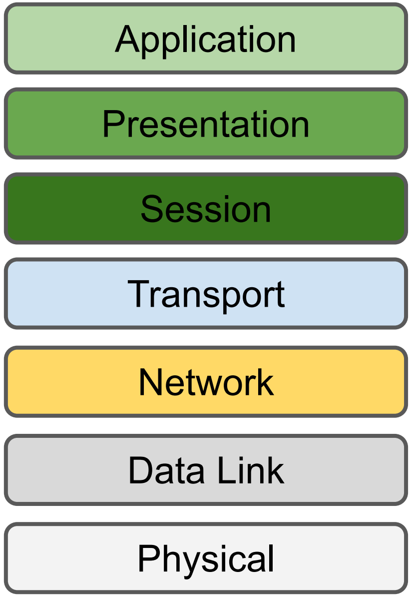
Network Fundamentals

Enrico RUSSO - <enrico.russo@unige.it>

ISO/OSI and TCP/IP

- ISO/OSI and TCP/IP represent the reference models for communication between different computers in the network. They both use a **layered** model
  - Separate networking functions into logical smaller pieces: network problems can more easily be solved through a **divide-and-conquer** methodology
  - Provide **modularity** and **clear interfaces**: they allows the standardization of interactions among devices
  - Allow **extensibility**: new network functions are generally easier to add to a layered architecture
- ISO/OSI model evolved as a **theoretical** model
- TCP/IP as a **practical** model, founded on widely used implementation of network functions

OSI Layers



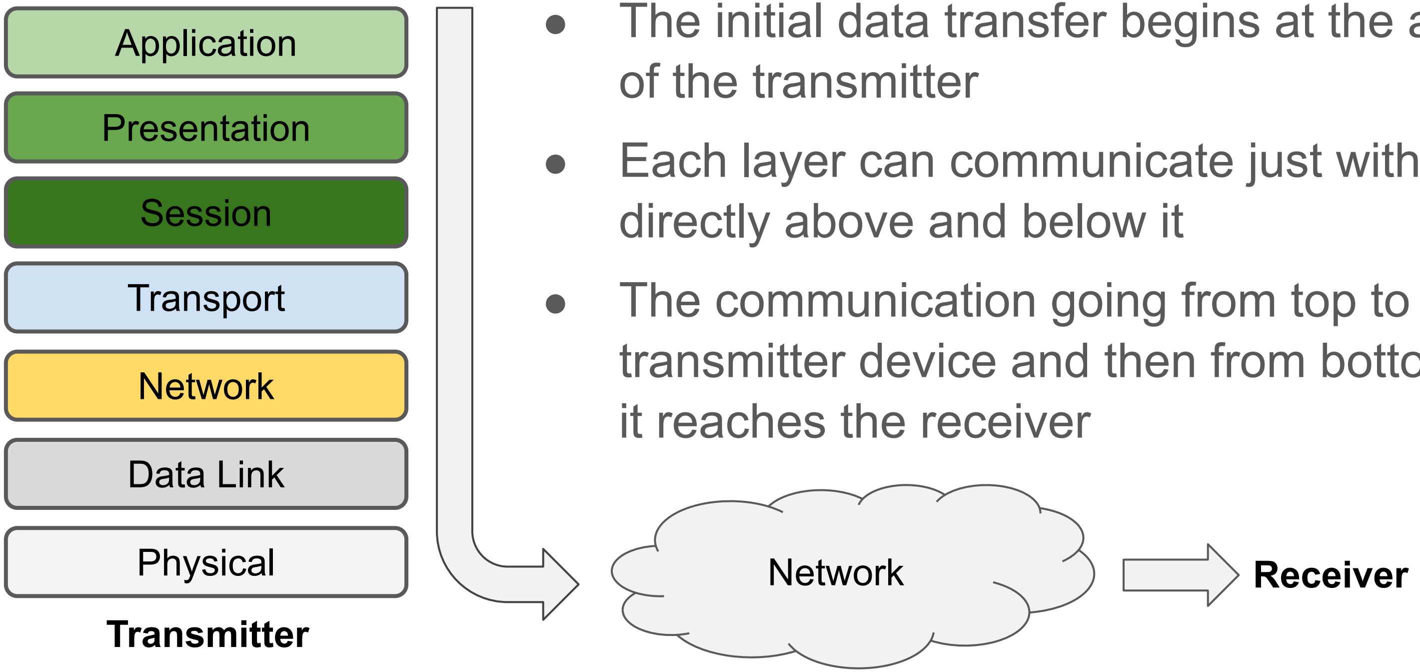
The Open Systems Interconnection (OSI) represents a guideline for network protocol design.

- A standard of the International Organization for Standardization (ISO)
- Seven layers

OSI Layers

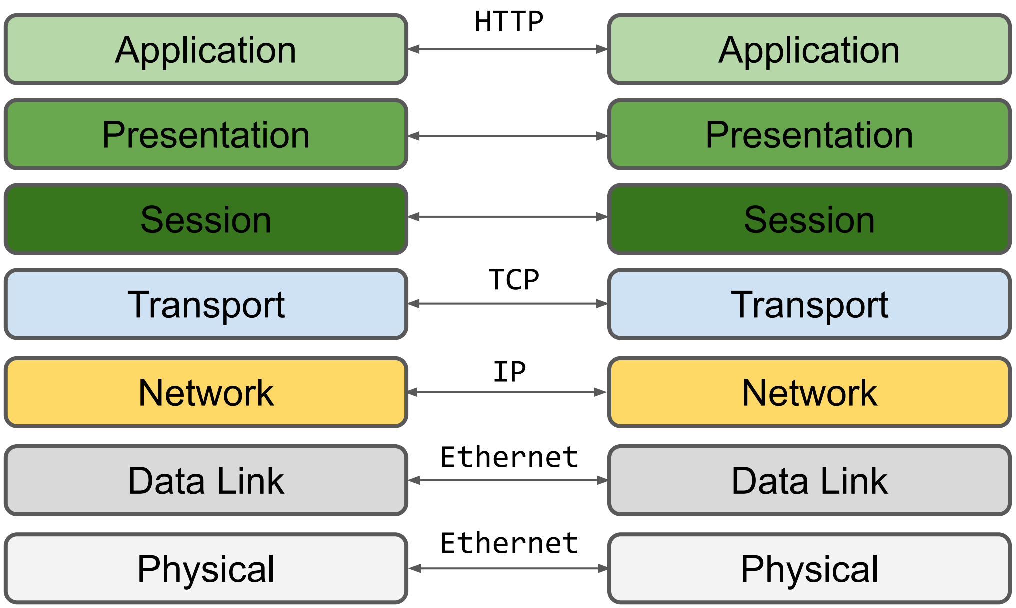
Application	It provides the services to the user
Presentation	It is responsible for the formatting of information (e.g., compression and encryption)
Session	It is responsible for establishing, managing, and terminating sessions
Transport	It provides message delivery from process to process
Network	It is responsible for moving the packets from source to destination
Data Link	It combines bits into a structure of data and provides their error-free transfer
Physical	It provides a physical medium through which bits are transmitted

OSI Layers: data transfer



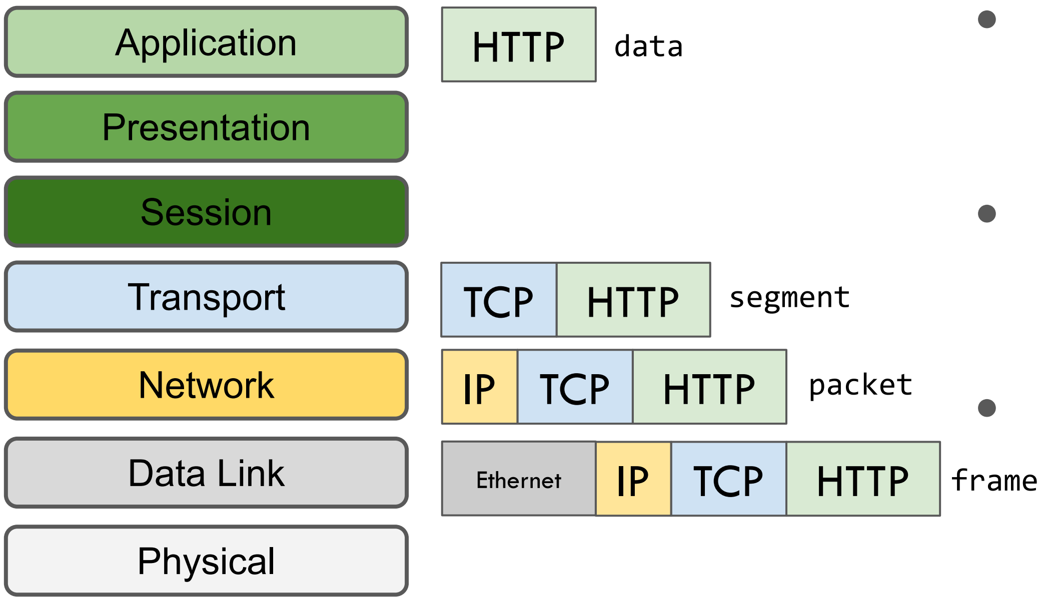
- The initial data transfer begins at the application layer of the transmitter
- Each layer can communicate just with the layers directly above and below it
- The communication going from top to bottom on the transmitter device and then from bottom to top when it reaches the receiver

