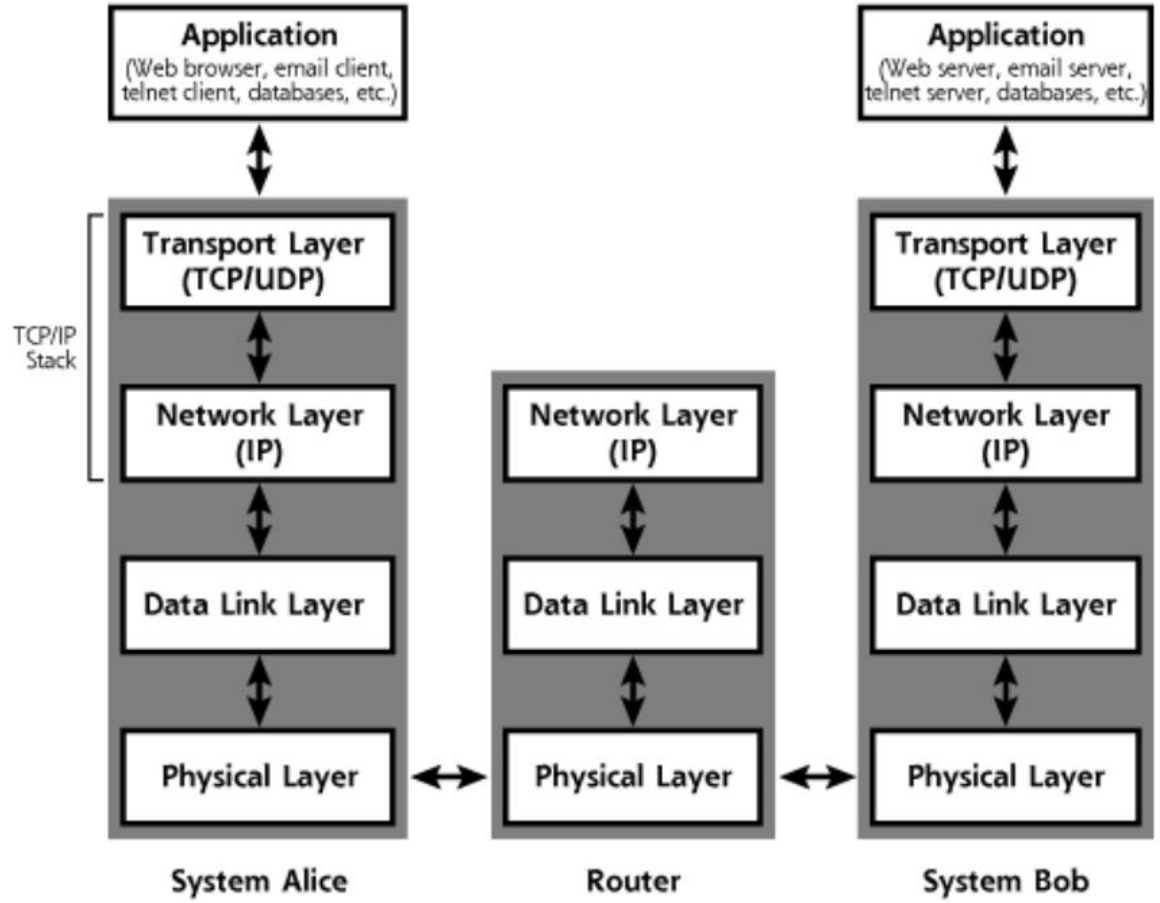
CS4238: Computer Security Practice

Lecture 6: Network Configuration, Traffic Analysis, Firewall

Networking Overview

(Chapter 2 of the Reference book 1)

TCP/IP Layers



Source: Skoudis & Liston, Counter Hack Reloaded

TCP

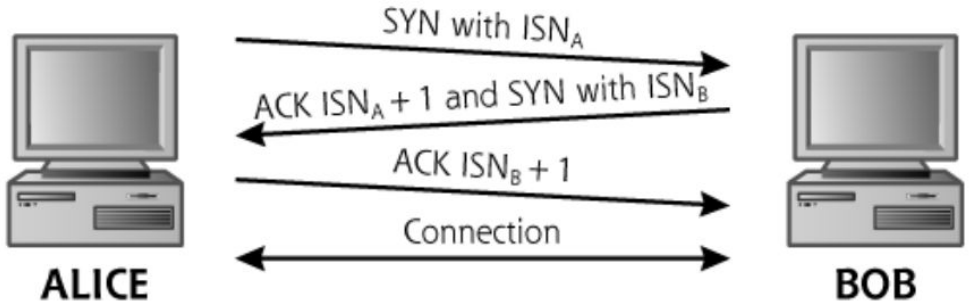
- TCP header format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Source port																Destination port															
Sequence number																															
Acknowledgment number (if ACK set)																															
Data offset			Reserved 0 0 0				NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size															
Checksum																Urgent pointer (if URG set)															
Options (if *data offset* > 5. Padded at the end with "0" bytes if necessary.) ...																															

Source: Wikipedia

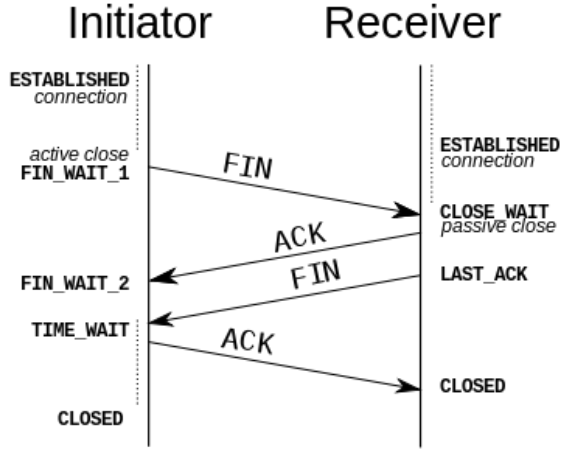
TCP Connection Management

- TCP three-way handshake:



Source: Skoudis & Liston, Counter Hack Reloaded

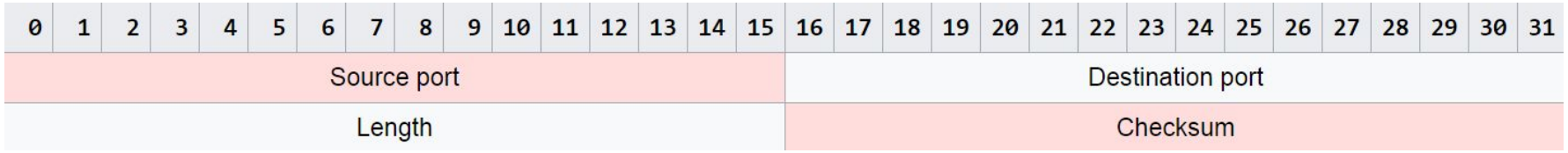
- TCP connection termination:



Source: Wikipedia

UDP

- Connectionless transport protocol
- UDP header format:



Source: Wikipedia

- Used among others by DNS (port 53), BOOTP/DHCP (port 67 & 68), TFTP (port 69), SNMP (port 161)

IP

- Importance of IP:
  - "Anything over IP and IP over anything"
  - The waist (glue point) of protocol-stack's hourglass
- IP header format:

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	Version				IHL				DSCP						ECN		Total Length															
32	Identification																Flags			Fragment Offset											
64	Time To Live								Protocol								Header Checksum															
96	Source IP Address																															
128	Destination IP Address																															
160	Options (if IHL > 5)																															
192																																
224																																
256																																

Source: Wikipedia

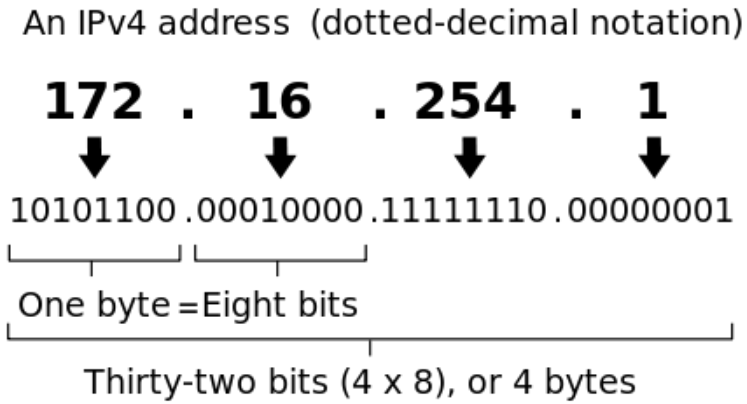
IP Packet Fragmentation & Reassembly

- **Goal**: To optimize packet length for various communication links with different maximum transmission unit (MTU)
- Two **flag bits** in IP header:
  - Don't Fragment bit
  - More Fragment bit
- Other related IP **header fields**:
  - Identification: set to a unique value
  - Fragment Offset: where a fragment needs to be positioned during the reassembly

Source: Wikipedia

IPv4 Address

- Dotted-decimal notation:



Source: Wikipedia

- Network address and host address components
- Classful network architecture (1981-1993):
  - Now only for default configuration of subnet masks
- Classless Inter-Domain Routing (CIDR):
  - Variable-length subnet masking (VLSM)
  - CIDR notation (e.g. 192.168.2.0/24)

IPv4 Address

- **Special IP** addresses:
  - Localhost address: 127.0.0.1
  - Private addresses:
    - **10.0.0.0 – 10.255.255.255**: 24-bit host ID (24-bit block)
    - **172.16.0.0 – 172.31.255.255**: 20-bit host ID (20-bit block)
    - **192.168.0.0 – 192.168.255.255**: 16-bit host ID (16-bit block)
    - Not routable on the public Internet
    - Usually used together with NAT or proxy
  - Automatic Private IP Addressing (APIPA) or auto-IP address: 169.254.1.0 – 169.254.254.255
    - E.g. when DHCP server is unavailable

Protocols on Top of IP

Some of the common payload protocols are:

Protocol Number	Protocol Name	Abbreviation
1	Internet Control Message Protocol	ICMP
2	Internet Group Management Protocol	IGMP
6	Transmission Control Protocol	TCP
17	User Datagram Protocol	UDP
41	IPv6 encapsulation	ENCAP
89	Open Shortest Path First	OSPF
132	Stream Control Transmission Protocol	SCTP

Source: Wikipedia

ICMP

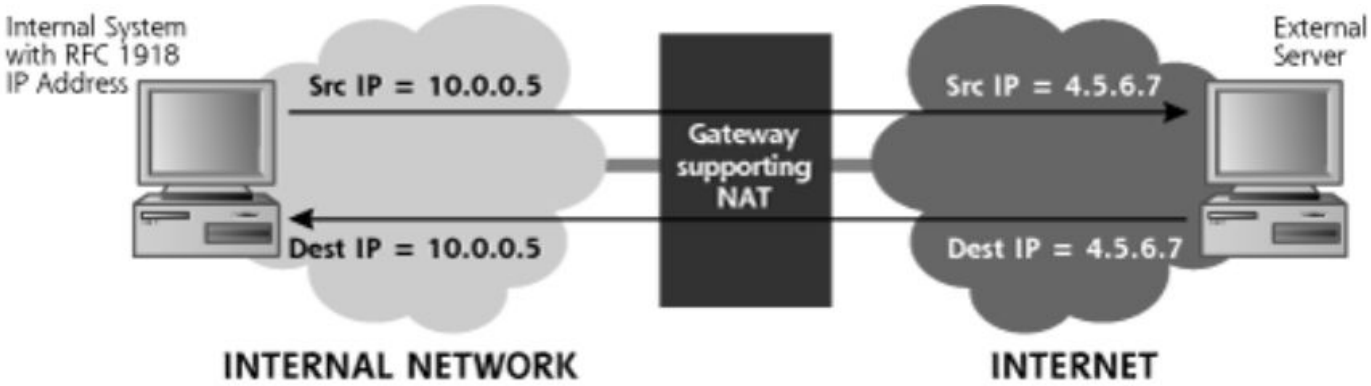
- A **supporting protocol** for sending error messages and operational information
- Used by ping and traceroute tools
- ICMP header format:

Source: Wikipedia

- Some control messages (with their **ICMP Types**):
  - Echo Reply (0), Destination Unreachable (3), Redirect Message (5), Echo Request (8), Time Exceeded (11), Parameter Problem: Bad IP header (12)

Network Address Translation (NAT)

- Necessary for private networks using private IP addresses to access the Internet
- Example:



Source: Skoudis & Liston, Counter Hack Reloaded

- Possible address mappings: to a single external IP address, 1-1 mapping, dynamic address mapping

Firewall

- Control flow of traffic *between* networks
- Different types of firewalls (based on network layer operations):
  - Traditional packet filters:
    - Check the following: source IP address, destination IP address, source TCP/UDP port, destination TCP/UDP port, TCP control bits, protocol in use, direction, interface
  - Stateful packet filters: keeps track of a state table
  - Proxy-based firewalls
- *Question: Differences with network-based IDS?*

Source: Wikipedia

Traceroute & Firewall: Extra Notes

- traceroute (UNIX):
  - Sends **UDP packets** by default
  - Can sends ICMP Echo Request (-I), or arbitrary protocol (-P)
- tracert (Windows):
  - sends **ICMP Echo Request** by default

- Firewalls **usually blocks** ICMP or unwelcome UDP!
- Other variants that use **TCP SYN** packets:
  - tcptraceroute (https://linux.die.net/man/1/tcptraceroute)
  - tctrace (http://manpages.ubuntu.com/manpages/cosmic/man1/tctrace.1.html)

Ethernet and 802.11

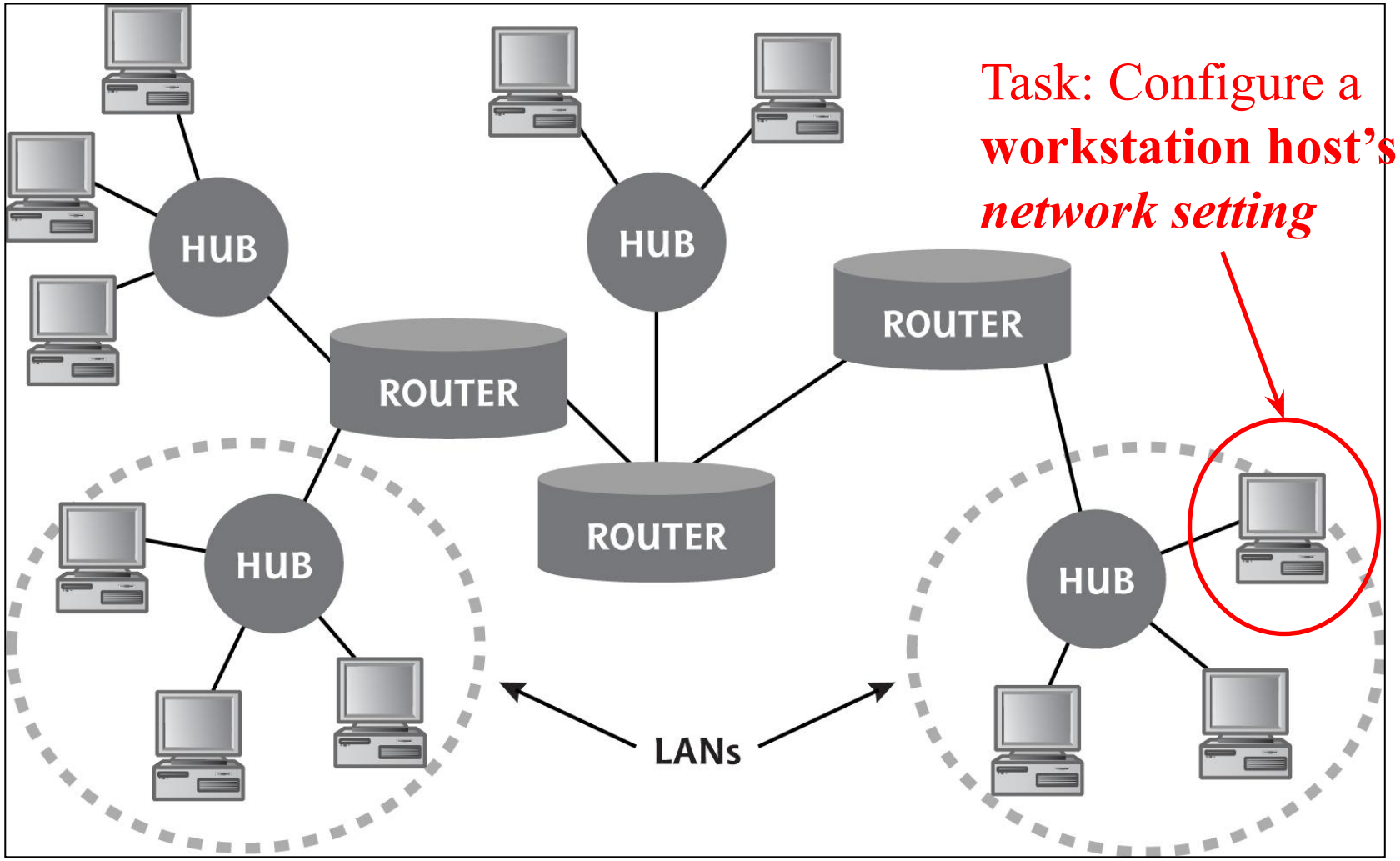
- Ethernet:
  - 48-bit MAC address
- Address Resolution Protocol (ARP):
  - Map logical IP address (layer 3) to physical MAC address (layer 2)
  - ARP Cache table for minimizing future ARP traffic
- Hubs vs switches:
  - Switches offer improved performance and better security
- 802.11: attacks on Ethernet are applicable too

Common Network Services

- telnet
- ssh
- ftp
- http
- r-commands: rlogin, rsh, rcp
- DNS
- NFS
- X Windows

Network Configuration:
Linux Desktop

Setting up a Computer

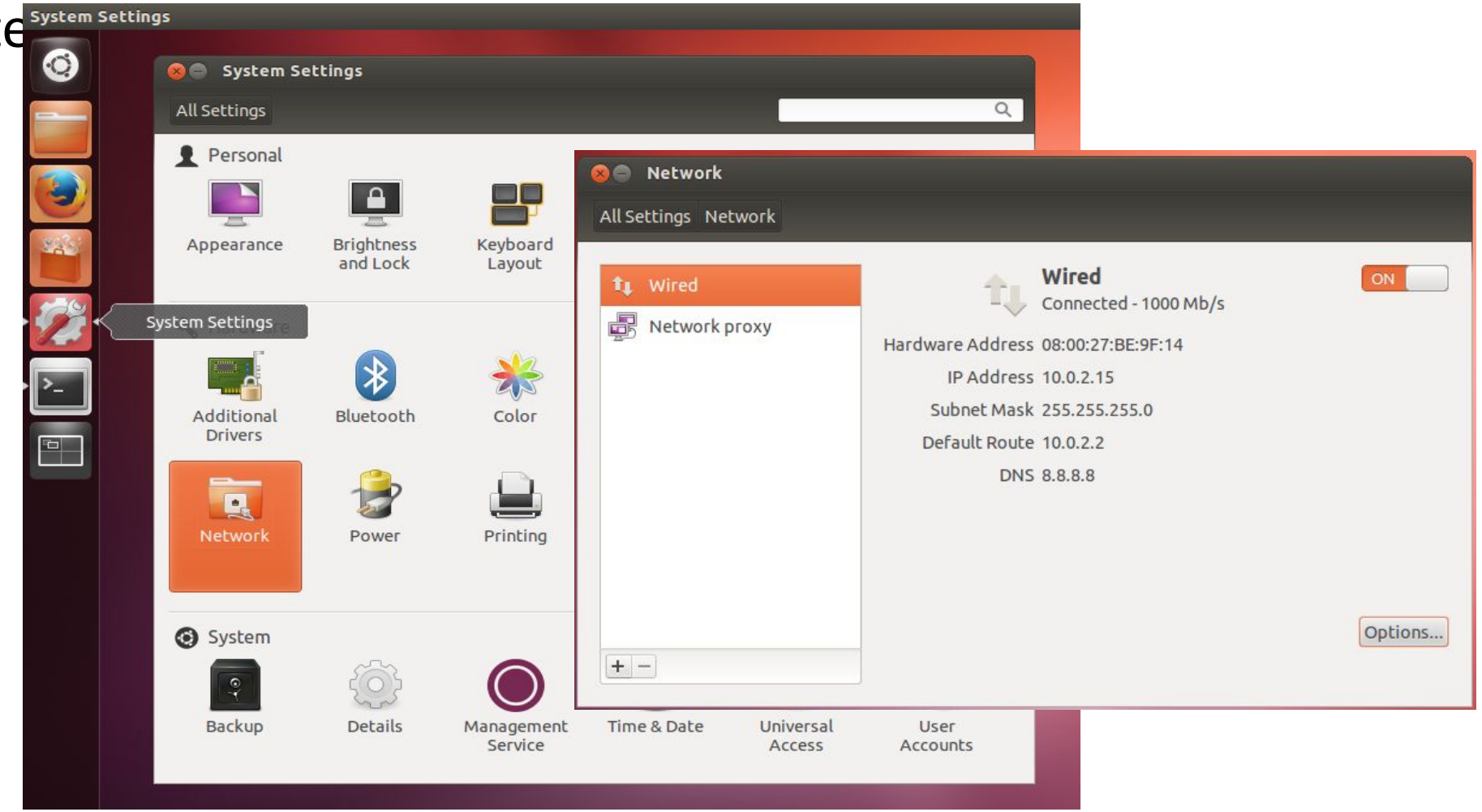


Computer Network Configuration

- **Information** needed to connect a computer to the Internet:
  - IP Address
  - Network mask
  - Gateway
  - DNS server
- *How to **obtain** such information?*
  - Automatic setting through DHCP
  - Manual setting

