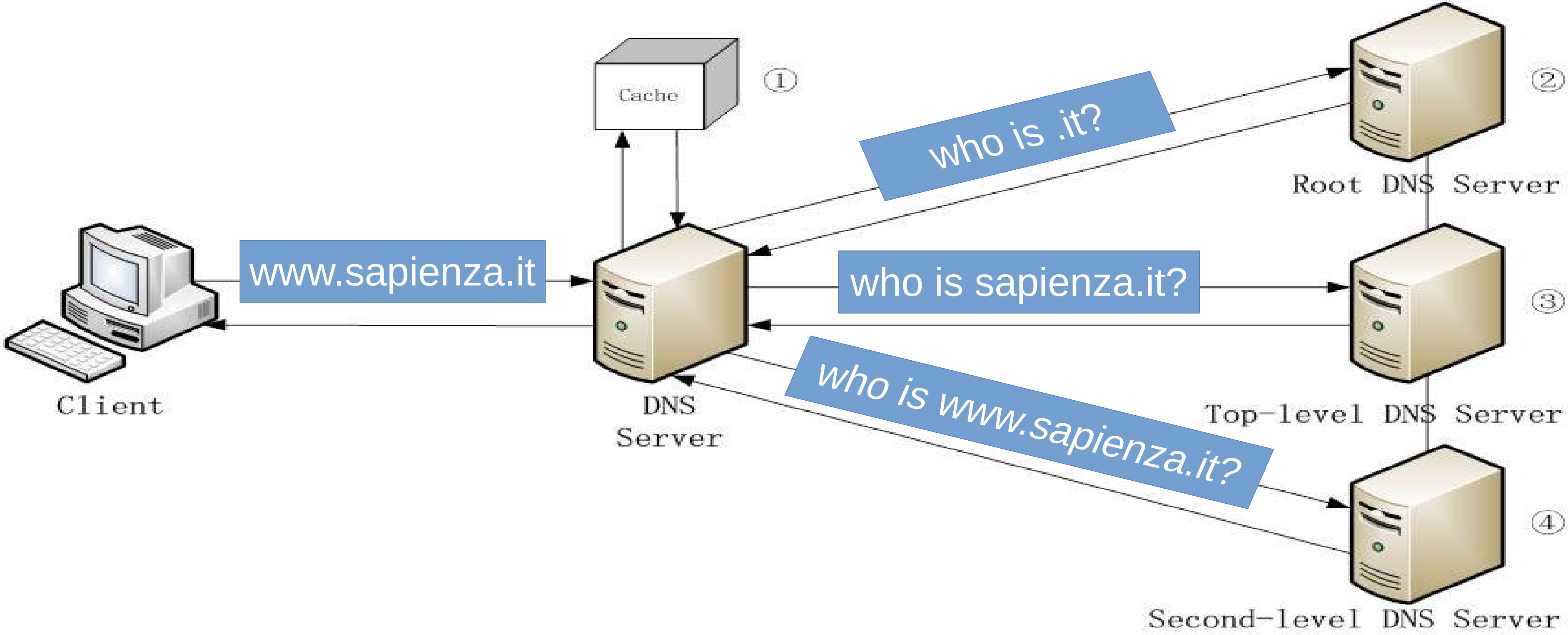
- DNS on port 53
- DHCP on ports 67 and 68
- TFTP on port 69
- SNMP on port 161
- RIP on port 520

DNS

- A service to get the IP address from an human-friendly domain name, like www.sapienza.it
- Hierarchy of entities responsible for domain names



DNS query example



Dive into packets

Capture packets

- Packets flow in the network, to capture them use a network traffic dump tool, like:
  - **dumpcap**
  - **wireshark/tshark** (https://www.wireshark.org/docs/)
  - **tcpdump**
- All based on the *pcap* (*winpcap* in Windows) library
- All of them can visualize and save the captured data
- Wireshark and tcpdump can also **analyze** *(decode)* the captured packets

