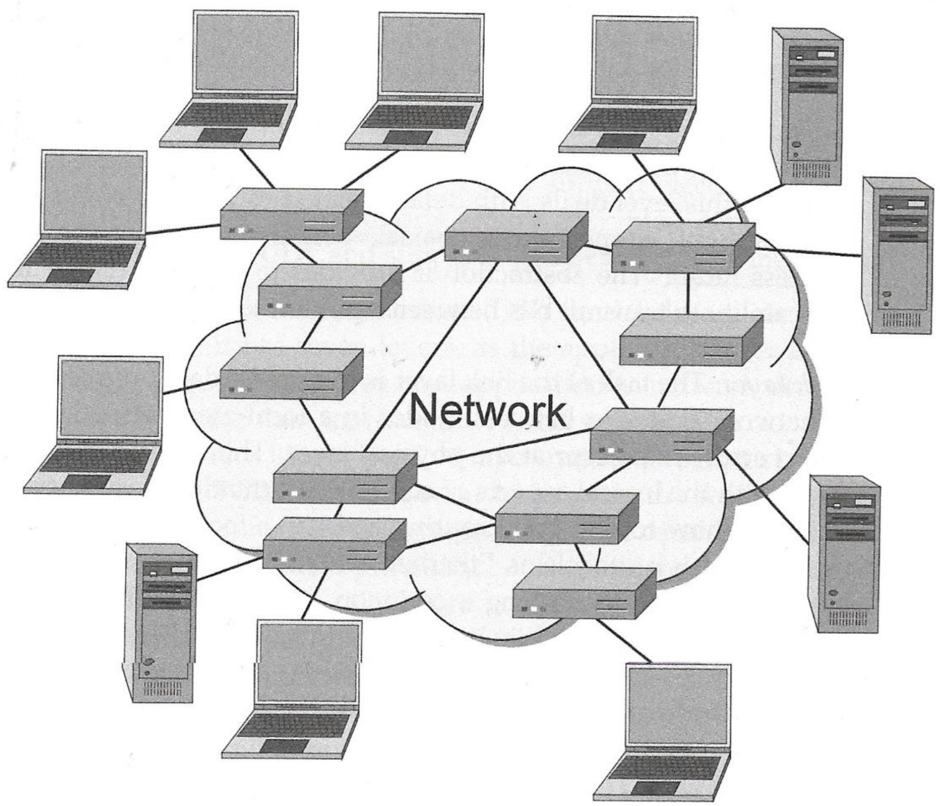
TK2100: Informasjonssikkerhet Lesson 05: Network 1

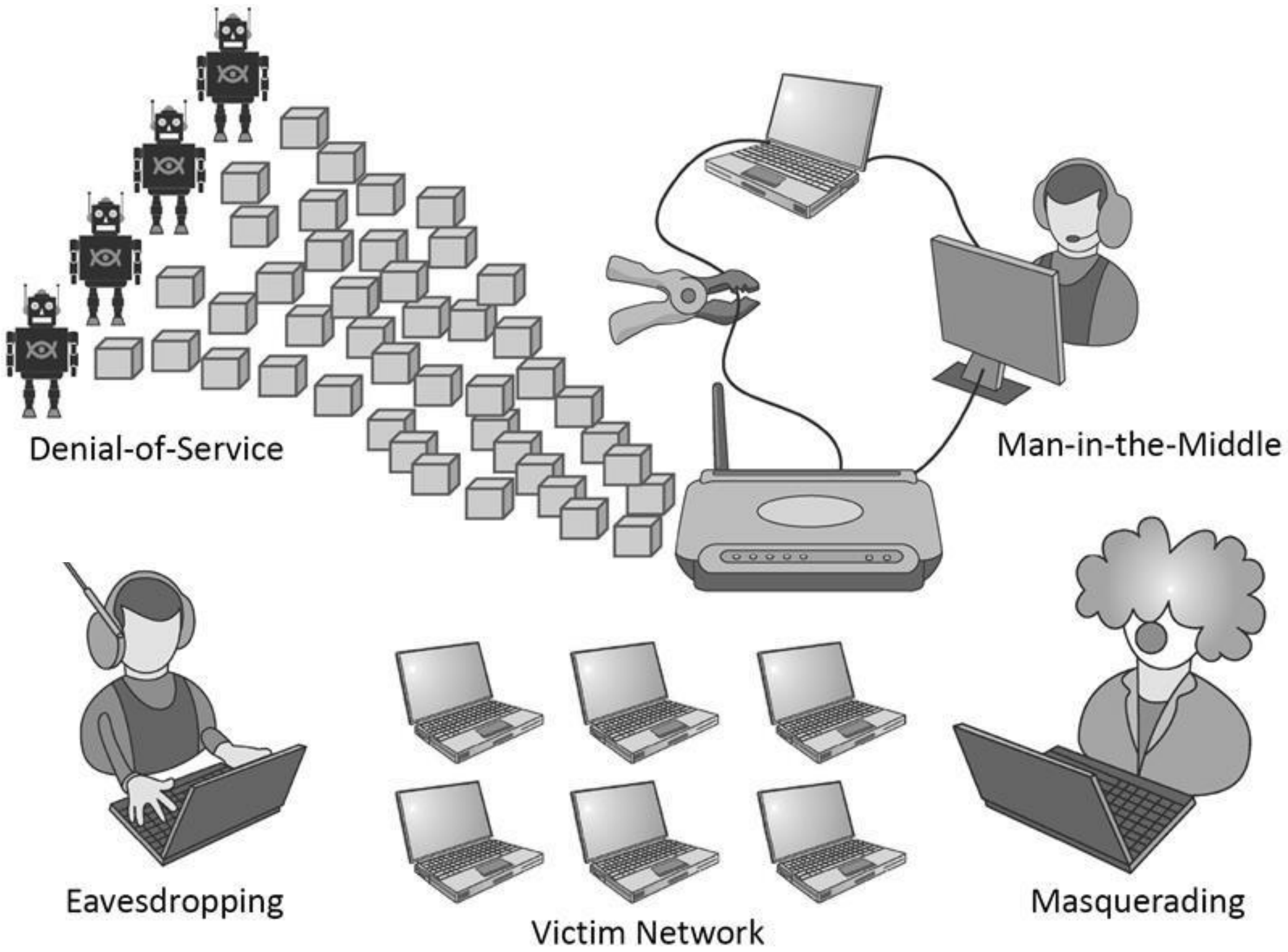
Dr. Andrea Arcuri
Westerdals Oslo ACT
University of Luxembourg

Goals

- Understand how computers can communicate over a network
  - eg, using browser to visit web pages
- Understand how an attacker can comprise communications over a network



Many Different Kinds of Attacks

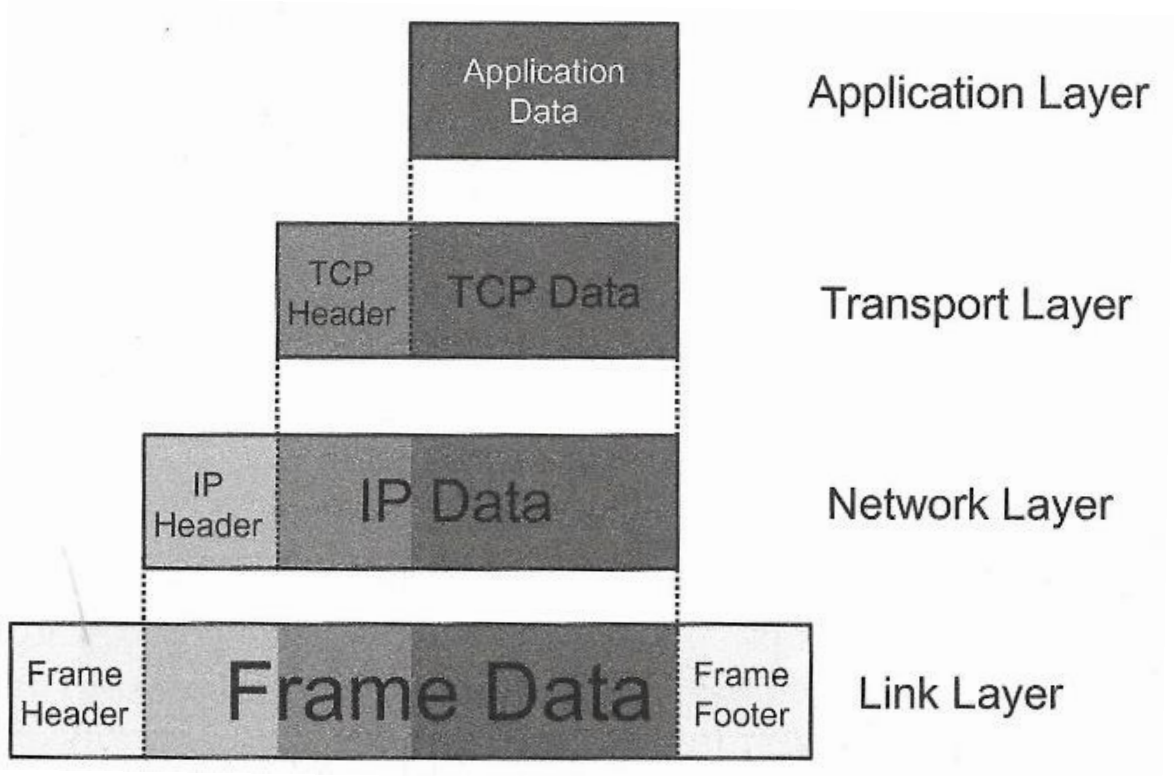


Internet Protocol Layers

- Communications over internet are very complex
- The sending/receiving of messages is abstracted in *layers*
- When programming at a certain level, do not need to know of the layers directly underneath
    - eg, when making a request to retrieve an HTML page using HTTP, do not need to have to deal with low level details of sending of packets over a fiber optic vs WiFi
- Not only this makes coding easier, but also enable the use of variegated protocols/hardware in the lower layers

5 Main Layers

- **Physical Layer** is the one in which the 0/1 bits of the data are sent
- 0/1 are sent in groups, ie in *packets*
- A packet contains a *header* and a *payload* (sometimes also a *footer*)
- The payload in one level is the entire packet (header+payload) of its above layer



Overview: Physical Layer

- Dealing with the low level details of the actual sending of 0/1 bits between two nodes
- Handling of connections on *copper wires*, *coaxial cables*, *optical fiber cables* or *wireless radio*
- *Best Effort:* cannot guarantee that bits arrive

Overview: Link Layer

- Dealing with communications between 2 nodes on same *local area network (LAN)*
  - eg, different machines connected to same WiFi router/switch
- Need to find good paths connecting the 2 nodes
- Protocols like *Ethernet*
- MAC addresses to identify nodes on the local network

Overview: Network Layer

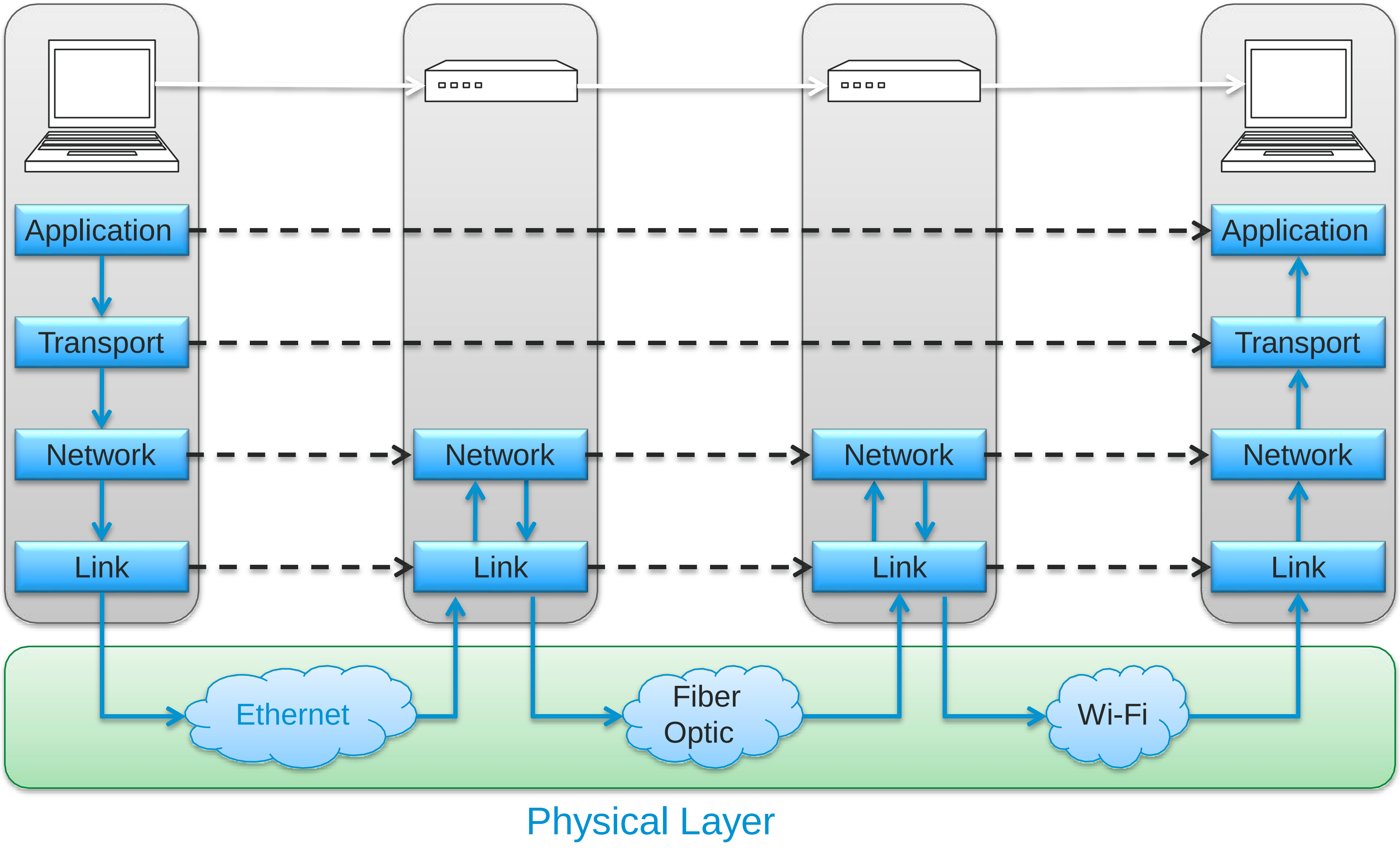
- Moving of packets between nodes on wide area networks, like *internet*
- Nodes identified with IP addresses, eg 127.0.0.1
- **IP** (Internet Protocol)
- *Best Effort*: cannot guarantee a packet is going be delivered without problems