OSI Layers: protocols



- The model itself does not provide specific methods of communication
- Actual communication is defined by various ***protocols***
- A protocol is a **standard procedure and format** that two data communication devices must understand, accept and use to be able to talk to each other

OSI Layers: Protocols Data Unit (PDU)



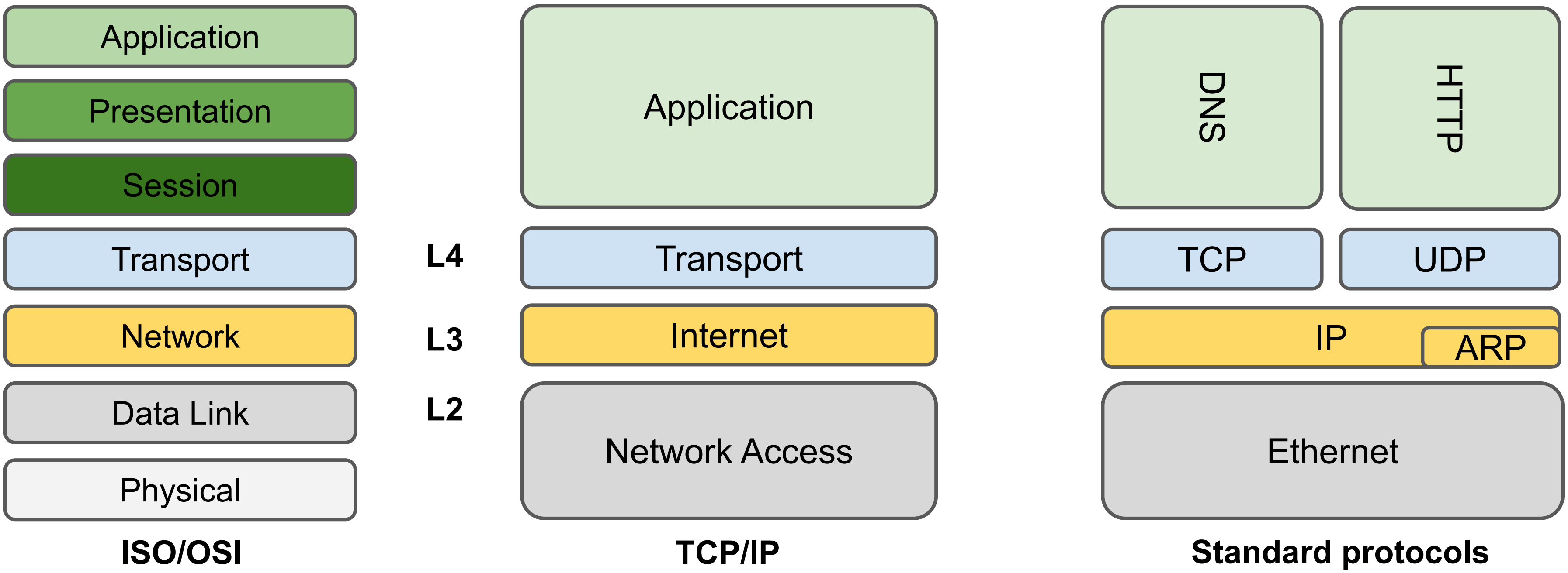
- The protocols at different layers exchange data with the aid of data encapsulation
- Each layer is responsible for adding a header or a footer to the data being transferred
- The encapsulation process creates a Protocol Data Unit (PDU), which includes the data being sent and all header or footer information added to it

TCP/IP

TCP/IP provides an alternative model used for the description of all network communications.

- is a four-layer model
- is based on standard protocols that the Internet has developed, and the name refers to the two widely used ones:
    - **Transmission Control Protocol** (TCP) which also implements the Transport layer of ISO/OSI model
    - **Internet Protocol** (IP) which also implements the Network layer of ISO/OSI model

TCP/IP model


Application
Presentation
Session
Transport
Network
Data Link
Physical
ISO/OSI
L4
L3
L2
Application
Transport
Internet
Network Access
TCP/IP
DNS
HTTP
TCP
UDP
IP
ARP
Ethernet
Standard protocols

Ethernet

Ethernet is a broadly deployed **layer 2 protocol**

- Encapsulate data and transmit them in the form of ***frames***
- Frames leverage the **Media Access Control** (MAC) addresses
  - **48 bits** burned in the adapter ROM (first 3-bytes: the ID of the manufacturer*)
  - Every Ethernet device (e.g., a server, a switch, or a router) has a **unique MAC address** on their local network
  - A Frame includes the MAC address of the **destination interface** on the target system as well the MAC address of the **source interface** on the sending system

*https://www.wireshark.org/tools/oui-lookup.html

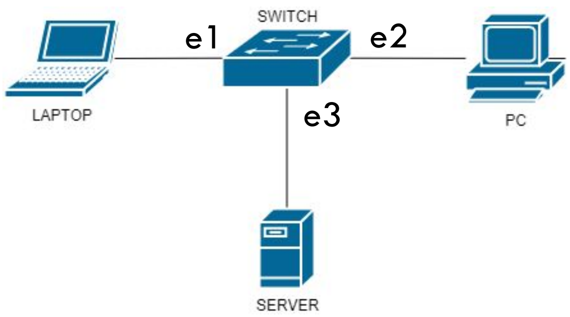
Bridges and Switches

Devices providing interconnectivity at Layer2 are called *(Transparent) Bridges or Switches*

- Analyze all frames received, find the **destination MAC address**, and **forward** to the appropriate port
- To determine where to forward the traffic uses a special table (**MAC address table**)

A basic switched network

- A switch device provides connection to a number of common devices
- Let's assume that all of the devices are powered on but have not sent any traffic
- In this case, the MAC address table of the switch would be empty

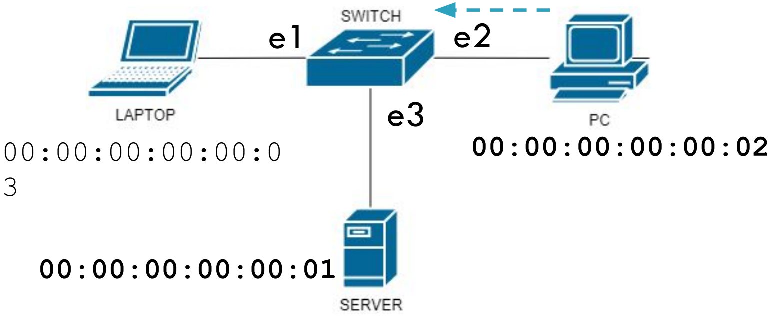


MAC address table (switch)

MAC address	Port

A basic switched network

- PC wants to send traffic to SERVER that has MAC address `00:00:00:00:00:01`
    - Creates a frame containing `00:00:00:00:00:02` as the source address and `00:00:00:00:00:01` as the destination address
    - Sends it off toward the switch

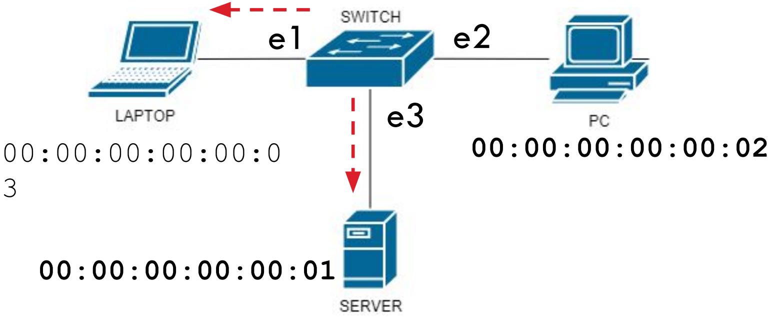


MAC address table (switch)

MAC address	Port

A basic switched network

- The switch receives the traffic
  - Creates a new entry in its MAC address table for PC MAC address (PC → e2)
  - Performs a lookup on its MAC address table to determine whether it knows which port to send the traffic to
  - Since no matching entries exist in the switch's tables, it would **flood** the frame out all of its interfaces except the receiving port (**broadcast**)

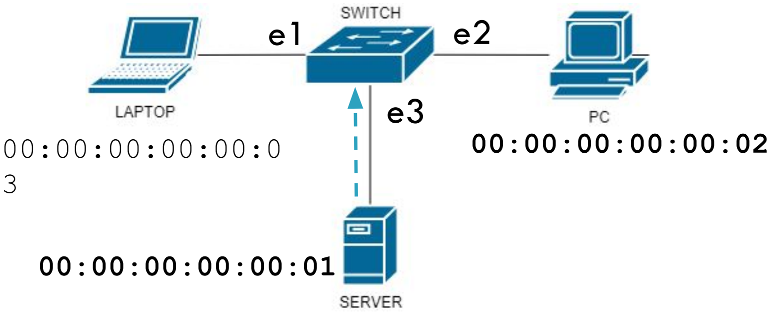


MAC address table (switch)

