IP Addressing, monitoring and packet analyzing

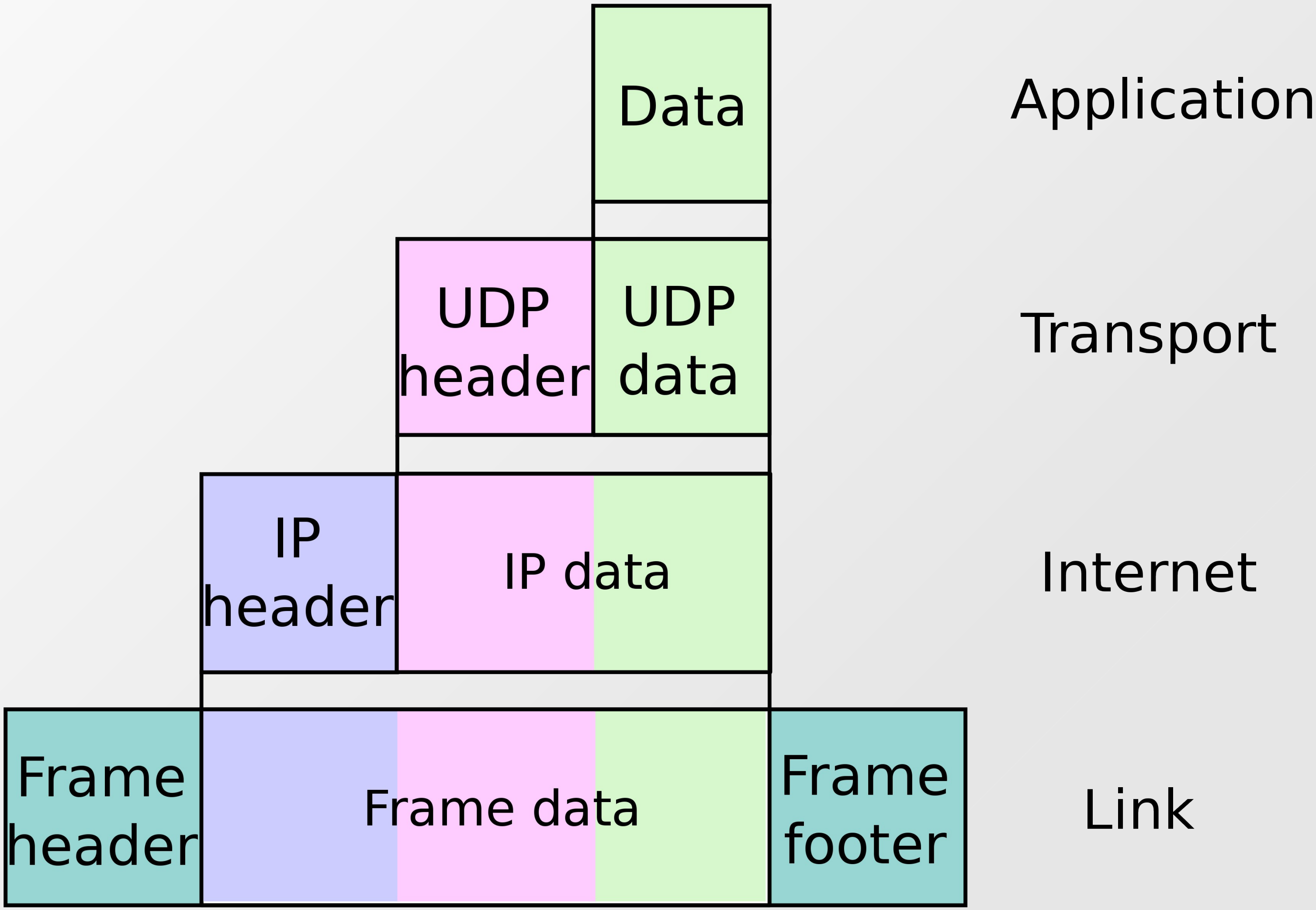
CS-335a

Fall 2012

Computer Science Department

Manolis Surligas
surligas@csd.uoc.gr

TCP/IP stack



Data
Application
UDP header
UDP data
Transport
IP header
IP data
Internet
Frame header
Frame data
Frame footer
Link