Overview: Transport Layer

- Establish connection between different processes/applications on different nodes
- An application address is identified with an IP address **AND** a *port* number
- Protocols like TCP and UDP
- Can guarantee delivery (or tells you when that does not happen) of packets (eg, by automatically resending the lost ones)

Overview: Application Layer

- Specific protocols for specific kinds of applications
- Eg, HTTP for web pages, and SMTP for emails
- By using a standard protocol, when you deploy your web applications to a conforming web server, then you know that all major browsers will be able to interact with them


Application
Transport
Network
Link
Network
Link
Network
Link
Application
Transport
Network
Link
Ethernet
Fiber
Optic
Wi-Fi
Physical Layer

Network Security Issues: CIA

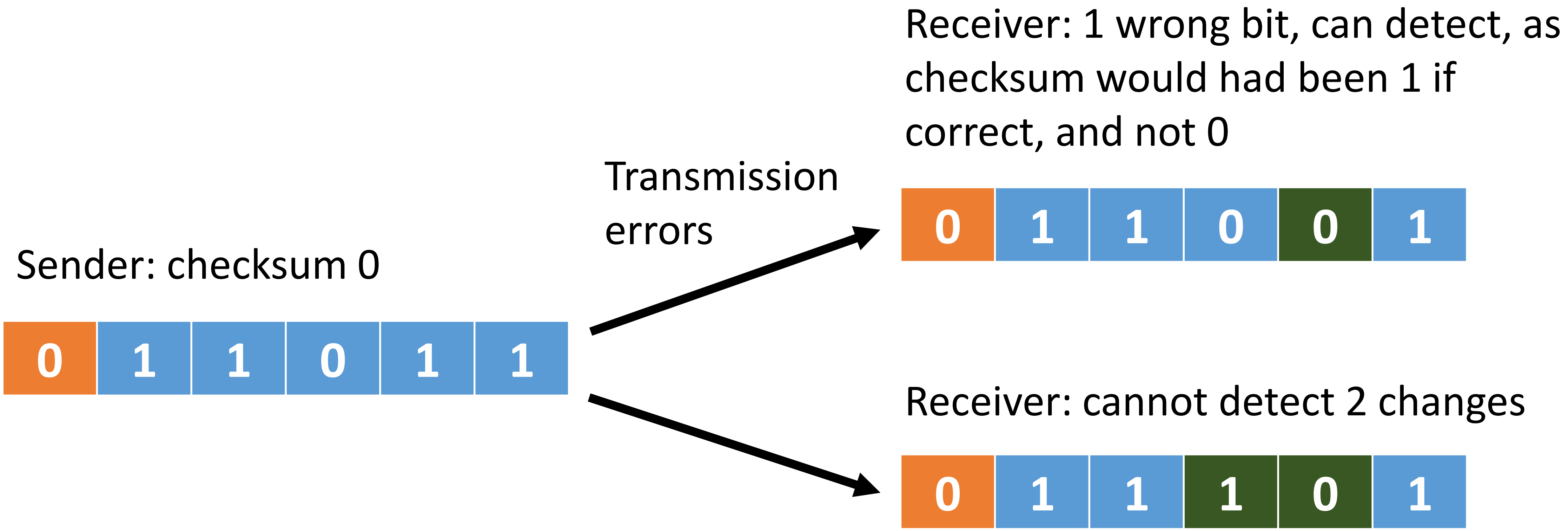
- *Confidentiality*: no requirement in any of the layers. If you need confidentiality, you need protocols that support encryption, like *HTTPS* and *IPsec*
- *Integrity*: support for basic *checksums*. Can detect small changes, but not cryptographically secure
- *Availability:* internet is designed to tolerate failure of routers and hosts, but cannot make strong guarantees, especially in the case of DDoS

Checksum

- Technique to check if data has been altered
- This could happen due to errors at physical layer, or if hacker altering messages
- Simple checksum: add one extra bit, checking if in the data the number of 1s is even or odd
- Receiver of message will check if checksum is correct or not
- In this latter case, message has been altered

Checksum Example

- A simple checksum (0 if even number of 1s, 1 otherwise) will only detect differences of 1 single bit change
- Tradeoff: message is now 1 bit longer



Network Security Issues: AAA

- *Assurance*: usually packets are allowed to be sent between 2 nodes. Preventing it has to be done explicitly, eg a firewall in a local network
- *Authenticity*: on internet protocol, no concept of *users*, as communications are between computers, not people. If needed authenticity, have to handle it at application layer
- *Anonymity*: built-in anonymity, as no concept of users. Agencies tracking users need to correlate with other source of info, like credit cards and mobile subscriptions

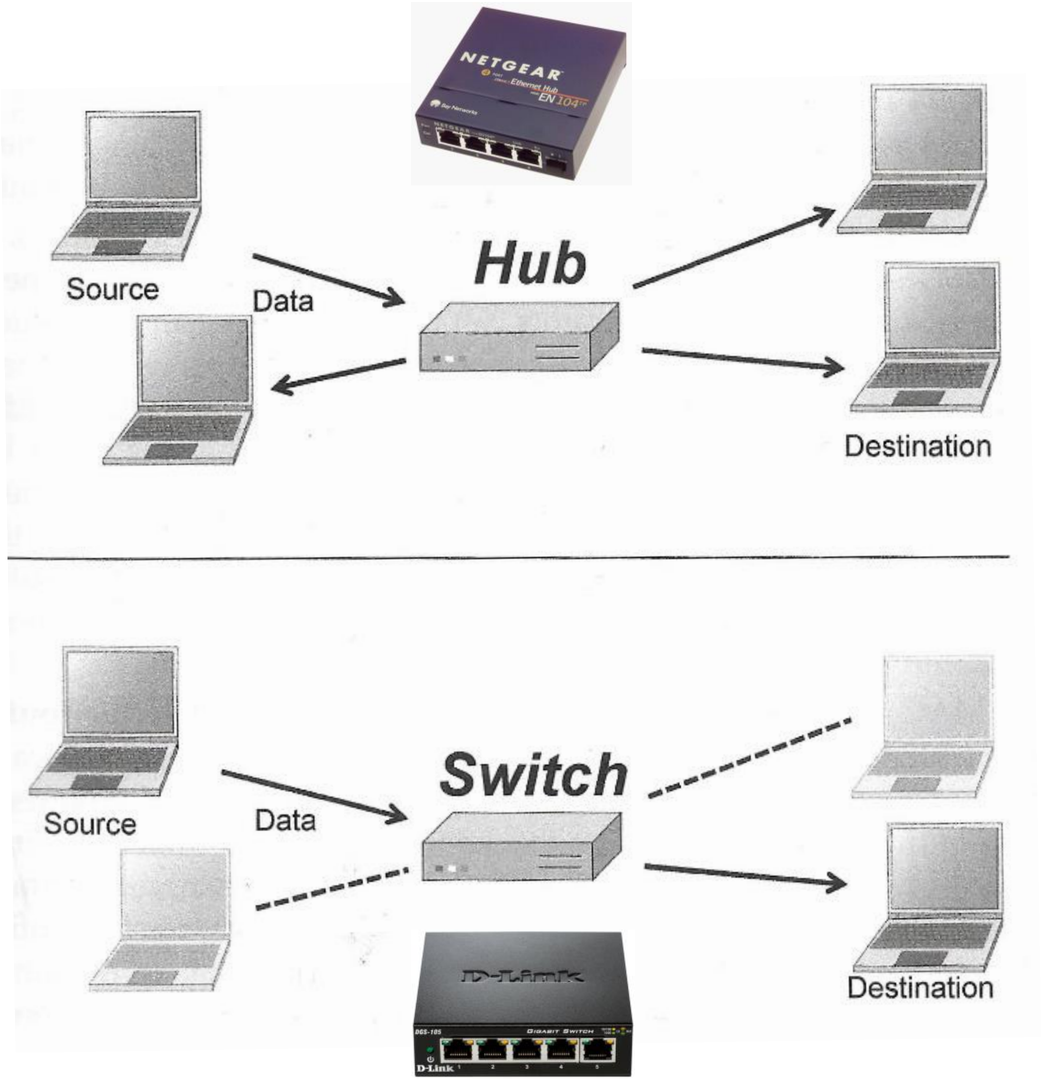
Link Layer

Ethernet

- A Link Layer protocol, standardized as IEEE 802.3
- Communications on cables
- Physical transmission: frames (ie messages) sent as electrical impulses
- If collisions (two machine sending impulses at same time), they will resend after a random amount of time

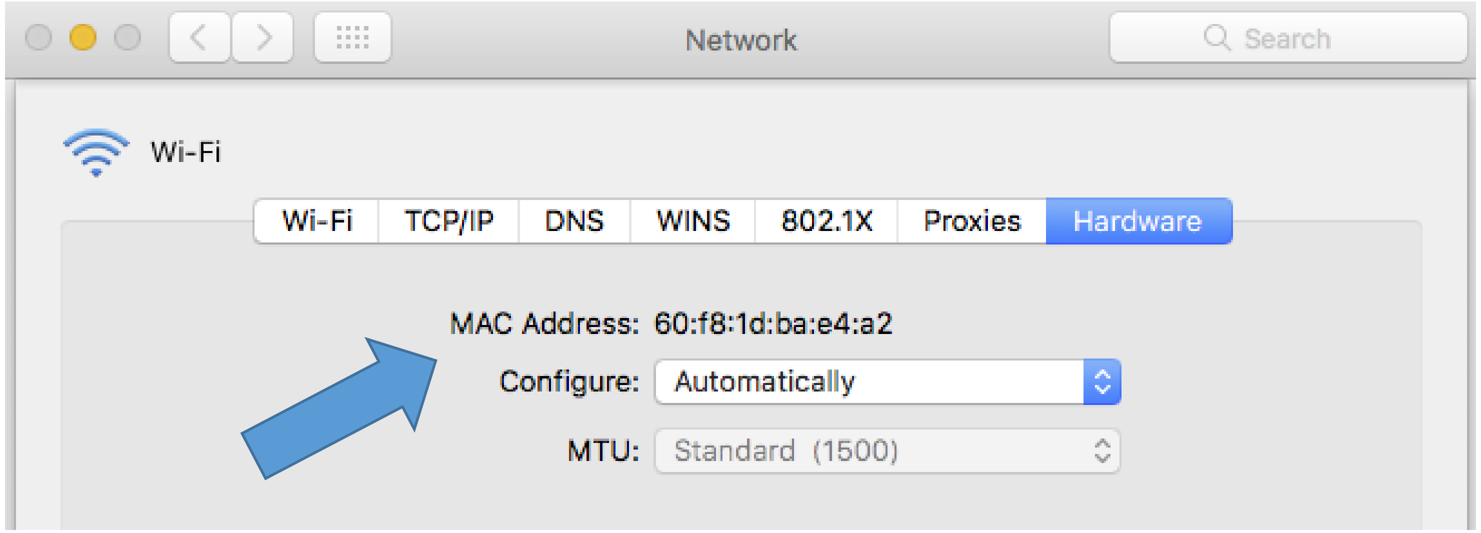
Hubs and Switches

- Hardware components
- *Hub*: does *broadcast* of all incoming electrical impulses, ie the frame packets
- *Switch*: start with broadcast, but, once learned the MAC addresses, only send frames to the right route
- Nowadays, *Switches* are used and not so many *Hubs,* as more efficient and not so expensive anymore