Wireshark

- Data from a network interface are “dissected” in frames, segments, and packets, understanding where they begin and end
- Then, they are interpreted and visualized in the context of the recognized protocol
- Best suited for
  - Looking for the root cause of a known problem
  - Searching for a certain protocol or stream between devices
  - Analyzing specific timing, protocol flags, or bits on the wire
  - Following a conversation between two devices
- It shouldn’t be the first tool thought of early on in discovering a problem, but solving a problem…

Logic of wireshark

- Frames are collected from the interface and passed to several, consecutive, “dissectors”, one for each layer
- Frames pass from bottom layer to upper layer
- Protocols can be detected in two ways:
  - directly, if a frame (e.g. Ethernet) has the field that states which protocol it is encapsulating
  - indirectly, with tables of protocol/port combinations and heuristics
    - Usually working, troubles when protocols are used in nonstandard ports

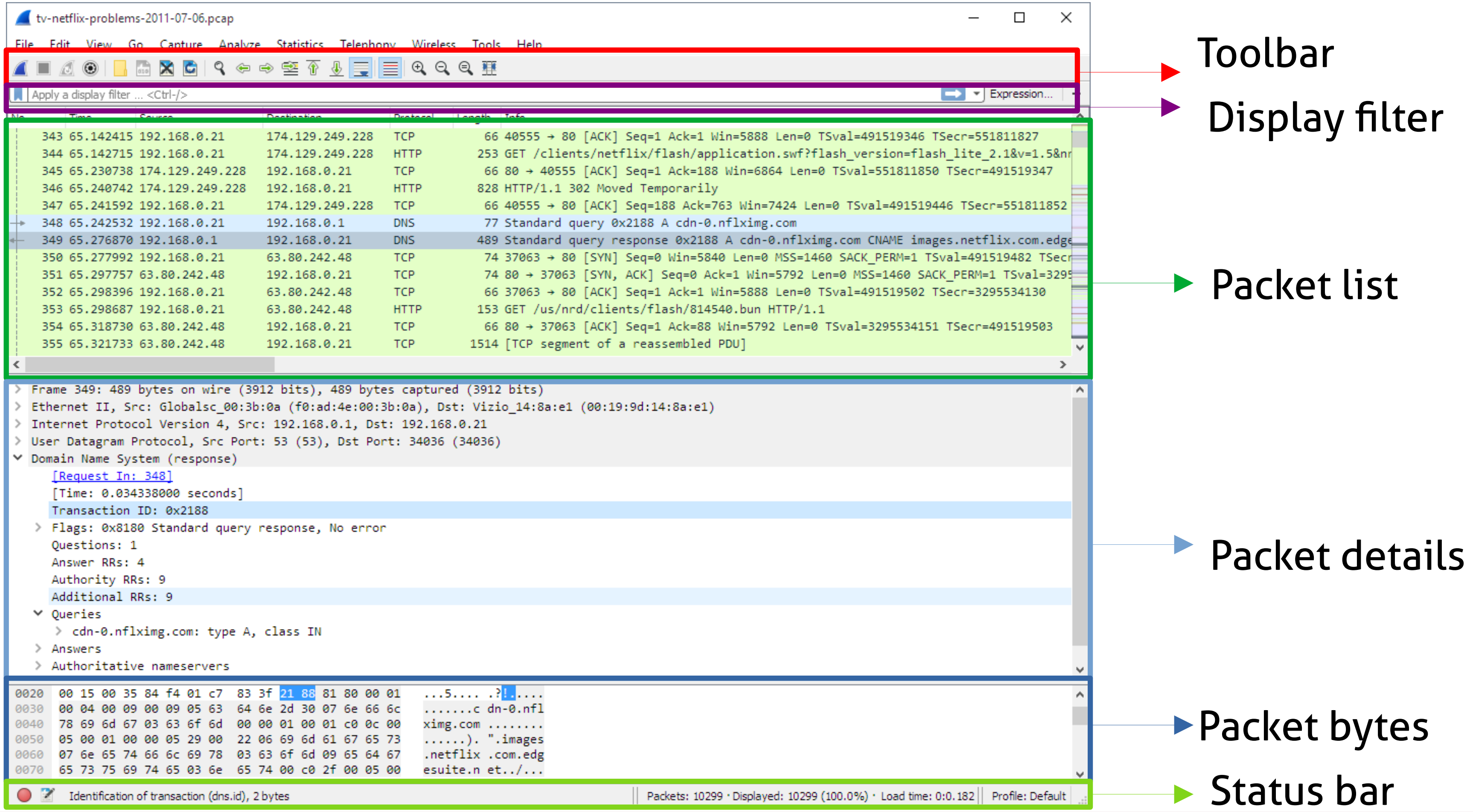
Alternative way to capture traffic info

- Traffic represented as “connections”
- Netflow
  - For statistics and monitoring
  - Netflow v9 https://www.ietf.org/rfc/rfc3954.txt
- Zeek (formerly known as Bro)
  - Framework for traffic inspection and monitoring
  - Scripting engine to enable immediate processing