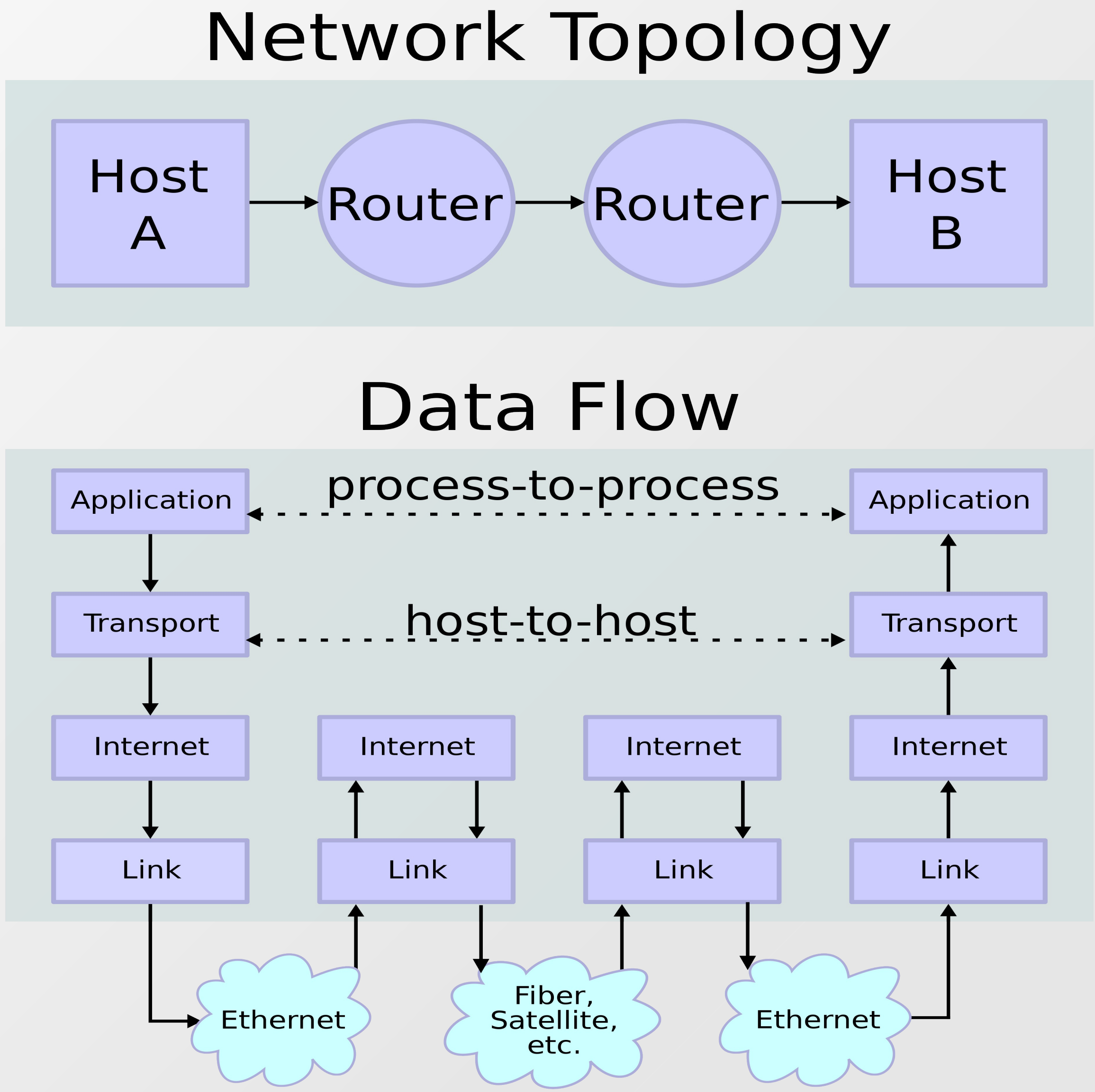
TCP/IP stack

- At sending:
  - Each layer adds information to the data that receives from the higher layer (headers, checksum, etc)
  - Propagates the new data to the next layer
- At receiving:
  - Each layer checks the data that received (headers, checksum, etc)
  - If header and checksum is correct, remove them and propagate the data to the next higher layer
  - Otherwise, packet is dropped

TCP/IP stack

- End-hosts implement all layers
- Intermediate nodes (hubs, switches, routers, etc) implement only some of them
  - Hubs → Physical layer
  - Switches → Physical, Link layer
  - Routers → Physical, Link and Network layer
- Question: Which of the above do you think that performs more complex tasks in a network?