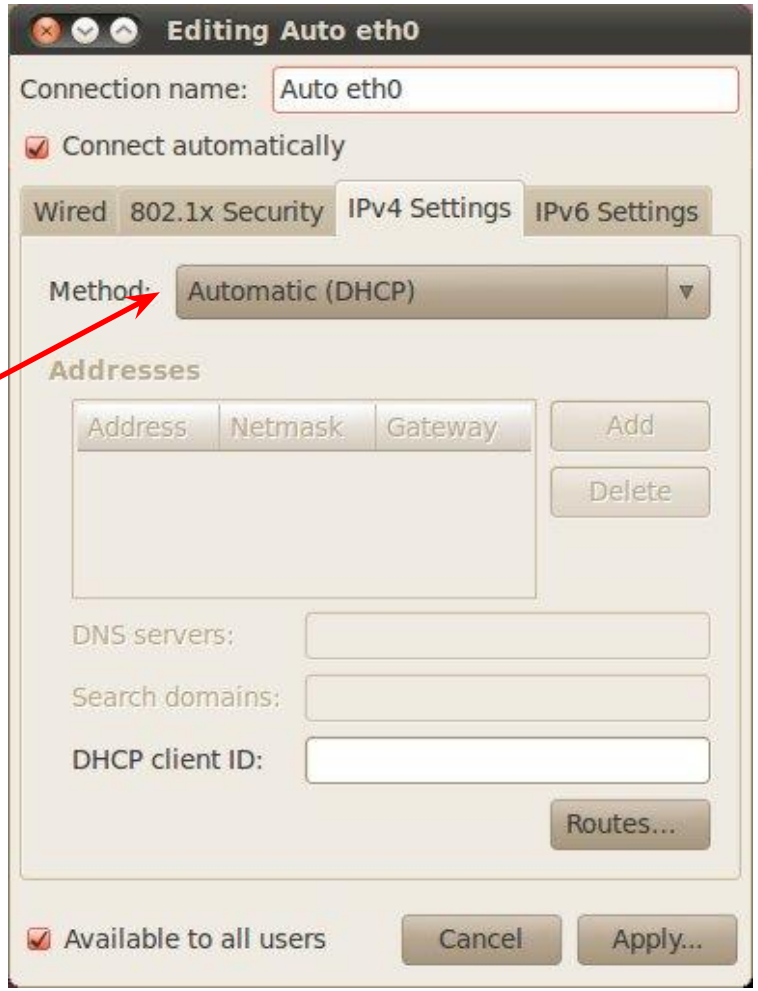
Configuration in Ubuntu Linux

- "Syste



Automatic Network Settings (DHCP)

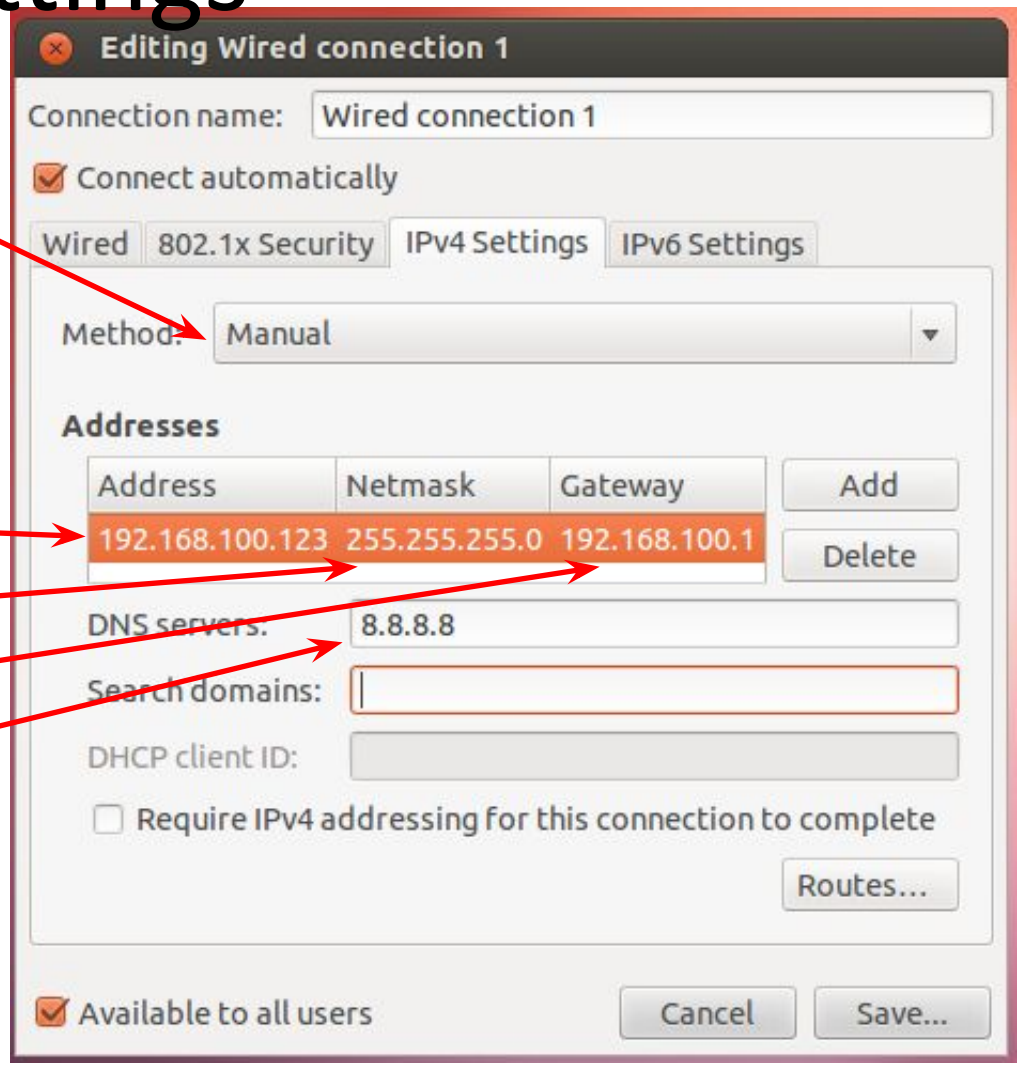
- Select your network interface, and click the "**Options**" button
- Select "**IPv4 Settings**" tab
- Set method to "**Automatic (DHCP)**" in order to automatically obtain network settings from **DHCP server**



Manual Network Settings

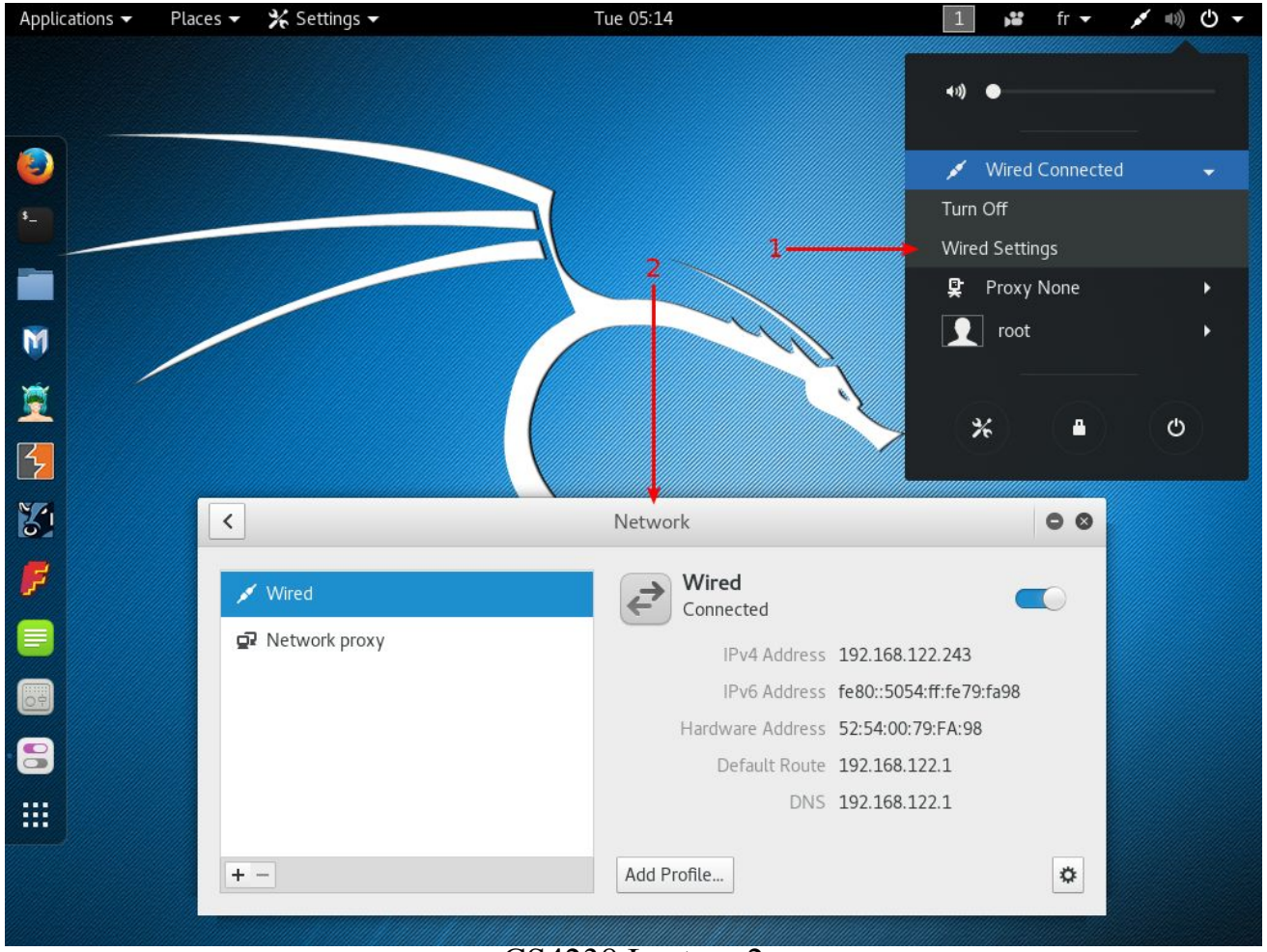
Set method to "**Manual**"

- IP Address
- Network mask
- Gateway
- DNS server



Configuration in Kali Linux

- *NetworkManager* setting interface:



Source: "Kali Linux Revealed", Hertzog et al., 2017

Network Setting File and Commands

- **Manual** network setting steps:
  - `ifdown` *<network-device>*
  - Modify **`/etc/network/interfaces`**
  - `ifup` *<network-device>*

- Setting `/etc/network/interfaces` for a plain **DHCP configuration**:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp
```

Network Setting File and Commands

- Setting `/etc/network/interfaces` for a **static IP** configuration:

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.0.3
netmask 255.255.255.0
broadcast 192.168.0.255
network 192.168.0.0
gateway 192.168.0.1
```

Configuring Kali Linux: Services

- Managing services:
  - E.g. ssh:
    - `systemctl start ssh`
    - `systemctl enable ssh`
    - `systemctl reload ssh`
  - E.g. Apache:
    - `systemctl start apache2`
    - `a2enmod` *module*
    - `a2dismod` *module*

Test Your Configuration

- If your setting steps, you should be able to **connect** to the Internet:
  - This slides are meant for the previous lab setup – you should already have it working!
- **Troubleshooting**: if your Internet connection doesn't work, try to diagnose it:
  - Can you reach your gateway? (use `ping` command)
    - Note that ping may not work for various reasons
  - Can you reach your DNS server? (use `ping` command)
  - Can you resolve a domain name? (use `nslookup`)

Some Useful Commands

- Check and start/stop network interfaces using **`ifconfig`** :
  - **List** network interfaces:
    - **All** interfaces (**up and down**) whose drivers are loaded:

```
$ ifconfig -a
```

    - All interfaces that are **up**:

```
$ ifconfig
```

    - A particular interface (e.g. eth0):

```
$ ifconfig eth0
```

  - **Start** and **stop** a network interface (e.g. eth0):

```
$ ifconfig eth0 down
$ ifconfig eth0 up
```

Consistent Network Device Naming

- A convention for naming **Ethernet adapters** in Linux
- Created ~2009 to replace the old eth*X* naming:
  - **Issues** on multihomed machines
  - NICs would be named based on **the order** in which they were found by the kernel as it booted
  - Removing existing or adding new interfaces?
- Device naming rules:
  - **Onboard** interfaces at firmware index nos: `eno`[1-*N*]
  - Interfaces at **PCI Express hotplug** slot nos: `ens`[1-*N*]
  - Adapters in the specified **PCI slot**, with slot index no on the adapter `enp`<*PCI-slot*>`s`<*card-index-no*>

Some Useful Commands

- Newer **`ip`** command from **iproute2**:
  - **List** network interfaces:
    - **All** interfaces (up and down) whose drivers are loaded:

      ```
      $ ip addr show
      ```

    - A particular interface (e.g. eth0):

      ```
      $ ip addr show eth0
      ```

    - IPv4 or IPv6 addresses only:
    - `$ ip -4|-6 addr show`
  - **Start** and **stop** a network interface (e.g. eth0):

    ```
    $ ip link set eth0 down
    $ ip link set eth0 up
    ```

Linux Network Commands: Deprecated and New

- Old-style network utilities from net-tools (`ifconfig`, `route`, …) are ***supposed*** to be replaced by iproute2:
  - `ifconfig` → `ip`
  - `route` → `ip`
  - `arp` → `ip`
  - `netstat` → `ss` (socket statistics)
- Sample command comparisons:
  - `route -n` vs `ip route show`
  - `route add default gw` *<gateway-IP-addr>* vs
    `ip route add default via` *<gateway-IP-addr>*

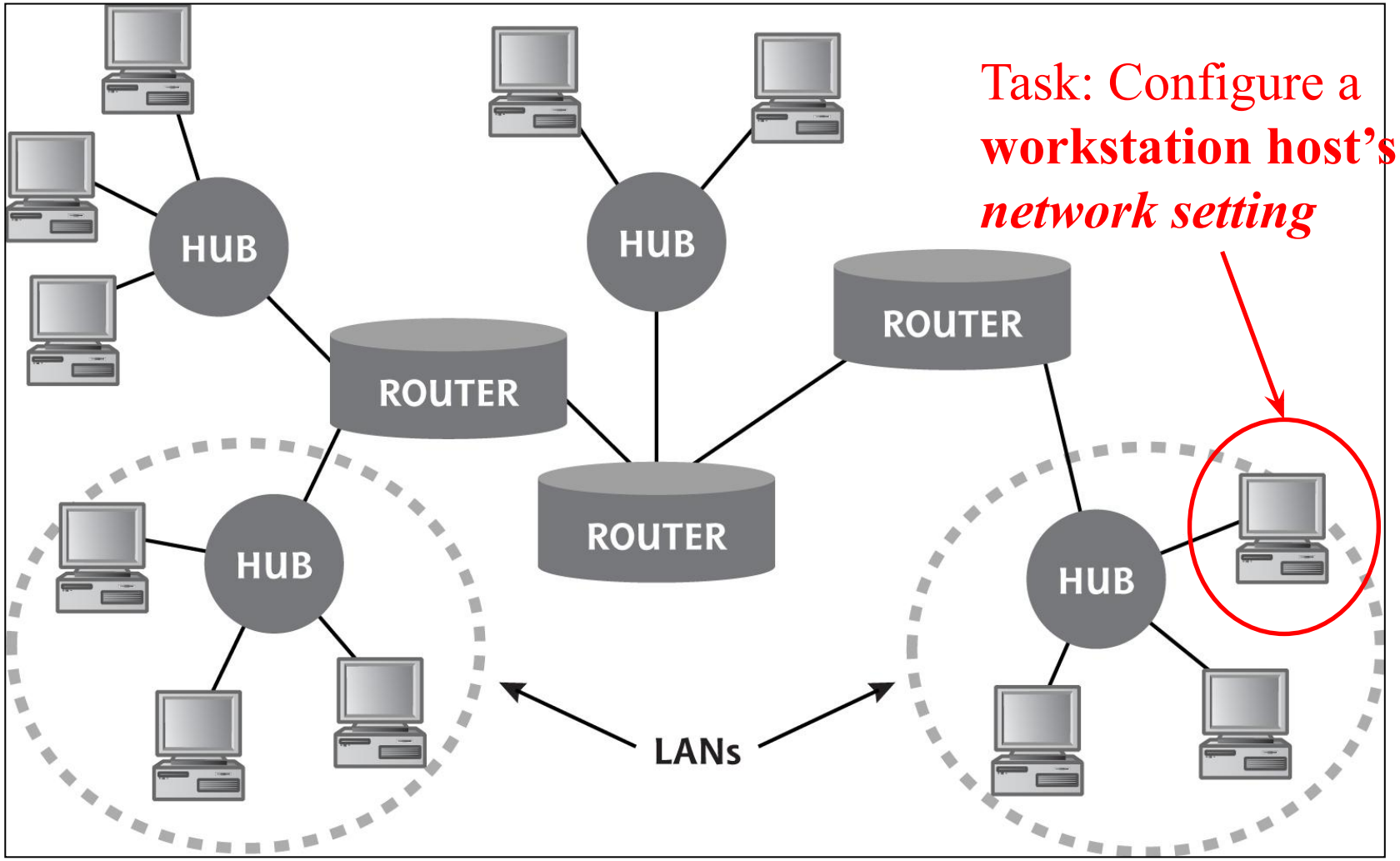
References: ip Command

- https://phoenixnap.com/kb/linux-ip-command-examples
- https://www.howtogeek.com/657911/how-to-use-the-ip-command-on-linux/