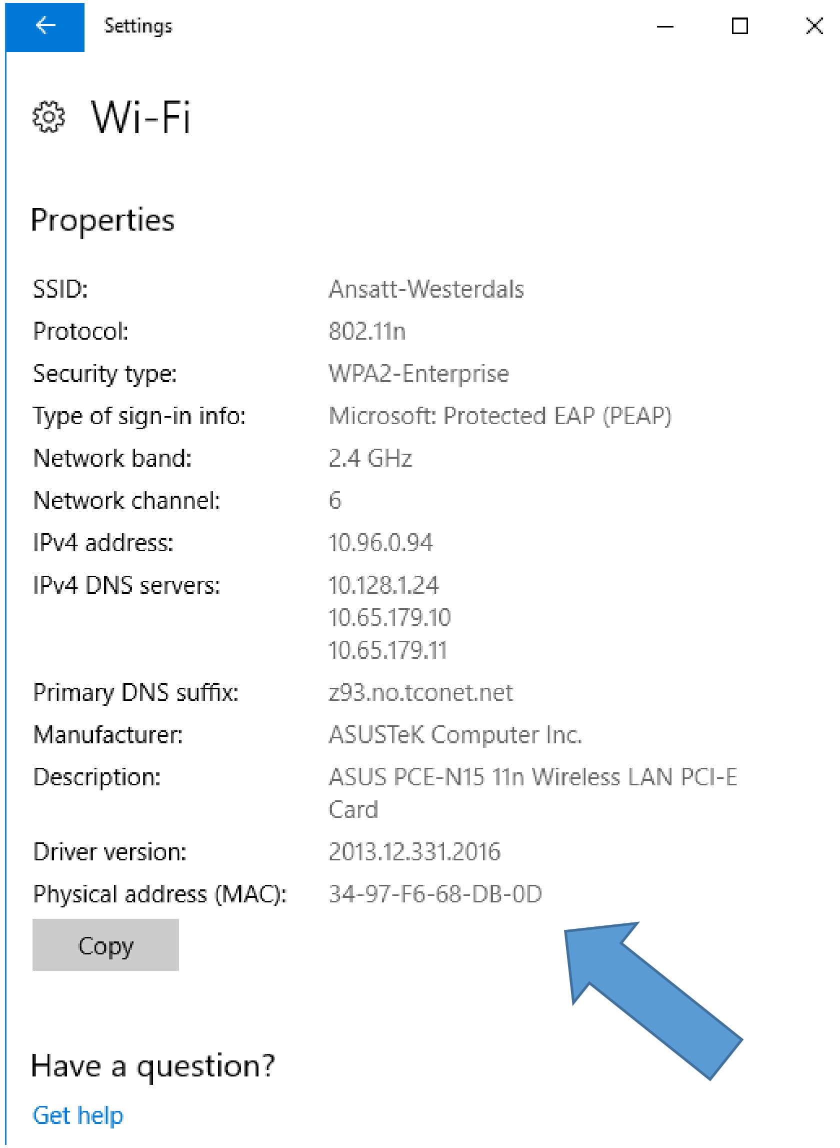
Media Access Control (MAC) Addresses

- *48-bit identifier*, usually represented with 6-pairs of hexadecimal digits, eg. 00:16:B7:29:E4:7D
  - recall, hexadecimal is 0-9-A-F, so 16 values, using 4 bits
  - 6 pairs is 12 digits, so 6 * 2 * 4 = 48 bits / 6 bytes
- Associated with physical device
- First 24 bits represent the manufacturer (eg Asus) and sometime the specific model
- Remaining 24 bits are for identifier of the actual device instance
- Note: $2^{24} = 16$ millions. Even if manufacturer starts to reuse MAC addresses, extremely unlikely to have two different devices on same local network with same MAC address


Network
Search
Wi-Fi
Wi-Fi TCP/IP DNS WINS 802.1X Proxies Hardware
MAC Address: 60:f8:1d:ba:e4:a2
Configure: Automatically
MTU: Standard (1500)

Mac

Windows


Settings
Wi-Fi
Properties
SSID: Ansatt-Westerdals
Protocol: 802.11n
Security type: WPA2-Enterprise
Type of sign-in info: Microsoft: Protected EAP (PEAP)
Network band: 2.4 GHz
Network channel: 6
IPv4 address: 10.96.0.94
IPv4 DNS servers: 10.128.1.24
10.65.179.10
10.65.179.11
Primary DNS suffix: z93.no.tconet.net
Manufacturer: ASUSTeK Computer Inc.
Description: ASUS PCE-N15 11n Wireless LAN PCI-E Card
Driver version: 2013.12.331.2016
Physical address (MAC): 34-97-F6-68-DB-0D
Copy
Have a question?
Get help

Ethernet Frame Format

- *Header*: eg, source and destination MAC addresses
- *Payload*: the actual data, which usually would be a Network Layer packet
- *Footer*: eg, a checksum

Bits	Field	
0 to 55	Preamble (7 bytes)	
56 to 63	Start-of-Frame delimiter (1 byte)	
64 to 111	MAC destination (6 bytes)	Header
112 to 159	MAC source (6 bytes)	
160 to 175	Ethertype/Length (2 bytes)	
176 to 543+	Payload (46-1500 bytes)	Payload
543+ to 575+	CRC-32 checksum (4 bytes)	Footer
575+ to 671+	Interframe gap (12 bytes)	

Address Resolution Protocol (ARP)

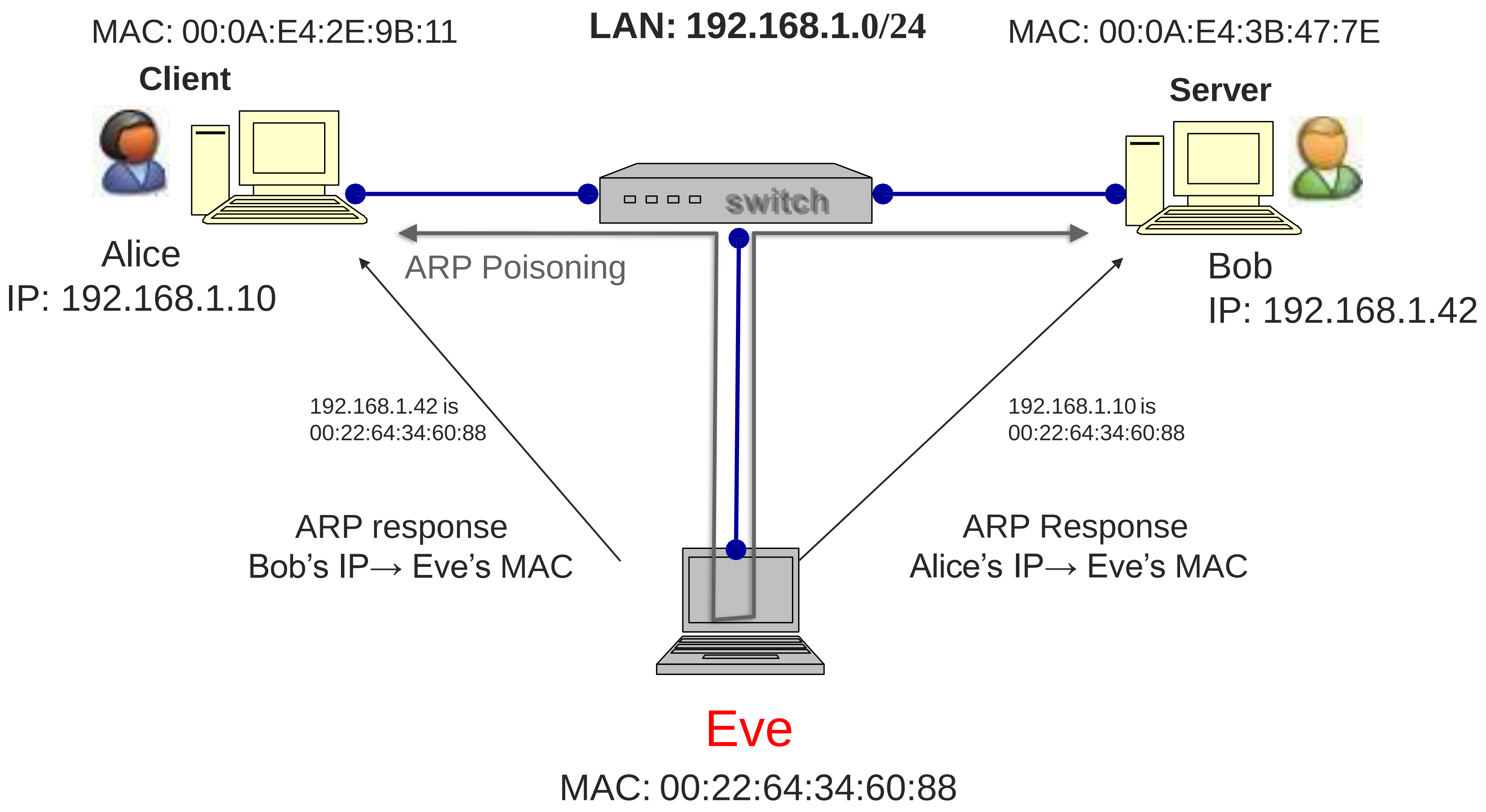
- The Link Layer works with MAC addresses
- But messages in the higher layers work with IP addresses
- ARP is a protocol to resolve an actual MAC address from the specified IP address

ARP Broadcast

- Assume ARP is used to resolve the MAC address for the IP 192.168.1.105
- ARP will send a *broadcast* on the LAN looking like: *“Who has the IP address 192.168.1.105?”*
- All machines on the LAN receives the broadcast, and the one with that IP address will *respond*
- ARP Cache: the mapping can be cached. So once a node finds out who has that IP, does not need to re-ask again

ARP Spoofing

- Any machine could reply to the ARP broadcast
- A malicious machine could lie and tell it is the one representing the requested IP address
- *Man-in-the-Middle* attack: trick Alice and Bob to send their messages to Eve



LAN: 192.168.1.0/24
MAC: 00:0A:E4:2E:9B:11
Client
Alice
IP: 192.168.1.10
MAC: 00:0A:E4:3B:47:7E
Server
Bob
IP: 192.168.1.42
switch
ARP Poisoning
192.168.1.42 is
00:22:64:34:60:88
192.168.1.10 is
00:22:64:34:60:88
ARP response
Bob’s IP→ Eve’s MAC
ARP Response
Alice’s IP→ Eve’s MAC
Eve
MAC: 00:22:64:34:60:88

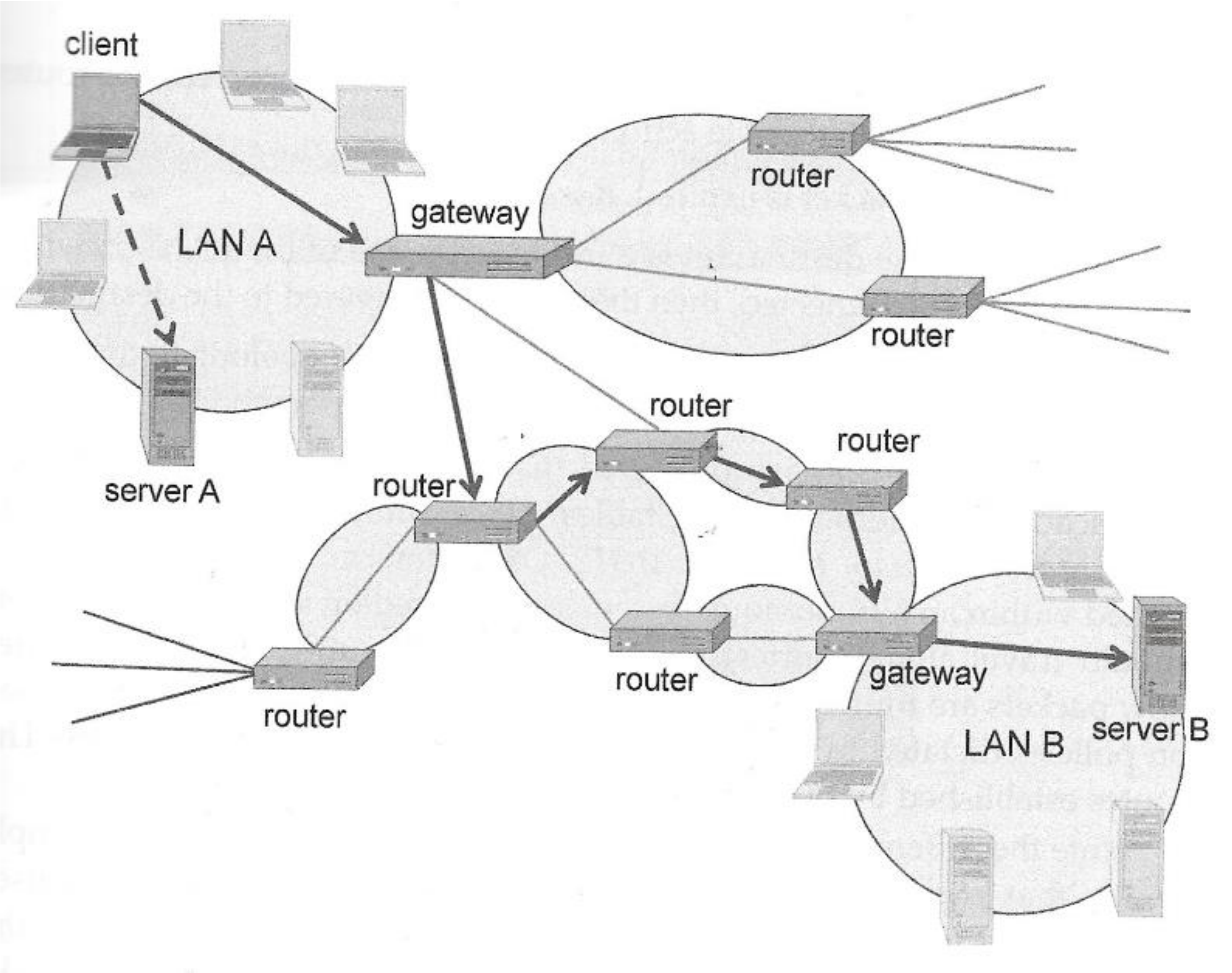
ARP Spoofing Prevention

- Different prevention mechanisms
- Simplest: check if more than one different machine answer to a ARP request
  - cannot know which is the true one, but at least can know if someone is messing up with the network
- Static ARP tables: manual registration of MAC addresses to static, fixed IP addresses on a local network