MAC address	Port
00:00:00:00:00:02	e2

A basic switched network

- The broadcast forwards the frame also to the target server
- (Assuming that the server wants to respond to PC) It sends a new frame back toward the switch containing `00:00:00:00:00:01` as the source address and `00:00:00:00:00:02` as the destination address
- The switch would receive the frame and create a new entry in its MAC address table for the Server MAC address (Server → e3)

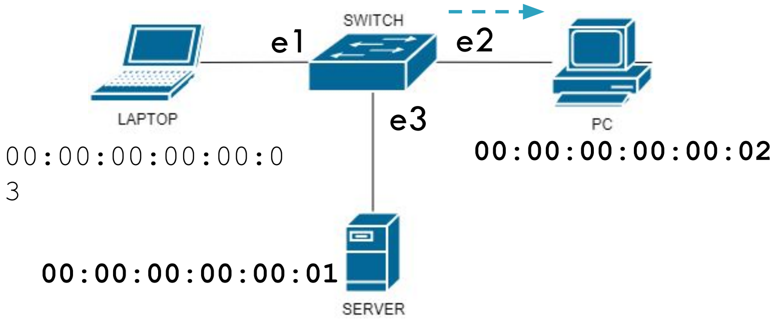


MAC address table (switch)

MAC address	Port
00:00:00:00:00:02	e2
00:00:00:00:00:01	e3

A basic switched network

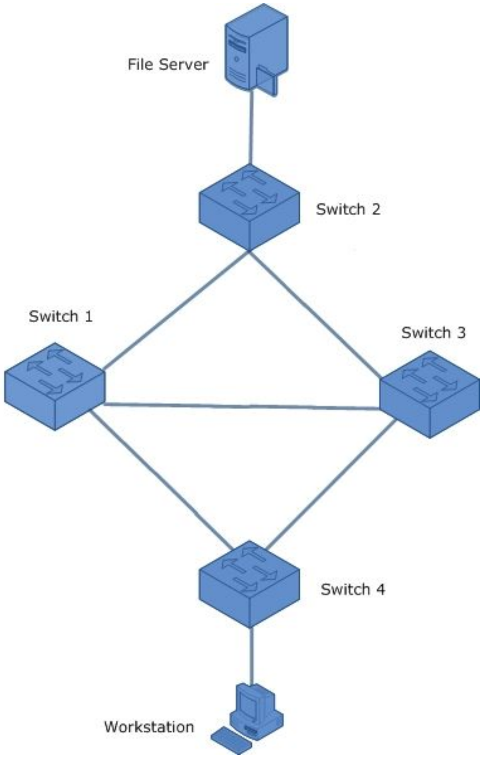
- Switch performs a lookup of its MAC address table to determine whether it knows which port to send the server frame to
- In this case, it does, so it sends the return traffic out only its e2 port (PC), without flooding



MAC address table (switch)

MAC address	Port
00:00:00:00:00:02	e2
00:00:00:00:00:01	e3

Loops



No **TTL** concept exists at Layer 2!

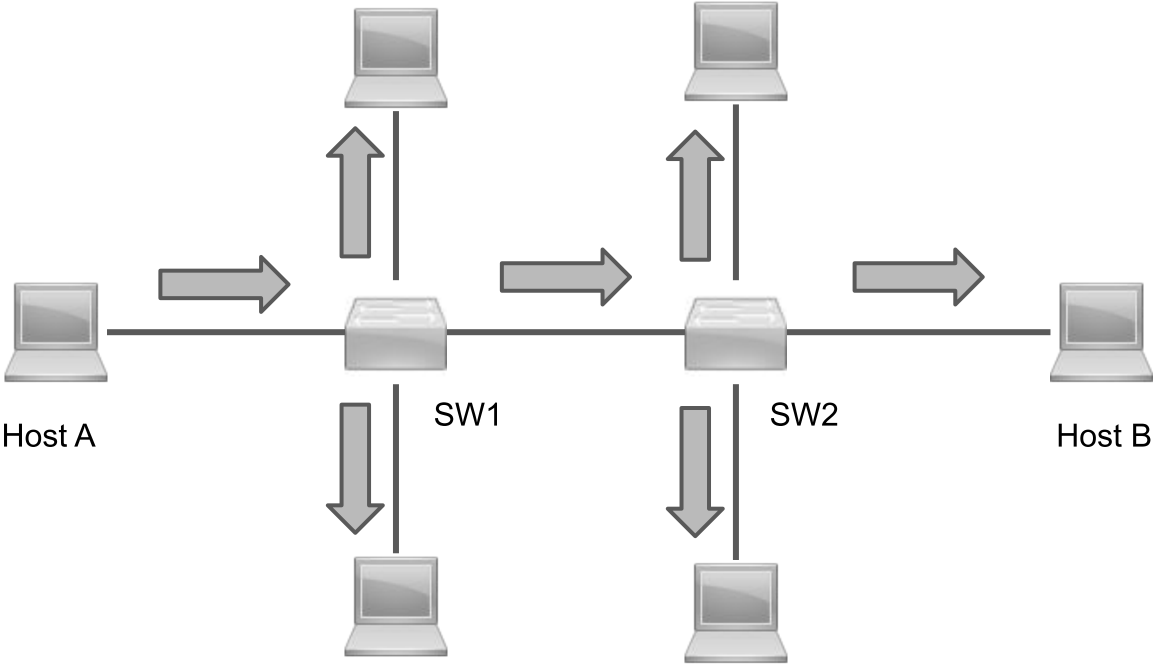
Virtual LANs (VLAN)

- are logical grouping of devices in the same broadcast domain
- are usually configured on switches by placing some interfaces into one broadcast domain and some interfaces into another
- can be spread across multiple switches, with each VLAN being treated as its own subnet or broadcast domain
- frames broadcasted onto the network will be switched only between the ports within the same VLAN.

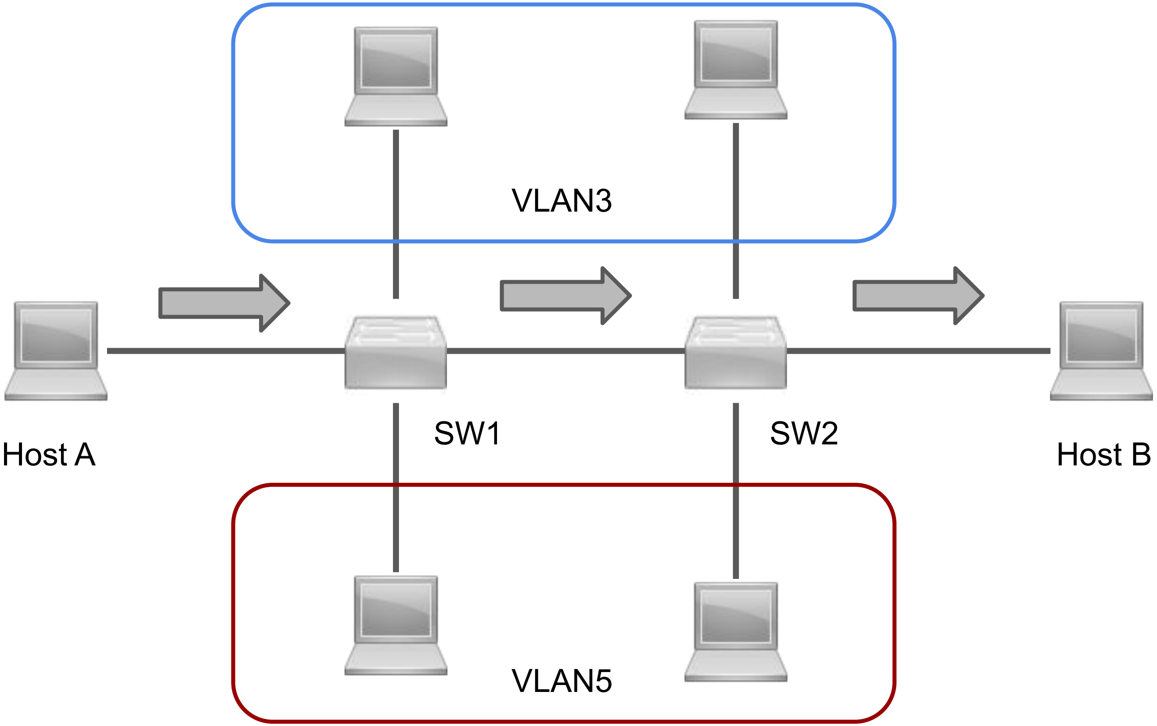
Advantages of VLANs

- increase the number of broadcast domains while decreasing their size
- reduce security risks by reducing the number of hosts that receive copies of frames that the switches flood
- you can keep hosts that hold sensitive data on a separate VLAN to improve security
- you can create more flexible network designs that group users by department instead of by physical location
- network changes are achieved with ease by just configuring a port into the appropriate VLAN