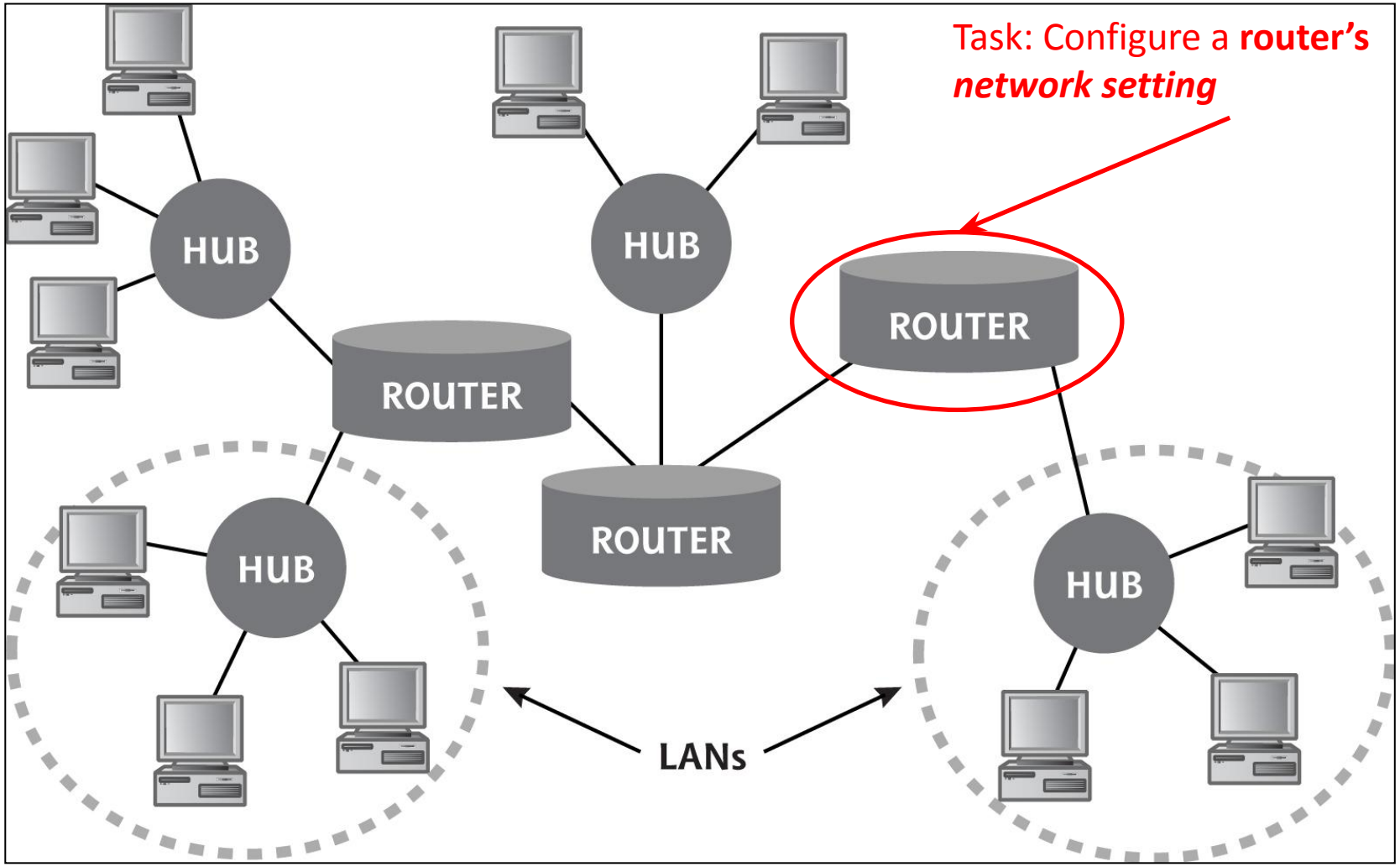
Network Configuration: Linux Router

Setting up a Computer

Done

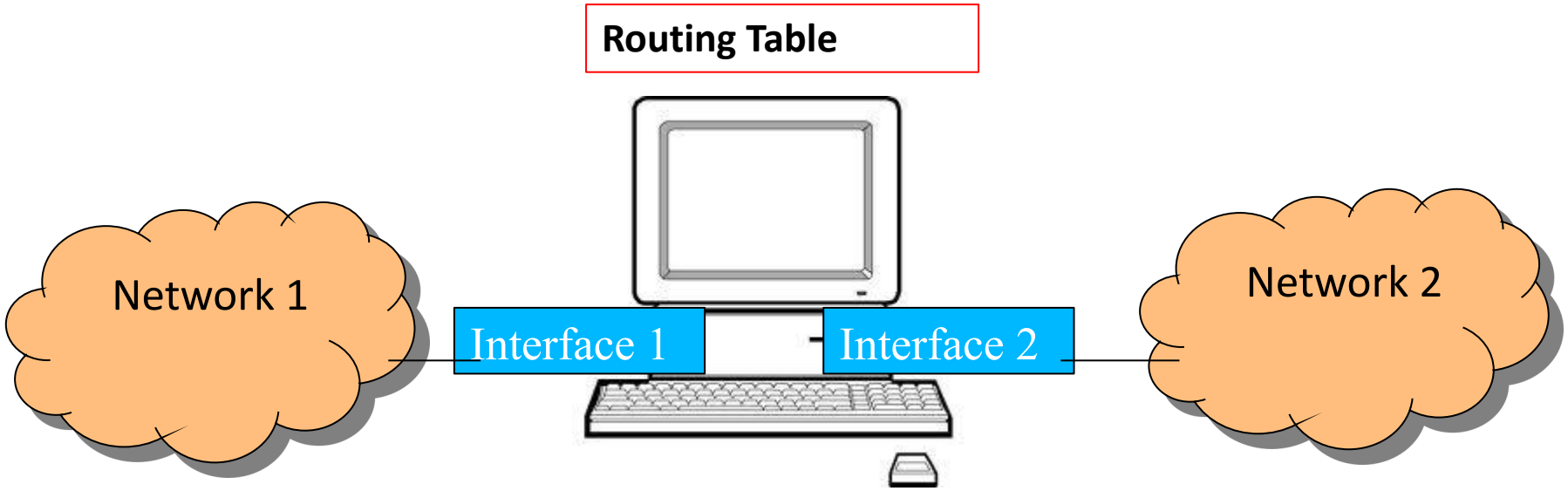


Setting up a Router



Configure a Computer as Router

- Router is a computer with ***multiple*** network interfaces
- After receiving a packet, it looks up its ***routing table*** to decide where to forward this packet



Enable Routing on Linux

- Linux kernel **already comes** with routing functionality:
  - **Check** whether it is enabled:

    ```
    $ sysctl net.ipv4.ip_forward
    ```

  - **Turn it on** by editing the configuration file `/etc/sysctl.conf` and uncomment the following line:

    ```
    net.ipv4.ip_forward = 1
    ```

  - **Reload** sysctl settings:

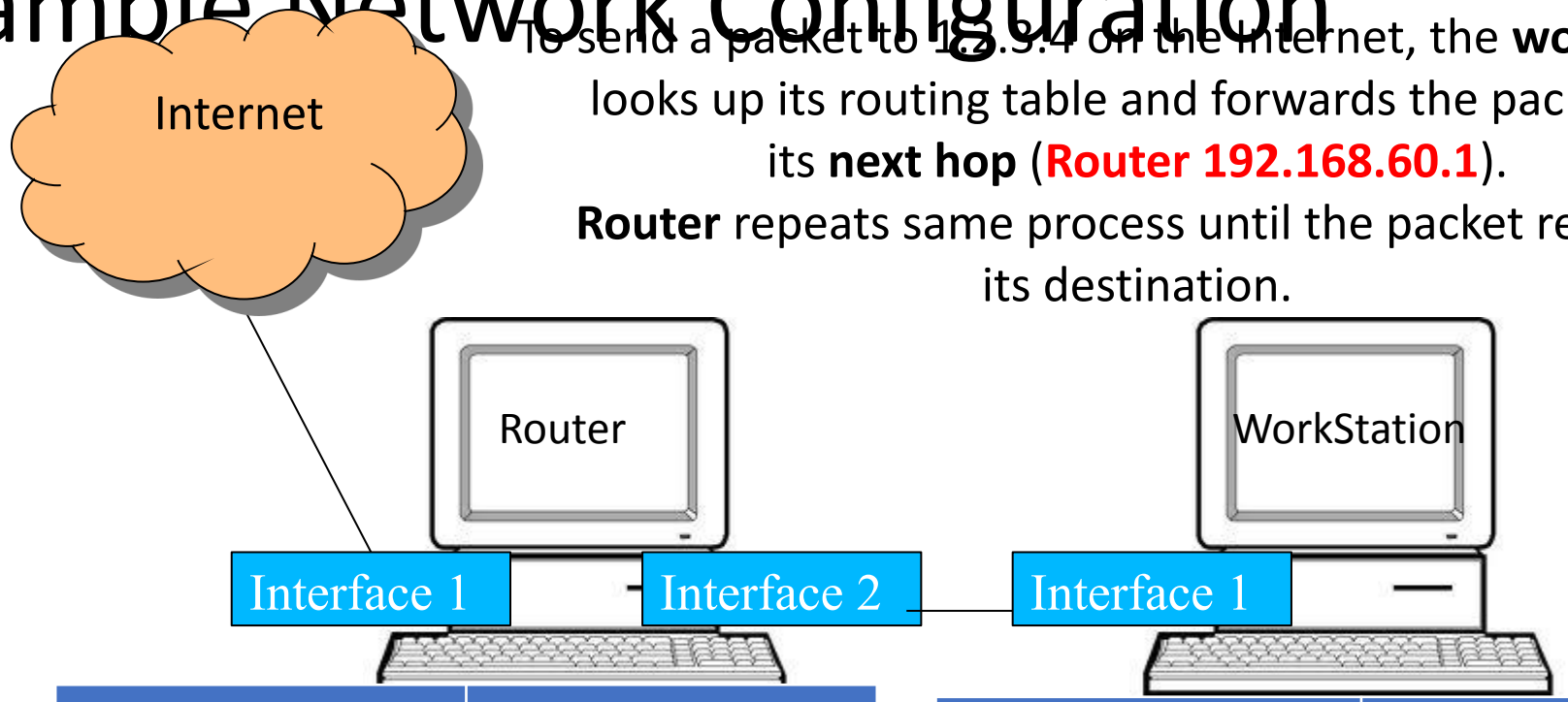
    ```
    $ sudo sysctl -p
    ```

- You usually don't need to configure the **routing table**.
  Use the command `ip` to display it (a newer alternative to `route -n`):

  ```
  $ ip route show
  ```

A Sample Network Configuration

To send a packet to 1.2.3.4 on the Internet, the **workstation** looks up its routing table and forwards the packet to its **next hop** (**Router 192.168.60.1**).
Router repeats same process until the packet reaches its destination.



Gateway Settings	
Interface 1 Netmask	172.26.184.101 255.255.255.0
Interface 2 Netmask	192.168.60.1 255.255.255.0
Gateway	172.26.184.1
DNS	137.132.94.2

Workstation	Settings
Interface 1 Netmask	192.168.60.101 255.255.255.0
Gateway	192.168.60.1
DNS	137.132.94.2

Network Traffic Analysis

Network Traffic Analysis

- How can we **capture** and **analyze** network traffic?
- Put a network interface controller (NIC) into ***promiscuous mode***: see all the traffic visible on that interfaces, including unicast traffic **not sent** to that NIC's MAC
- **Wireshark**: a free and open source packet analyzer
- **No 1** in the "Top 125 Network Security Tools" of SecTools.Org
- Widely used by both network administrators and attackers

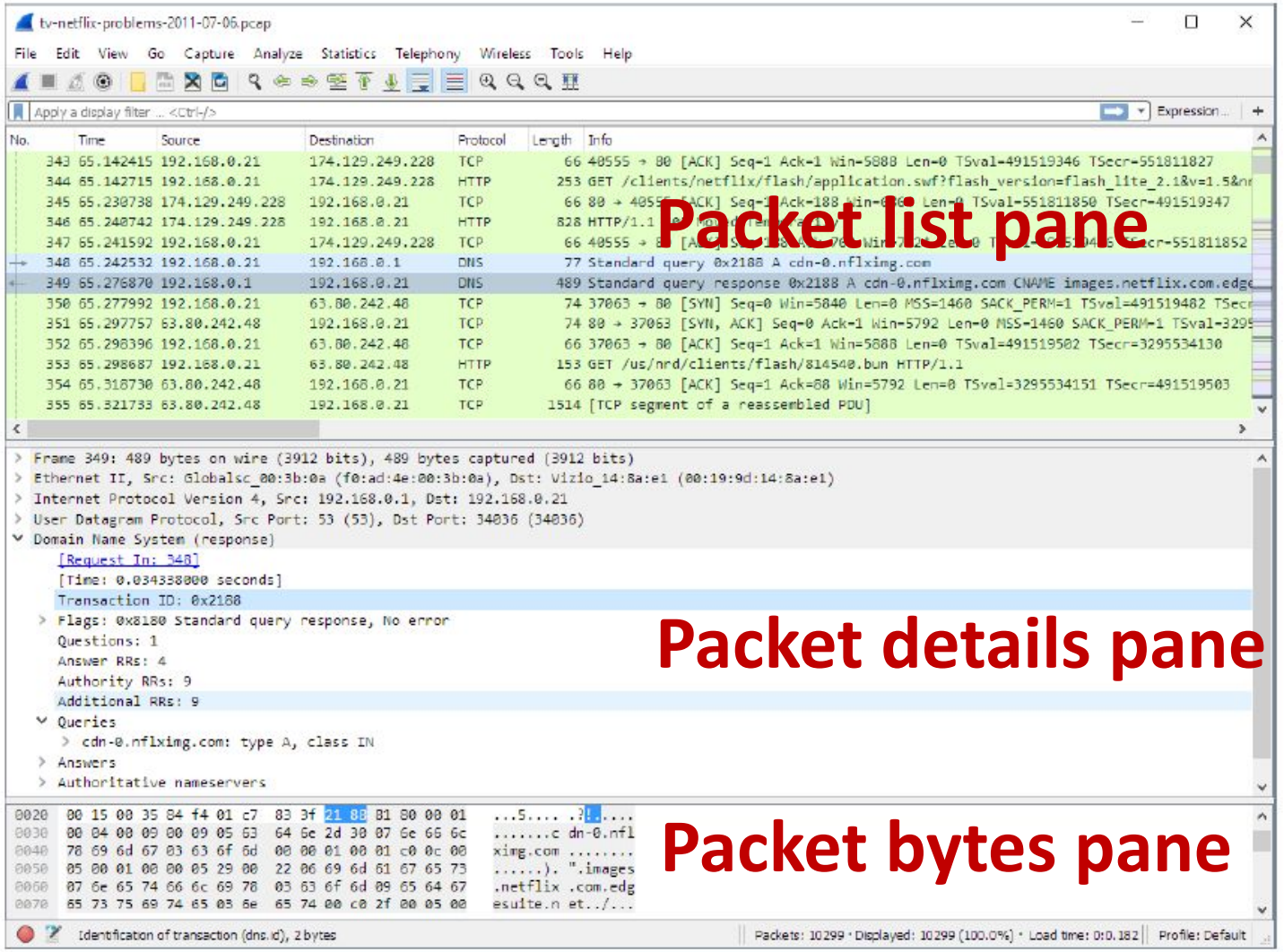
Wireshark Background

- **History**:
  - July 1998: *Ethereal* version 0.2.0
  - 2006: the project moved house and re-emerged under a new name *Wireshark*
  - 2008: Wireshark version 1.0
  - 2015: Wireshark 2.0
  - 2018: Version 2.9.0
- Wireshark uses **pcap** to capture packets: **libpcap** library (UNIX/Linux), **WinPcap** (Windows)
- Other alternative tools: tcpdump, snoop, **tshark**, …

Wireshark Features

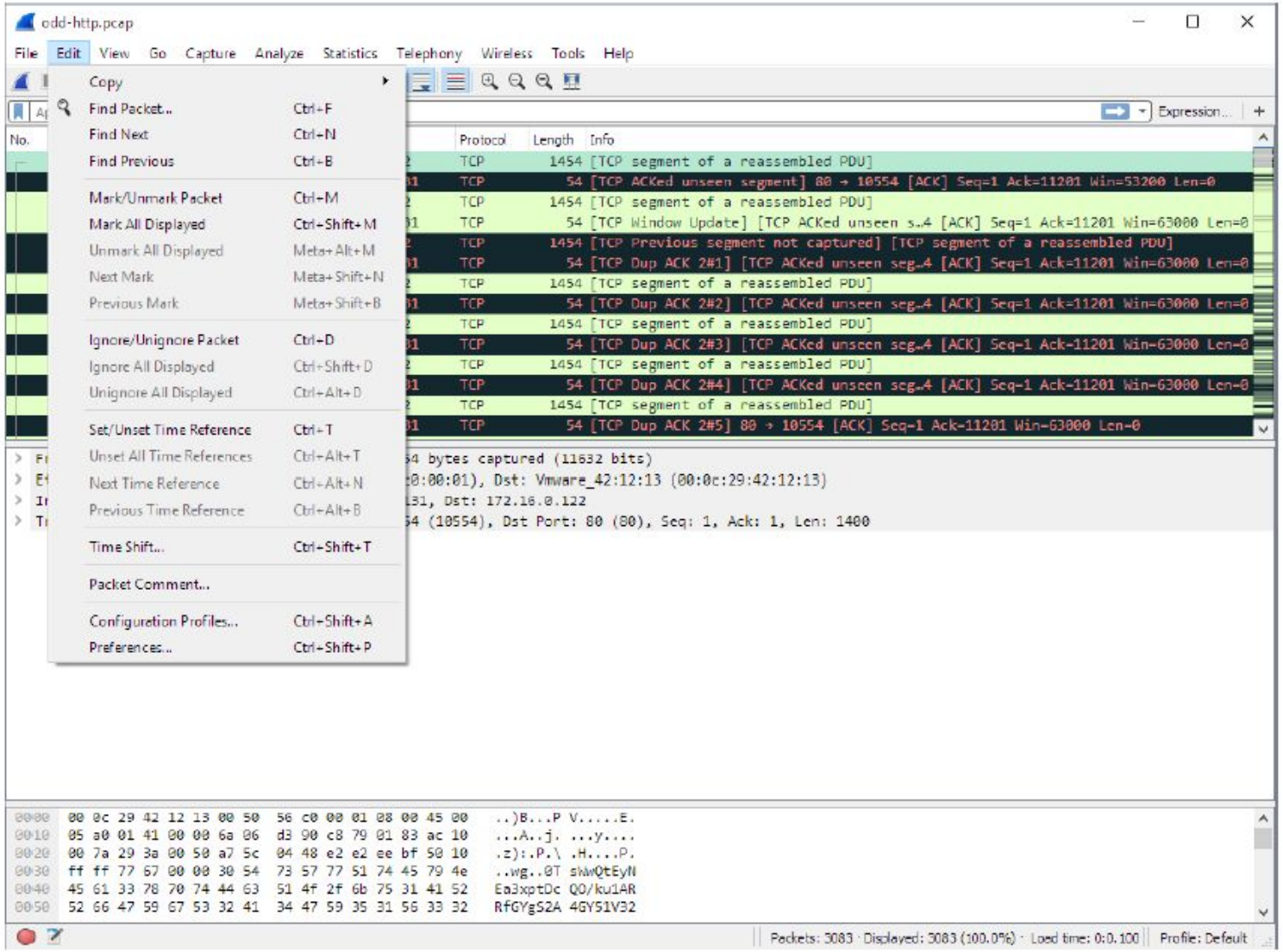
- Some **good features**:
  - Import files from other capture programs
  - Various protocol **dissectors**
  - **Filter** packets on many criteria
  - Search for packets on many criteria
  - Colorize packet display based on filters
- What Wireshark ***is not***?
  - Wireshark is not an IDS
  - Wireshark will not manipulate things on the network, it will only “measure” things from it