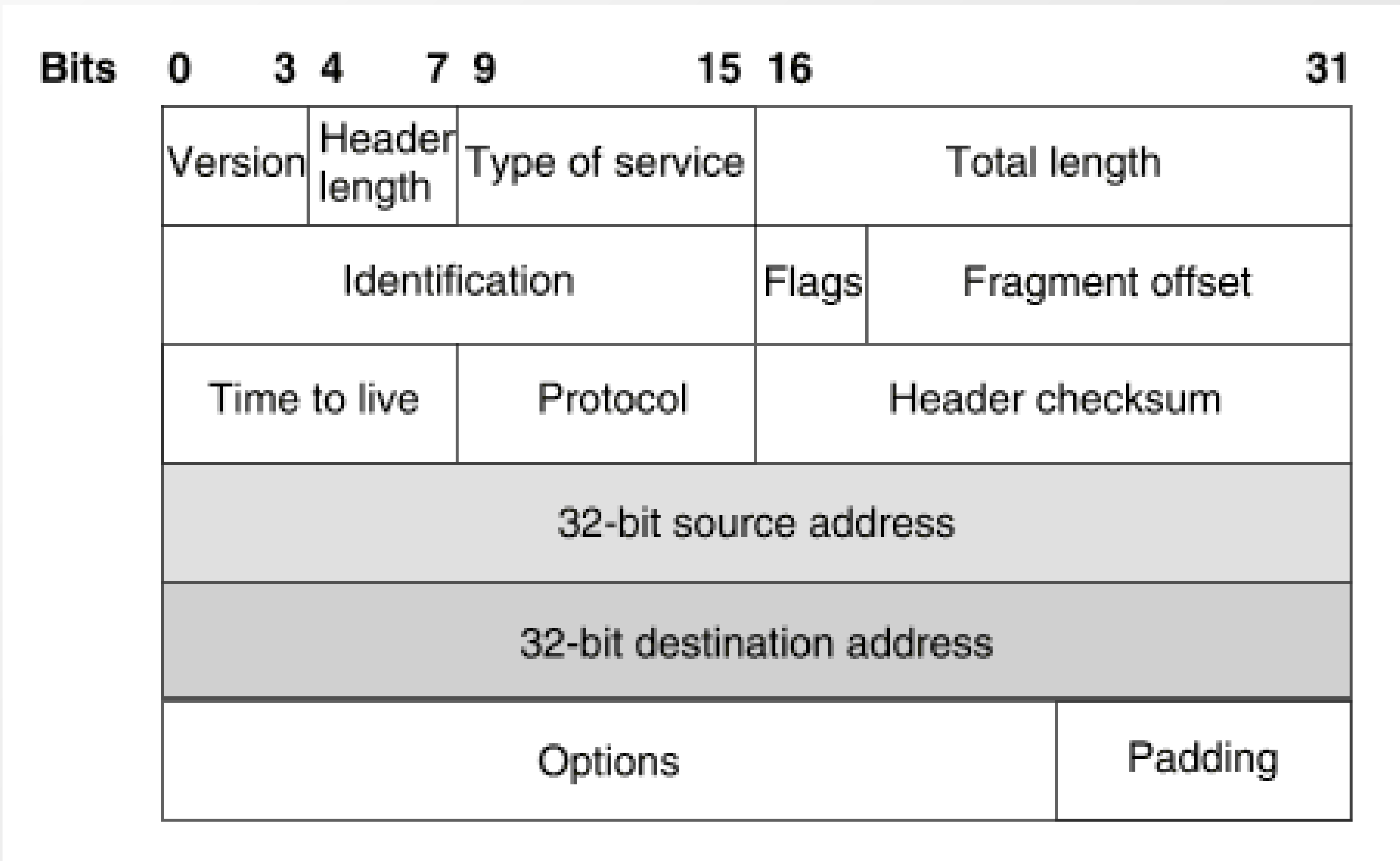
TCP/IP stack



Network Topology
Host A
Router
Router
Host B
Data Flow
process-to-process
host-to-host
Application
Application
Transport
Transport
Internet
Internet
Internet
Internet
Link
Link
Link
Link
Ethernet
Fiber, Satellite, etc.
Ethernet