Single VLAN topology

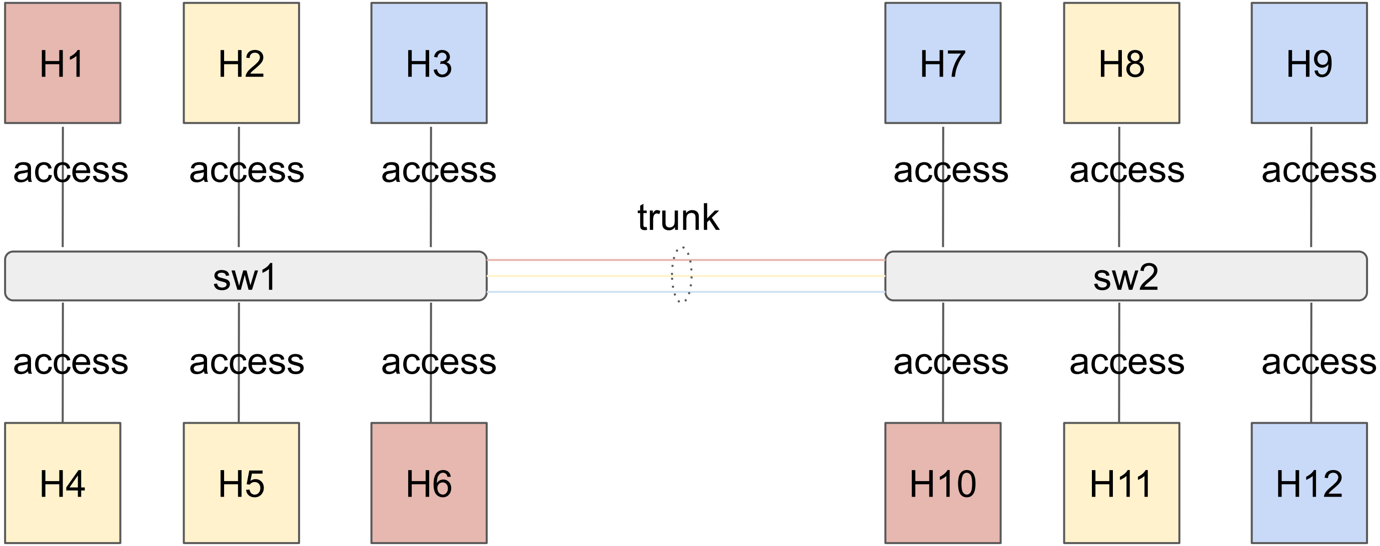


Multiple VLAN topology



Layer 2: VLAN (example)

VID	sw1	sw2
10	H1,H6	H10
20	H2,H4,H5	H8,H11
30	H3	H7,H9,H12



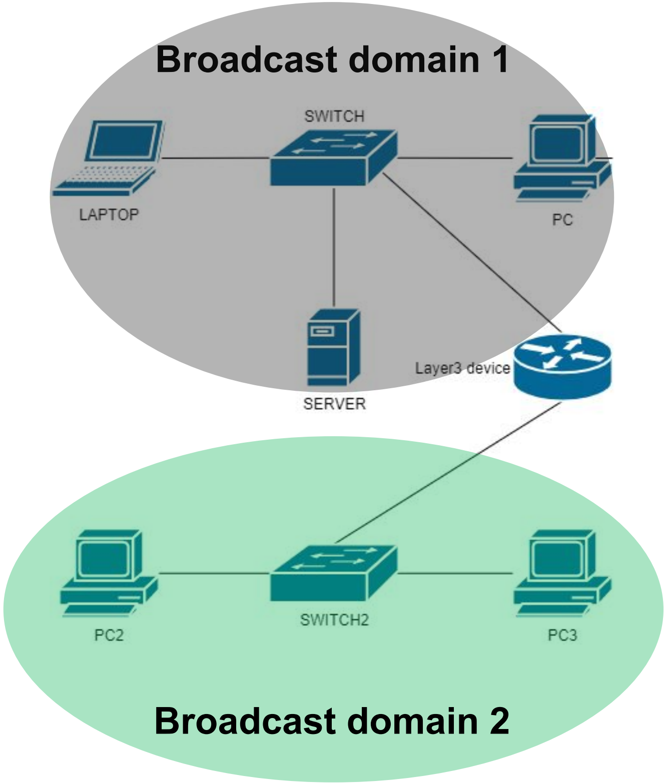
Access/Trunk port

Each port on a switch can be configured as either an access or a trunk port

- an **access port** is a port that can be assigned to a single VLAN. This type of interface is configured on switch ports that are connected to devices with a normal network card, for example a host on a network
- a **trunk interface** is an interface that is connected to another switch. This type of interface can carry traffic of multiple VLANs

Broadcast domains

- Switching relies on broadcasts.
- All network nodes that can be reached at Layer 2 share the same **broadcast domain**.
- Layer 3 devices form boundaries between these domains.



Internet Protocol (IP)

The most significant protocol at layer 3 is the *Internet Protocol* or IP

- The standard for routing packets across interconnected networks (hence, the name internet)
- Encapsulate data and pass that data in the form of *packets*

IP addressing

- An Internet Protocol address is also known as an **IP address**.
- A numerical label which assigned to each device connected to a computer network that uses the IP for communication.
- Two versions: IPv4 and IPv6
  - IPv6 is the new version that is being deployed to fulfill the need for more Internet addresses.
  - In this module, we focus on IPv4 (currently the most widely used).

IP addressing

- IPv4 address
  - 32 bits
  - Grouped 8 bits at a time (octet)
  - Each of the four octets is separated by a dot and represented in decimal format (dotted decimal notation)

11000000 10101000 01100100 11001000

192 . 168 . 100 . 200

IP addressing - Home addressing

2^{32}

IPv4
available
addresses

IP addressing - Home addressing

192.168.100.200
(host address)

35
(house number)

2^{32}

IPv4
available
addresses

IP addressing - Home addressing

192.168.100.200
(host address)

35
(house number)

via Dodecaneso
(street name)

2^{32}

IPv4
available
addresses

IP addressing - Home addressing

192.168.100.200
(host address)

192.168.100
(network address)

2^{32}

IPv4
available
addresses

35
(house number)

via Dodecaneso
(street name)

IP address and Netmask

- An IP address has two components: a *network* component (street name), and a *host* component (house number)
- The purpose of the netmask is to split the IP address into the two components
- When you combine, using a logical AND, the IP address and the netmask you reveal the network component

11000000 10101000 01100100 11001000 address
192 . 168 . 100 . 200
11111111 11111111 111111111 00000000 netmask (**/24**)
255 . 255 . 255 . 0
network 11000000 10101000 01100100 00000000 host
192 . 168 . 100

Reserved IP addresses

- In every network, two addresses are used for special purposes. <u>These addresses are not available for nodes</u>
- **Network address**: is the first address in the network (all the host bits are 0) and it is used for identifying the network (always even)
- **Broadcast address**: is the last address in the network (all the host bits are 1). An IP packet having the broadcast address as the destination address is sent to all nodes of the IP network (always odd)

11000000 10101000 01100100 11001000 address

192 . 168 . 100 . 200

11111111 11111111 111111111 00000000 netmask (**/24**)

255 . 255 . 255 . 0

11000000 10101000 01100100 **00000000** network addr.

192 . 168 . 100 . 0

11000000 10101000 01100100 **11111111** broadcast addr.

192 . 168 . 100 . 255

Default Netmasks

- Default netmasks have all ones (255) or all zeroes (0) in an octet

Address Class	Total # Of Bits For Network ID / Host ID	Default Subnet Mask			
Class A	8/24	255	0	0	0
Class B	16/16	255	255	0	0
Class C	24/8	255	255	255	0

Non-default Netmasks (example)

- 192.168.100.x/**25**, 7 bits for hosts ⇒ 126 addresses + network addr. + bcast addr.
- **first** network: 192.168.100.0-127
  - 192.168.100.0: network address
  - 192.168.100.1: first host
  - 192.168.100.126: last host
  - 192.168.100.127: broadcast address
- **second** network: 192.168.100.128-255
  - 192.168.100.128: network address
  - 192.168.100.129: first host
  - 192.168.100.254: last host
  - 192.168.100.255: broadcast address

```
  192    .    168    .    100
11000000 10101000 01100100 00000000
11000000 10101000 01100100 00000001
11000000 10101000 01100100 01111110
11000000 10101000 01100100 01111111

11000000 10101000 01100100 10000000
11000000 10101000 01100100 10000001
11000000 10101000 01100100 11111110
11000000 10101000 01100100 11111111
```

Private IP addresses

Private IP addresses are **not routed on the Internet,** and traffic cannot be sent to them from the Internet

- They are supposed to work within the local network, only.
  - Range from 10.0.0.0 to 10.255.255.255 — a 10.0.0.0 network with a 255.0.0.0 or an /8 (8-bit) mask
  - Range from 172.16.0.0 to 172.31.255.255 — a 172.16.0.0 network with a 255.240.0.0 (or a 12-bit) mask
  - A 192.168.0.0 to 192.168.255.255 range, which is a 192.168.0.0 network masked by 255.255.0.0 or /16

Neighbor Table and Address Resolution Protocol (ARP)