Demo & Practice:
Traffic Analysis using WireShark



Source: Wireshark User's Guide

Useful Wireshark Tips: Edit Menu



Source: Wireshark User's Guide

Useful Wireshark Tips: Find Packet

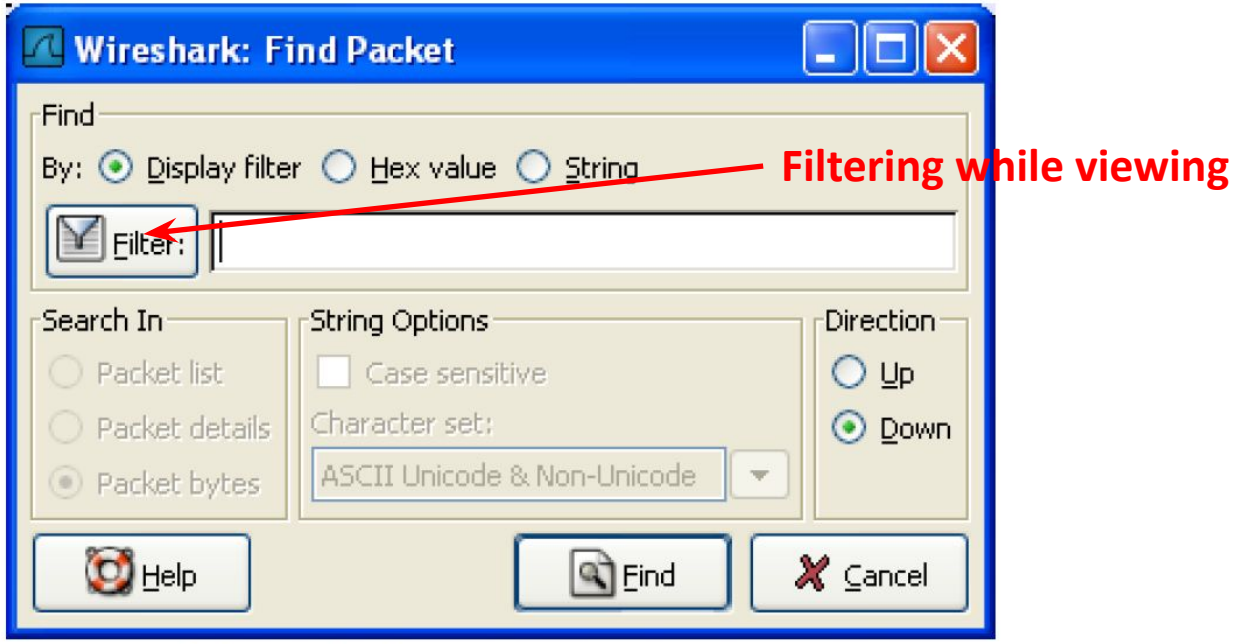
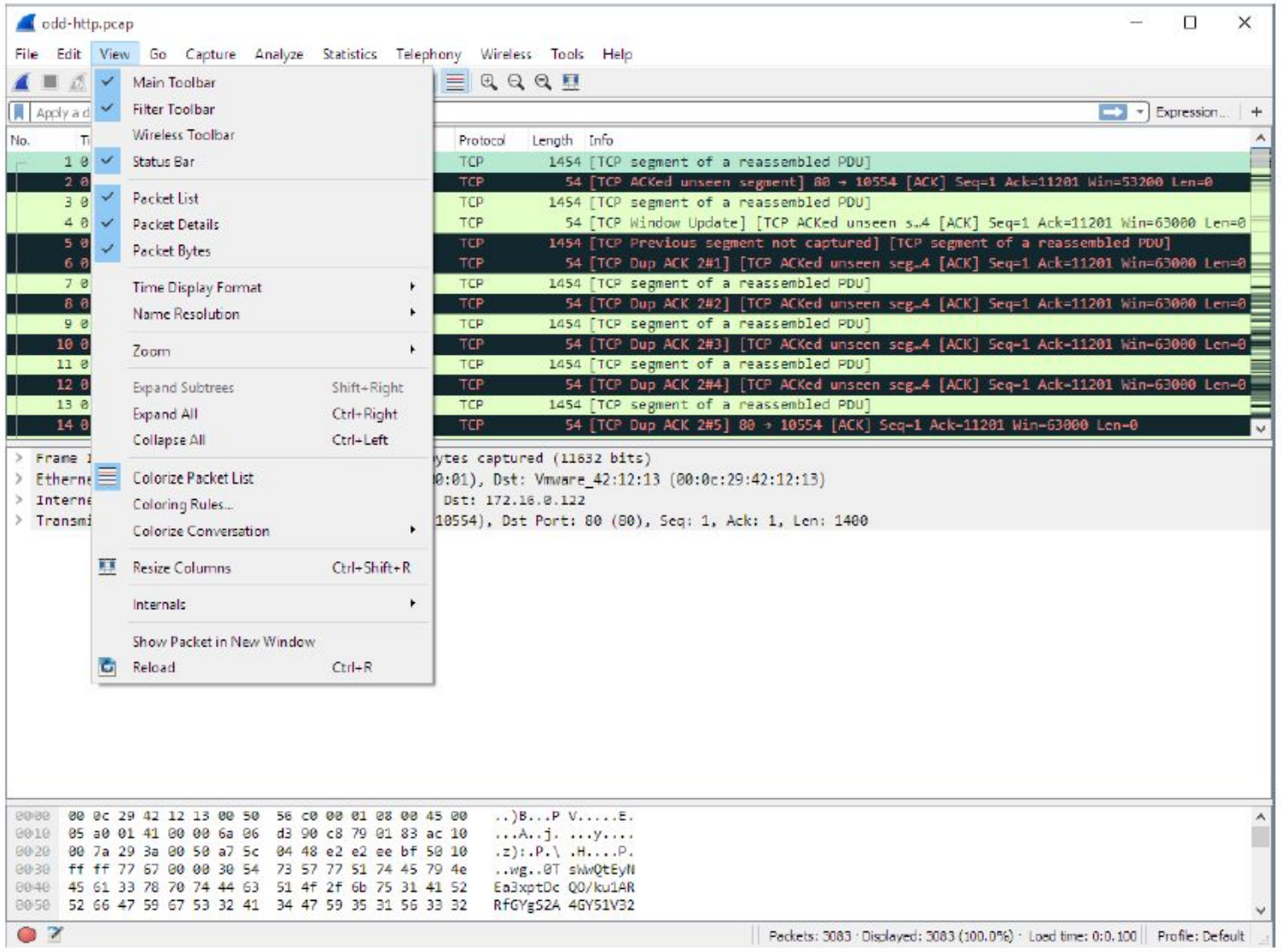


Figure 64. The "Find Packet" dialog box

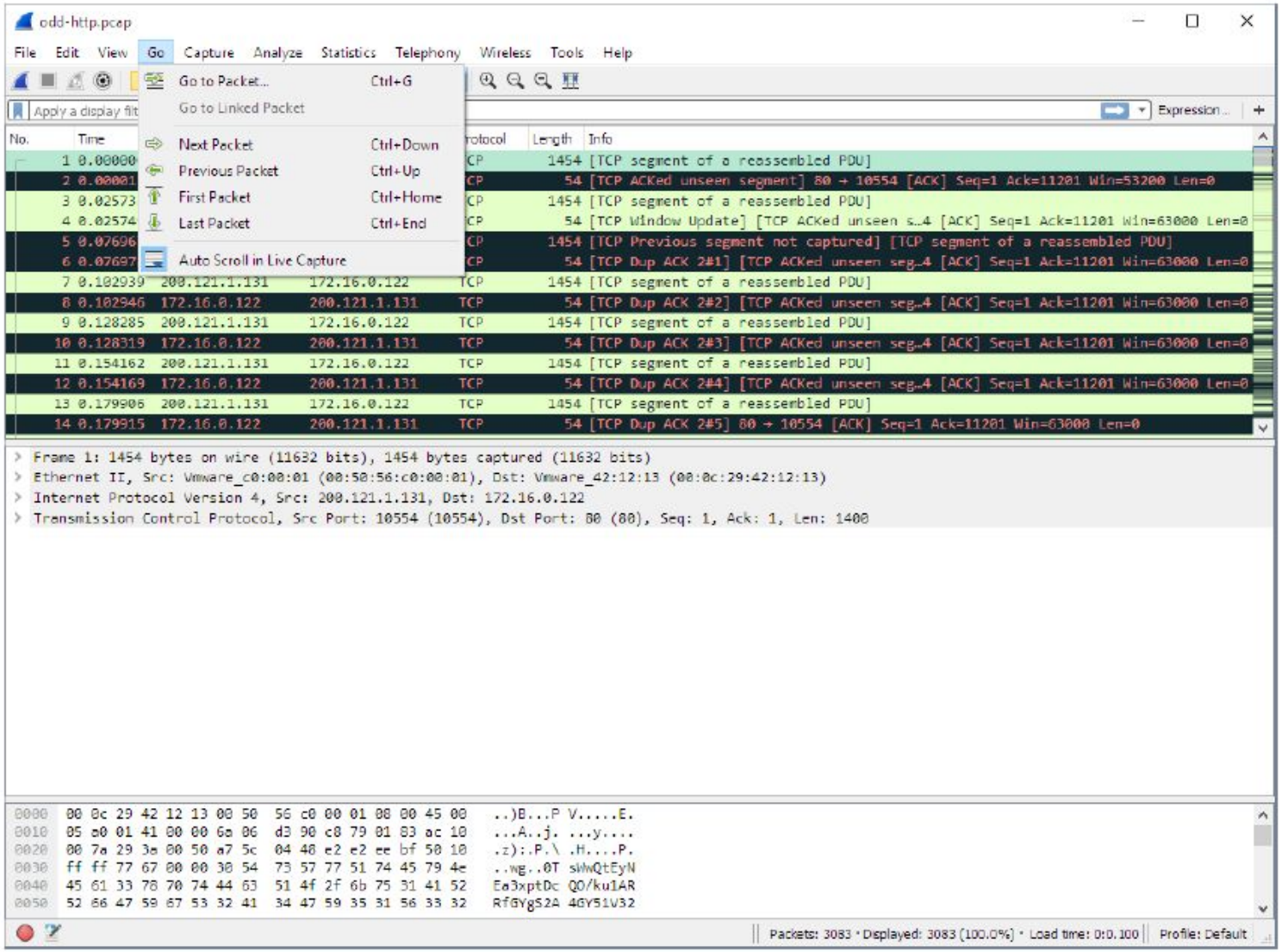
Source: Wireshark User's Guide

Useful Wireshark Tips: View Menu



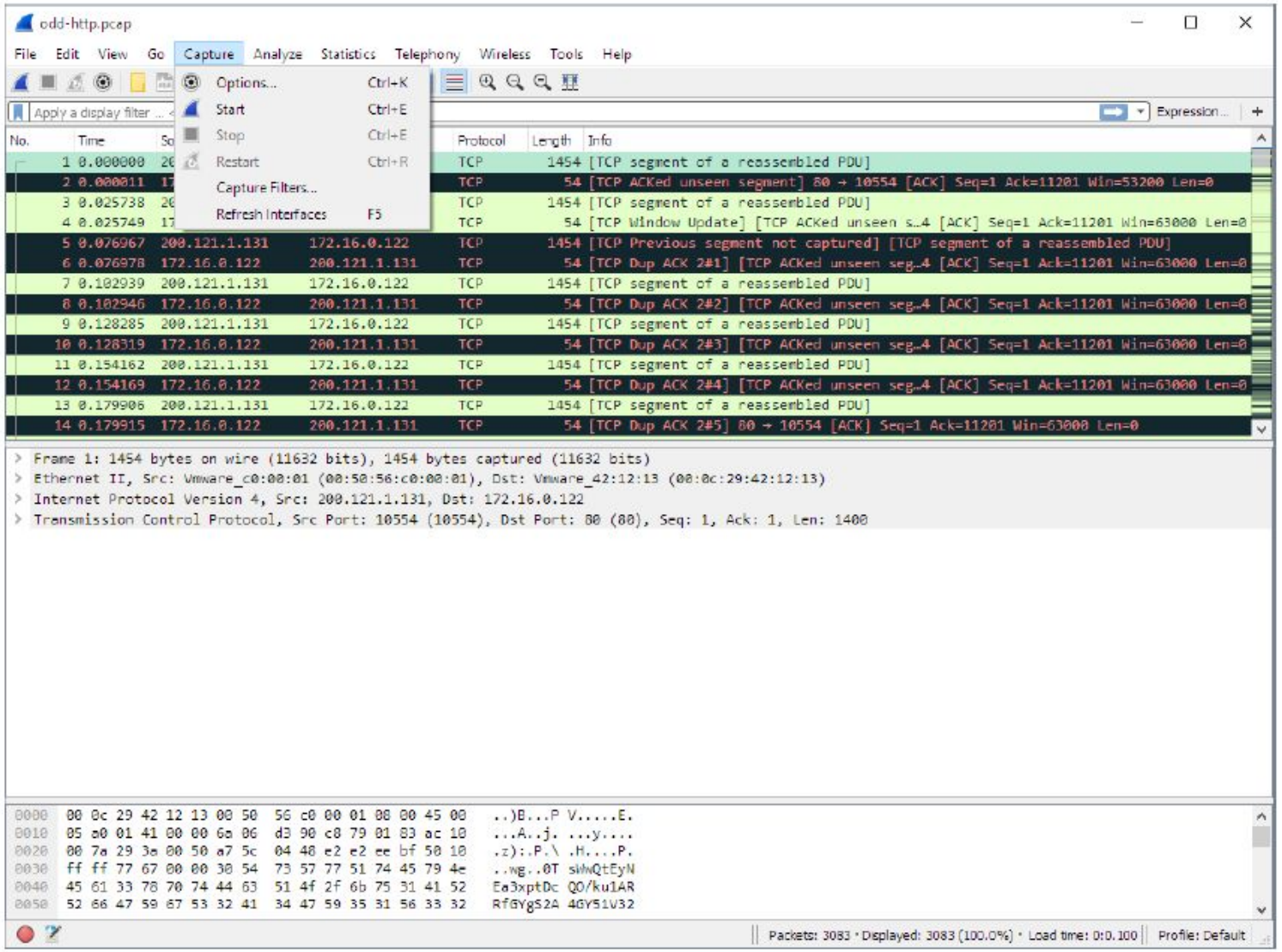
Source: Wireshark User's Guide

Useful Wireshark Tips: Go Menu



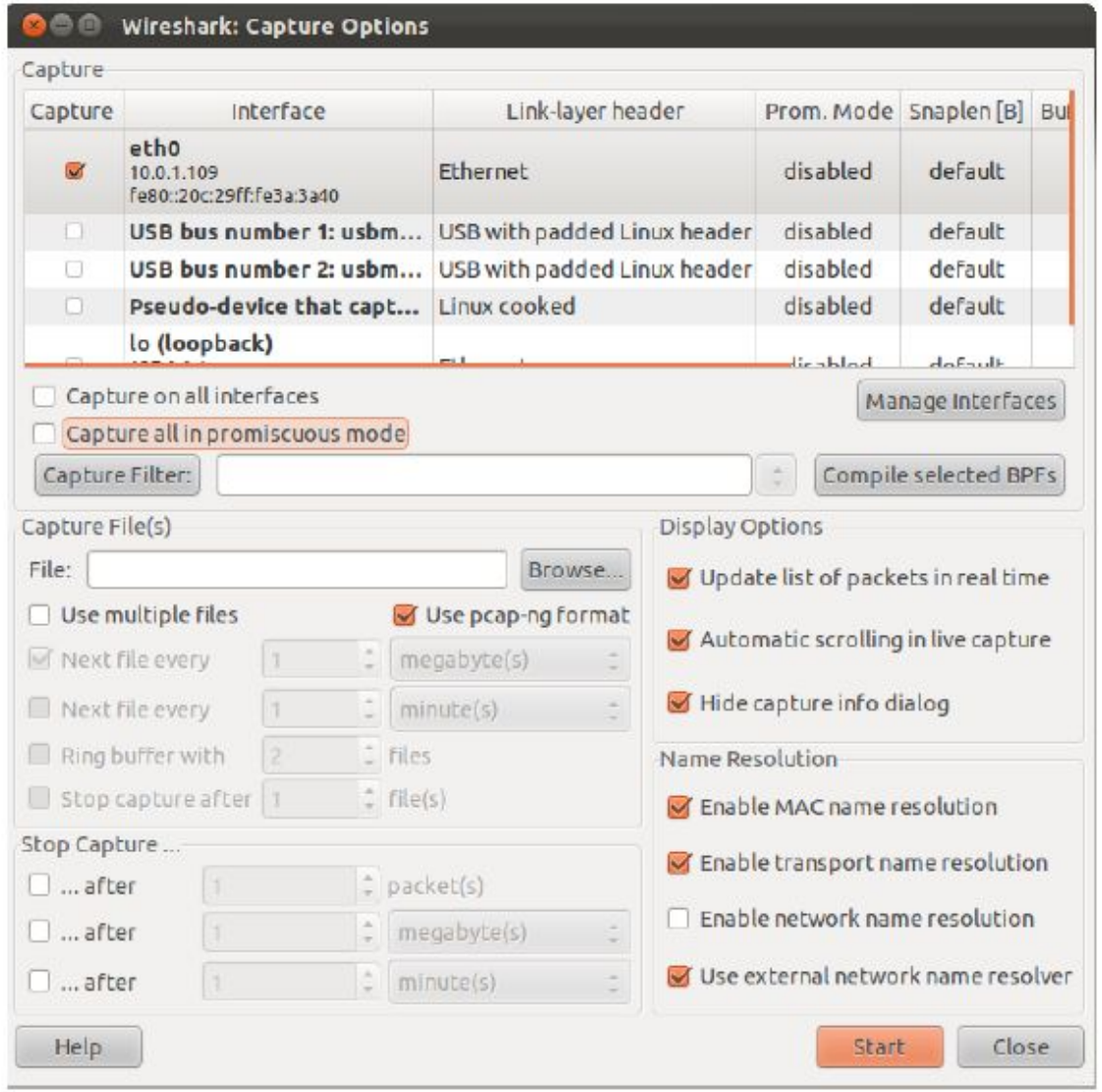
Source: Wireshark User's Guide

Useful Wireshark Tips: Capture Menu



Source: Wireshark User's Guide

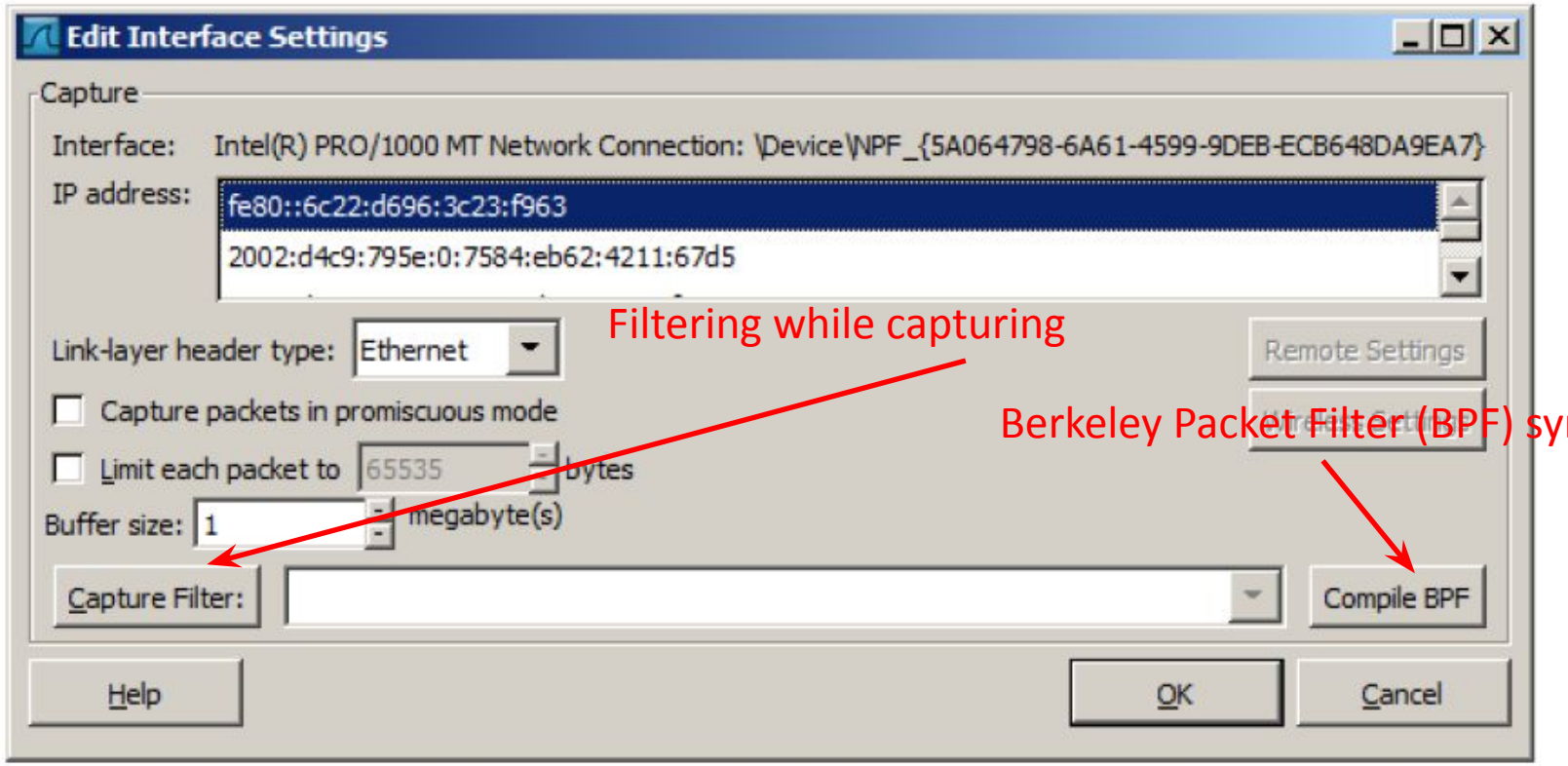
Useful Wireshark Tips: Capture Options



Source: Wireshark User's Guide

Useful Wireshark Tips

Interface setting:

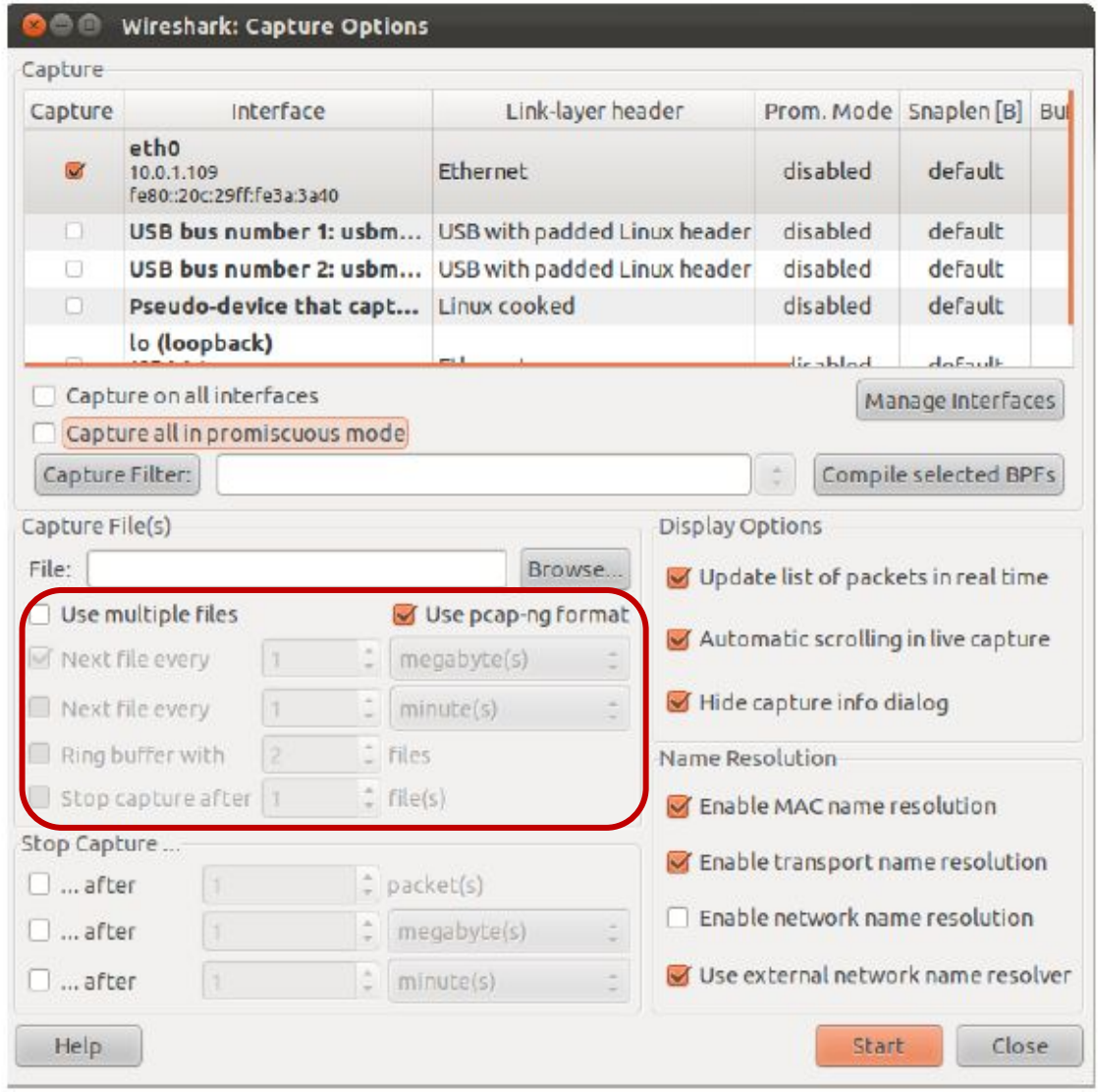


Source: Wireshark User's Guide

Examples of Capture Filters

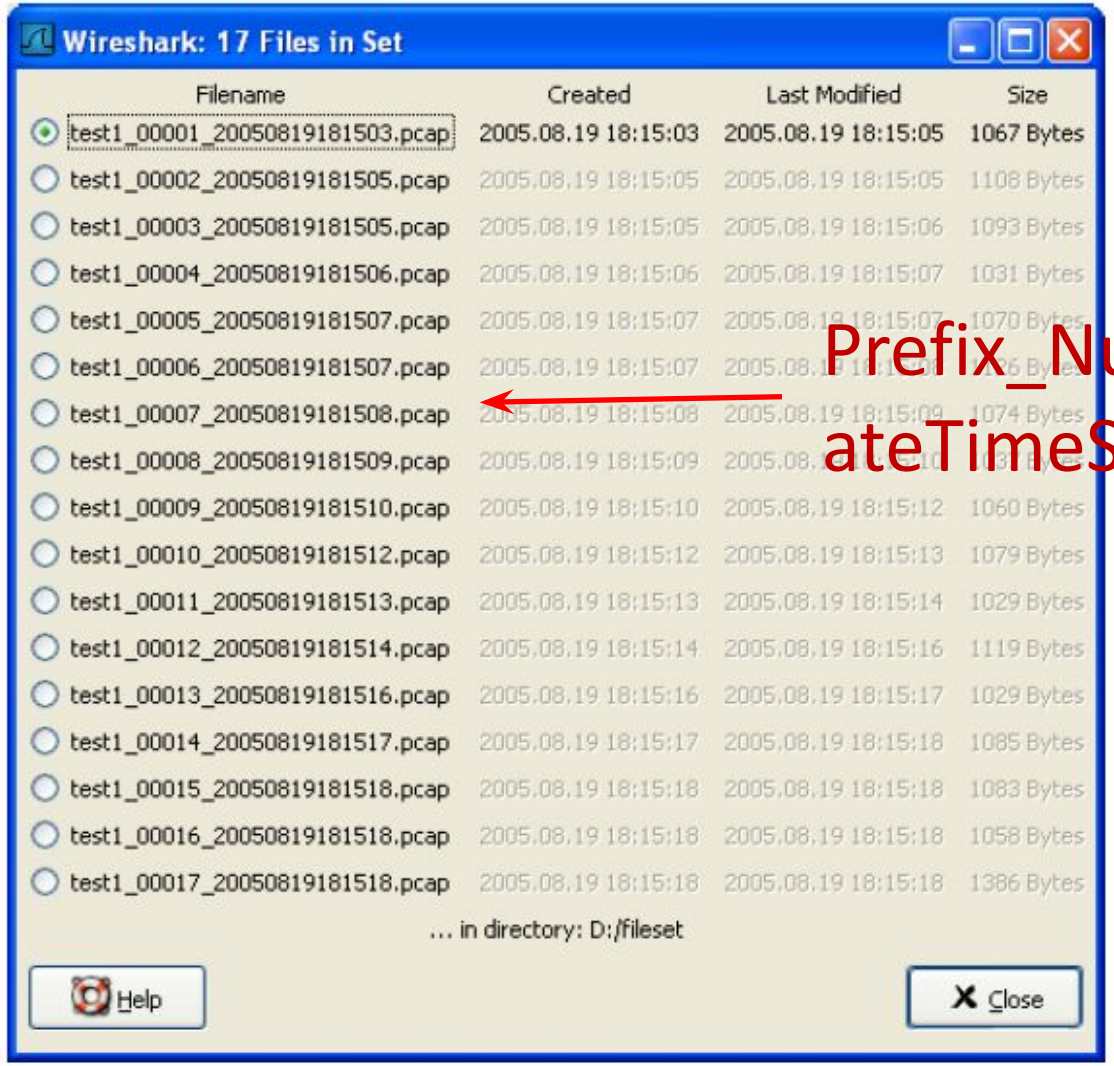
• Ethernet address <*address*>	`ether host` <*address*>
• Ethernet type 0x0806 (ARP)	`ether proto 0x0806`
• No ARP	`not arp`
• IPv4 only	`ip`
• IPv4 address <*address*>	`host` <*IPv4-address*>
• IPv6 only	`ip6`
• IPv6 address <*address*>	`host` <*IPv6-address*>
• TCP only	`tcp`
• UDP only	`udp`
• TCP or UDP port 80 (HTTP)	`port 80`
• HTTP TCP port (80)	`tcp port http`

Useful Wireshark Tips: Capture Options



Source: Wireshark User's Guide

Useful Wireshark Tips: Multiple Files



Source: Wireshark User's Guide

Useful Wireshark Tips: Popup Menu 1

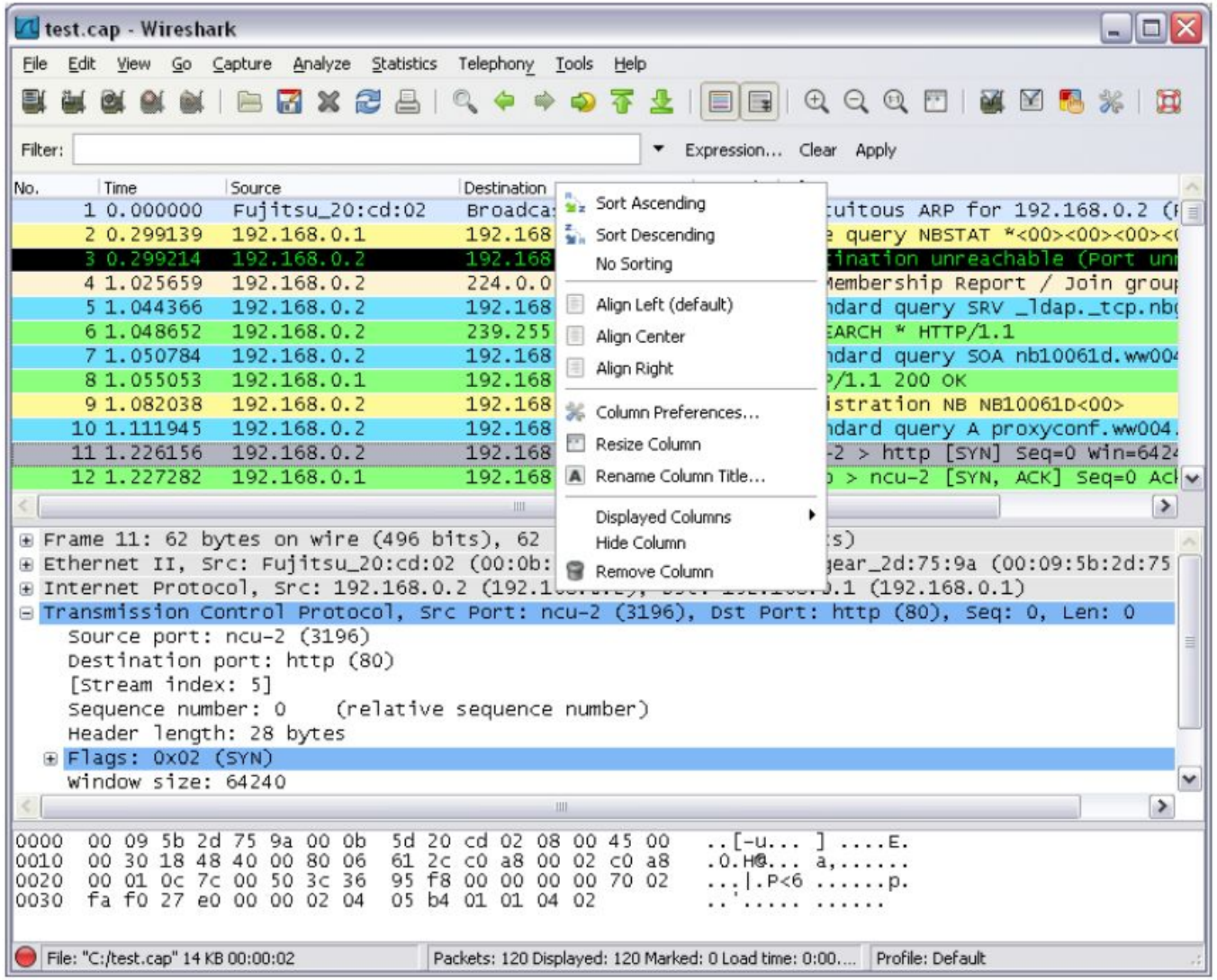


Figure 58. Pop-up menu of the "Packet List" column header

Source: Wireshark User's Guide

Useful Wireshark Tips: Popup Menu 2

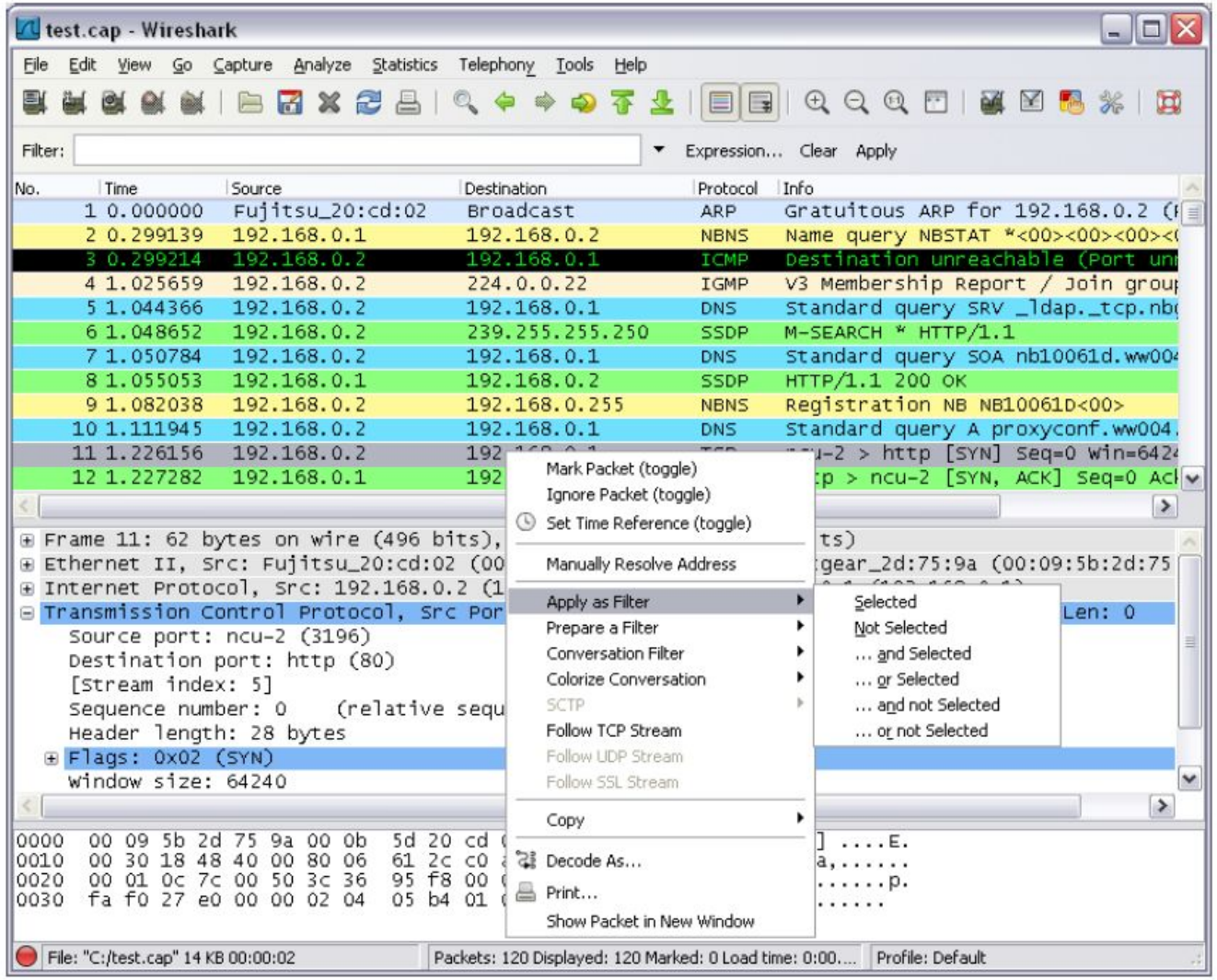


Figure 59. Pop-up menu of the "Packet List" pane

Source: Wireshark User's Guide

Useful Wireshark Tips: Popup Menu 3

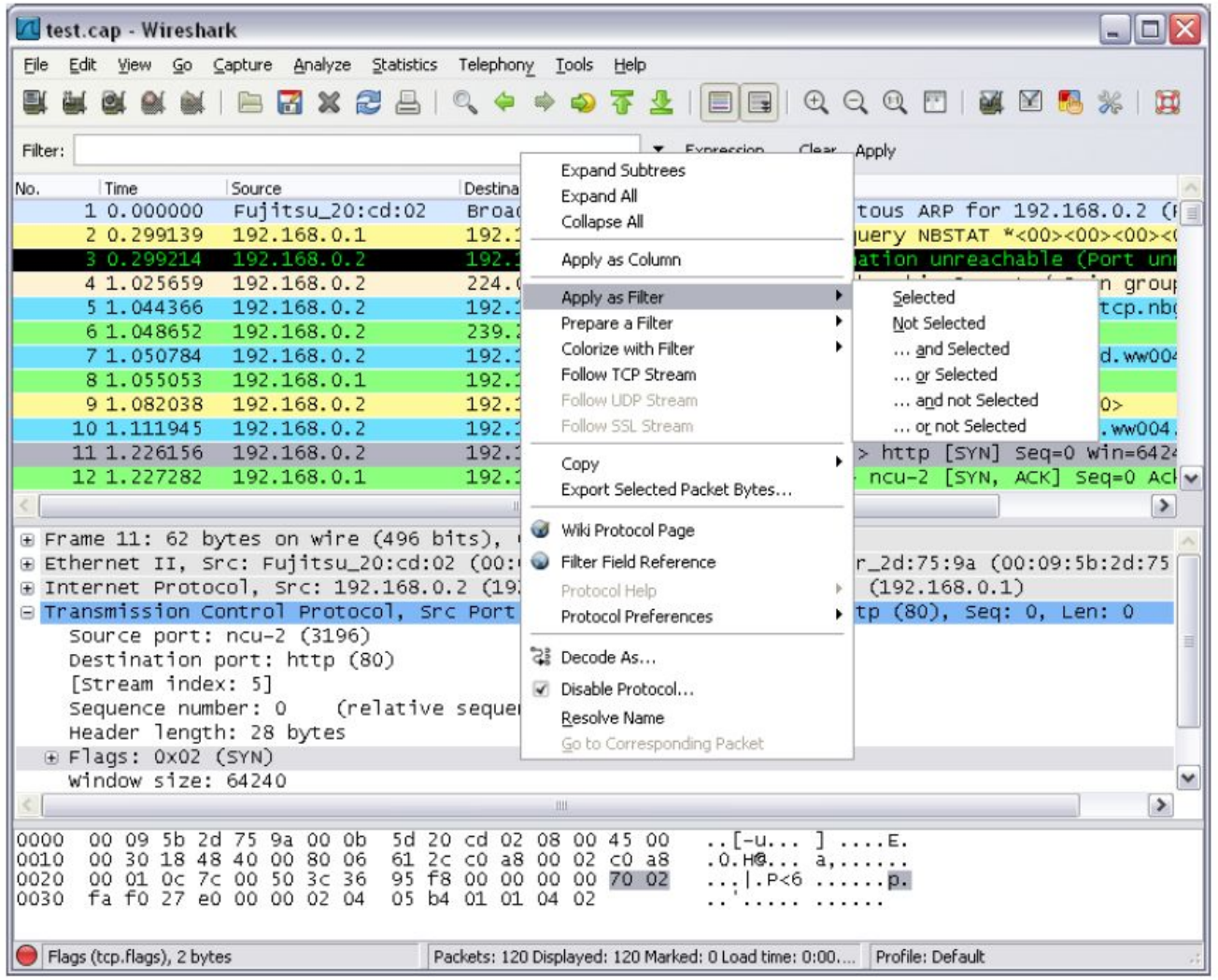
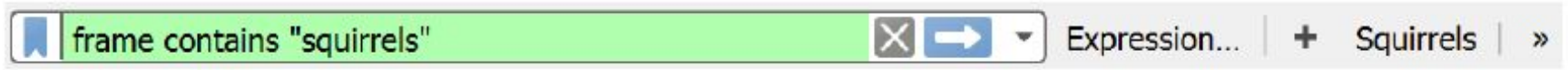