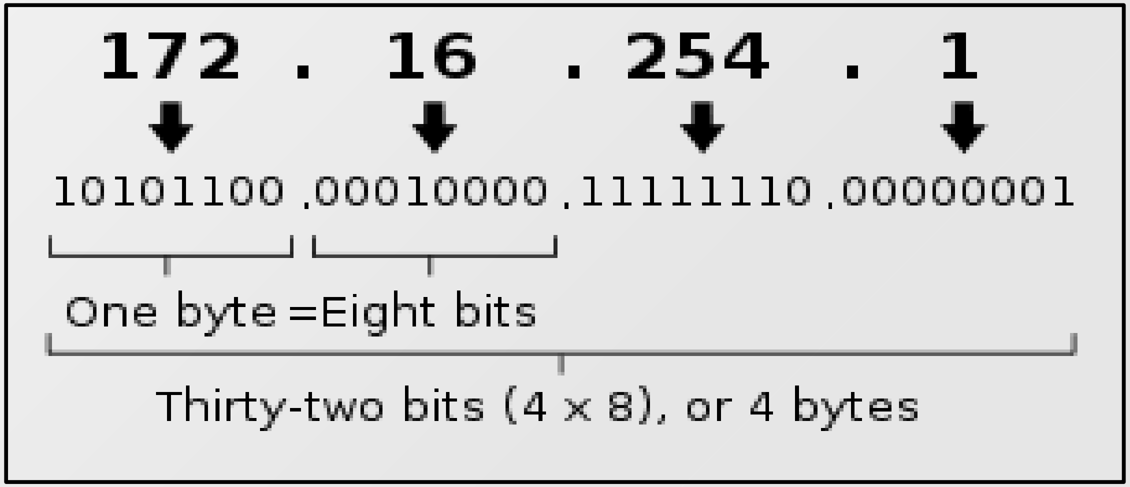
IPv4 Addressing

- Lets first take a look at the IPv4 header



- 32-bit addresses → 2^32 different IP addresses
- Not all of them can be used
- The address space of IP addresses is controlled by a global organization, the IANA (http://www.iana.org)
- IPv4 address assignment can be found at the IANA resource pages
  http://www.iana.org/assignments/ipv4-address-space/ipv4-address-space.xml
- IPv4 addresses are exhausted
- Solution: NAT, IPv6

- In order IP addresses to be easily remembered, the decimal dot-notation is used
- 32 bits are divided into four octets
- We calculate the number of each octet
- After each octet (except the last!) we place a dot (.)
- Eg:



Classless Inter-Domain Routing

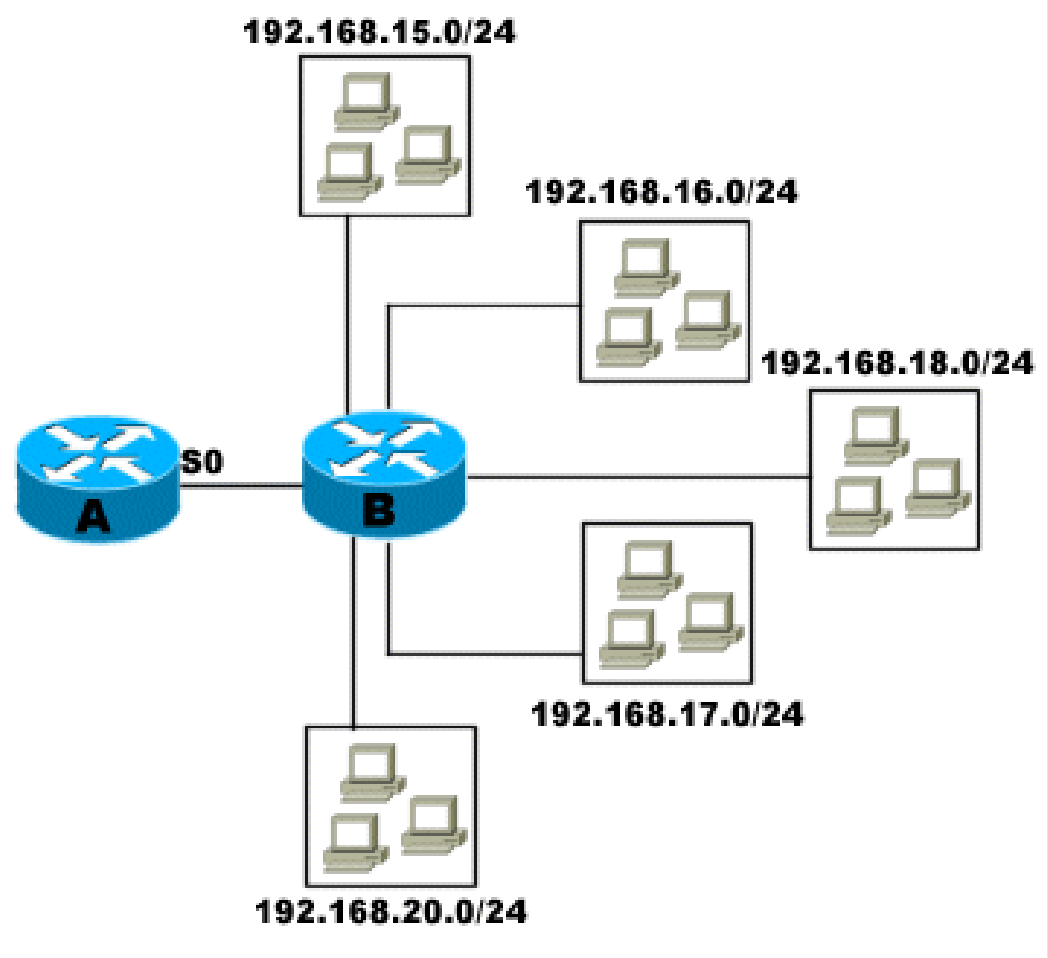
- CIDR → Classless Inter-Domain Routing
- CIDR is a method for allocating IP addresses and routing IP packets efficiently
- An IP address is devided into two parts
  - Most significant bits, are called network address or subnet
  - Lest significant bits are the host identifier
- CIDR notation: 192.168.1.0/24
  - 192.168.1.0 is the network ID
  - /x part indicates the number (x) of the most significant bits