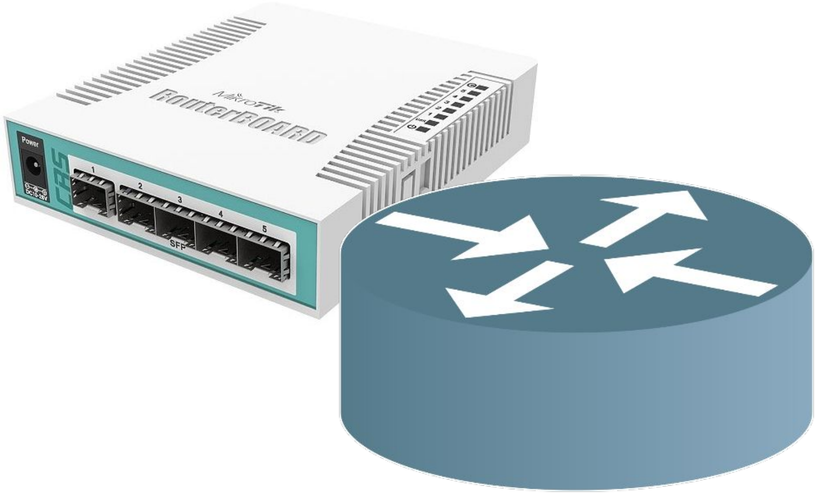
- An IP node wants to communicate with a system in the same layer 2 domain
  - It looks in its neighbor table, or **ARP table** (IP → MAC), to determine how to construct the Ethernet frame.
  - If the desired destination IP address is not in the ARP table, the node issues an ARP request, which is **broadcast** to everyone in the layer 2 domain, that asks “*Please tell me the MAC address for the node with IP address X.X.X.X.*”.
  - Assuming the target device is available, the node with that IP address will respond.
  - An ARP request for a non-existing host takes a fixed number of retries (after a timeout) before concluding that the host isn't reachable.

IP Routing

- IP routing is the process of sending packets from a host on one network to another host <u>on a different remote network</u>
    - Nodes examine the destination IP address of a packet, determine the next-hop address, and forward the packet
    - Nodes use **routing tables** to determine a next hop address to which the packet should be forwarded

Router

- A router is the Layer 3 device that forwards data packets between computer networks.
- A router is connected to two or more data lines from different IP networks.



Internetworking: Routing Table

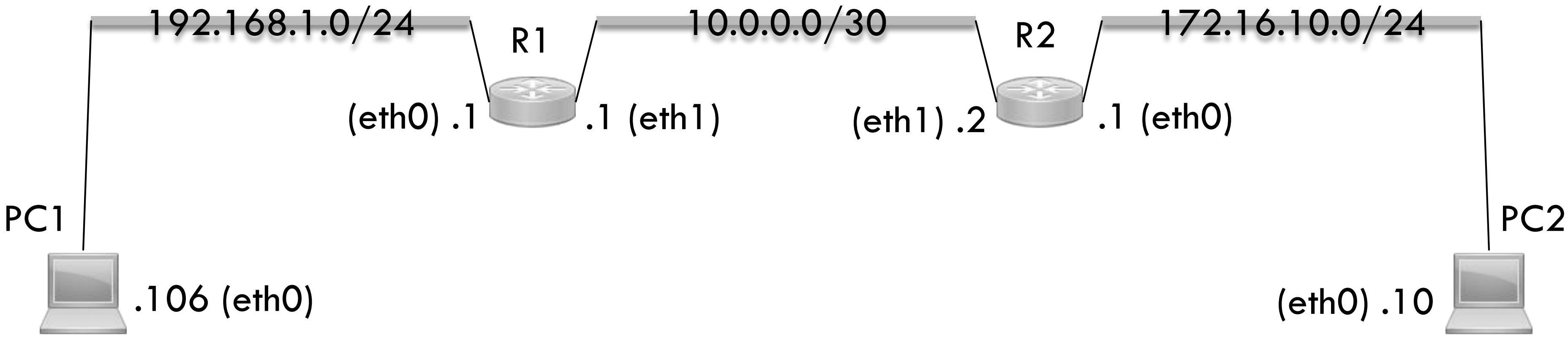
A routing table is used by nodes to determine the path to the destination network

- Each routing table consists, at least, of the following entries:
  - **Network destination and subnet mask** – specifies a range of IP addresses
  - **Remote router** – IP address of the router used to reach that network
  - **Outgoing interface** – outgoing interface the packet should go out to reach the destination network

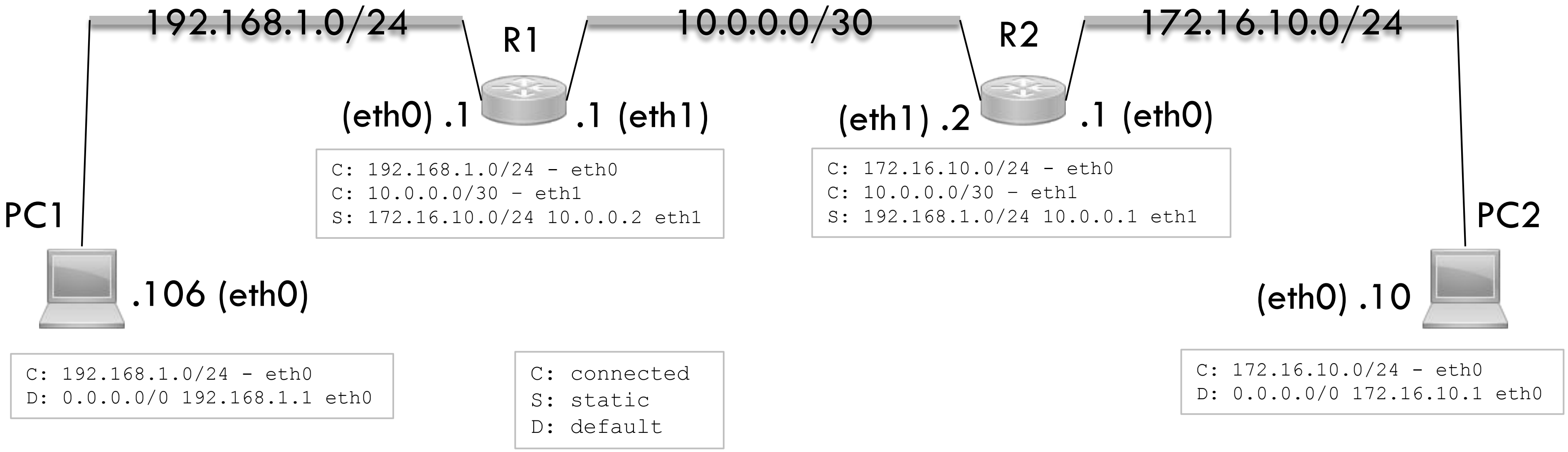
Connected, static and default routes

- Routing table entries can originate from the following sources:
  - **connected**: subnets directly connected to a node's interface are added to the node's routing table (interface has to have an IP address configured and must be in the up state)
  - **static**: by adding static routes, a node can learn a route to a remote network that is not directly connected to one of its interfaces. Static routes are configured manually specifying DESTINATION_NETWORK SUBNET_MASK NEXT_HOP_IP_ADDRESS
  - **default**: a forwarding rule for packets when no specific address of a next-hop host is available from the routing table

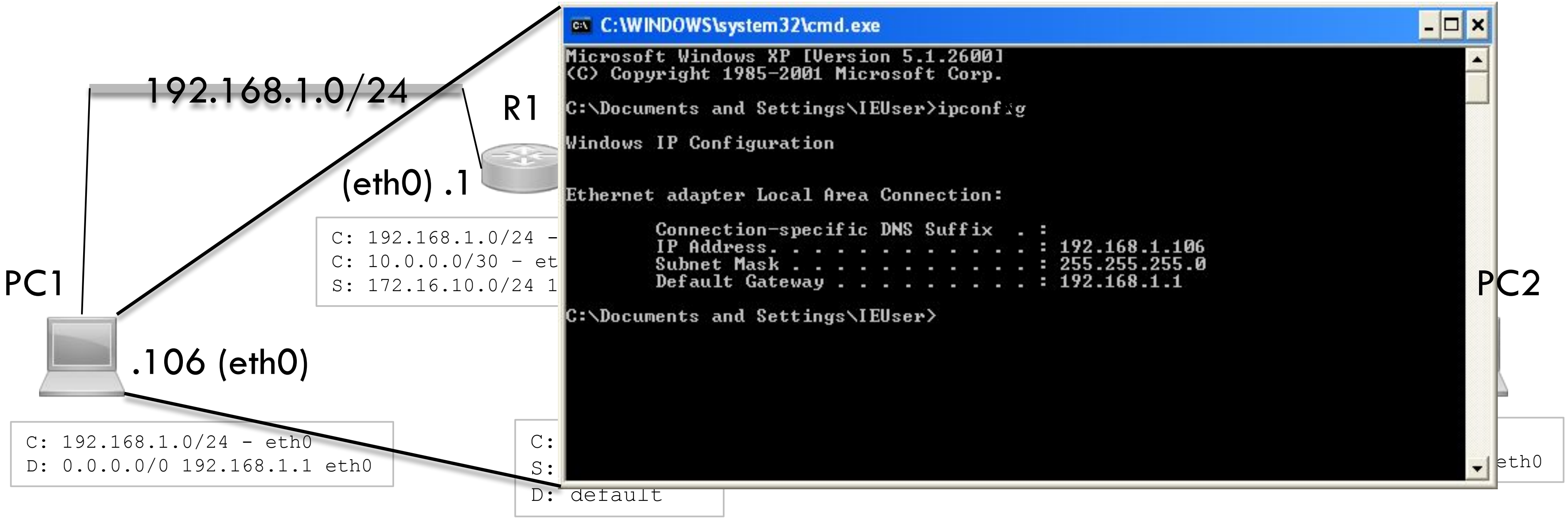
Routing tables (example)