Wireshark

https://www.wireshark.org/docs/wsug_html/

Wireshark

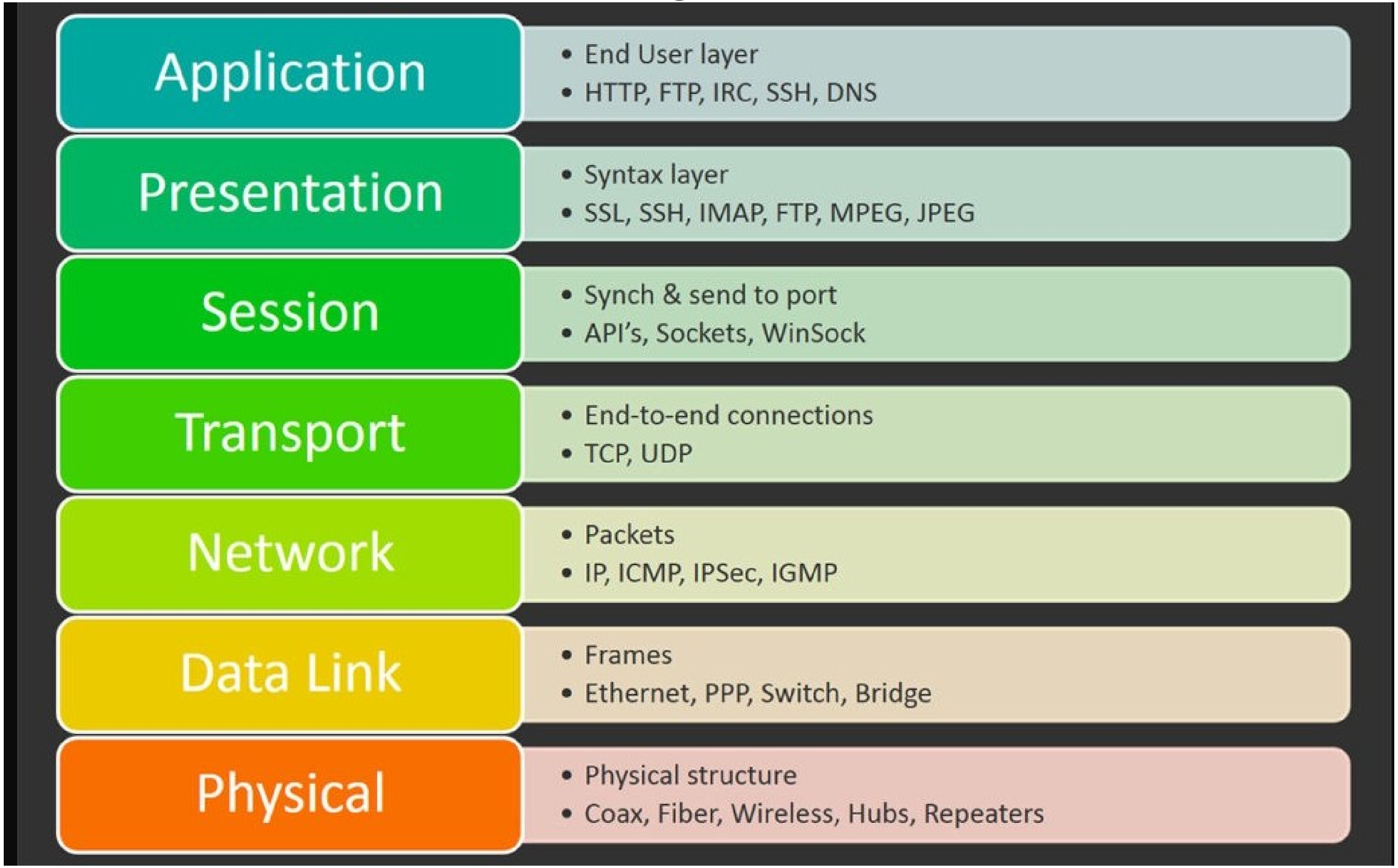


tv-netflix-problems-2011-07-06.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
343 65.142415 192.168.0.21 174.129.249.228 TCP 66 40555 → 80 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=491519346 TSecr=551811827
344 65.142715 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
Toolbar
Display filter
Packet list
Packet details
Packet bytes
Status bar

Logic of wireshark