Reserved IP addresses

Range	Description
0.0.0.0/8	Current network (only valid as source address)
10.0.0.0/8	Private network
100.64.0.0/10	Shared Address Space
127.0.0.0/8	Loopback
169.254.0.0/16	Link-local
172.16.0.0/12	Private network
192.0.0.0/24	Reserved (IANA)
192.0.2.0/24	TEST-NET-1, documentation and examples
192.88.99.0/24	IPv6 to IPv4 relay
192.168.0.0/16	Private network
198.18.0.0/15	Network benchmark tests
198.51.100.0/24	TEST-NET-2, documentation and examples
203.0.113.0/24	TEST-NET-3, documentation and examples
224.0.0.0/4	IP multicast (former Class D network)
240.0.0.0/4	Reserved (former Class E network)
255.255.255.255	Broadcast

Supernetting

- A subnet may contain several smaller subnets
- Easier routing, administration and more robust topologies
- Isolation (CSD may have its own smaller subnet rather that be in the "huge" \16 subnet of UoC)



CIDR Examples

- Question 1: 147.52.0.0/16 Which is the network ID? Which is the subnet? Which is the start and the end IP?
- Question 2: A \16 network how many IP addresses may have?
- Question 3: How many \24 subnets can have, a \16 subnet?
- Question 4: A network has a range of IP's from 10.0.0.0 – 10.255.255.255. Which is the network ID and the subnet?

IP Assignment

- IP addresses can be assigned statically
  - Human interaction is needed
  - If the topology or the network change, modifications must be done again

- IP addresses can be also assigned dynamically

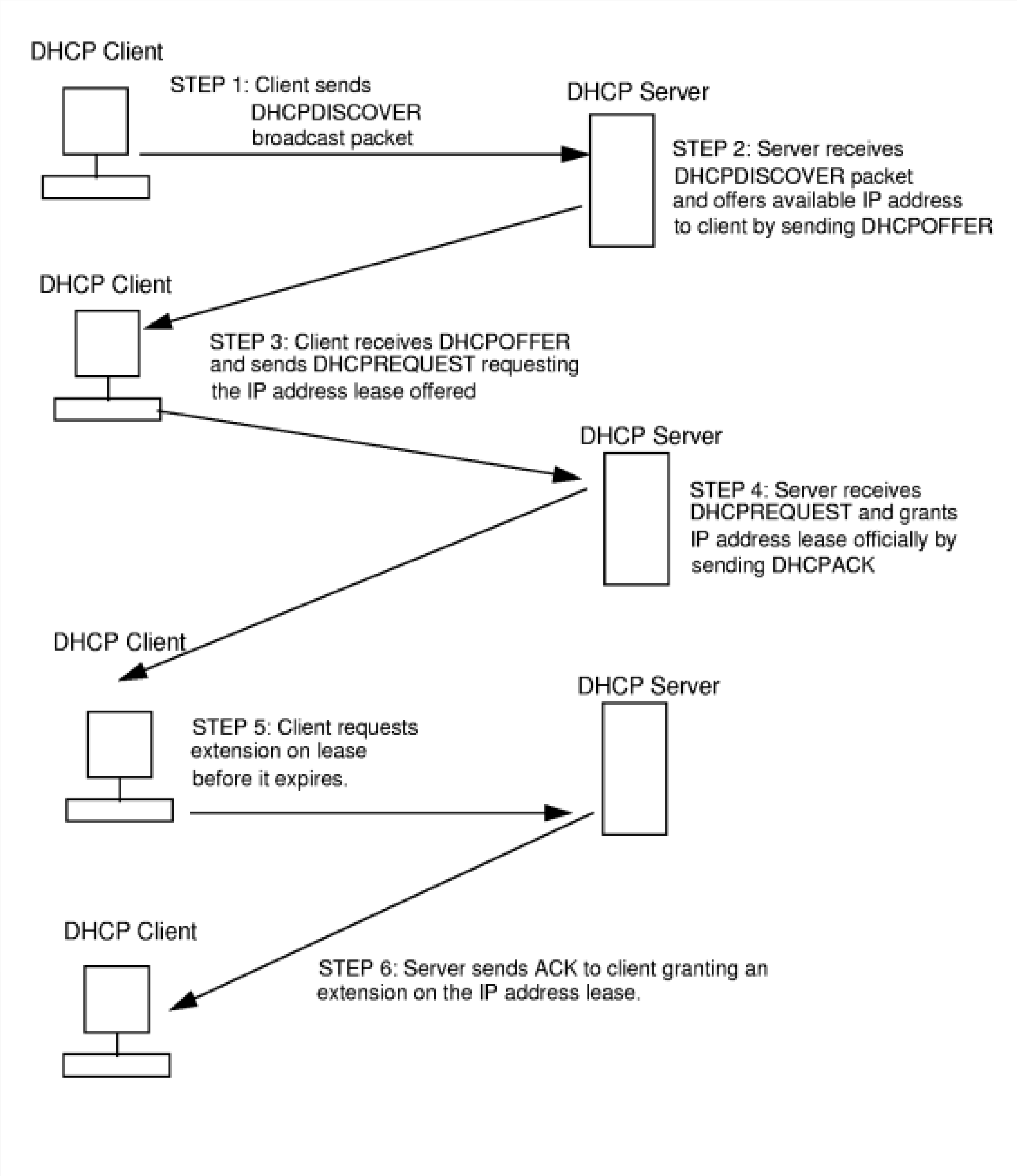
- DHCP (Dynamic Host Configuration Protocol)

