Computer Networks
@CS.NCTU

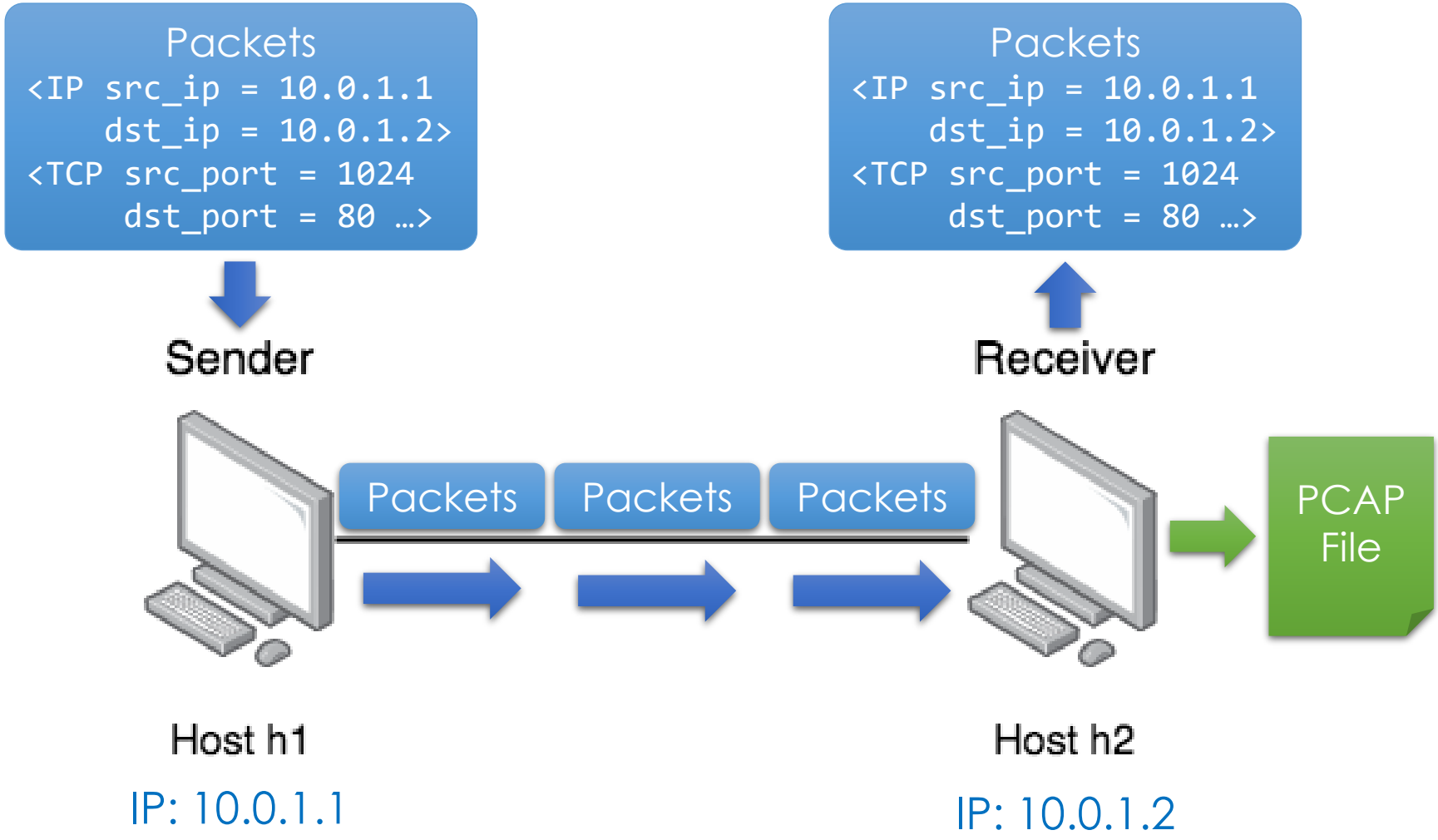
Lab. 1: Packet Manipulation via Scapy

Due Oct 25 (Fri) 23:59

Objectives

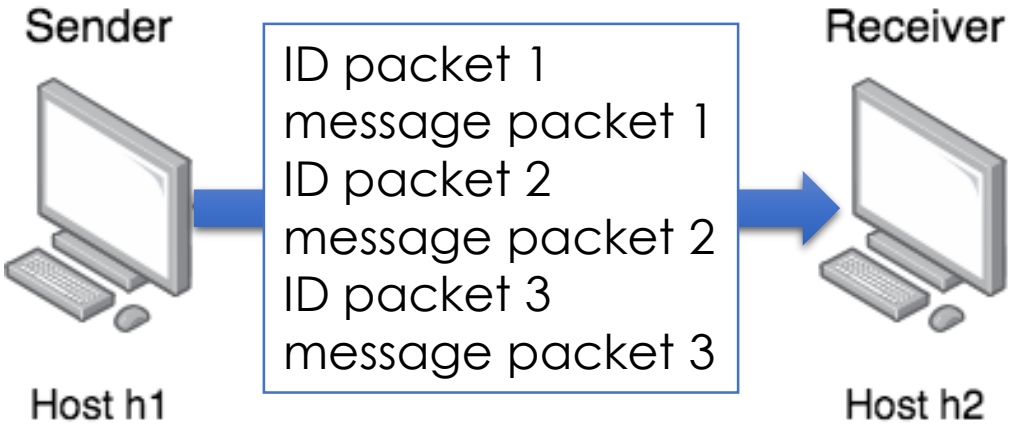
1. Learn how to use **Scapy** to define your own protocol and generate a packet payload
2. Learn how to use **Wireshark** to filter packets and find your wanted information