Figure 60. Pop-up menu of the "Packet Details" pane

Source: Wireshark User's Guide

Useful Wireshark Tips: Filter

You need to specify a good (while-viewing) filter:



Filter comparison operators

Table 20. Display Filter comparison operators

English	C-like	Description and example
eq	==	Equal. `ip.src==10.0.0.5`
ne	!=	Not equal. `ip.src!=10.0.0.5`
gt	>	Greater than. `frame.len > 10`
lt	<	Less than. `frame.len < 128`
ge	>=	Greater than or equal to. `frame.len ge 0x100`
le	<=	Less than or equal to. `frame.len <= 0x20`
contains		Protocol, field or slice contains a value. `sip.To contains "a1762"`
matches	~	Protocol or text field match Perl regualar expression. `http.host matches "acme\.(org|com|net)"`
bitwise_and	&	Compare bit field value. `tcp.flags & 0x02`

Source: Wireshark User's Guide

Useful Wireshark Tips: Export Object

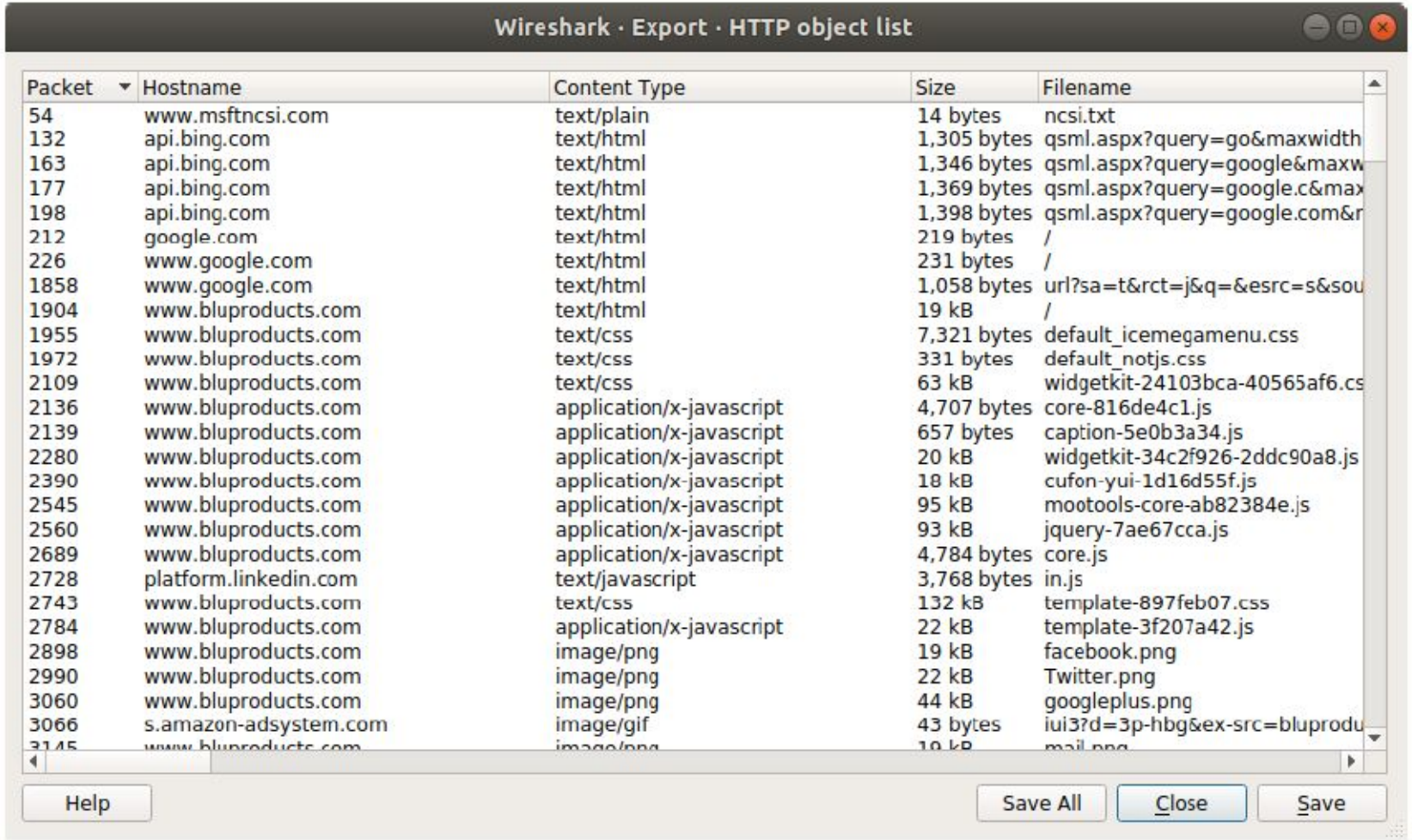


Figure 52. The "Export Objects" dialog box

Source: Wireshark User's Guide

Useful Wireshark Tips: Follow TCP Stream

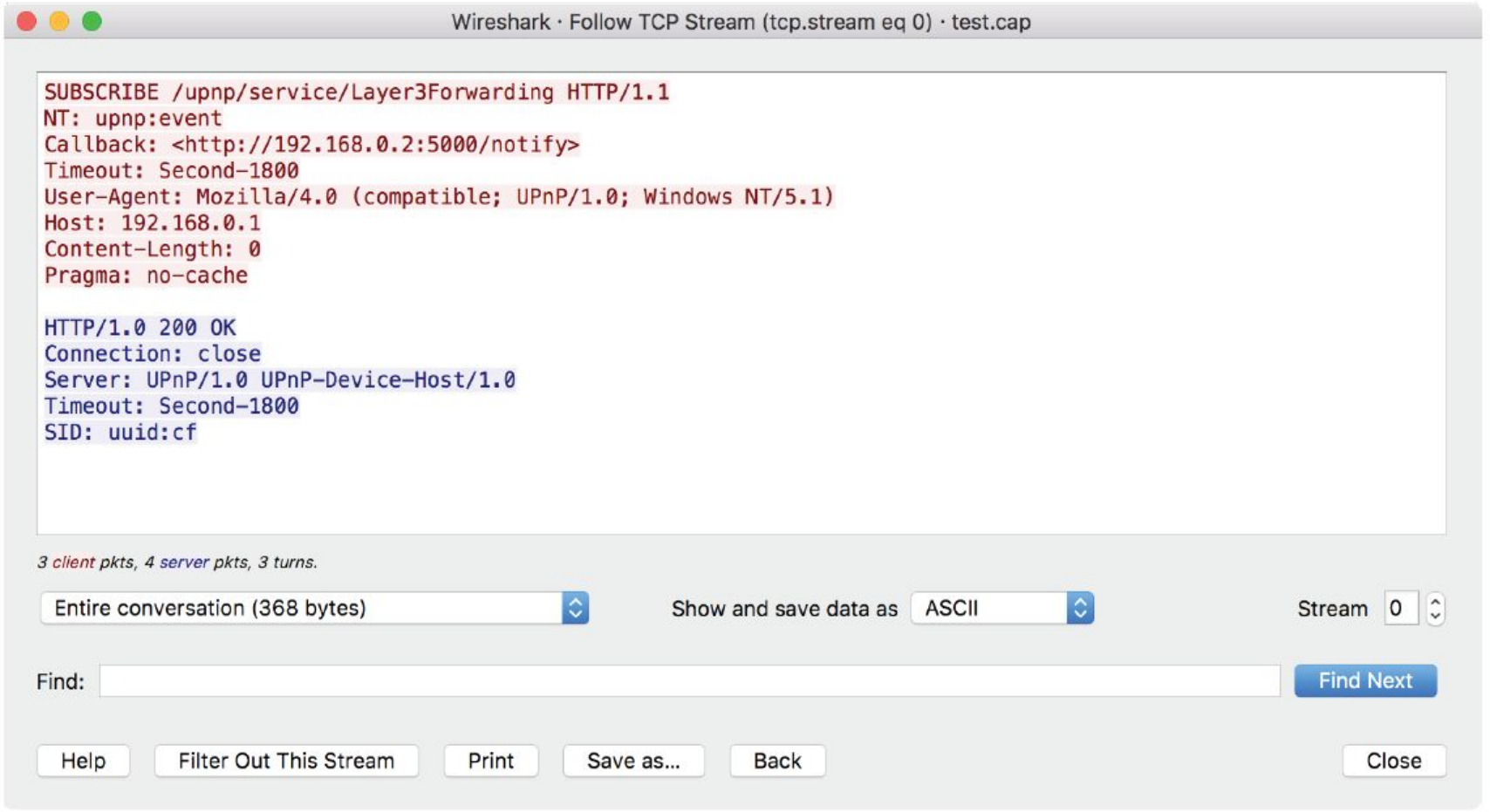
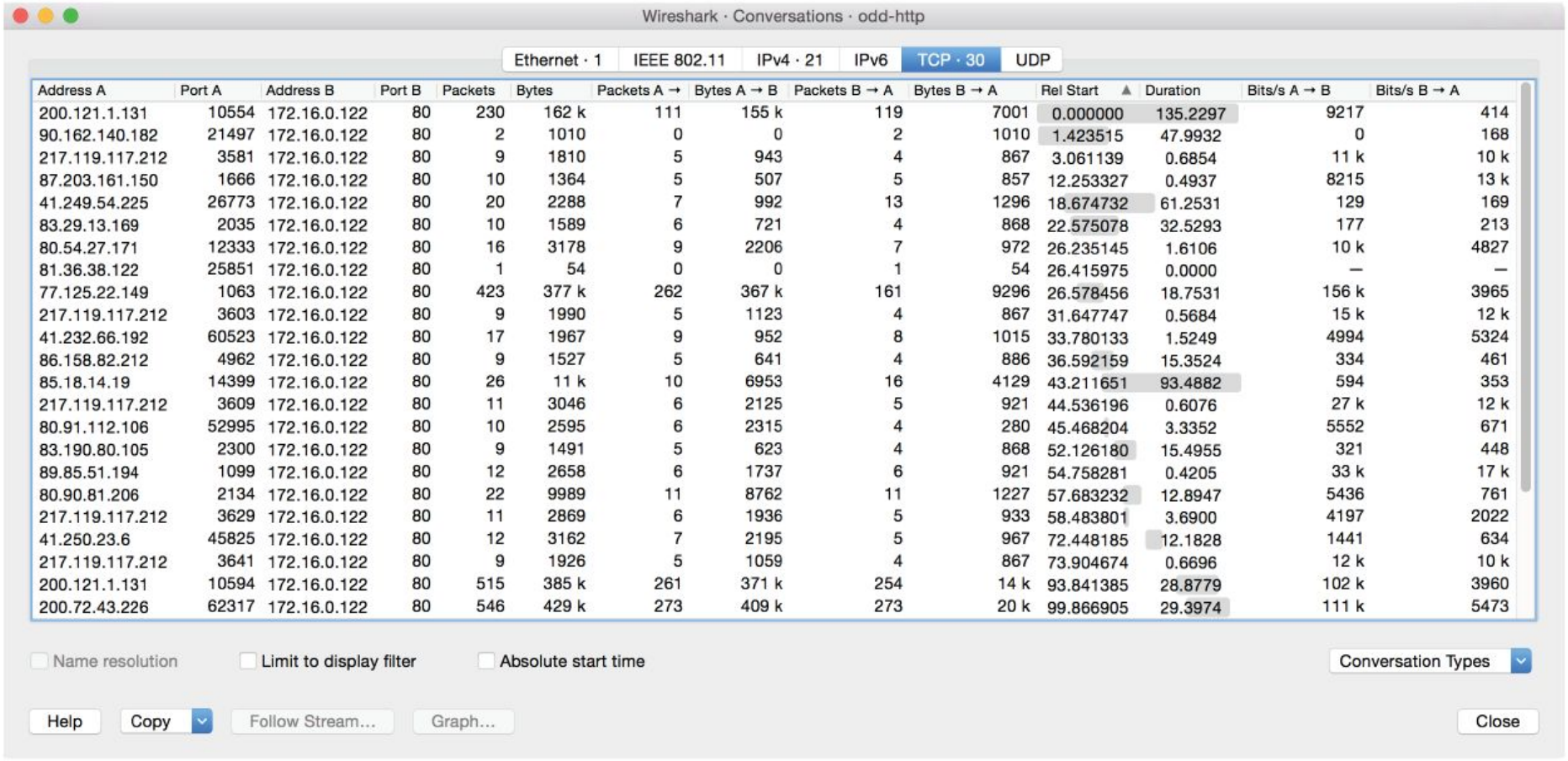


Figure 67. The "Follow TCP Stream" dialog box

Source: Wireshark User's Guide

Useful Wireshark Tips: Statistics



Address A	Port A	Address B	Port B	Packets	Bytes	Packets A →	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
200.121.1.131	10554	172.16.0.122	80	230	162 k	111	155 k	119	7001	0.000000	135.2297	9217	414
90.162.140.182	21497	172.16.0.122	80	2	1010	0	0	2	1010	1.423515	47.9932	0	168
217.119.117.212	3581	172.16.0.122	80	9	1810	5	943	4	867	3.061139	0.6854	11 k	10 k
87.203.161.150	1666	172.16.0.122	80	10	1364	5	507	5	857	12.253327	0.4937	8215	13 k
41.249.54.225	26773	172.16.0.122	80	20	2288	7	992	13	1296	18.674732	61.2531	129	169
83.29.13.169	2035	172.16.0.122	80	10	1589	6	721	4	868	22.575078	32.5293	177	213
80.54.27.171	12333	172.16.0.122	80	16	3178	9	2206	7	972	26.235145	1.6106	10 k	4827
81.36.38.122	25851	172.16.0.122	80	1	54	0	0	1	54	26.415975	0.0000	—	—
77.125.22.149	1063	172.16.0.122	80	423	377 k	262	367 k	161	9296	26.578456	18.7531	156 k	3965
217.119.117.212	3603	172.16.0.122	80	9	1990	5	1123	4	867	31.647747	0.5684	15 k	12 k
41.232.66.192	60523	172.16.0.122	80	17	1967	9	952	8	1015	33.780133	1.5249	4994	5324
86.158.82.212	4962	172.16.0.122	80	9	1527	5	641	4	886	36.592159	15.3524	334	461
85.18.14.19	14399	172.16.0.122	80	26	11 k	10	6953	16	4129	43.211651	93.4882	594	353
217.119.117.212	3609	172.16.0.122	80	11	3046	6	2125	5	921	44.536196	0.6076	27 k	12 k
80.91.112.106	52995	172.16.0.122	80	10	2595	6	2315	4	280	45.468204	3.3352	5552	671
83.190.80.105	2300	172.16.0.122	80	9	1491	5	623	4	868	52.126180	15.4955	321	448
89.85.51.194	1099	172.16.0.122	80	12	2658	6	1737	6	921	54.758281	0.4205	33 k	17 k
80.90.81.206	2134	172.16.0.122	80	22	9989	11	8762	11	1227	57.683232	12.8947	5436	761
217.119.117.212	3629	172.16.0.122	80	11	2869	6	1936	5	933	58.483801	3.6900	4197	2022
41.250.23.6	45825	172.16.0.122	80	12	3162	7	2195	5	967	72.448185	12.1828	1441	634
217.119.117.212	3641	172.16.0.122	80	9	1926	5	1059	4	867	73.904674	0.6696	12 k	10 k
200.121.1.131	10594	172.16.0.122	80	515	385 k	261	371 k	254	14 k	93.841385	28.8779	102 k	3960
200.72.43.226	62317	172.16.0.122	80	546	429 k	273	409 k	273	20 k	99.866905	29.3974	111 k	5473