- Its a server-client protocol

- Only the server must be configured

- Clients automatically get the proper IP (and other info) from the server

DHCP


DHCP Client
STEP 1: Client sends DHCPDISCOVER broadcast packet
DHCP Server
STEP 2: Server receives DHCPDISCOVER packet and offers available IP address to client by sending DHCPOFFER
DHCP Client
STEP 3: Client receives DHCPOFFER and sends DHCPREQUEST requesting the IP address lease offered
DHCP Server
STEP 4: Server receives DHCPREQUEST and grants IP address lease officially by sending DHCPACK
DHCP Client
DHCP Server
STEP 5: Client requests extension on lease before it expires.
DHCP Client
STEP 6: Server sends ACK to client granting an extension on the IP address lease.

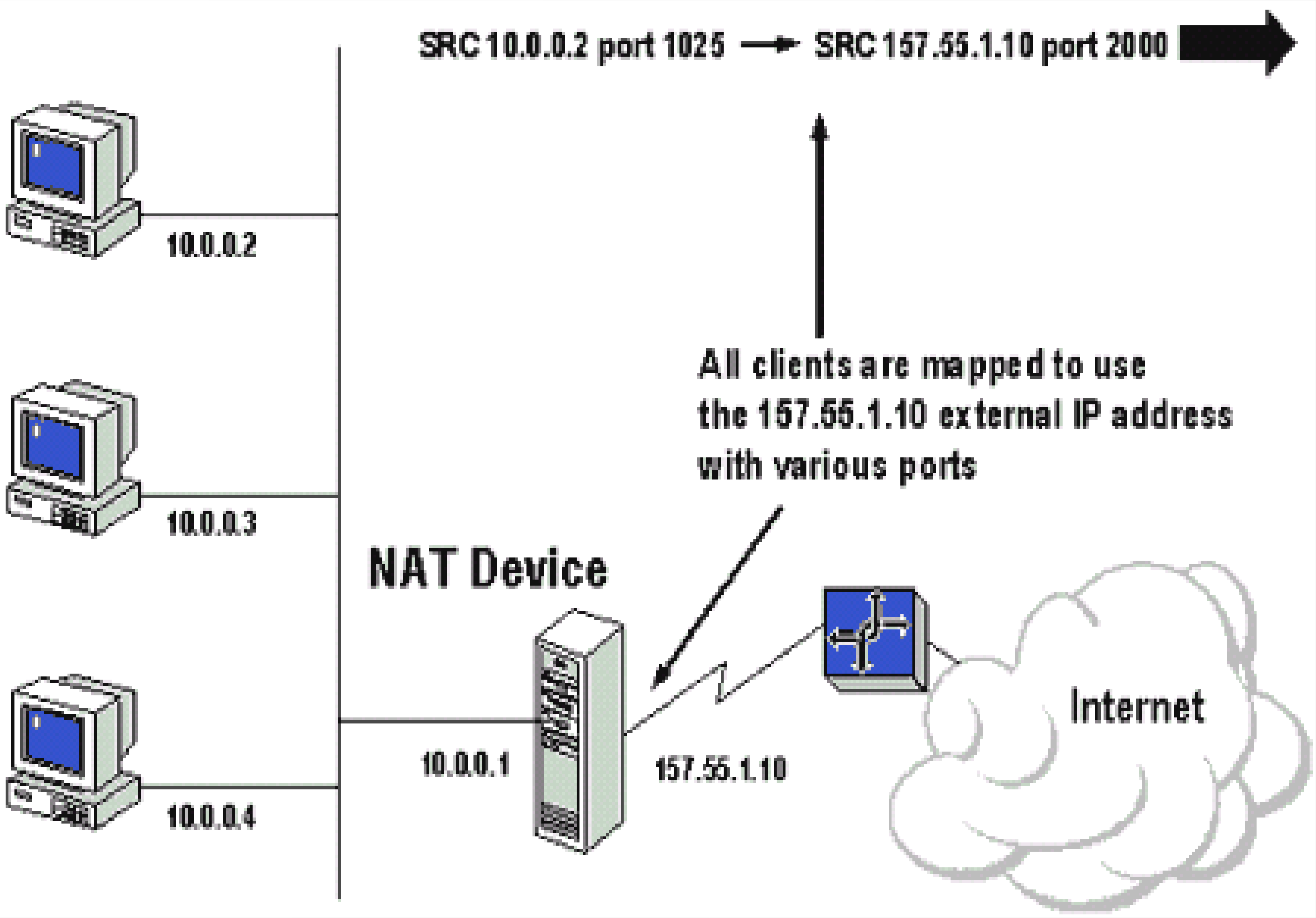
Network address translation (NAT)