Overview


Packets
<IP src_ip = 10.0.1.1
dst_ip = 10.0.1.2>
<TCP src_port = 1024
dst_port = 80 …>
Packets
<IP src_ip = 10.0.1.1
dst_ip = 10.0.1.2>
<TCP src_port = 1024
dst_port = 80 …>
Sender
Receiver
Packets
Packets
Packets
PCAP File
Host h1
IP: 10.0.1.1
Host h2
IP: 10.0.1.2

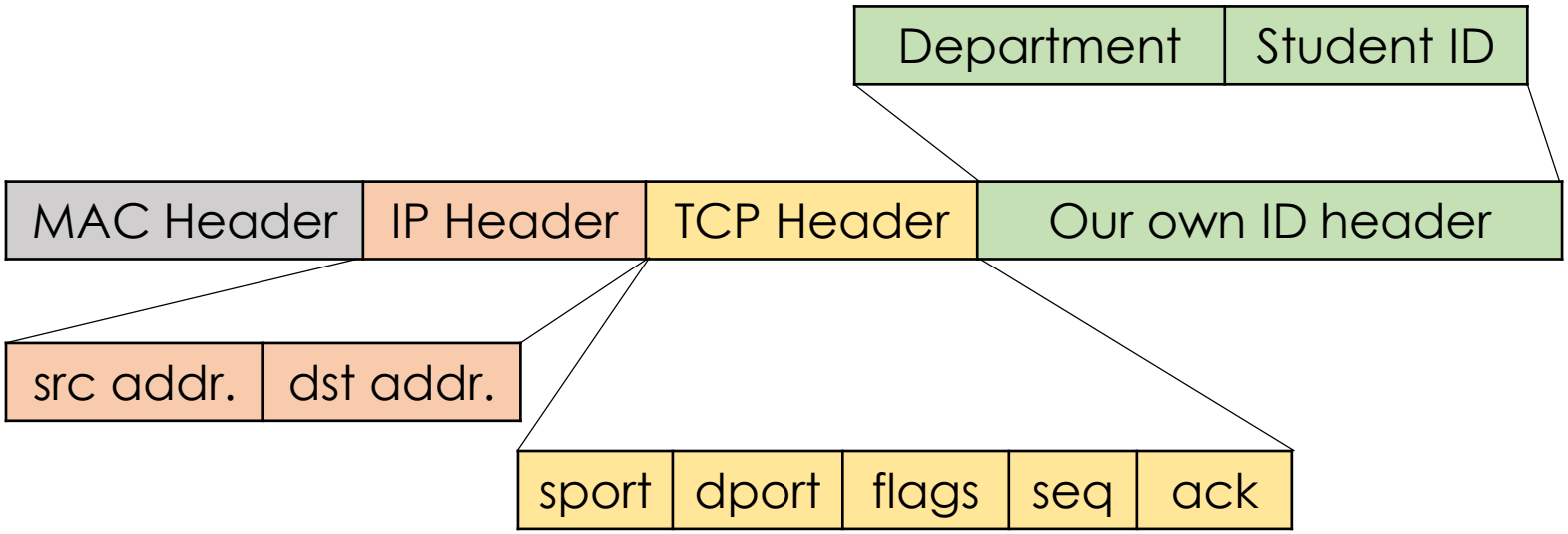
Overview (cont.)