Figure 75. The "Conversations" window

Source: Wireshark User's Guide

Useful Wireshark Tips: Statistics

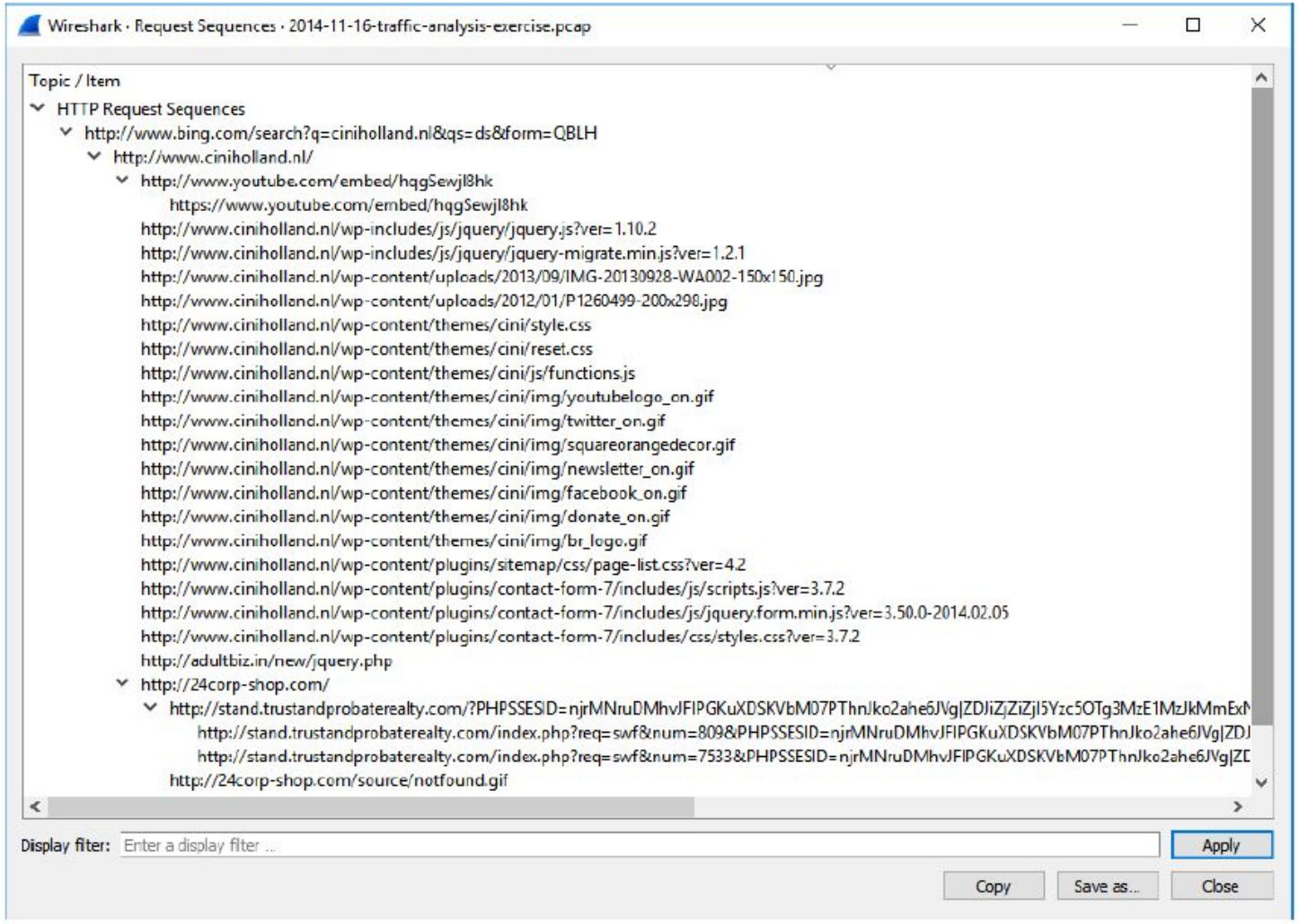
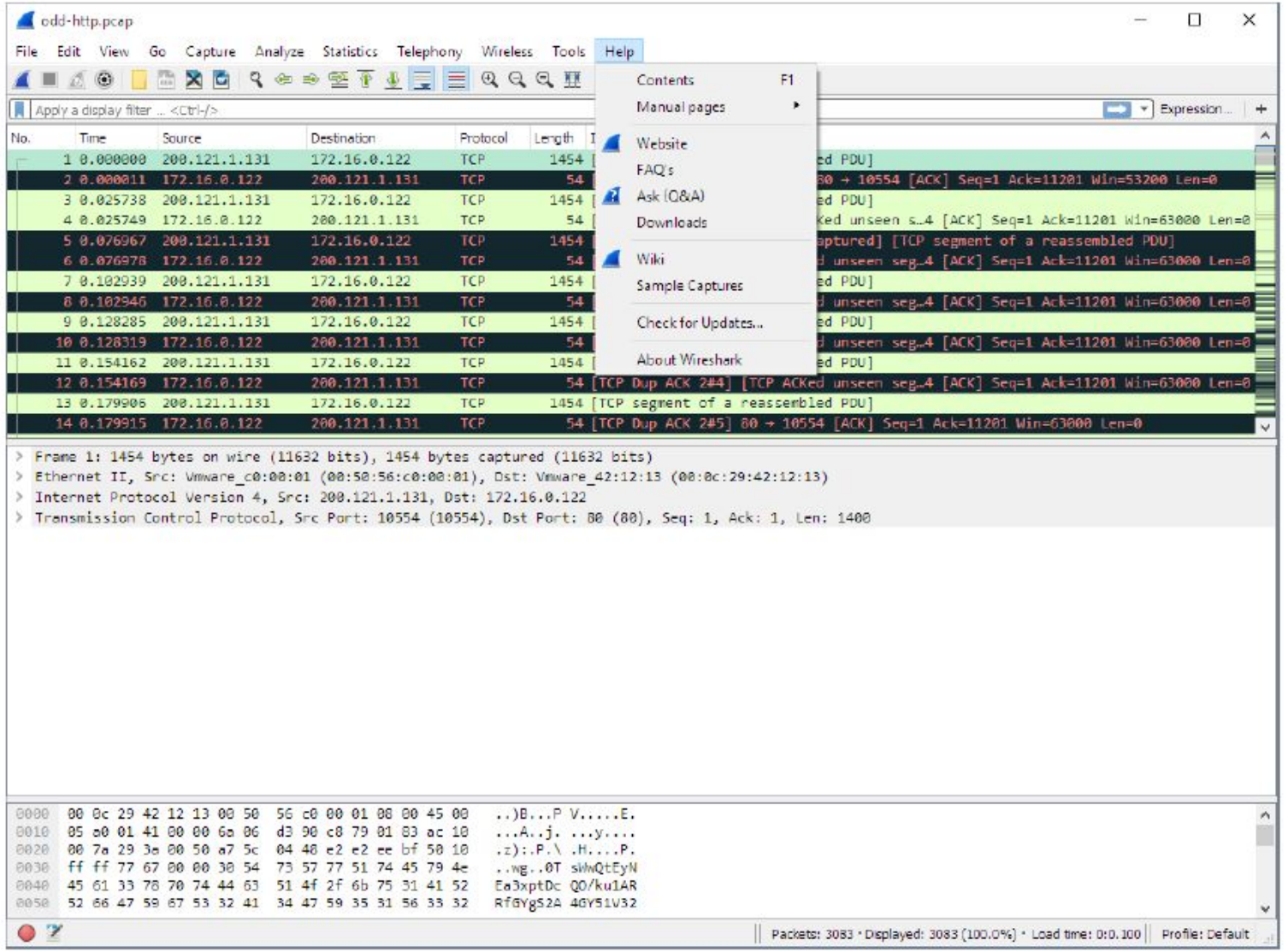


Figure 80. The "HTTP Request Sequences" window

Source: Wireshark User's Guide

Useful Wireshark Tips: Help



Source: Wireshark User's Guide

Some Wireshark "*Kungfu*"

Firewalls

Firewalls

- **Firewalls** are tools to control the flow of traffic going **between** networks:
  - Sitting at border between networks
  - Looking at services, addresses, etc. of traffic
  - Deciding whether a packet should be **allowed**, **dropped**, or **logged** (other actions are possible)
- General **principles**:
  - Don't open a **private service** to public unless it is necessary (i.e. *default-deny*)
  - Allow internal hosts to access a **public service** unless it is undesirable (i.e. *default-allow*)

Review: Types of Firewall

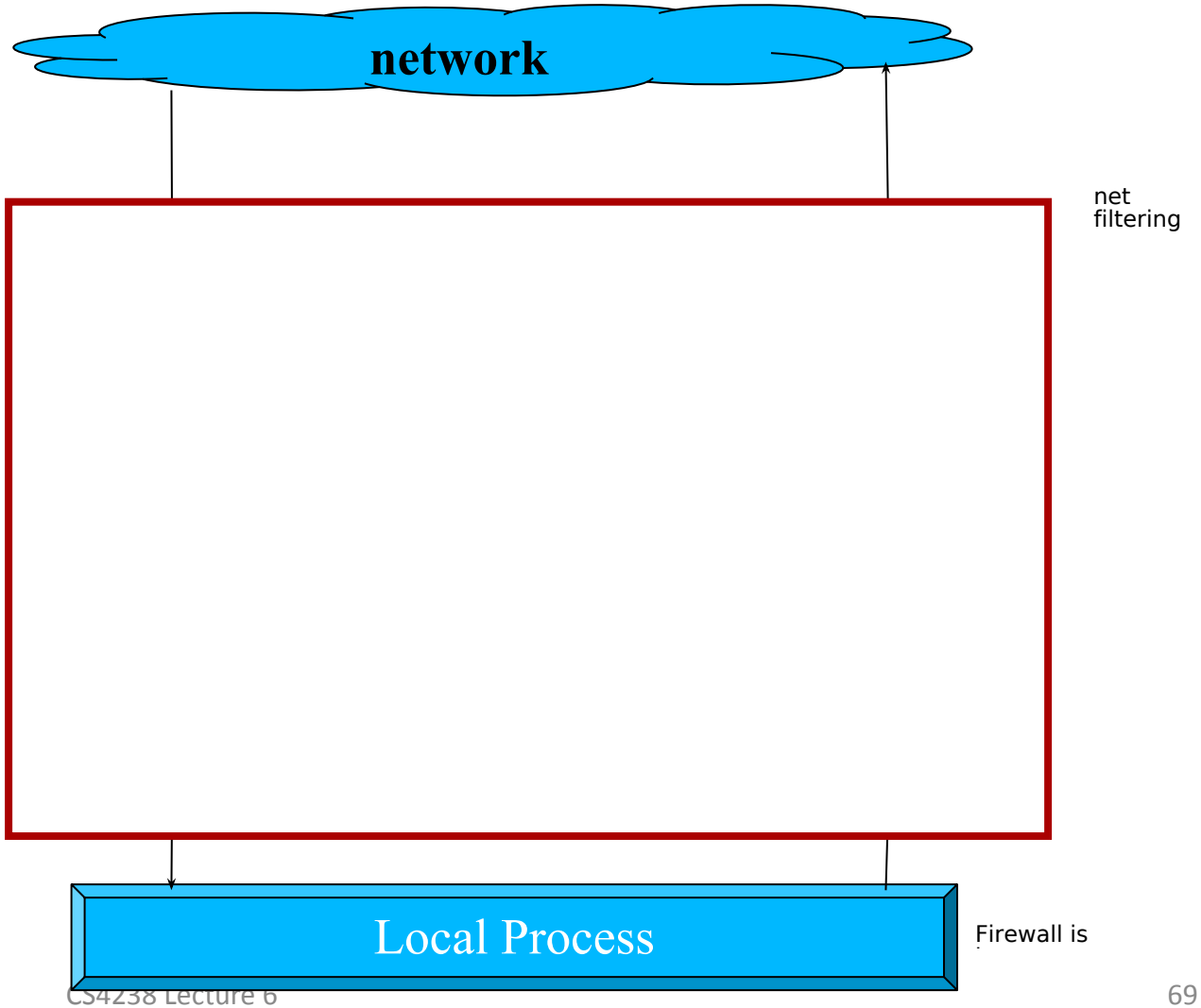
- **Traditional packet filters**:
  - Applying rules to packets in/out of firewall
  - Based on information in packet header
- **Stateful packet filters (SPFs)**:
  - Maintaining a state table of all active connections
  - Filtering packets based on connection states
- **Proxy-based firewalls:**
  - Understanding application logic
  - Acting as a relay of application-level traffic

Linux Firewalls: Netfilter Framework

- **`Netfilter`** is the packet-filtering framework in Linux kernel
- Supersedes `ipchains` (Linux kernel 2.2.x) and `ipfwadm` (Linux kernel 2.0.x)
- What can `netfilter/iptables` do?
  - Build Internet firewalls using stateless and stateful **packet filtering**
  - Use **NAT** and **masquerading**
  - Aid the tc and iproute2 systems to build sophisticated QoS and policy routers
  - Do further packet manipulation (***mangling***) like altering the TOS/DSCP/ECN bits of the IP header

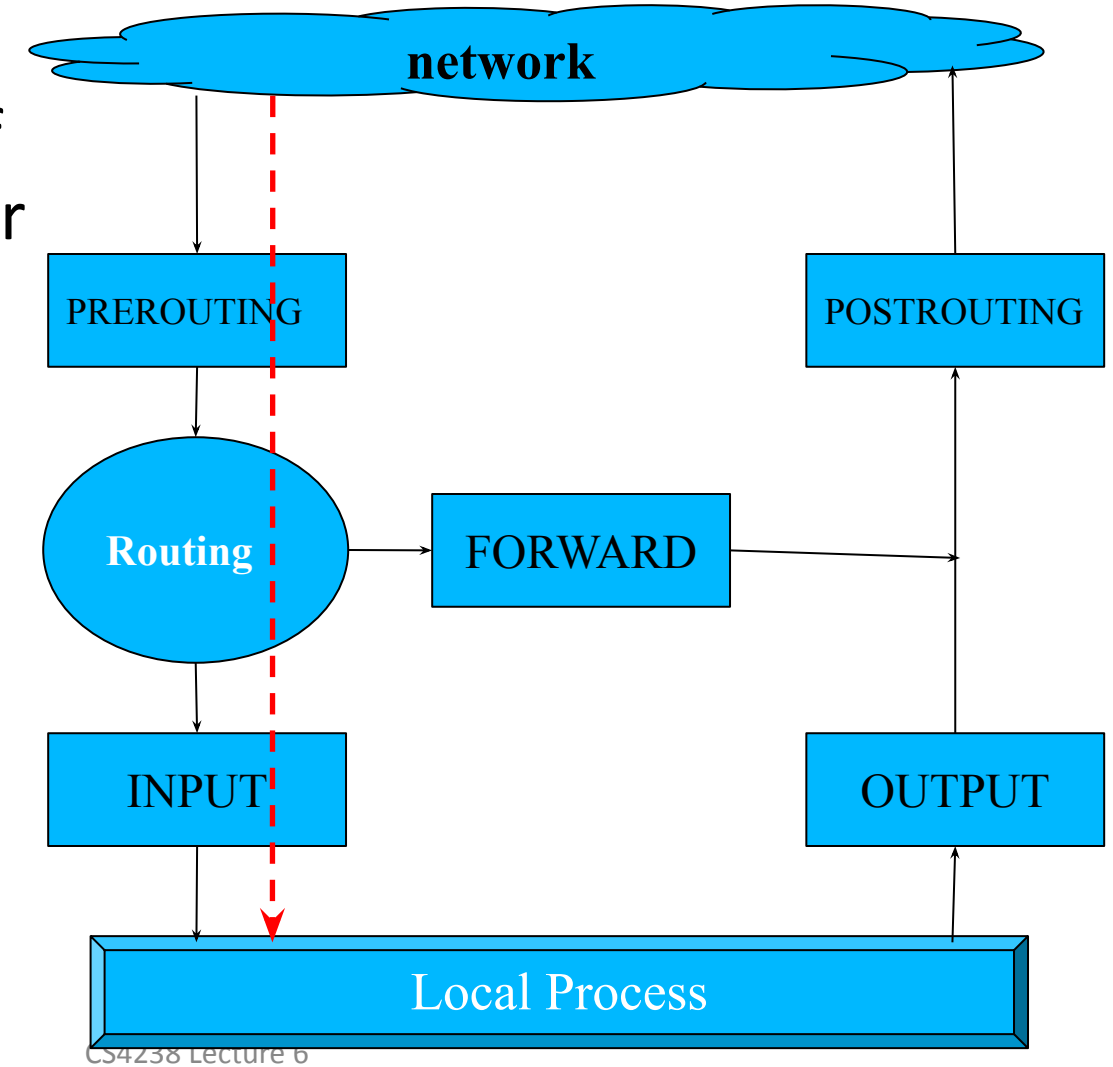
Linux Firewalls: Netfilter Framework

- Five ***hooks/chains*** **(filtering points)** to decide the fate of a packet:
  - Prerouting
  - Postrouting
  - Forward
  - Input
  - Output



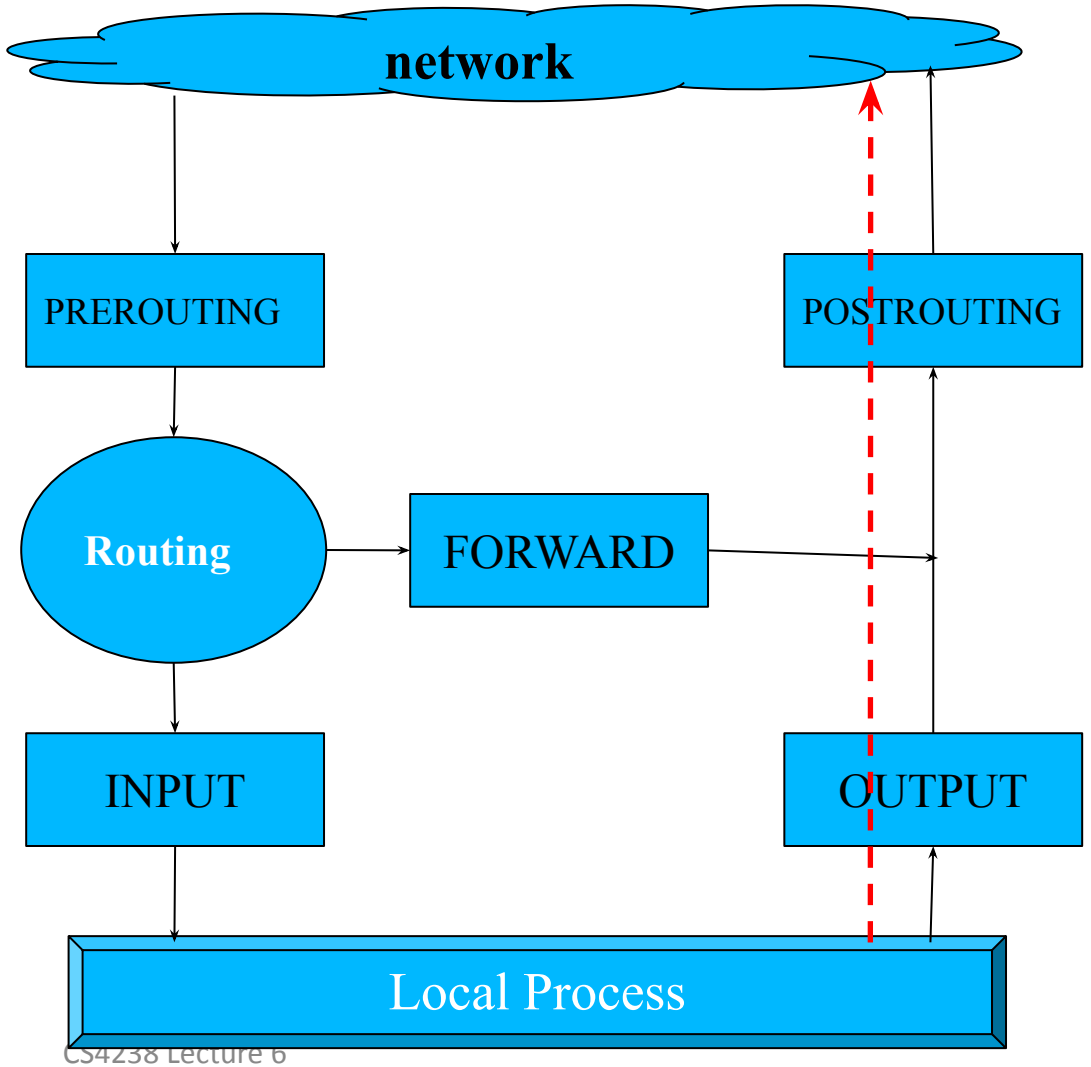
Path of Packets: Receiving Packets

- Packets *received* by **local processes** of the firewall computer pass through two hooks:
  - Prerouting
  - Input



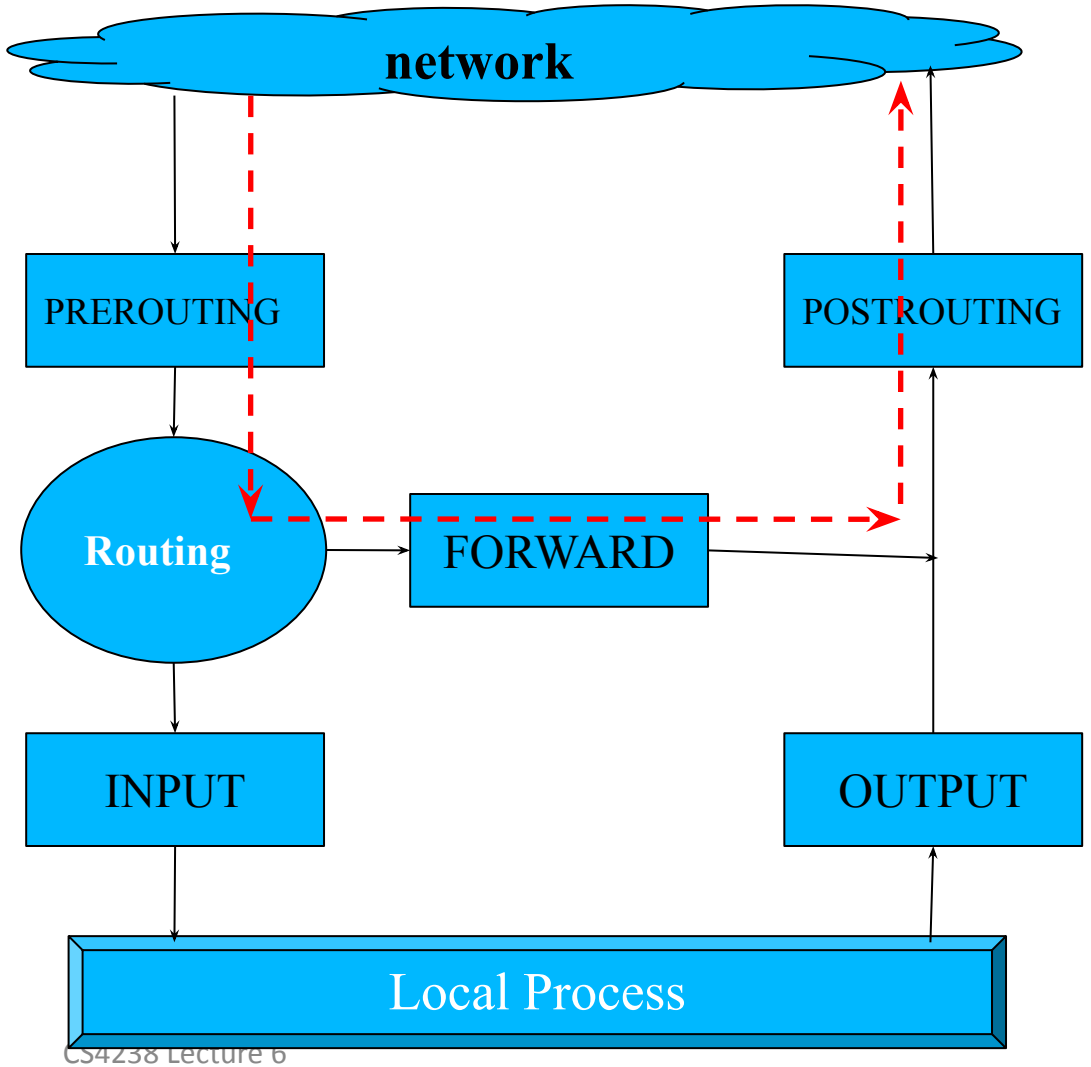
Path of Packets: Sending Packets

- Packets *sent out* by **local processes** of the firewall computer pass through two hooks:
  - Output
  - Postrouting