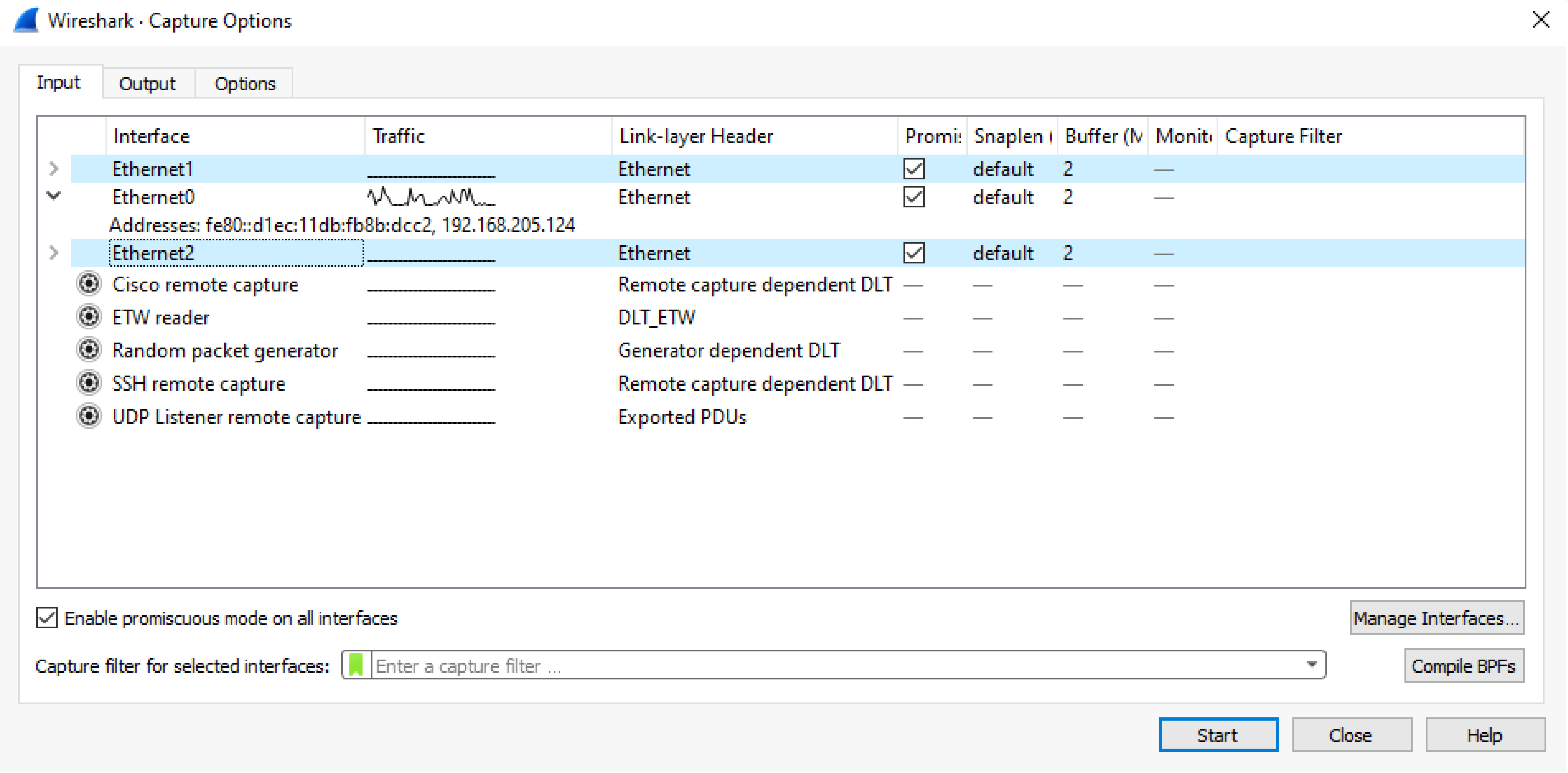
- Frames are collected from the interface and passed to several, consecutive, “dissectors”, one for each layer
- Frames pass from bottom layer to upper layer
- Protocols can be detected in two ways:
  - directly, if a frame (e.g. Ethernet) has the field that states which protocol it is encapsulating
  - indirectly, with tables of protocol/port combinations and heuristics
    - Usually working, troubles when protocols are used in nonstandard ports