- Define our own proprietary protocol
- In this protocol, we will iteratively send data to a server
  1. **ID packet**:
     your (department + student ID)
  2. **Message packet**:
     add message to packet payload
- The above procedure will repeat 3 times



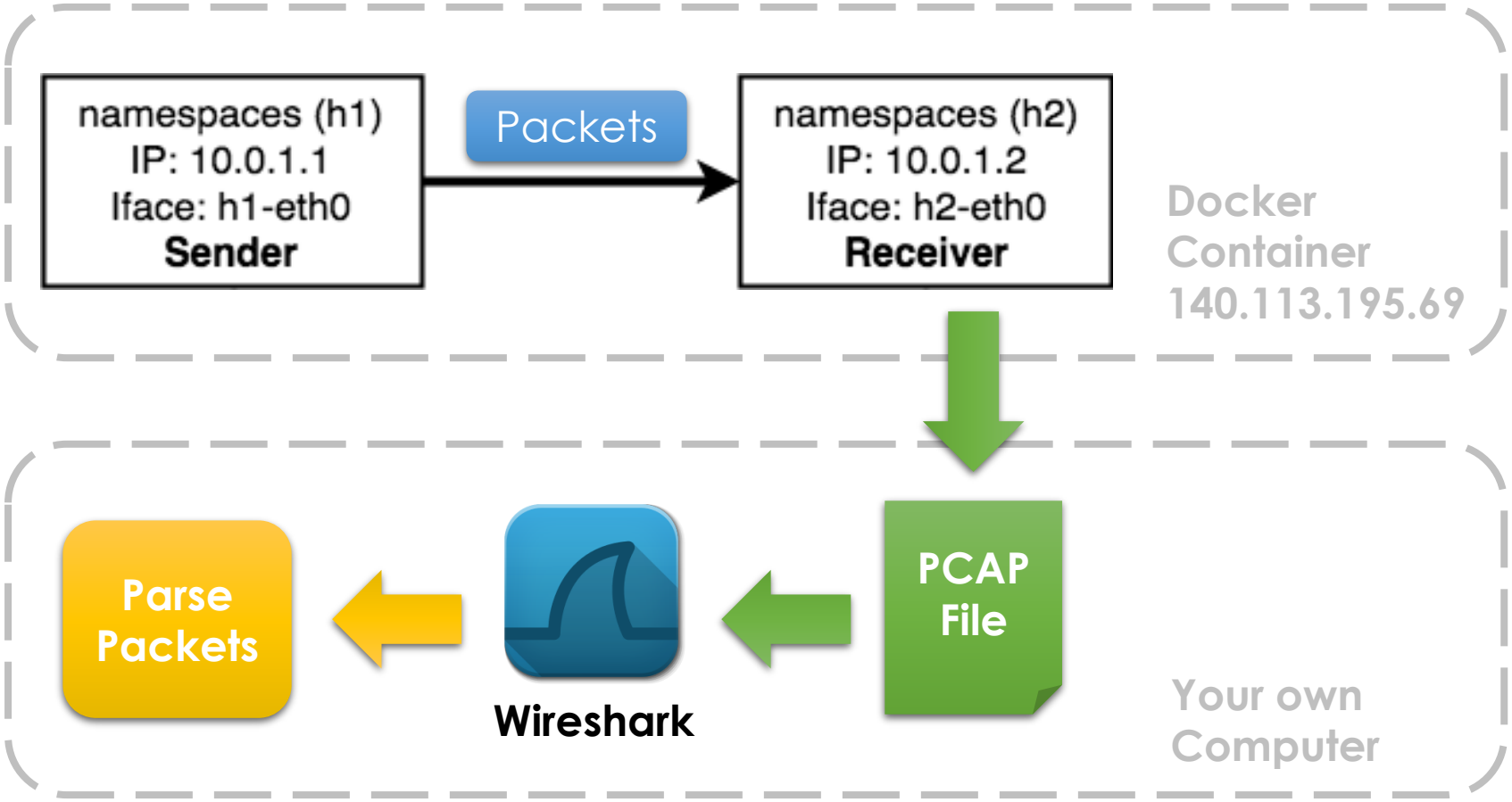
Overview (cont.)

- Packet format
  - **ID packet**



  - **Message packet**

Overview (cont.)



Scapy

- **Scapy** is an interactive packet manipulation tool in Python
  - forge or decode packets of protocols,
  - send packets to wire,
  - capture packets,
  - match requests and replies, etc.