Path of Packets: Routing Packets

- When *the firewall computer* works as a **router**, routed packets pass through three hooks:
  - Prerouting
  - Forward
  - Postrouting



Tables in Netfilter

- Three **main tables** for 3 different functionalities:
  - `filter` (the default if no `-t` option is passed): for **packet filtering**
  - `nat`: for **network address translation**
  - `mangle`: for packet content **alteration**
- Additional tables:
  - `raw`: for configuring exemptions from connection tracking in combination with the NOTRACK target
  - `security`: for Mandatory Access Control (MAC) networking rules

See also: http://ipset.netfilter.org/iptables.man.html

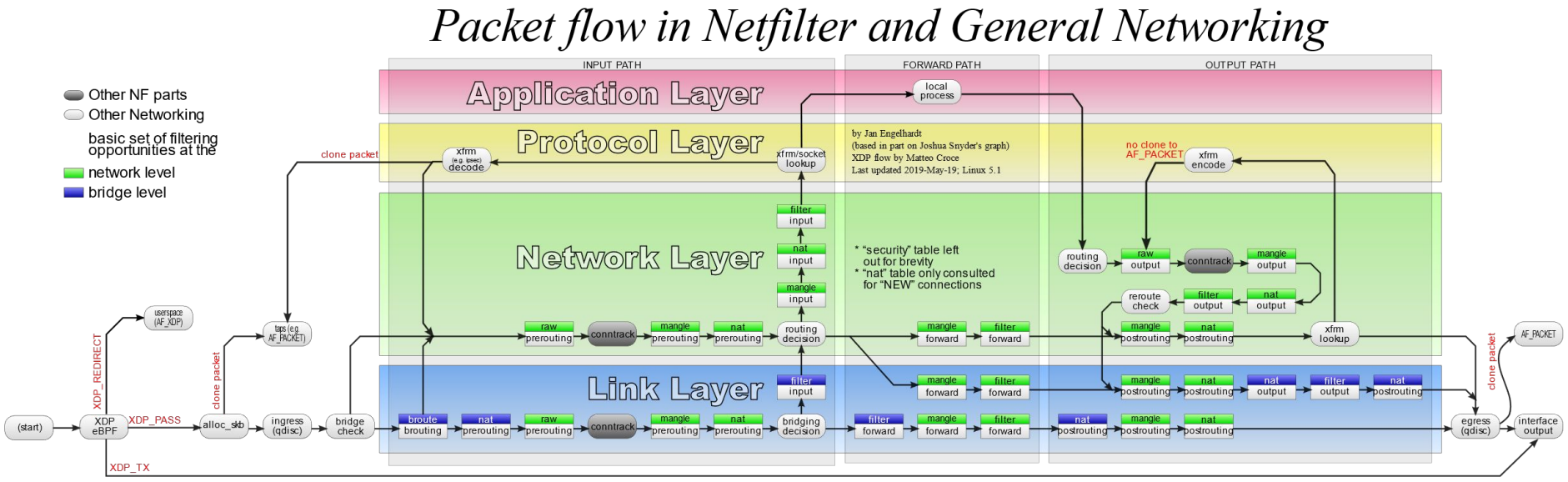
Tables and Hooks/Chains in Netfilter

- Tables and applicable hooks/chains:

Filtering point	Table		
	filter	nat	mangle
INPUT	✗		✗
FORWARD	✗		✗
OUTPUT	✗	✗	✗
PREROUTING		✗	✗
POSTROUTING		✗	✗

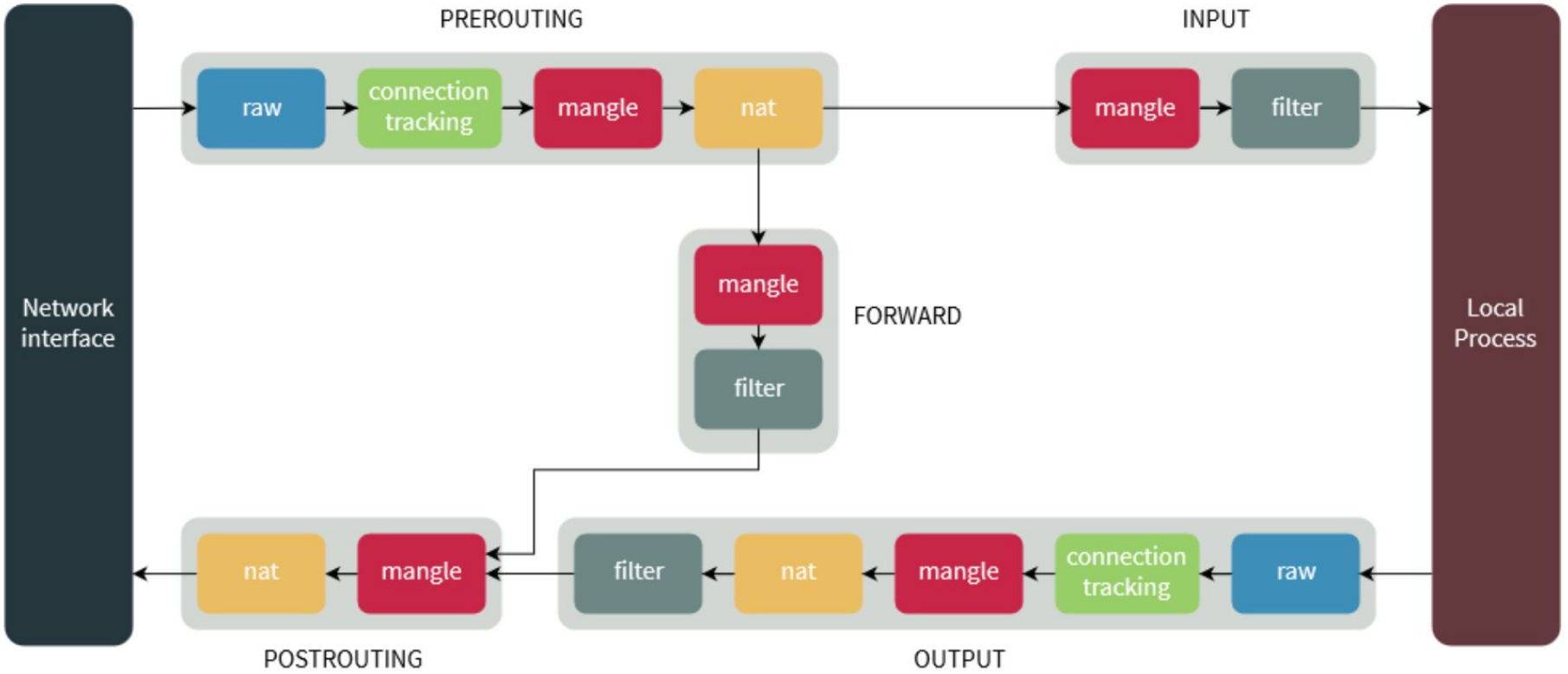
Source: http://tips.itliveweb.com

Packet Flow in Netfilter



Source: Wikipedia

Packet Flow in Netfilter (Simpler)



Source:
https://www.booleanworld.com/depth-guide-iptables-linux-firewall/

Table's Components

- A ***table***:
  - Contains multiple (applicable) **chains**
- Each ***chain***:
  - Corresponds to a Netfilter hook
  - Has multiple **rules**
  - Additionally has a **default policy**, which can implement default-deny and default-allow policies
- Each ***rule***:
  - Has ***conditions***:
    - Implicit logical AND operator connecting all the conditions
  - Specifies ***action/target*** to jump when conditions apply

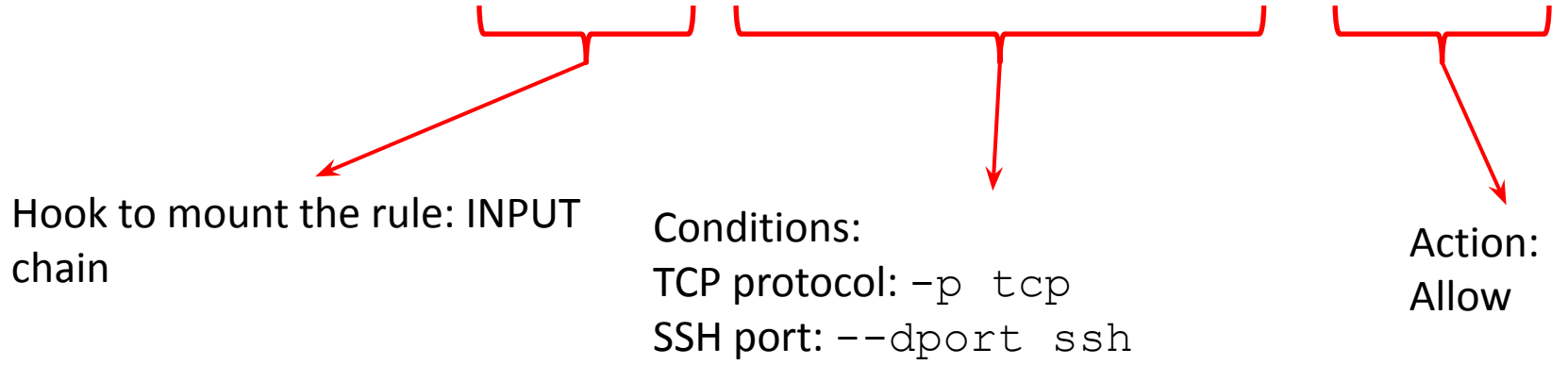
Firewall Rules

- Main components of a firewall rule:
  - The **table** to use
  - The **chain/hooks** to mount the rule:
    - Filtering packets for processes on the firewall computer: INPUT, OUTPUT
    - Filtering packets for other computers connected to the firewall: FORWARD
    - Network address translation: PREROUTING, POSTROUTING
  - **Rule conditions**:
    - IP address, ports, network interface, connection state, …
  - **Rule actions/targets**:
    - **Terminating**: accept, drop, reject (drop + send error packet), change packet information
    - **Non-terminating**: log

The `iptables` Utility

- The **Linux program** to maintain firewall rules in the `Netfilter` framework
- Example: allowing incoming `ssh` connections to this computer (using the default `filter` table)

```
$ sudo iptables -A INPUT -p tcp --dport ssh -j ACCEPT
```



The `iptables` Utility

- Some commonly-used iptables **flags**:
  - `-t` *<table>*: select **table** (useful for `nat`, `mangle` tables)
  - `-L` *<chain>*: **list** all rules of a chain
    (use `-n` to avoid long reverse DNS lookups)
  - `-A` *<chain>*: **append** a rule to the end of a chain
  - `-I` <*chain*> [*index*]: **insert** a rule to the specified index
  - `-D` <*chain*> [*index*]: **delete** a rule in a chain
  - `-R` <*chain*> [*index*]: **replace** a rule in a chain
  - `-F` *<chain>*: remove/**flush** all rules of a chain
  - `-P` <*chain*> <*target*>: set default **policy** for a chain
    (e.g. DROP, ACCEPT)

The `iptables` Utility (Cont)

- Some commonly-used flags for ***rule conditions***:
  - `-s:` **source** IP address
  - `-d:` **destination** IP address
  - `-p:` **protocol** (e.g. `tcp`, `udp`, `icmp`)
  - `--sport:` **source port** no (available with tcp/udp module)
  - `--dport:` **destination port** no (available with tcp/udp)
  - `-i:` **input** interface (e.g. `eth0`)
  - `-o:` **output** interface (e.g. `eth0`)
  - `-j:` action/**jump** (e.g. `ACCEPT, DROP, REJECT, LOG`)
  - `-m:` **match** (i.e. extension **module**) that tests for a specific property (*covered more later*)

Logging ICMP Example

```
iptables -t filter -A INPUT -p icmp --icmp-type
echo-request -j LOG --log-prefix="ICMPIN-REQ:"

iptables -t filter -A INPUT -p icmp --icmp-type
echo-reply -j LOG --log-prefix="ICMPIN-RPY:"

iptables -t filter -A OUTPUT -p icmp --icmp-type
echo-request -j LOG --log-prefix="ICMPOUT-REQ:"

iptables -t filter -A OUTPUT -p icmp --icmp-type
echo-reply -j LOG --log-prefix="ICMPOUT-RPY:"

iptables -t filter -A FORWARD -p icmp --icmp-type
echo-request -j LOG --log-prefix="ICMPFWD-REQ:"

iptables -t filter -A FORWARD -p icmp --icmp-type
echo-reply -j LOG --log-prefix="ICMPFWD-RPY:"
```

iptables-extensions

- ***iptables-extensions***: list of **extensions** in the standard iptables distribution

- How can iptables **match** an extension module?

  - **Explicit** module matching: using **`-m`** or **`--match`** option, e.g. **`-m state`**
  - **Implicit** module matching: if the **`-p`** was specified, and if and only if **an unknown option** is encountered, iptables will try to match a module of **the same name** as the protocol,
    e.g. `-p` **`icmp --icmp-type echo-request`**
    `-p` **`tcp --dport ssh`**

iptables-extensions

- Example: **`state`** extension module
  - A subset of the **`conntrack`** module
  - Allows access to the connection-tracking state for this packet
  - Option: `--state` *<state>*
    (`--ctstate` in **`conntrack`**)
  - *<state>* is a comma separated list of the connection states to match: `INVALID`, `ESTABLISHED`, `NEW`, `RELATED`, `UNTRACKED`
- For other extension modules, see:
  http://ipset.netfilter.org/iptables-extensions.man.html

Stateful Filtering Example

- Goal: allowing all **outgoing HTTP** connections and **incoming** response to **these connections**, and *nothing else* is allowed
  - On the **OUTPUT** chain:

```
iptables –A OUTPUT –p tcp --dport 80 –j ACCEPT

iptables –A OUTPUT –j DROP
```

  - On the INPUT chain:

```
iptables –A INPUT
  -m state --state ESTABLISHED,RELATED
  -j ACCEPT

iptables –A INPUT –j DROP
```

This condition matches packets in established and related connections

Automatically Loading Firewall Rules

- `/etc/rc.local` is a **start-up script**, which will be executed on every system start-up
- Put the `iptables` commands that set firewall rules into this file
- Alternatively, **dump** the current firewall configuration into a file using:

```
iptables-save > <file>
```

and use `iptables-load <` *<file>* to load the saved configuration in the start-up script

IP Address Sharing using NAT

Sharing an IP Address

- If the workstation's IP address is *a* **public address**, which is allocated by Internet Assigned Numbers Authority (IANA), then we are **done** with the configuration:
  - But IANA has (basically) run out of IPv4 addresses
  - Computers on a private network, such as ours, often need to share *one IP address* (or use IPv6, but >10% adoption as of Jan 2016)
- Technique: use ***Network Address Translation (NAT)***
  - Assign **private IP addresses**, e.g. 192.168.60.1, to workstations
  - Modifies IP address in workstations' packets **before leaving** the organization's gateway
  - A private IP address is not directly accessible from the Internet (but it does not function as a firewall, and only hides private network in some situations)

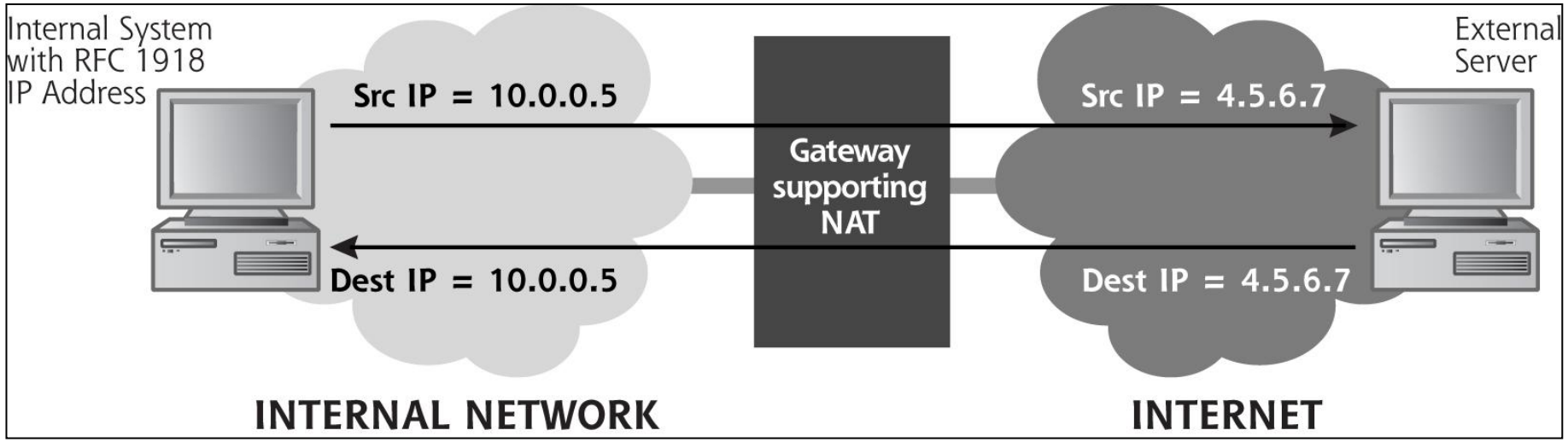
Private IPv4 Addresses (Review)

- **Private address** ranges:
  - 10.0.0.0 – 10.255.255.255
  - 172.16.0.0 – 172.31.255.255
  - 192.168.0.0 – 192.168.255.255
- Not routable on the public Internet
- Used together with NAT or proxy

Network Address Translation

- ***Before forwarding*** the workstation's request to Internet, the gateway changes the packet's source address (**private address 10.0.0.5**) to the **gateway's public address (4.5.6.7)**

- ***After receiving*** the response of this packet, the gateway changes the **response packet's destination address (4.5.6.7)** back to the **workstation's address (10.0.0.5)**, and forward it to the workstation.



Network Address Translation in Linux

- In Linux, network address translation is supported by its **firewall**
- Use the following command to turn on **IP sharing** (suppose that the router uses the interface `eth0` to connect to the Internet):
  - `$ sudo iptables` **`-t nat -A POSTROUTING -o eth0 -j MASQUERADE`**

Summary

- Linux network configuration:
  - Configuring a router
- Network traffic analysis
- Linux firewall:
  - Kernel-land `Netfilter` framework
  - User-land `iptables` utility
  - Packet filtering and NAT
- References: Chapter 2 (Textbook)

Additional References

- Cheswick, Bellovin, Rubin, "*Firewalls and Internet Security, Repelling the Wily Hacker*", 2nd Edition
- Noonan and Dubrawsky, "*Firewall Fundamentals*", Cisco Press
- Netfilter: http://www.netfilter.org/
- Ubuntu iptables howto: https://help.ubuntu.com/community/IptablesHowTo
- The Beginner's Guide to iptables, the Linux Firewall: https://www.howtogeek.com/177621/the-beginners-guide-to-iptables-the-linux-firewall/
- Iptables Essentials: Common Firewall Rules and Commands, https://www.digitalocean.com/community/tutorials/iptables-essentials-common-firewall-rules-and-commands
- Oskar Andreasson, Iptables Tutorial 1.2.2 (a long/comprehensive document) , https://www.frozentux.net/documents/iptables-tutorial/