- A temporal solution for the IPv4 address space exhaustion

- Used also for isolation and security

- The idea:
  - Use only one IP for the outer world (public IP)
  - Inside the LAN use other IP addresses (private IP)
  - For every connection from a host in the LAN with the outer world use a different combination of the public ip and a port number
  - These combinations are stored in a table at the NAT enabled device
  - At an incoming connection the table can be used to find the LAN's appropriate host

NAT

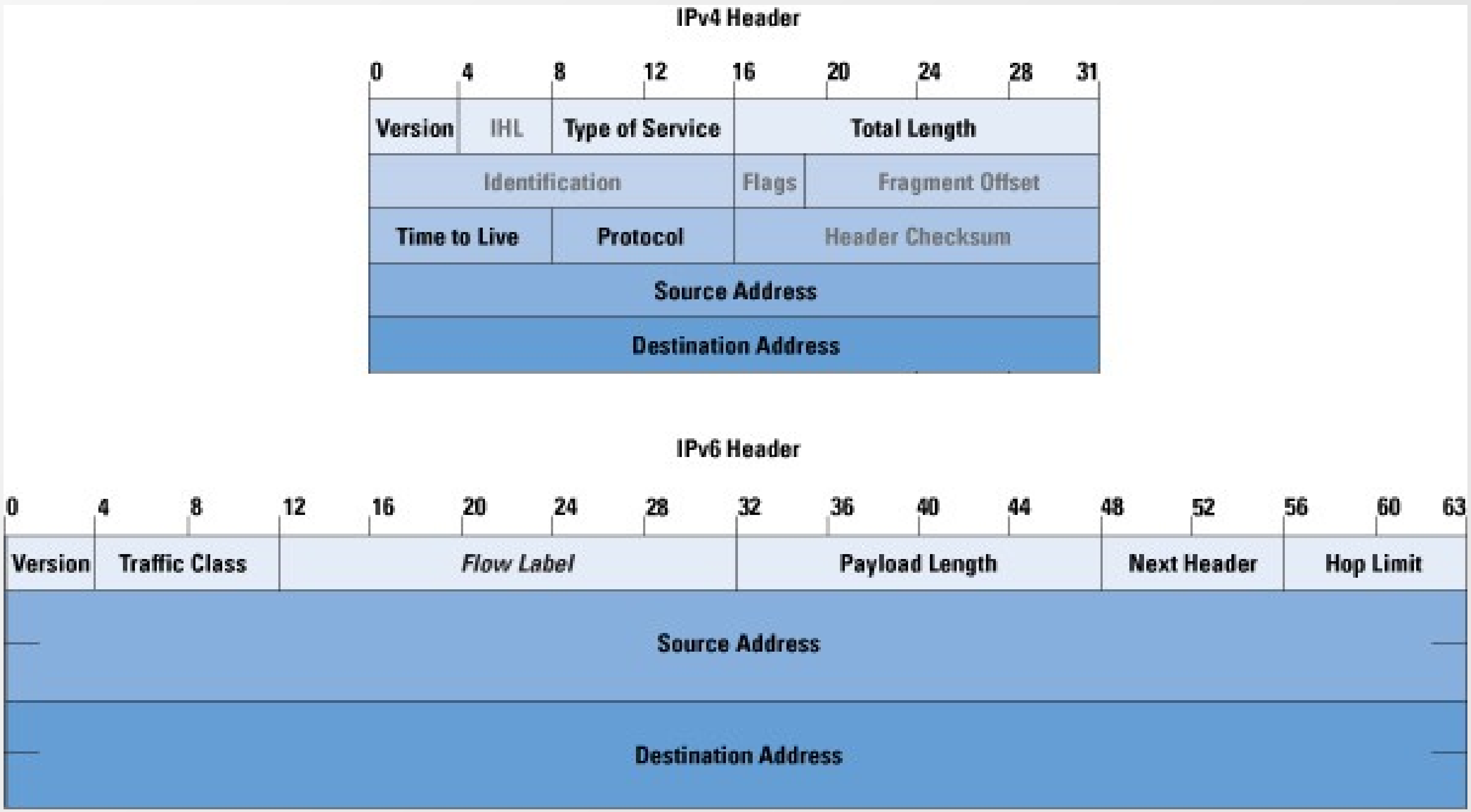

SRC 10.0.0.2 port 1025
SRC 157.55.1.10 port 2000
10.0.0.2
10.0.0.3
10.0.0.4
All clients are mapped to use the 157.55.1.10 external IP address with various ports
NAT Device
10.0.0.1
157.55.1.10
Internet

IPv6

- "An IPv4 address walks into a bar and says: *"Quick, give me a drink. I am exhausted!""*
- "*An IPv6 packet walks into a bar. Nobody talks to him."*
- Several years ago, the IPv4 exhaustion problem was known
- IPv6 was introduced to solve the problem of 32 bits addresses
- Even today only a small part of the ISP, organizations and universities are IPv6 ready

IPv6

- IPv6 uses 128 bit for source and destination address
- 2^128 different addresses!!!



Questions

Packet Monitoring

- Why do I need to monitor packets?
  - Debugging network applications
  - Traffic analyzing
  - Hacking :)
  - Find problematic links
  - Many, many others...

- Tcpdump and wireshark are two famous packet monitors and analyzer tools

Get the software

- Tcpdump can be downloaded from http://www.tcpdump.org for both Windows and Linux

- Most Linux distributions include tcpdump in their standard packages so you do not need to compile it from the source. Just type as **root**:

  - apt-get install tcpdump (Debian based distributions like Ubuntu)

  - zypper install tcpdump (openSuse)

  - yum install tcpdump (Fedora)

Get the software

- Wireshark is a graphical tool for capturing and analyzing easily packets
- Can be downloaded for Windows from http://www.wireshark.org
- Most Linux distros have it on their standard package, so just type as **root**:
  - apt-get install wireshark (Debian based distros)
  - zypper install wireshark (openSuse)
  - yum install wireshark (Fedora)

Linux? Oh noooooooooooooo!

- It is highly recommended to do your projects and your captures on Linux machines