Network Layer

Internet Protocol (IP)

- A network protocol to move packets between 2 nodes on the internet
- Internet can be seen as a collection of independent LANs connected via routers and gateways
- Link Layer works on a LAN, but here we need communications with nodes on different LANs



IP Addresses

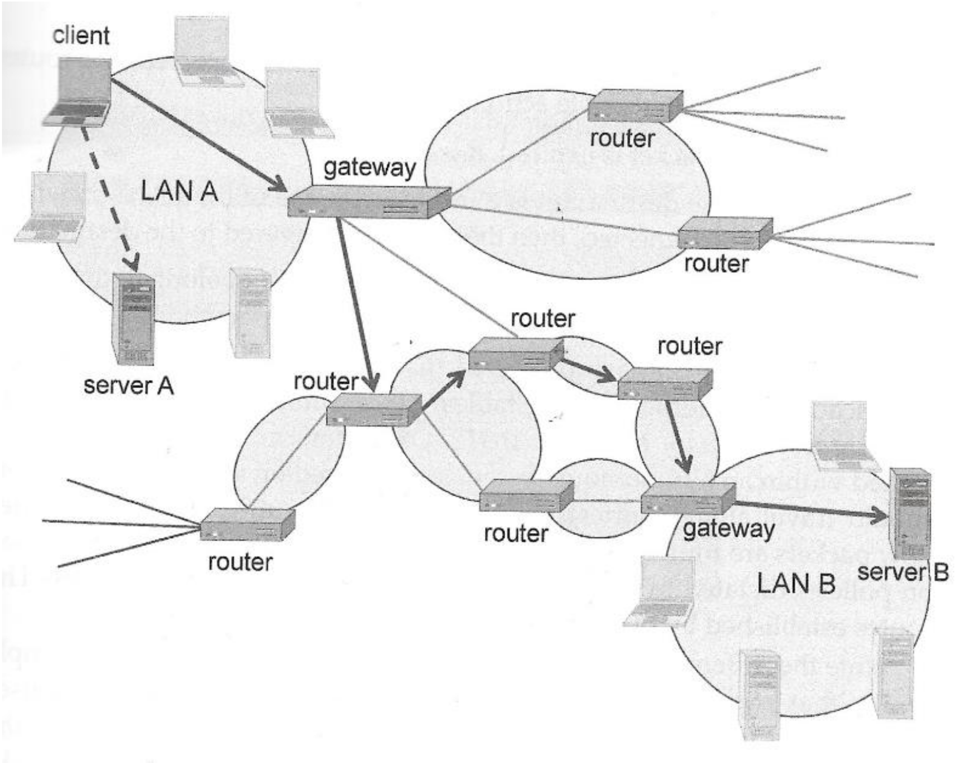
- 2 main versions, IPv4 and IPv6
- *IPv4*: 32-bits, currently most used, but running out of available new addresses for new machines
  - Represented with by 4 byte numbers (0-255 values) separated by ".", eg 127.0.0.1
- *IPv6*: 128-bits, *slowly* trying to replace IPv4…

Routing IP Packets

- If destination IP address is on same LAN, use ARP to get MAC address, and send it directly via Link Layer
- If destination IP address is on different LAN, need to send packet to the LAN *Gateway*
- Client needs to know IP address of Gateway, and then use ARP to get its MAC to send packet to it
- When Gateway receives packet for node not in the LAN, needs to route it out toward other routers

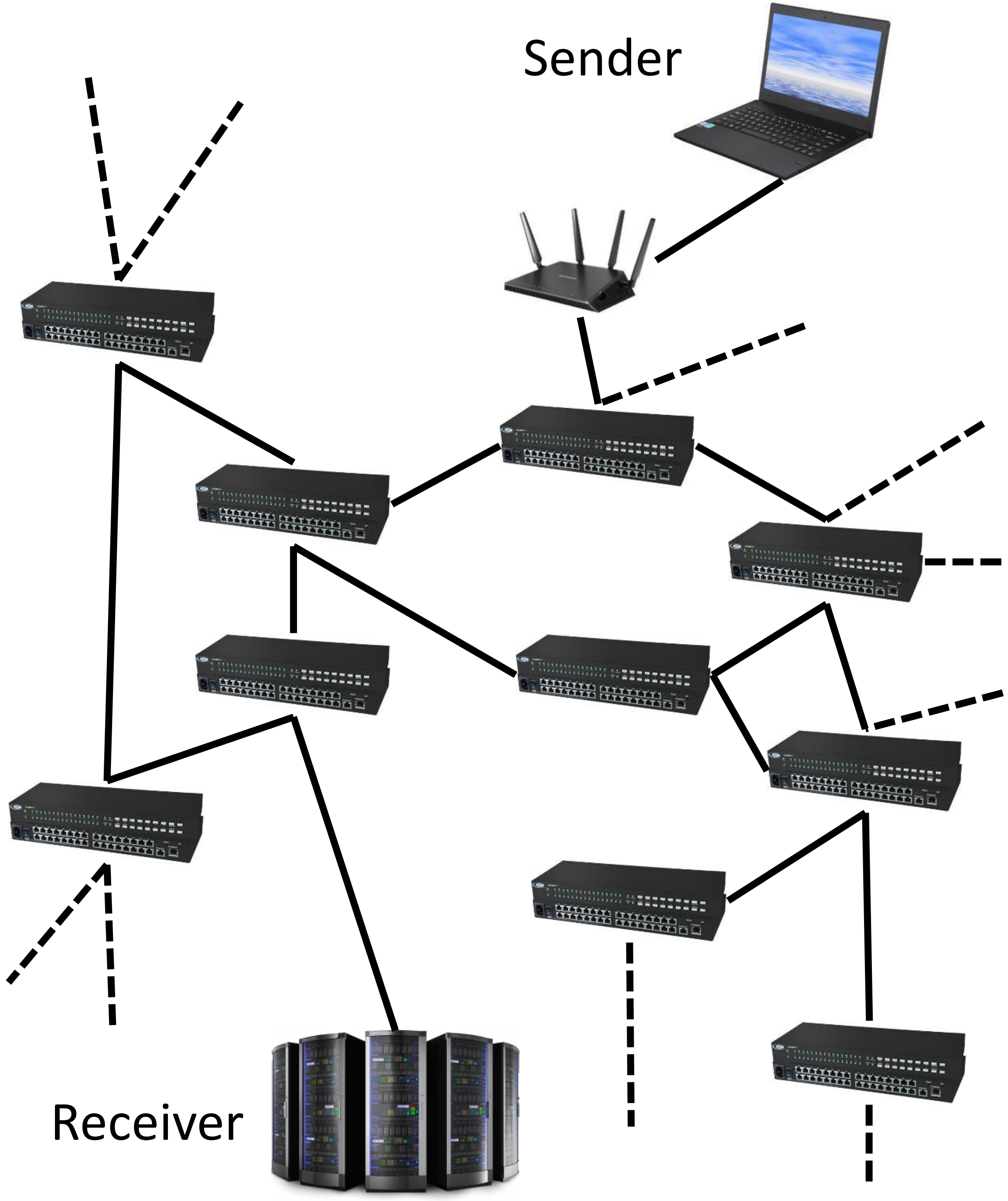
Gateways

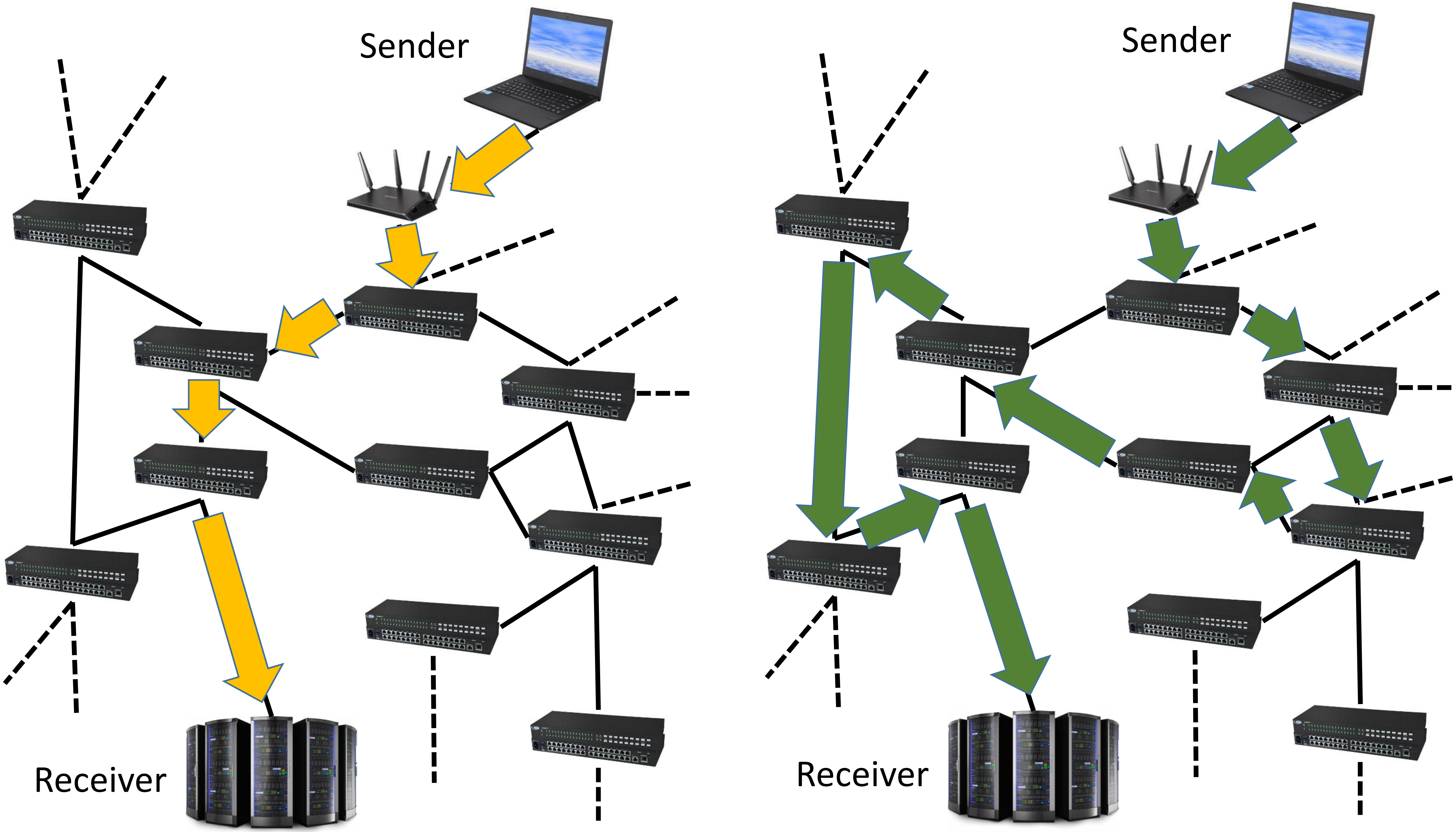
- Each LAN has 1 *gateway* (e.g., a WiFi router), which is the entry/exit point for the LAN
- A packet might go through several *routers* before reaching the gateway of the destination LAN
- *Very complex* algorithms to find out to which router to send packet next



Paths

- Many paths from a sender to a receiver
- Ideally, want the *shortest* one
- Problems: (1) routers can come and go; (2) a single router does not know the whole network topology; (3) some routers could be misconfigured
- Note: finding shortest path in a graph is something that some of you will see in *Algorithms* course next semester