Dissection in network layers



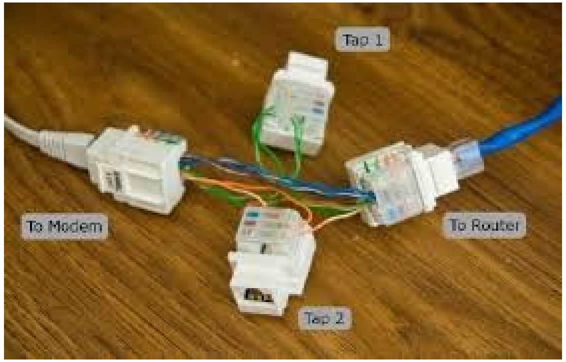
Raza, M., 2018. 7 Layers Of The OSI Model

Realtime capture



How to capture network traffic

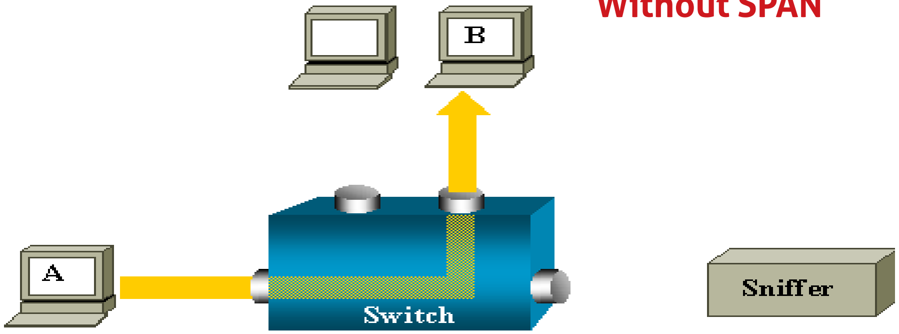
- Promiscuous mode
  - Limitations?
  - Remember the difference between hubs and switches!
- Physical tap
- Port mirroring on a managed switch
- More “aggressive” approaches:
  - ARP cache poisoning
  - MAC flooding
  - DHCP redirection
  - Redirection and interception with ICMP
- NOTICE: on virtualized environments and SDN, this can be easier or harder