Routing tables (example)



Routing tables (example)



Internet Control Message Protocol (ICMP)

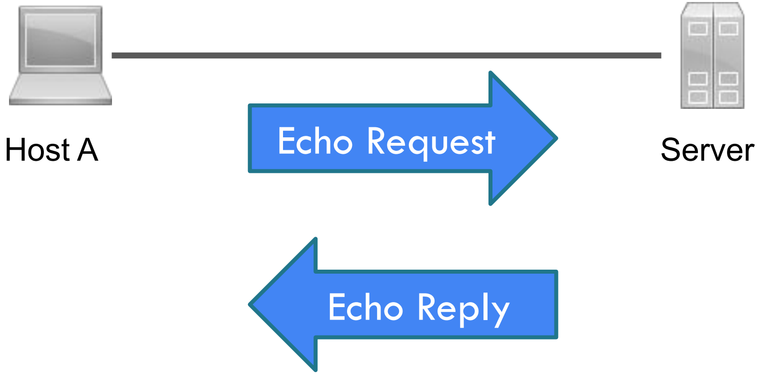
A network layer protocol that reports errors and provides information related to IP packet processing

- is used by network devices to send error messages indicating, for example, that a requested service is not available or that a host isn't reachable
- ICMP messages are encapsulated in IP datagrams, which means that they don't use higher level protocols (such as TCP or UDP) for transmission

ICMP - Ping

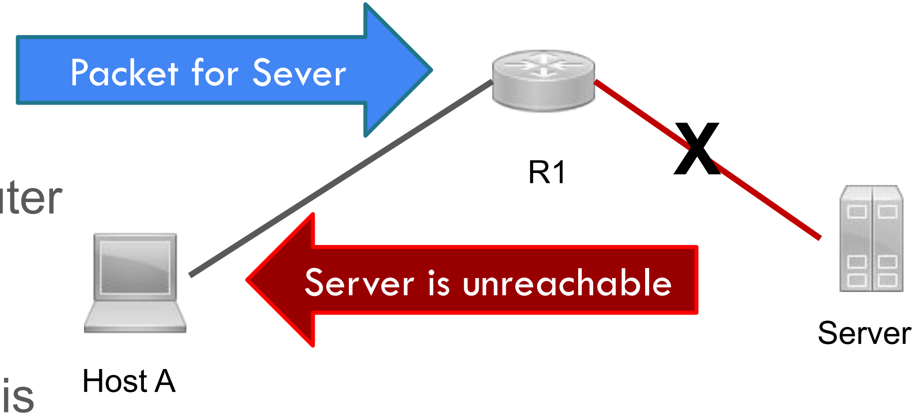
Host A wants to test whether it can reach Server over the network.

- Host A will start the ping utility that will send ICMP Echo Request packets to Server.
- If Server is reachable, it will respond with ICMP Echo Reply packets.
- If Host A receives no response from Server, there might be a problem on the network.



ICMP - Destination unreachable

- Host A sends a packet to Host B
- Because the Host B is down, the router will send an ICMP Destination host unreachable message to Host A, informing it that the destination host is unreachable



ICMP - Traceroute

Traceroute is a command-line interface based tool used to identify the path used by a packet to reach its target.Traceroute sends a series of ICMP echo request packets to a destination.

- First series of messages has a Time to Live (TTL) parameter set to 1, which means that the first router in a path will discard the packet and send an ICMP Time Exceeded message.
- TTL is then increased by one until the destination host is reached and an ICMP echo reply message is received.
- Originating host can then use received ICMP messages to identify all routers in a path.

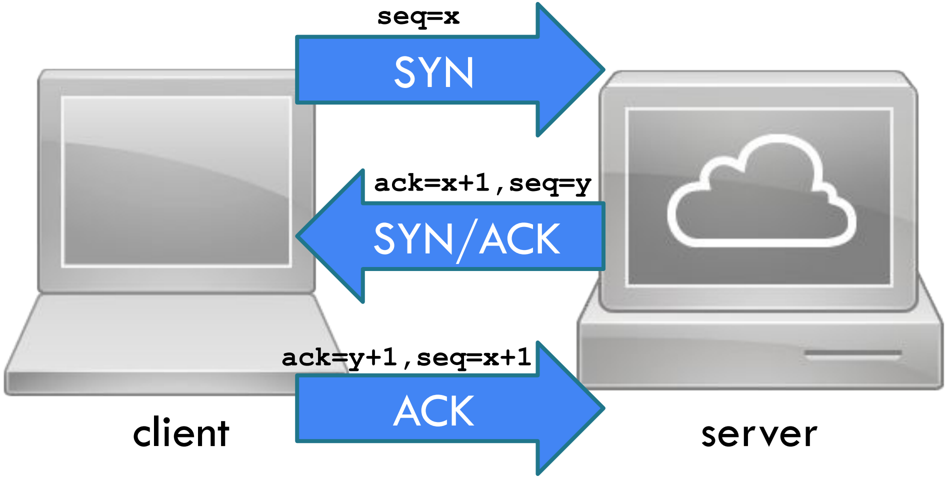
TCP vs UDP

TCP and UDP are the most common Layer 4 protocols

TCP	UDP
creates a connection	connectionless
error checking (checksums)	error checking (checksums)
error recovery	-
rearranges data packets in the original order	-
-	small packet header/overhead
require all transmitted data to arrive	“lossy” applications (e.g., streaming audio and video)

Three-Way Handshake

- TCP uses a ***three-way handshake*** to **establish** a reliable **connection**
- The use of **sequence** (seq) and **acknowledgment** (ack) numbers allows both sides to **detect** missing or **out-of-order segments**



Layer 4 addressing: ports

- Layer 4 is in charge of the **process-to-process** communication. Transmitter and receiver are identified using **ports**
  - 16-bit unsigned integer (0-65535, 0 reserved) ***conventionally*** divided into:
    - **Well-known ports** (0-1023): used by **system processes** that provide widely used types of network services (requires **superuser privileges**)
    - **Registered ports** (1024-49151): **assigned** by a central authority (the Internet Assigned Numbers Authority, IANA) for specific services
    - **Ephemeral ports** (49152–65535): contains **dynamic** or **private ports** that cannot be registered with IANA

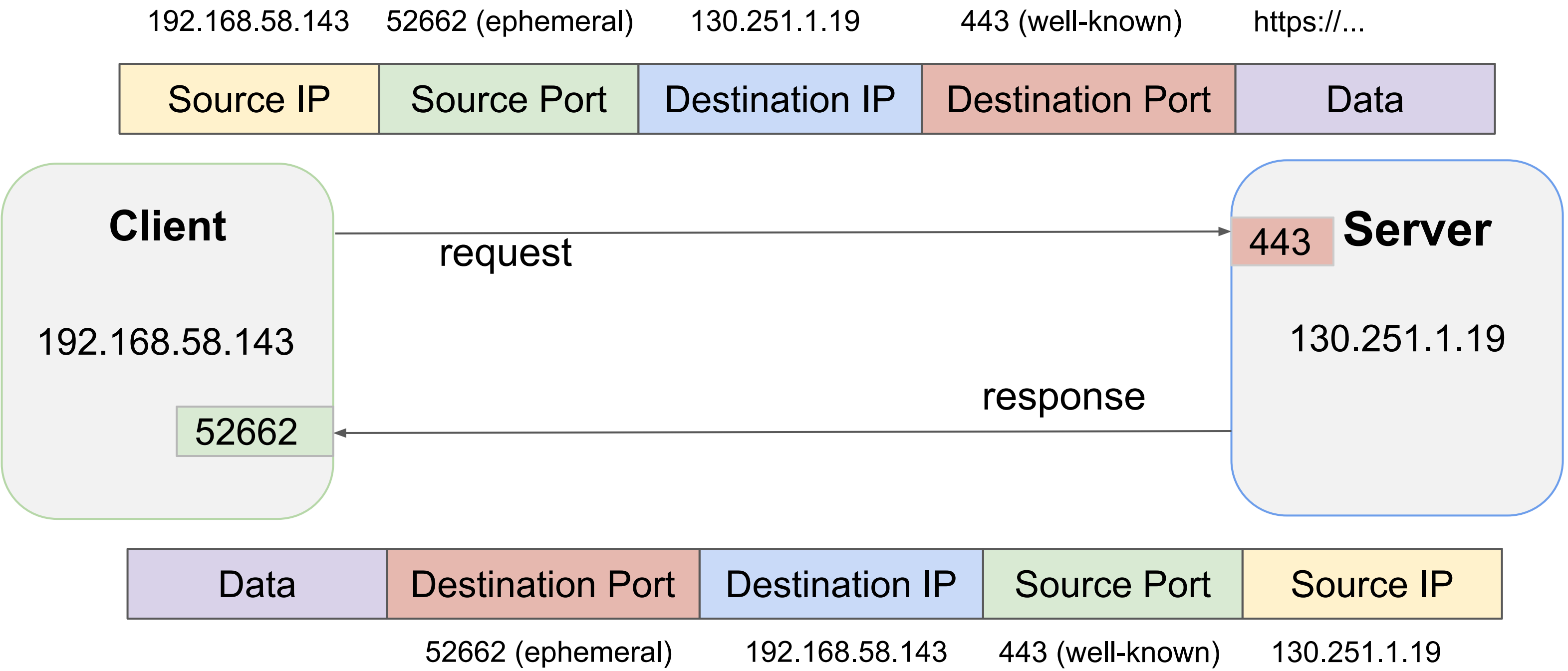
Layer 4 addressing: ports

- The use of well-known and registered ports allows the requesting process to easily **locate** the corresponding **server application processes** on other hosts
  - For example, a web browser knows that the web server process listens on port 80/TCP
- Despite these agreements, **any service can listen on any port**
  - For example, a web server process can listen on port 8080/TCP instead of the well-known one