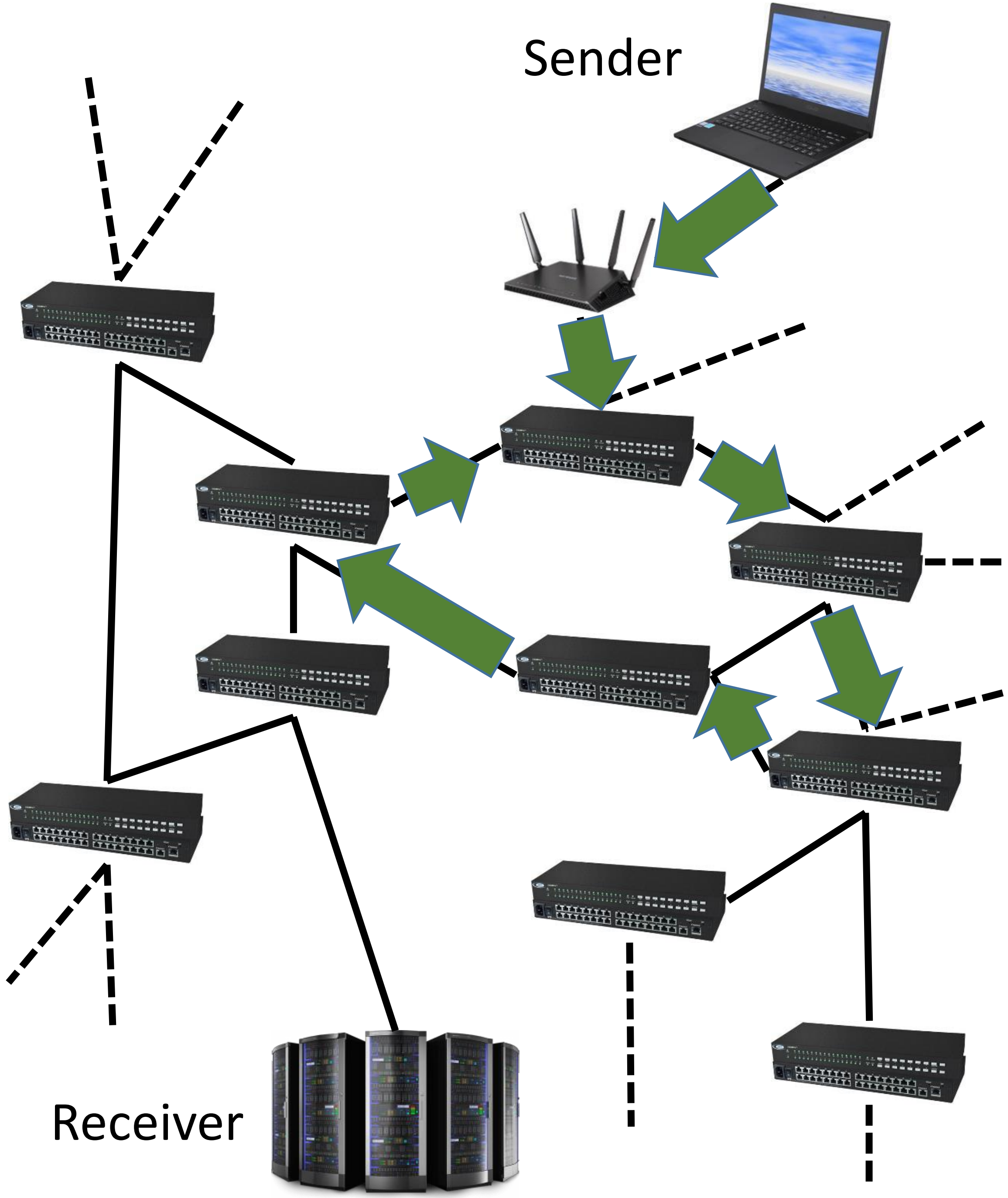




Sender
Receiver
Sender
Receiver

Cycles

- Due to *misconfigurations*, or *dynamic changes in topology*, a packet could be sent back-and-forward in a *cycle*
- Packet will never be delivered, and client will just wait for nothing, in eternity
- Internet protocol has a mechanism to avoid such issue

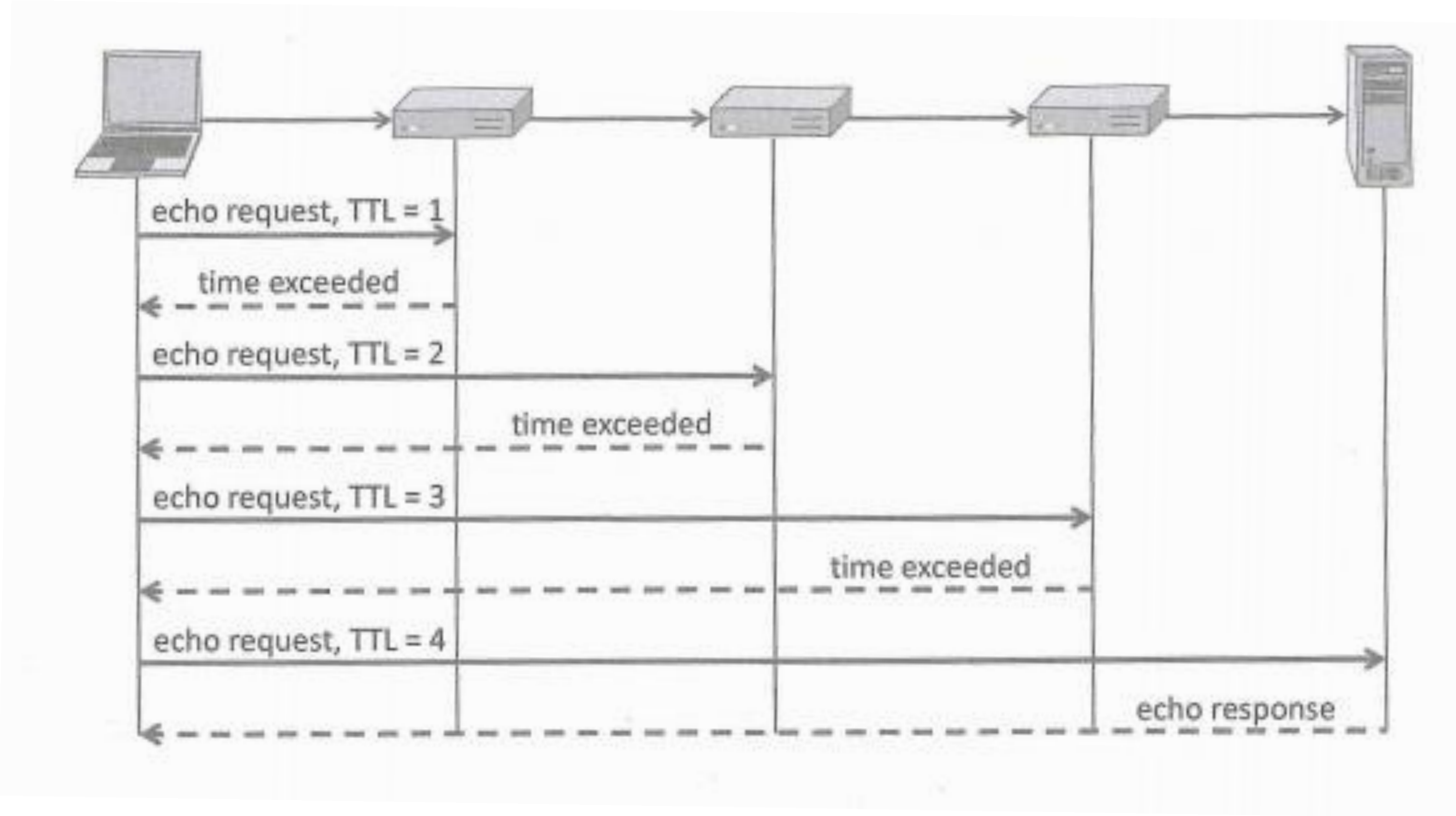


Time-To-Live (TTL)

- Each IP packet has a TTL counter, eg set to N=50
- Each time a router receives a packet, it *decreases* the counter by 1
- When a router sees a TTL=1, it *discards* the packet, and send an error message back to the client sender
- At most a path would be N long, and so impossible to have infinite cycles

But what if low TTL on purpose?

- Start sending IP packets with TTL=1, and increase by 1 at each new request
- Each time getting back an error message from a router in the path
- This way can find all routers toward destination



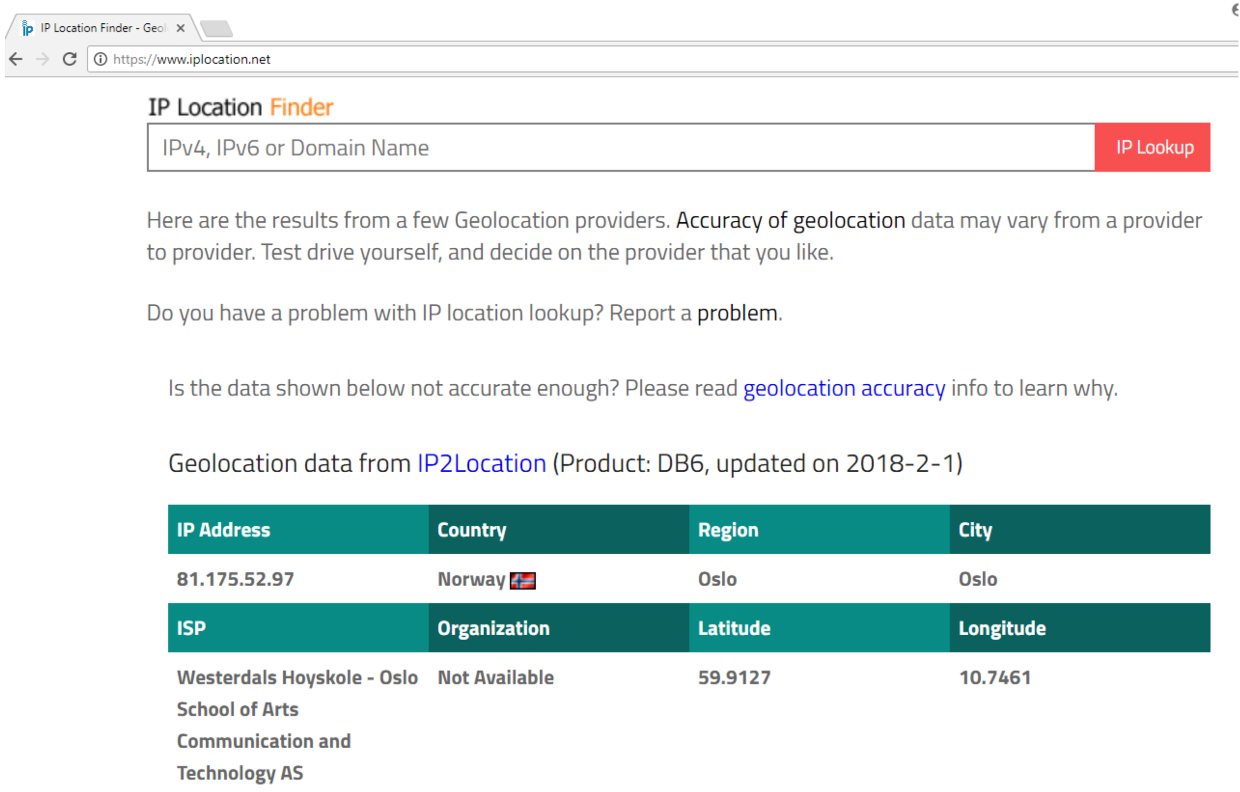
13 machines from Fjerdingen to *google.com*

MINGW64:/e/WORK/code/teaching

```
arcur@DESKTOP-IR7IFID MINGW64 /e/WORK/code/teaching
$ tracert google.com

Tracing route to google.com [216.58.209.142]
over a maximum of 30 hops:

  1     1 ms     1 ms     1 ms  10.96.16.2
  2    <1 ms    <1 ms    <1 ms  10.240.33.1
  3    <1 ms    <1 ms    <1 ms  97.sopp10.party.fredrikstadlan.no [81.175.52.97]
  4    <1 ms    <1 ms    <1 ms  77.88.111.60
  5     1 ms    <1 ms    <1 ms  po5-vl3008.ooe121-060.as41572.net [81.175.32.117]
  6     5 ms     3 ms     4 ms  ten-2-1-vl1504.ooe121-070.as41572.net [81.175.32.226]
  7     1 ms     1 ms     1 ms  oso-b3-link.telia.net [62.115.12.37]
  8     8 ms     8 ms     8 ms  s-bb4-link.telia.net [62.115.137.252]
  9     8 ms     8 ms     8 ms  s-b5-link.telia.net [62.115.133.27]
 10     9 ms     9 ms     9 ms  google-ic-314684-s-b5.c.telia.net [62.115.61.30]
 11     8 ms     8 ms     8 ms  216.239.49.13
 12     9 ms     8 ms     8 ms  216.239.49.217
 13     8 ms     8 ms     8 ms  arn09s05-in-f142.1e100.net [216.58.209.142]

Trace complete.
```