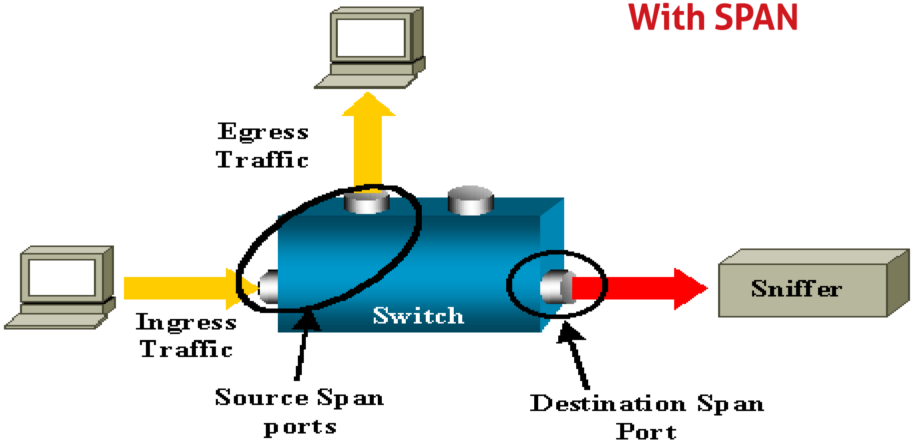
Port mirroring

- Switched Port Analyzer (SPAN) or Roving Analysis Port (RAP)

Without SPAN



With SPAN



Less conventional approaches for sniffing

- ARP cache poisoning (or spoofing)
  - Unsolicited ARP replies to steal IP addresses (ettercap, cain&abel)
- MAC flooding
  - Fill the CAM of the switch to make it acting as a hub (macof)
- DHCP redirection
  - Rogue DHCP server: it exhausts the IP addresses of the pool
  - Then pretends to be the default gateway of the network with the new DHCP requests (Gobbler, DHCPstarv, Yersinia)
- Redirection and interception with ICMP
  - ICMP type 5 (redirect) used to indicate a better route (ettercap)

How to prevent packet capture

- Dynamic address inspection
  - Implemented in switches: Dynamic Address Resolution Inspection (DAI) validates ARP packets
  - IP-to-MAC address binding inspection, drop invalid packets
- DHCP snooping
  - Implemented in switches: distinguishes between trusted and untrusted ports and uses a database of IP-to-MAC
  - Ports that show rogue activity can also be automatically placed in a disabled state

Using wireshark

- Capturing is way too easy... Too many packets!
  - https://wiki.wireshark.org/CaptureSetup/CapturePrivileges
- To survive, use filters!
  - They allow to only focus on requested packets or certain activity by network devices
- Two kinds of filters: **display filters** and **capture filters**
  - Capture filters to limit the amount of network data that goes into processing and is getting saved
  - Display filters to inspect only the packets you want to analyze once the data has been processed

Display filters – wireshark

- Display only captured packets matching the filters
  - Packets are not discarded or lost
- Easy but refined syntax: only packets evaluating true are displayed
  - Comparison operators
  - Filters use types (strings where numbers are required return errors)
  - Common logical operators
- Filters can be built interacting with the packets

Capture filters – wireshark/tcpdump

- Limit the traffic captured and, optionally, analyzed
  - Packets not captured are lost!
- Berkeley Packet Filter (BPF) syntax (man pcap-filter)

protocol direction type

  - Protocol: ether, tcp, udp, ip, ip6, arp
  - Direction: src, dst
  - Type: host, port, net, portrange
  - Other primitives: less, greater, gateway, broadcast
  - Operators: equals → eq / ==, not equal → ne / !=, greater than → gt / > less than → lt / <
  - Combinations with operators: and (&&), or (||), not (!)