The client-server model

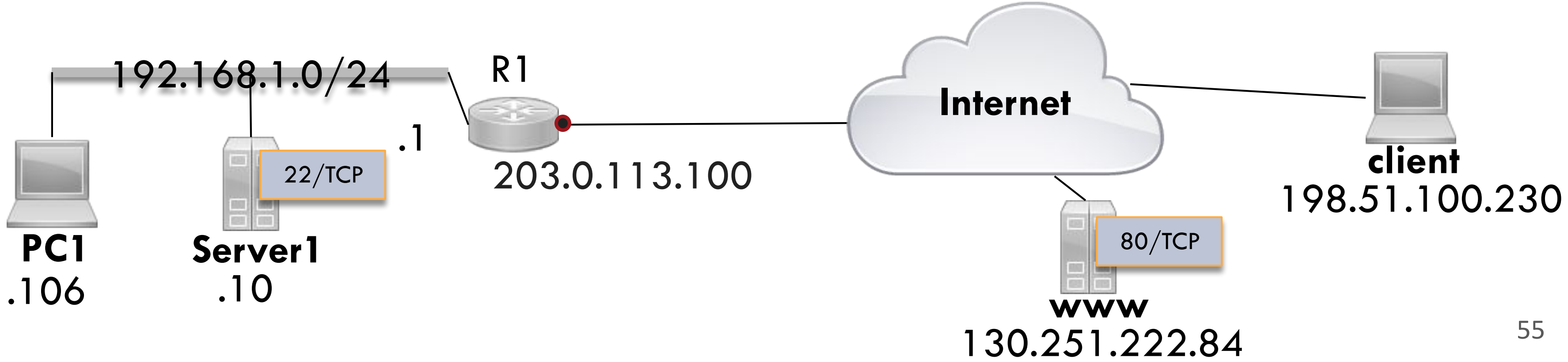
- TCP/IP relies on the **client-server** model for enabling the process communication between network nodes
  - is a relationship in which one program (client) requests a service or resource from another program (server)
  - the **client** needs to know of the existence of and the **address of the server**
  - the **server** does **not need to know the address** of (or even the existence of) the **client** prior to the connection being established

The client-server model (example)



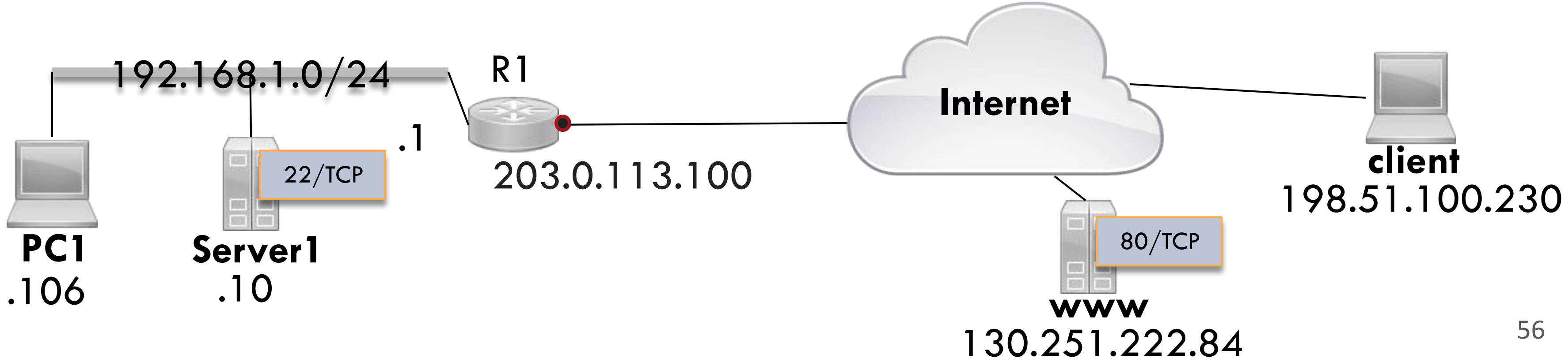
Network Address Translation

- Network Address Translation (NAT) generally involves **rewriting the source** and/or destination addresses of IP packets as they pass through a router or firewall
  - 192.168.1.0/24 is a **private network** and it is **<u>not routable</u>** on the Internet



Source NAT and Masquerade

- Masquerade is a **source NAT** rule, i.e., it is related to the source address of a packet
- The popular usage of NAT Masquerade is to **translate a private address** range to a **single public IP address**



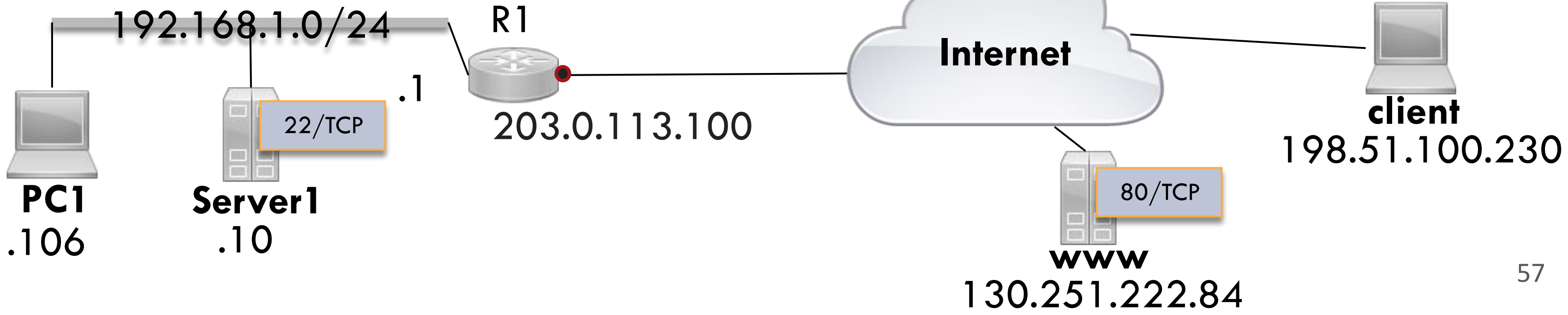
Source NAT and Masquerade (example request)

- PC1 and Server1 accessing www (request)

SNAT table (dynamic)

203.0.113.100	52000,80	192.168.1.106
203.0.113.100	53000,80	192.168.1.10

SRCIP	SRCPORT	DSTIP	DSTPORT		SRCIP	SRCPORT	DSTIP	DSTPORT
192.168.1.106	52000	130.251.222.84	80	R1 →	203.0.113.100	52000	130.251.222.84	80
192.168.1.10	53000	130.251.222.84	80	R1 →	203.0.113.100	53000	130.251.222.84	80



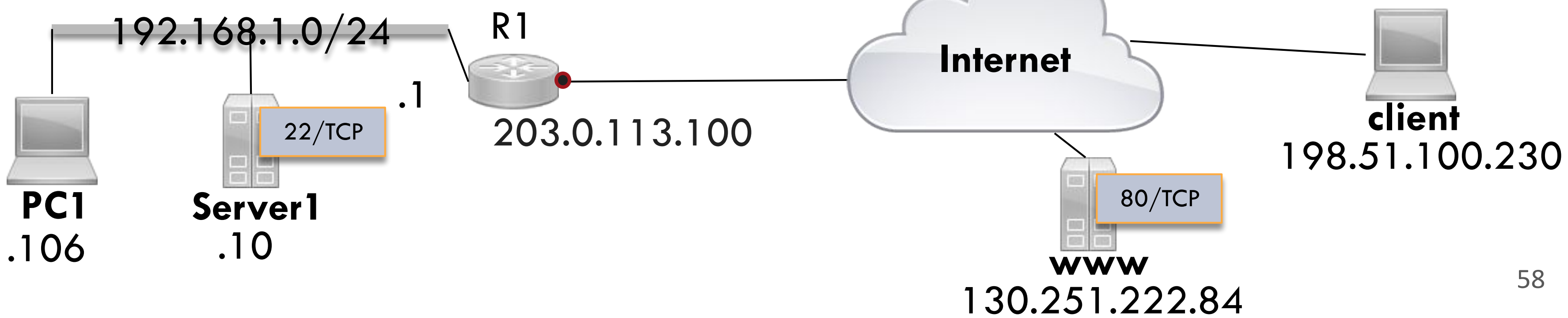
Source NAT and Masquerade (example response)