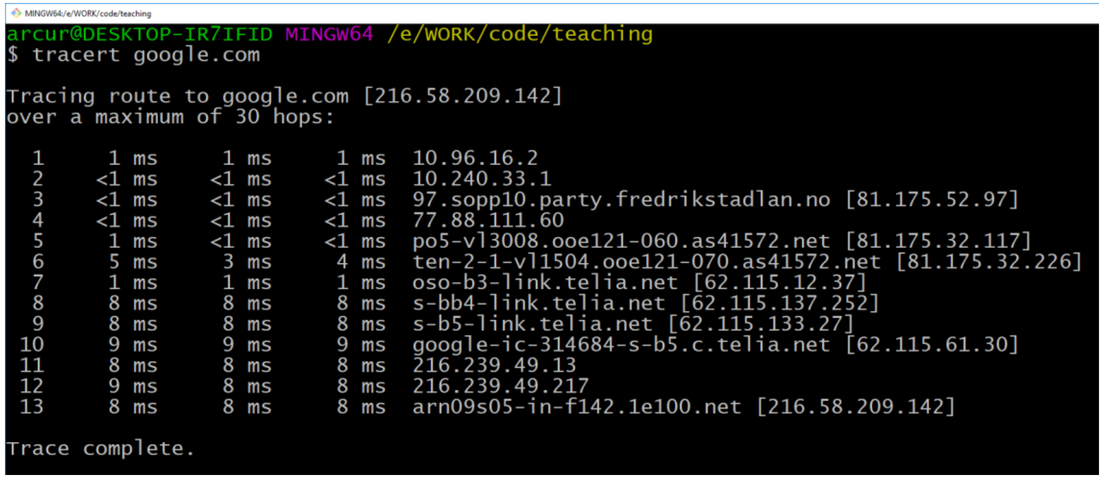
IP Registration

- IP addresses are allocated/sold to companies by a standard internet organization
- Given an IP address, can find *owner* and estimation of the *location*
- Different services can tell you that, eg *www.iplocation.net*



On the route to *google.com*

- *10.x.x.x*: local IP (discussed later)
- *81.175.52.97*: Westerdals, Oslo
- *77.88.111.60*: backbone router
- *62.115.137.252*: Telia Company AB, Germany
- *216.58.209.142*: Google, California
- In other words, *before* reaching server in *California*, packets go first through *Germany*…



Internet Control Message Protocol (ICMP)

- Protocol used to diagnostic a network
- *Ping*: utility/program that checks if a host is reachable, and how long it takes to reach it
- *Traceroute*: utility/program that checks the path toward a host
  - ie, what seen in previous slides based on TTL counters

Pinging *google.com:* 8 milliseconds

MINGW64:/c/Users/arcur

```
arcur@DESKTOP-IR7IFID MINGW64 ~
$ ping google.com

Pinging google.com [172.217.21.174] with 32 bytes of data:
Reply from 172.217.21.174: bytes=32 time=8ms TTL=51
Reply from 172.217.21.174: bytes=32 time=8ms TTL=51
Reply from 172.217.21.174: bytes=32 time=8ms TTL=51
Reply from 172.217.21.174: bytes=32 time=8ms TTL=51

Ping statistics for 172.217.21.174:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 8ms, Maximum = 8ms, Average = 8ms
```

IP Spoofing

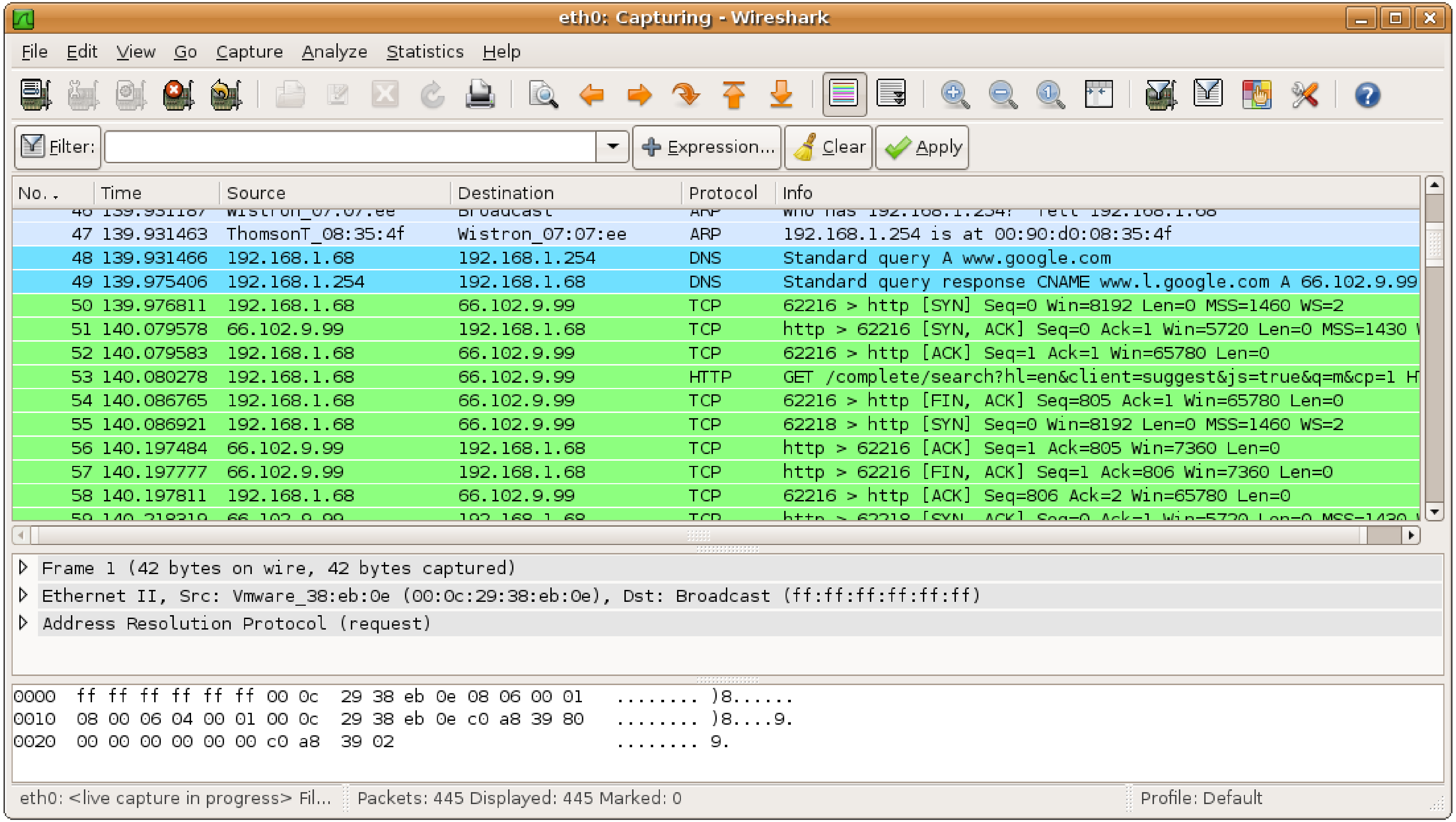
- IP Packet contains “*Destination*” and “*Source*” addresses
- But when you send a packet, you can alter the *Source* by using the IP of another machine
  - there is no check that Source address is the real one
- You will not get the reply, the other machine specified in Source will get it
- Why? Trick machine to send messages to another one (eg, DoS), or to circumvent firewall policies based on IP sources…

Bit Offset	0-3	4-7	8-15	16-18	19-31	
0	Version	Header length	Service Type	Total Length		
32	Identification			Flags	Fragment Offset	
64	Time to Live		Protocol	Header Checksum		Header
96	Source Address					
128	Destination Address					
160	(Options)					
160+	Data Data Data Data Data Data Data Data Data Data Data Data ...					Payload

Packet Sniffing

- When using *hub,* packets are *broadcasted* to all connected device
- In *WiFi*, it is the same kind of *broadcasting*
- A computer will receive *all messages*, but then *discard* the ones meant for other MAC addresses
- You can analyze all packets, even the ones meant for others
    - Note: some network hardware cards do not support it, and neither some OS by default (eg, not trivial to get it to work on Windows…)

Wireshark


eth0: Capturing - Wireshark
File Edit View Go Capture Analyze Statistics Help
Filter:
Expression... Clear Apply
No. Time Source Destination Protocol Info
47 139.931463 ThomsonT_08:35:4f Wistron_07:07:ee ARP 192.168.1.254 is at 00:90:d0:08:35:4f
48 139.931466 192.168.1.68 192.168.1.254 DNS Standard query A www.google.com
49 139.975406 192.168.1.254 192.168.1.68 DNS Standard query response CNAME www.l.google.com A 66.102.9.99
50 139.976811 192.168.1.68 66.102.9.99 TCP 62216 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=2
51 140.079578 66.102.9.99 192.168.1.68 TCP http > 62216 [SYN, ACK] Seq=0 Ack=1 Win=5720 Len=0 MSS=1430
52 140.079583 192.168.1.68 66.102.9.99 TCP 62216 > http [ACK] Seq=1 Ack=1 Win=65780 Len=0
53 140.080278 192.168.1.68 66.102.9.99 HTTP GET /complete/search?hl=en&client=suggest&js=true&q=m&cp=1
54 140.086765 192.168.1.68 66.102.9.99 TCP 62216 > http [FIN, ACK] Seq=805 Ack=1 Win=65780 Len=0
55 140.086921 192.168.1.68 66.102.9.99 TCP 62218 > http [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=2
56 140.197484 66.102.9.99 192.168.1.68 TCP http > 62216 [ACK] Seq=1 Ack=805 Win=7360 Len=0
57 140.197777 66.102.9.99 192.168.1.68 TCP http > 62216 [FIN, ACK] Seq=1 Ack=806 Win=7360 Len=0
58 140.197811 192.168.1.68 66.102.9.99 TCP 62216 > http [ACK] Seq=806 Ack=2 Win=65780 Len=0
Frame 1 (42 bytes on wire, 42 bytes captured)
Ethernet II, Src: Vmware_38:eb:0e (00:0c:29:38:eb:0e), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Address Resolution Protocol (request)
0000 ff ff ff ff ff ff 00 0c 29 38 eb 0e 08 06 00 01 ....)8......
0010 08 00 06 04 00 01 00 0c 29 38 eb 0e c0 a8 39 80 ....)8....9.
0020 00 00 00 00 00 00 c0 a8 39 02 .... 9.
eth0: <live capture in progress> Fil... Packets: 445 Displayed: 445 Marked: 0 Profile: Default

Please, do **NOT**…

- … go to a place with a public WiFi…
- … use Wireshark in promiscuous/monitor mode…
- … and then check if any #@^%&* person is accessing not-encrypted web sites or mobile apps…
- And first of all, do **NOT** be that kind of #@^%&* person!!!
  - Especially if you see someone running Wireshark on their screens…

Transport Layer

Applications

- When application *X* wants to send a message to application *Y* on different machine *S*, needs to know its IP address
- But on *S* there can be *different applications* running
- How can we know on which application in *S* should the messages of *X* go to?
- IP address of *S* is not enough… we also needs *port* numbers

Ports

- Ports are 16-bit unsigned numbers, so in 0-65535 ($2^{16} - 1$)
- An application wanting to receive messages, needs to open a *listening* port on its side
- Given the same Transport Layer protocol, no 2 applications can open the same port at the same time
  - combination of protocol + port should be unique
  - eg, starting 2 web servers on same port would have the second failing