Wireshark packet colors

<table>
<tr><th>Name</th><th>Filter</th></tr>
<tr><td>Bad TCP</td><td>tcp.analysis.flags && !tcp.analysis.window_update</td></tr>
<tr><td>HSRP State Change</td><td>hsrp.state != 8 && hsrp.state != 16</td></tr>
<tr><td>Spanning Tree Topology Change</td><td>stp.type == 0x80</td></tr>
<tr><td>OSPF State Change</td><td>ospf.msg != 1</td></tr>
<tr><td>ICMP errors</td><td>icmp.type eq 3 || icmp.type eq 4 || icmp.type eq 5 || icmp.type eq 11 || icmpv6.type eq 1 || icmpv6.type eq 2 || icmpv6.type eq 3 || icmpv6.type eq 4</td></tr>
<tr><td>ARP</td><td>arp</td></tr>
<tr><td>ICMP</td><td>icmp || icmpv6</td></tr>
<tr><td>TCP RST</td><td>tcp.flags.reset eq 1</td></tr>
<tr><td>SCTP ABORT</td><td>sctp.chunk_type eq ABORT</td></tr>
<tr><td>TTL low or unexpected</td><td>(! ip.dst == 224.0.0.0/4 && ip.ttl < 5 && !pim && !ospf) || (ip.dst == 224.0.0.0/24 && ip.dst != 224.0.0.251 && ip.ttl != 1 && !(vrrp || carp))</td></tr>
<tr><td>Checksum Errors</td><td>eth.fcs.status=="Bad" || ip.checksum.status=="Bad" || tcp.checksum.status=="Bad" || udp.checksum.status=="Bad" || sctp.checksum.status=="Bad"</td></tr>
<tr><td>SMB</td><td>smb || nbss || nbns || netbios</td></tr>
<tr><td>HTTP</td><td>http || tcp.port == 80 || http2</td></tr>
<tr><td>DCERPC</td><td>dcerpc</td></tr>
<tr><td>Routing</td><td>hsrp || eigrp || ospf || bgp || cdp || vrrp || carp || gvrp || igmp || ismp</td></tr>
<tr><td>TCP SYN/FIN</td><td>tcp.flags & 0x02 || tcp.flags.fin == 1</td></tr>
<tr><td>TCP</td><td>tcp</td></tr>
<tr><td>UDP</td><td>udp</td></tr>
<tr><td>Broadcast</td><td>eth[0] & 1</td></tr>
<tr><td>System Event</td><td>systemd_journal || sysdig</td></tr>
</table>

Additional setup

- Configure the GeoIP resolver
  - https://wiki.wireshark.org/HowToUseGeoIP
  - Sign and download the GeoLite2 MaxMind free database(s)
    - Alternative link
  - Unzip the files in a directory
- In wireshark:
  - Edit→Preferences→Name Resolution
  - Select MaxMind database directories
- Now you can use filters like

```
ip.geoip.country eq "China"
```

Working with PCAP files

- Wireshark can read in previously saved capture files
- Handles many capture formats
- It can also merge, manipulate and dump data
- It can extract any info from the captured files
- It can also go trough encrypted connections…

Challenge!

- Try to solve with wireshark the CTF of Hack3rCon 3 conference (2012)
- http://sickbits.net/other/hc3.pcap-04.cap
  - https://drive.google.com/file/d/1ANd0t_U7Ya8R1fppcHhi51WYq9FjltM6/view?usp=drive_web&authuser=0

References

- Wireshark for Security Professionals: Using Wireshark and the Metasploit Framework
  - Bullok, Parker, Wiley ed.
- The Network Security Test Lab: A Step-by-Step Guide
  - Gregg, Wiley e.
- https://www.wireshark.org/docs/wsug_html/