- You can avoid several Windows restrictions
- Powerful command line
- More capabilities with your network interfaces
- If you haven't a Linux OS installed, you can use a Linux Live DVD
- Use BackTrack (comes with most tools pre-installed)

Start capturing packets

- Although Wireshark has the ability to capture packets, it consumes lot of memory
- Better to capture packets with tcpdump, split the trace file in smaller files
- Then analyze easily one by one the smaller files
- With this way we avoid:
  - system and memory getting overload
  - waiting Wireshark to process large files

Start capturing packets

- In a console run:
  tcpdump -i eth0 -s 0 -w filename.pcap
- -i: Specifies the name of the interface in which tcpdump will start capturing packets
  - To list all your available interfaces run:
    ifconfig -a
- -w: Give the name of the file in which the packets should be saved. Should end with .*pcap* extension
- When you are finished press Ctrl+C to stop
- Some systems may need to run these commands as root

Spitting the trace file into smaller

- As we said before it is a good practice to split large traces into smaller
- To do that run:
  tcpdump -r old_file -w new_file -C file_size
  - file_size unit is 1.000.000 bytes (e.g. -C 10 will split the trace file in files with size 10.000.000 bytes)
  - The files that are created have names new_file1, new_file2 etc
- Do that if you trace file has size larger than the 1/4 of your physical memory

Analyzing with Wireshark

- Open Wireshark
- Go File->Open... and select one of your trace files
- You can see the packets that you captured
- If you click on one of them, you can see below more info about it, like its Transfer Protocol or even if the data that contains!!!

Apply filters

- You can apply several filters, in order to categorize your captured packets
- In the Filter field type for example tcp and click apply.
- These should list all the TCP packets of your trace
- Some other filters keywords are: http, arp, udp etc
- You can also specify and combinations (e.g http and arp, tcp and not arp, etc)

More info

- This was the begging. You should experiment a lot by your own
- man tcpdump
- Tcpdump online documentation http://www.tcpdump.org/#documentation
- Use the mailing-list (hy335a-list@csd.uoc.gr) for questions

Questions