Ports to know

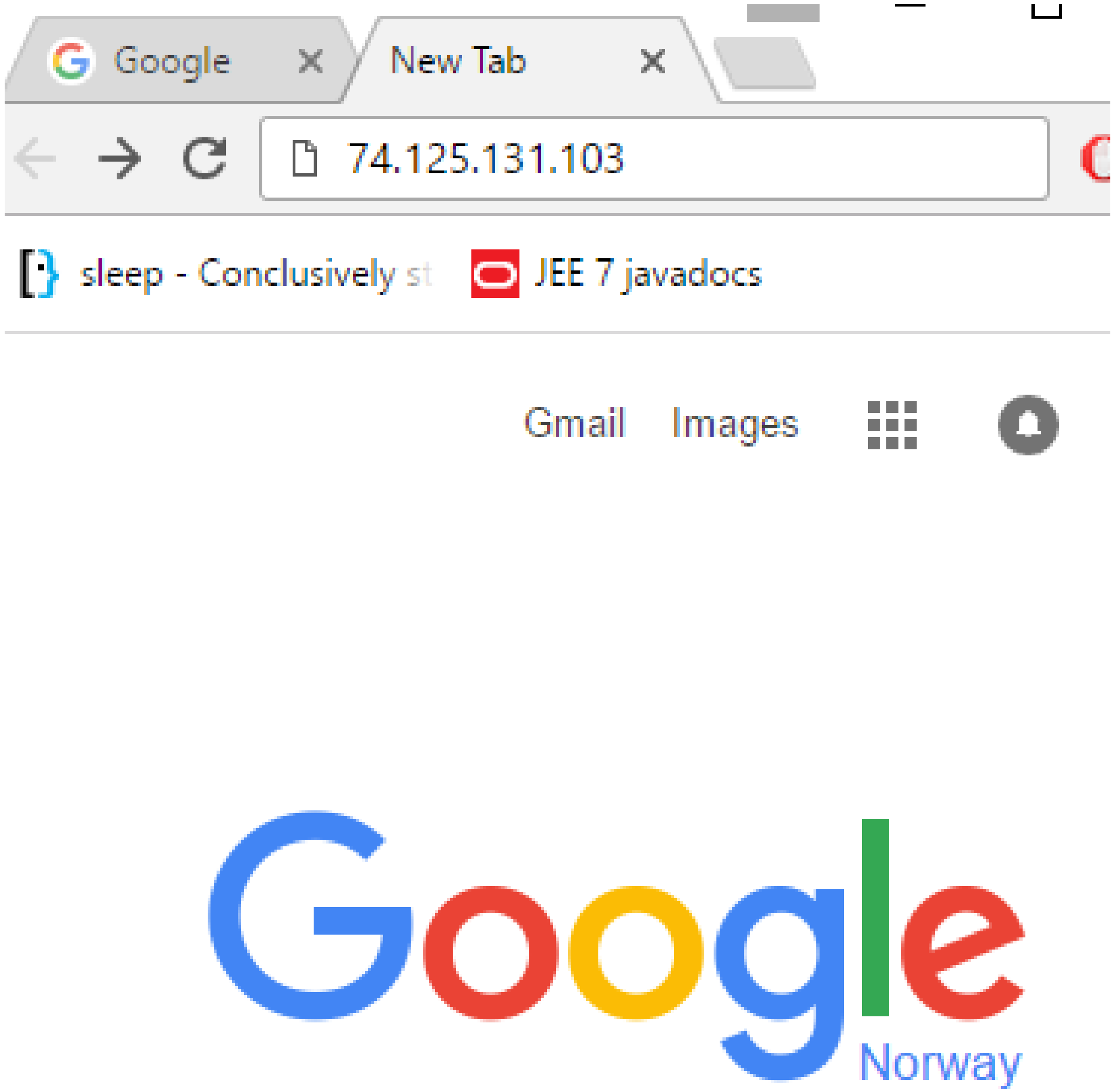
- 0: dynamically allocated (see next slide)
- 22: for SSH connections
  - Very common when you need to connect to a remote server or an IoT device using a terminal
- 80: for a HTTP connection
  - When browsing the web without encryption
- 443: for HTTPS, ie secure/encrypted HTTP over TLS/SSL
  - More and more common nowadays, even when no user authentication
- 8080: unreserved port.
  - “Typically” used by tools when running a HTTP server locally on your machine
- Note: HTTP is an Application Layer protocol specific for the web

Ephemeral/Dynamic ports

- “Typically” in range 49152–65535
  - But will vary based on the OS
- For reading responses from the other machine
  - when you establish a connection to a remote machine, a port will be open locally to read the responses
- During development, when using port 0
  - Get a dynamic port which is free, not used
  - Eg, if you want to start a server locally for debugging/testing, you might want to avoid conflicts with other applications using the same port

Defaults

- When nothing specified, browsers do default to known ports for the given *application* protocol
  - Default protocol is HTTP, and default resource is the root “/”
- So typing *www.google.com* is equivalent to ***http://****www.google.com****:80/***
- Typing http**s**://www.google.com is equivalent to http**s**://www.google.com**:443/**
- Note: the page you request might not be the one you will get, as you could get a HTTP *3xx redirection*



- If you know the IP address, you can type it directly
- Same “name” can be mapped to different IP addresses, ie different servers
- The mapping can change
- We will go in more details on DNS next class

Before going into the details…

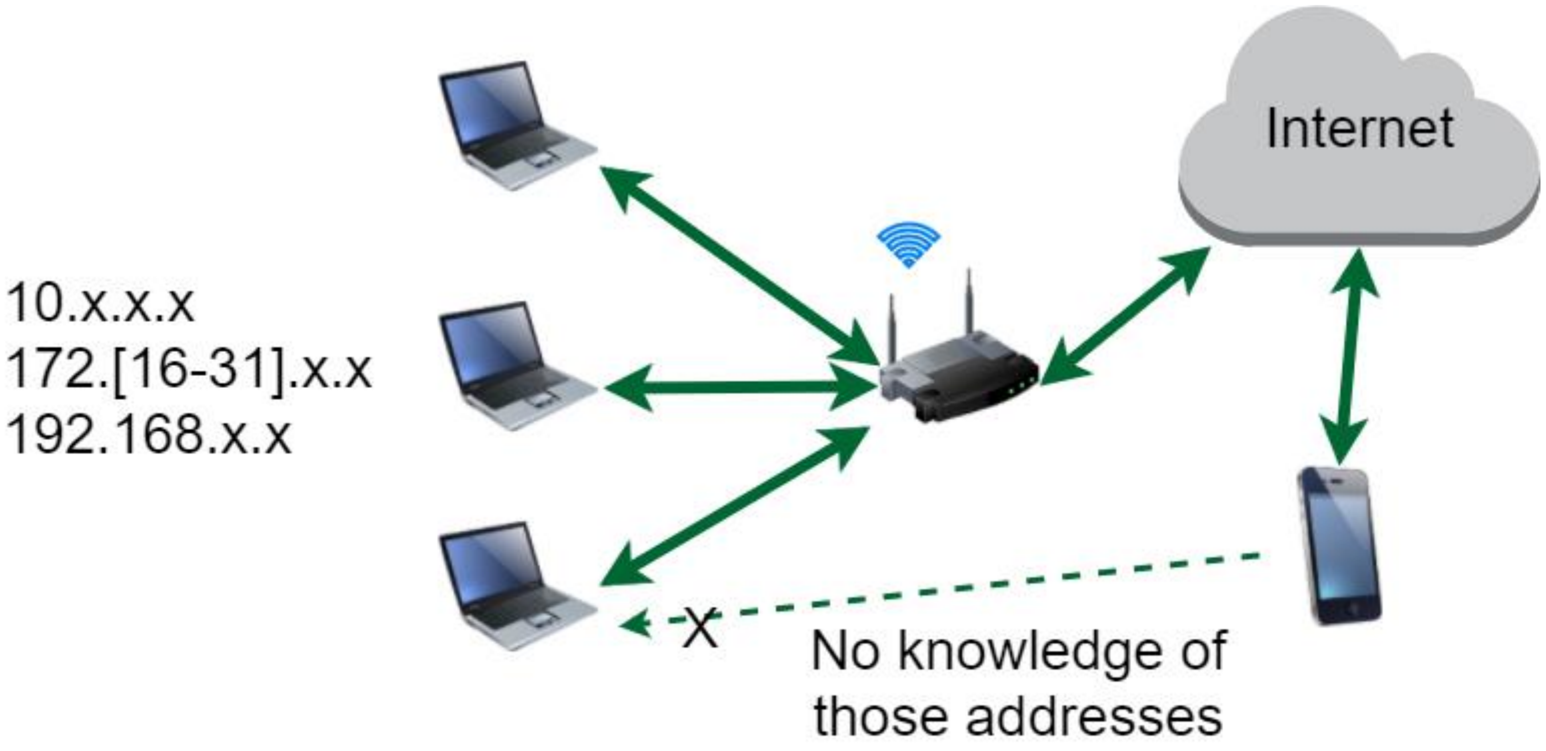
- … of Transport Layer protocols like **TCP** and **UDP**, let’s go back a moment to IP addresses
- *How many devices are connected to the internet?*

Some Estimations

- 2016: around 300 **M**illions *new* devices connected *each month*
- 2020: estimated around 50 **B**illions devices connected
- *Why is this a problem?*
- IPv4 are 32-bits, so can only represent 4 **B**illion devices
- What to do until IPv6 is widely supported?

Local Networks

- Router has IP accessible from internet
- Machines connected to it have *local IP* not visible from outside
- Cannot use mobile to connect to such machines, unless special settings on router, or WiFi directly to same router

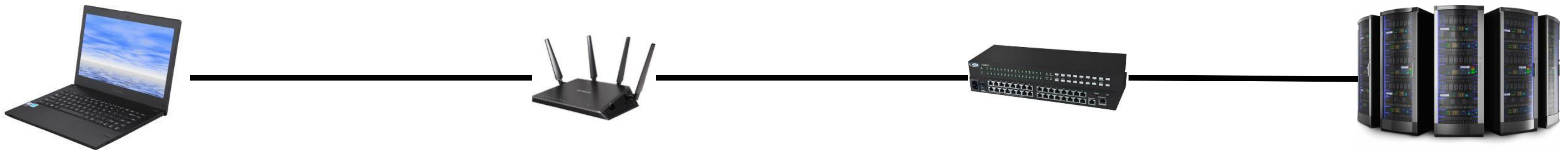


Network Address Translation (NAT)

- A LAN will have IP addresses in
  - 10.x.x.x, $2^{24} = 16m$ numbers, class A network
  - 172.[16-31].x.x , $2^{20} = 1m$ numbers, class B network
  - 192.168.x.x, $2^{16} = 65k$ numbers, class C network
- These local IP addresses are NOT used on internet
- Different LANs can use same local IP addresses with **NO** conflicts
- Only the gateway has an IP address that is reachable from outside
  - So behind a single IP address, there could be millions of devices
- Gateway has to handle the mapping/translation of IP:port

NAT Example

- Let’s see a HTTP connection to *google.com:80*
- Packet: source 192.168.0.1:61234, where the port is ephemeral
- Gateway does *change* the packet content by replacing source with itself, with an ephemeral port that can be different (eg 12345)
- Google will send back HTML page to the gateway at 81.175.52.97:12345
- When gateway receives message on port 12345, it knows that it has to route it to 192.168.0.1:61234
- Google server does NOT know about the IP and port of the laptop in the LAN



192.168.0.1:61234 81.175.52.97:12345 216.58.209.142:80

IP Addresses to Know

- 127.0.0.1
  - loopback address. Virtual one bypassing the network card. Useful for debugging, when running network applications (eg a web server) on your machine
- 10.x.x.x, 172.[16-31].x.x, 192.168.x.x
  - local LAN addresses
- 255.255.255.255
  - broadcast address. LANs might have it disabled for security reasons (discussed later)
- 0.0.0.0
  - all addresses on local machine. Useful when starting a server, and want it to listen to all possible incoming messages (a machine can have more than a network card, eg a cable and a WiFi)

UDP (User Datagram Protocol)

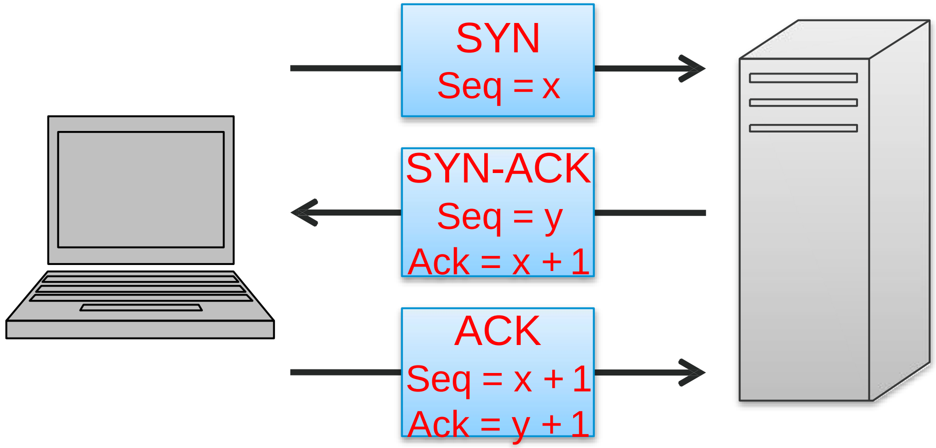
- One of the main Transport Layer protocols
- Single *independent* packets
- No guarantee of delivery
- No guarantee that packets are received in same order of sending
  - might go through different routing paths
- Useful in *time-sensitive* applications when lost of some data is acceptable

TCP (Transmission Control Protocol)

- The main protocol used for the web
  - eg, when using a browser to access HTML pages
- Cannot guarantee delivery, *but* will try to resend packets if detects that were not received
  - ie, getting back an acknowledgement when packets are received
- Packets are logically linked together in a *session*, each one having an increasing index number
- Buffer of packets on receiving side, to re-order packets if arrived in wrong order
- Can automatically slow down sending of packets if too many are lost

Three-Way Handshake

- X and Y are originally chosen at random
- Needed to decide increasing numbers for ordering of packets
- If packet is received with an unexpected number, then buffered until the previous packets arrive