- Example of Scapy (Scapy's documentation)

```
Welcome to Scapy (2.4.0)
>>> a = IP(dst="172.16.1.40")
>>> a
<IP dst=172.16.1.40 |>
```

Wireshark

- **Wireshark** is a widely-used network protocol analyzer
  - Deep inspection of hundreds of protocols
  - Live capture and offline analysis
  - Most powerful display filter
  - Read/write many different capture file formats
- Examples of DisplayFilter
  - Load a PCAP file
  - Show only SMTP (port 25) or ICMP

```
>>> tcp.port eq 25 or icmp
```

  - Show any traffic to or from 10.0.0.1

```
>>> ip.addr == 10.0.0.1
>>> ip.src == 10.0.0.1 or ip.dst == 10.0.0.1
```

Wireshark Filtering Rules

- Filter the packets that match some conditions
  - For example, to find TCP packets with a port number of 80, you can use **tcp.port == 80**
- For more filter instructions, please reference to:
  - Building display filter expressions
  - DisplayFilters
- Frequently used:
  - ip.src, ip.dst, ip.addr, …(IP address)
  - tcp.port, tcp.srcport, tcp.dstport, … (port)
  - eth.src, eth.dst, eth.addr, ... (MAC address)

Installation

- **Wireshark**
  - Windows / MacOS (Wireshark 3.0.5)
  - Ubuntu Linux

```
$ sudo apt-get install wireshark
```

- **Others**
  - PieTTY (for Windows)

Tasks

1. Environment Setup/ [modify `Protocol.py`] Define protocol via Scapy
2. [modify `sender.py`]: Send packets
3. [modify `receiver.py`]: Sniff packets
4. [execute]: Run sender and receiver
5. Push your modified files to GitHub
6. Download the PCAP file from GitHub and load PCAP via Wireshark
7. Filter the target packets
8. Report

GitHub

- Join the GitHub Classroom Lab1
  - https://classroom.github.com/a/32hLe5bW
- If you haven't filled out this form yet, NCTU CN 2019 - GitHub Account

Task 1. Environment setup (cont.)

- Login to your **Docker** container using **SSH**
- **For Windows**
  - Open PieTTY and connect to the Docker container
    - IP address: `140.113.195.69`
    - Port: port list
  - Login as root

```
Login: root
Password: cn2019
```

- **For Windows, MacOS and Ubuntu**
  - Use terminal to connect to the Docker

```
$ ssh root@140.113.195.69 -p xxxxx
Password: cn2019
```

Task 1. Environment setup

- Download required files from GitHub

```
$ git clone
https://github.com/chenyang14/Lab1_Packet_Manipulation.git
```

- Get and set repository or global options

```
$ cd Lab1_Packet_Manipulation/
$ git config --global user.name "<NAME>"
$ git config --global user.email "<EMAIL>"
```

- Set a new remote URL to your repository

```
$ git remote set-url origin
https://github.com/nctucn/lab1-<GITHUB_ID>.git
```

- Push your repository to GitHub

```
$ git push origin master
```

Task 1. Environment setup (cont.)

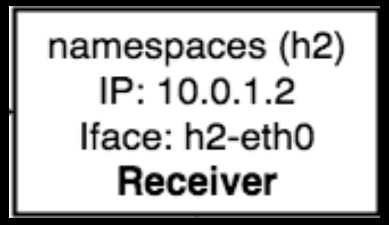
Structure of the packet manipulation project

```
Lab1_Packet Manipulation/      # This is ./ in this repository
|--- src/                      # Source code
     |--- out/                 # Output files
     |--- scripts/             # Networks configuration
          |--- main.sh         # Scripts for building namespace
     |--- sender.py            # Send packets
     |--- receiver.py          # Receive and sniff packets
     |--- Protocol.py          # Define your own protocol
|--- Lab1_info.pdf
```

Task 1. Environment setup (cont.)

- Create network namespace: ./src/scripts/main.sh

```
# Create h2 network namespaces
ip netns add h2
# Delete h2 network namespaces
ip netns del h2
# Bring up the lookup interface in h2
ip netns exec h2 ip link set lo up
# Set the interface of h2 to h2-eth0
ip link set h2-eth0 netns h2
# Delete the interface of h2-eth0
ip link delete h2-eth0
# Activate h2-eth0 and assign IP address
ip netns exec h2 ip link set dev h2-eth0 up
ip netns exec h2 ip link set h2-eth0 address 00:0a:00:00:02:02
ip netns exec h2 ip addr add 10.0.1.2/24 dev h2-eth0
# Disable all IPv6 on h2-eth0
ip netns exec h2 sysctl net.ipv6.conf.h2-eth0.disable_ipv6=1
# Set the gateway of h2 to 10.0.1.254
ip netns exec h2 ip route add default via 10.0.1.254
```



Task 1. Environment setup (cont.)