- PC1 and Server1 accessing www (response)

SNAT table (dynamic)

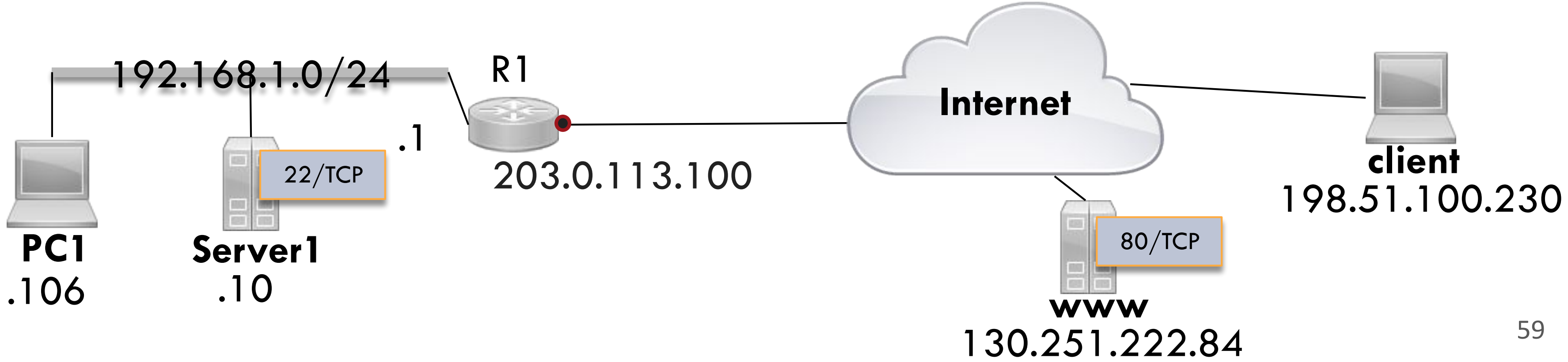
203.0.113.100	52000,80	192.168.1.106
203.0.113.100	53000,80	192.168.1.10

SRCIP	SRCPORT	DSTIP	DSTPORT		SRCIP	SRCPORT	DSTIP	DSTPORT
130.251.222.84	80	192.168.1.106	52000	R1 ←	130.251.222.84	80	203.0.113.100	5200
130.251.222.84	80	192.168.1.10	53000	R1 ←	130.251.222.84	80	203.0.113.100	5300



Port forwarding

- Port forwarding is a **destination NAT** rule, i.e., it is related to the destination address of a packet
- Maps **external IP addresses** and **ports** to **Internal IP addresses** and **ports** allowing access to internal services from the Internet



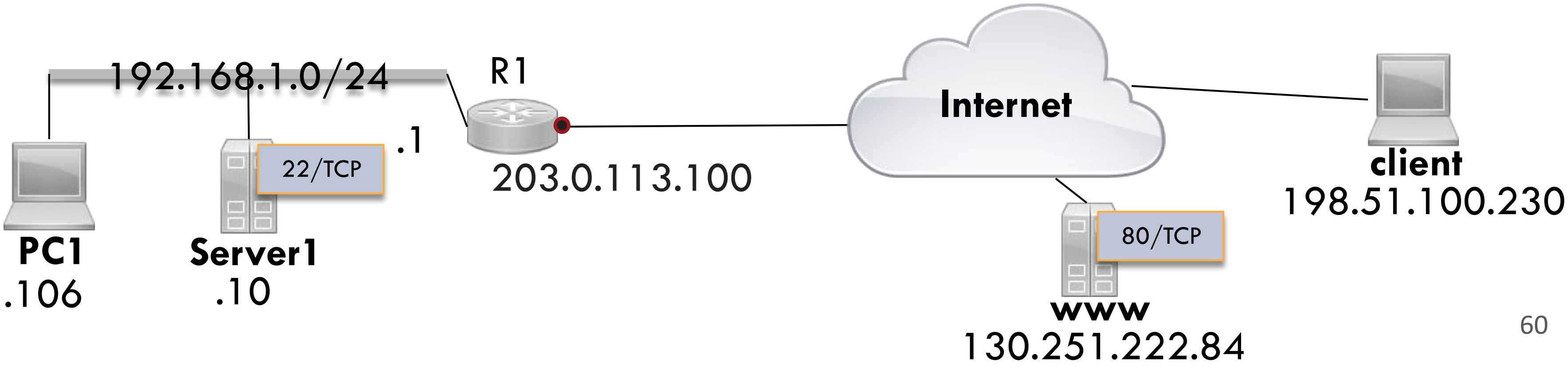
Port forwarding (example request)

- Client connecting to Server1 (request)

DNAT table (static)

Public IP address	Ext. port	Private IP address	Int. Port
203.0.113.100	2222	192.168.1.10	22

SRCIP	SRCPORT	DSTIP	DSTPORT	R1	SRCIP	SRCPORT	DSTIP	DSTPORT
198.51.100.230	54000	192.168.1.10	22	←	198.51.100.230	54000	203.0.113.100	2222



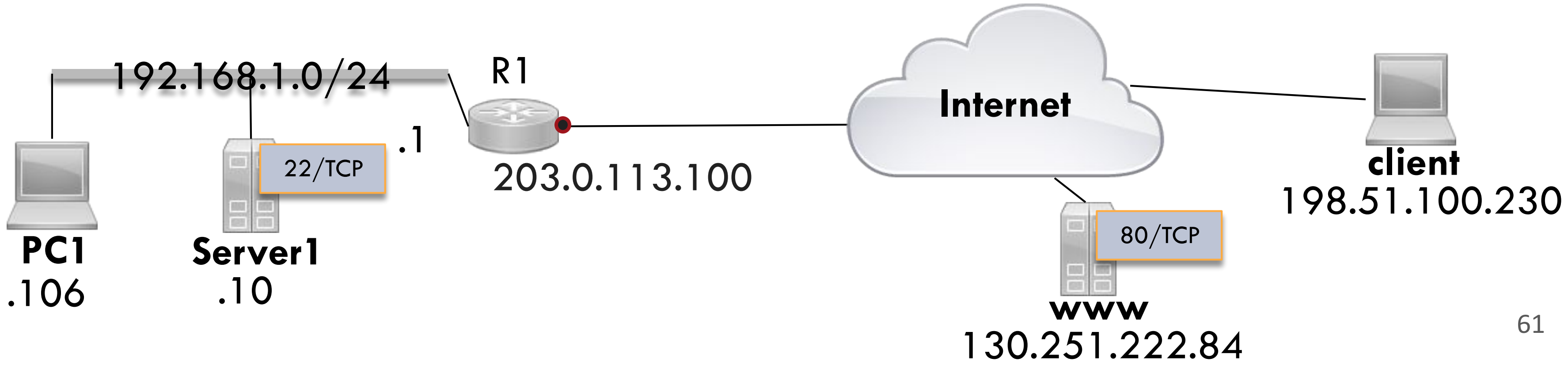
Port forwarding (example response)

- Client connecting to Server1 (response)

DNAT table (static)

Public IP address	Ext. port	Private IP address	Int. Port
203.0.113.100	22	192.168.1.10	22

SRCIP	SRCPORT	DSTIP	DSTPORT	R1	SRCIP	SRCPORT	DSTIP	DSTPORT
192.168.1.10	22	198.51.100.230	54000	→	203.0.113.100	2222	198.51.100.230	54000



Linux and NAT: Netfilter

The Linux kernel has a **packet filter framework** called **netfilter**

- This framework enables a **Linux machine** with an appropriate number of network cards (interfaces) to become a **firewall/router capable of NAT**
- The command utility **iptables** is used to create complex rules for the modification and filtering of packets
- List of rules are named **chains**

Netfilter and NAT

Rules regarding **NAT** are found in the **nat-table**.
This table has two *main* **predefined chains**.

- **PREROUTING**: is responsible for packets that **just arrived** at the network interface (**no routing decision** has taken place).
- **POSTROUTING**: **just before the forwarded packet leaves** the machine it passes the POSTROUTING chain and then leaves through the network interface.

Linux commands: masquerade

- You can control the **masquerade** (source NAT) configuration using the following command.

  ```
  iptables -t nat -A POSTROUTING -o [devicename]  --source
  [sourcenet_ipaddress/netmask] -j MASQUERADE
  ```

  `devicename` represents the **network interface** configured with the **IP address** that the system should use for **masquerading the source** address of each packet. This rules is applied only for the packets having a source address coming from sourcenet.

Linux commands: port forwarding

- You can control the **port forwarding** (destination NAT) using the following command

  ```
  iptables -t nat -A PREROUTING -p [protocol] -d [source_ip] --dport
  [source_port] -j DNAT --to-destination [dest_ip]:[dest_port]
  ```

  For example, the command:

  ```
  iptables -t nat -A PREROUTING -p tcp -d 203.0.113.100 --dport 2222 -j DNAT
  --to-destination 192.168.1.10:22
  ```

  configure the port forwarding so that each TCP connection to the IP address 203.0.113.100 and port 2222 is forwarded to the IP address 192.168.1.10 and port 22