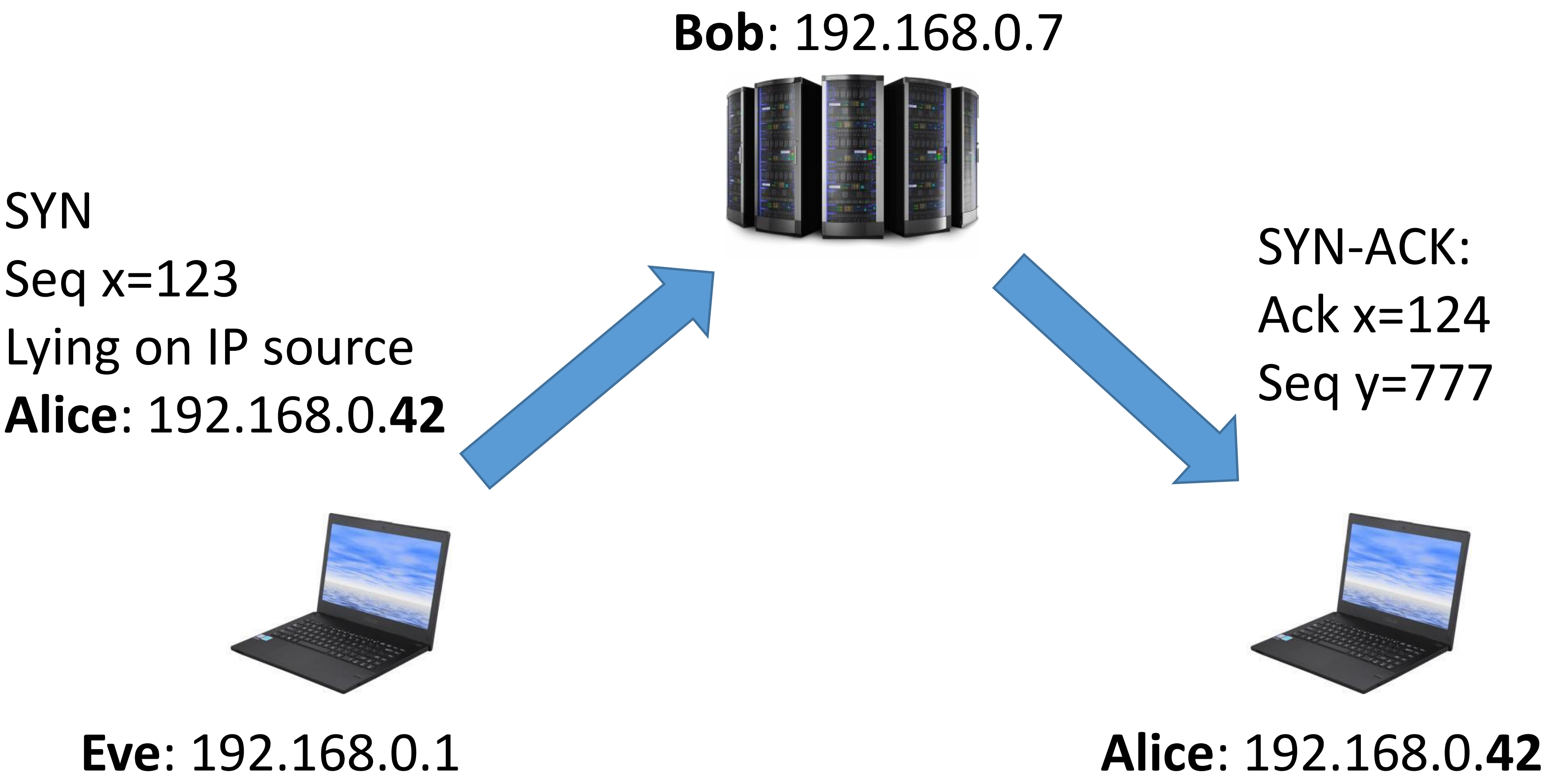
TCP vs. UDP

"Hi, I'd like to hear a TCP joke."
"Hello, would you like to hear a TCP joke?"
"Yes, I'd like to hear a TCP joke."
"OK, I'll tell you a TCP joke."
"Ok, I will hear a TCP joke."
"Are you ready to hear a TCP joke?"
"Yes, I am ready to hear a TCP joke."
"Ok, I am about to send the TCP joke. It will last 10 seconds, it has two characters, it does not have a setting, it ends with a punchline."
"Ok, I am ready to get your TCP joke that will last 10 seconds, has two characters, does not have an explicit setting, and ends with a punchline."
"I'm sorry, your connection has timed out.
...Hello, would you like to hear a TCP joke?"

TCP Session Hijacking

- Old TCP implementations used X-Y starting from fixed value, eg 0
- Attacker can hijack a TCP connection, by sending packet with IP of attacked machine Z, and predict TCP session numbers
  - Note: the attacker will not see the responses (*blind*), because sent to Z, and that is the reason why it needs to find out the correct session numbers
  - Can do man-in-the-middle attack
  - Random session numbers prevent such issue
- If on same LAN, can use Wireshark to see the session numbers
- Have to use encryption to prevent this kind of attack

Example: after sending the first SYN by spoofing the IP of Alice, Eve needs to find out the Seq y=777 of Bob before sending the next TCP packet



Denial-of-Service (DoS) Attacks

Limits

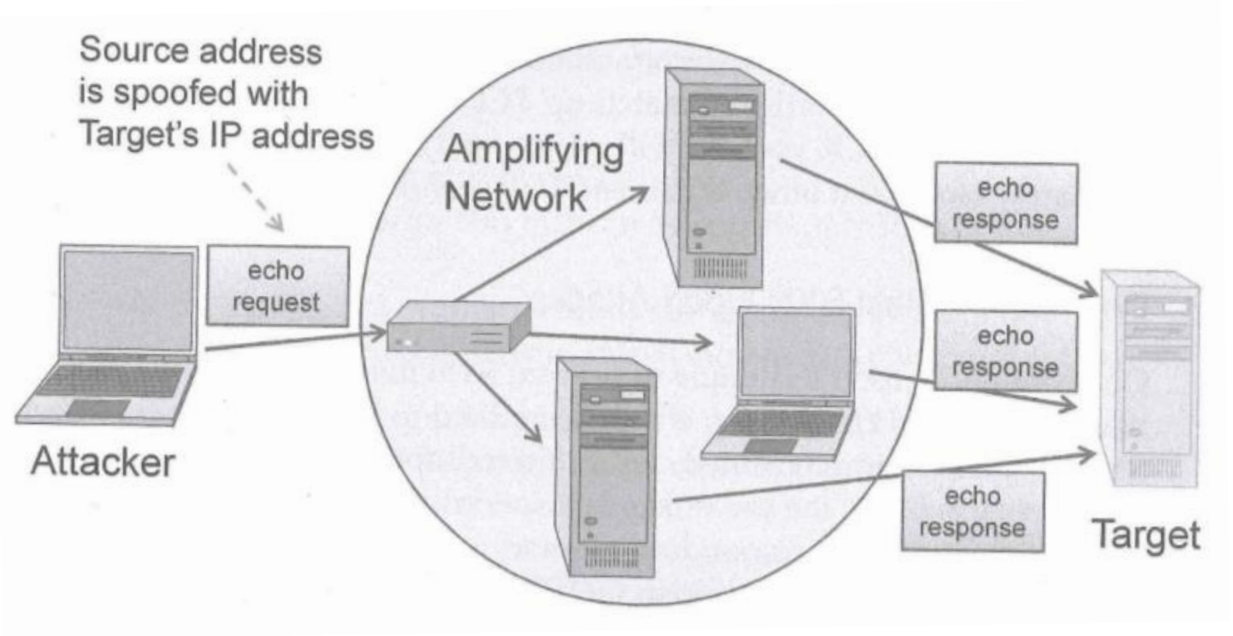
- Servers have limits on:
  - Memory
  - CPUs
  - Bandwidth
  - etc.
- If receiving too many requests from an attacker, then server cannot answer legitimate users, ie a so called *denial-of-service*

Ping Flood Attack

- Simple attack
- Just do a lot of Ping/ICMP requests against a target
- Only works if attacking machine has more *resources* than the victim one

Smurf Attacks

- Many LANs do have a broadcast address (e.g., *255.255.255.255*)
- A packet sent to such address will be broadcast to all machines on the network
- If ICMP packet, the machines will reply to the source of that packet (eg Ping)
- But you can spoof the IP of a target machine, and so use the whole network to DoS the target



WARNING!!!

- A Smurf Attack is quite simple to do…
  - and just to be on safe side, I will not show the actual code, but you could google it…
- Do **NOT** do this kind of attack (or any attack), nor any broadcasting, on the WiFi network in school…
  - due to some services needing it, broadcasting is not disabled
- WiFi connections of students are logged, and this kind of disruption of school services (eg abuse of broadcasting or DoS) can be ground for *degree termination* and *expulsion*
  - this is explicitly stated in the contract you signed…
- If you want to try it, do it at *home* with a private router you own…

SYN Flood

- One of the most typical types of DoS attack
- In TCP connection, server needs to keep some *state*
  - eg, the X/Y session numbers
- An attacker might send many SYN requests, never followed by the ACK
- Each time, different IP source (can be spoofed)
- Server can run out of memory, because would keep state of connection until timeout
- Some attempts to prevent this type of attack (eg SYN Cookies), but nothing widely used/supported

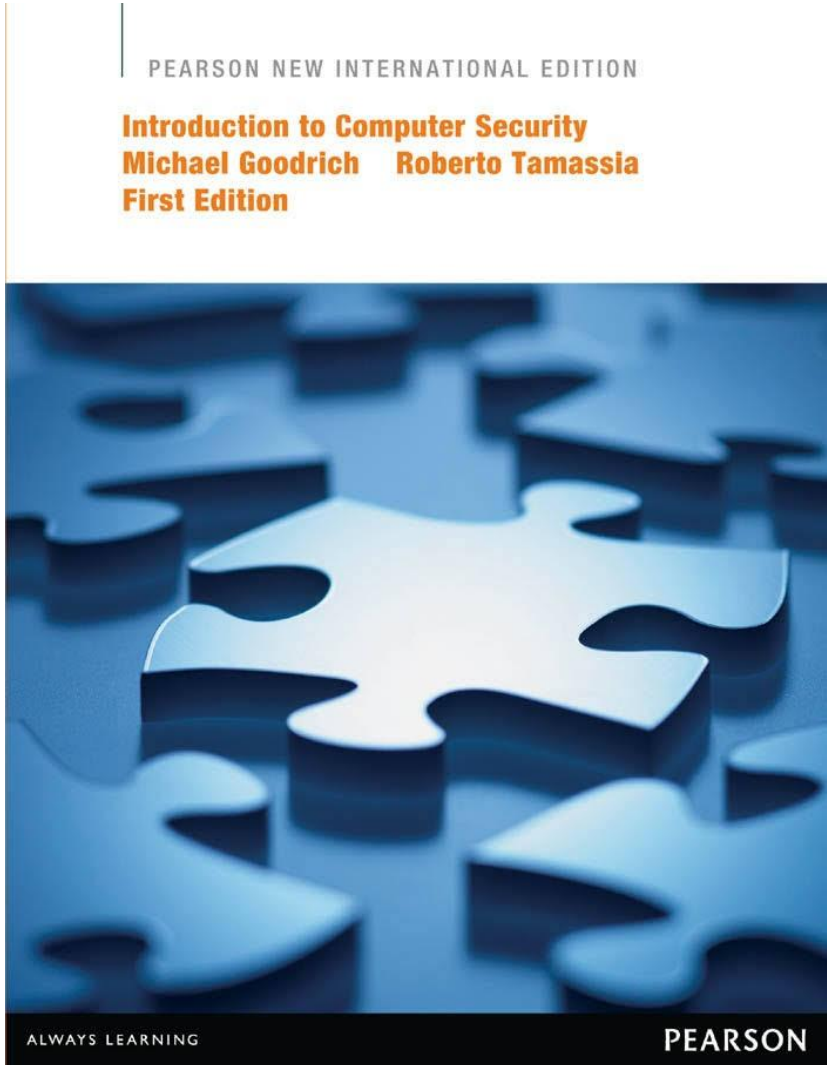
Distributed DoS (DDoS)

- A single machine is unlikely to have the same resources of an enterprise system (eg, Amazon, Google or Facebook)
- But with Malwares, can have a bot-army that can do SYN Floods
- Servers of large corporations can be taken down

DDoS Prevention

- If see that there are many requests coming from same IP, such IP can be *blocked*
- But that is usually *futile*, as IP addresses can be easily spoofed
- Many different techniques proposed to detect if packets are coming from the same source *without* relying on IP addresses
- But because they require changes on all routers on internet, only little is really used in practice…

For Next Week



- Book pages: 221-263
- Note: when I tell you to **study** some specific pages in the book, it would be good if you also *read* the other pages in the same chapter at least once
- Exercises for Lesson 5 on GitHub repository