- Run main.sh to build the namespace

```
$ sudo chmod +x main.sh
$ ./main.sh net
```

- You will get the following result if succeed

```
/bin/bash: warning: setlocale: LC_ALL: cannot change locale (en_US.UTF-8)
[INFO] Create h1 and h2 network namespaces
[INFO] Bring up the lookup interface in h1 and h2
[INFO] Build the link: h1-eth0 <-> h2-eth0
[INFO] Activate h1-eth0 and assign IP address
[INFO] Activate h2-eth0 and assign IP address
[INFO] Disable all IPv6 on h1-eth0 and h2-eth0
net.ipv6.conf.h1-eth0.disable_ipv6 = 1
net.ipv6.conf.h2-eth0.disable_ipv6 = 1
[INFO] Set the gateway to 10.0.1.254 in routing table
```

Task 1. Define protocol via Scapy

- **TODO:** Define customized protocol: **ID header format**
  - ./src/Protocol.py

```
class Protocol(Packet):
    # Set the name of protocol
    name = 'Student'
    # Define the fields in protocol
    # [TODO] Add the 2nd string field called “id” with default
    #        value '0000000'
    fields_desc = [
        StrField('dept', 'cs', fmt = 'H', remain = 0),
        # TODO here
    ]
```

Task 2. Send packets

- **TODO**: modify ./src/sender.py to define packet information

```
# [TODO] Set source and destination IP address (Task 2.)
src_ip = ''
dst_ip = ''
# Set source and destination port
src_port = 1024
dst_port = 80
# Define IP header
ip = IP(src = src_ip, dst = dst_ip)

# [TODO] Add 'id' field in customized header here (Task 2.)
#        and fill in your department
student = Protocol(dept = 'YOUR_DEPT')
# [TODO] Fill in the message payload (Task 2.)
msg = ['Anything you want to say','...','...']
```

Task 2. Send packets (cont.)

- Send packets (TCP 3-way handshaking): ./src/sender.py

```
# TCP connection - SYN / SYN-ACK
tcp_syn = TCP(sport = src_port, dport = dst_port, flags =
'S', seq = 0)
packet = ip / tcp_syn
tcp_syn_ack = sr1(packet)
print '[INFO] Send SYN and receive SYN-ACK'

# TCP connection - ACK
ack = tcp_syn_ack.seq + 1
tcp_ack = TCP(sport = src_port, dport = dst_port, flags =
'A', seq = 1, ack = ack)
packet = ip / tcp_ack
send(packet)
print '[INFO] Send ACK'
```

Task 2. Send packets (cont.)

- Send packets: ./src/sender.py

```
# Send packet with customized header
ack = tcp_ack.seq + 1
tcp = TCP(sport = src_port, dport = dst_port, flags = '',
seq = 2, ack = ack)
packet = ip / tcp / student
send(packet)
print '[INFO] Send packet with customized header'

# Send packet with payload
ack = tcp.seq + 1
tcp = TCP(sport = src_port, dport = dst_port, flags = '',
seq = 3, ack = ack)
payload = Raw(msg[i])
packet = ip / tcp / payload
send(packet)
print '[INFO] Send packet with payload'
```

Task 3. Sniff packets

- **TODO:** modify ./src/receiver.py
- Set the source IP address (h1):

```
# [TODO] Set source IP address (Task 3.)
src_ip = ''
```

- Sniff received packets and write into pcap file

```
# Sniff the packets from specific source IP
print '[INFO] Start to sniff'
filter_msg = "ip src " + src_ip + " and tcp"
packets = sniff(filter=filter_msg, prn = lambda x:
packetHandler(x))
# Dump the sniffed packet into PCAP file
print '[INFO] Stop sniffing ... Write into PCAP file'
filename = './out/lab1_' + id + '.pcap'
wrpcap(filename, packets)
```

Task 4. Run sender and receiver

- Open tmux (tutorial) with horizontal two panes

```
# Keep your path in ./src/
# Open tmux
$ tmux
# Open new pane in horizontal
Ctrl-b
Shift-%
# Switch between two panes
Ctrl-b
Arrow-left/right key
```

- Switch into two namespaces

```
# Run namespace h1 in your left pane
$ ./scripts/main.sh run h1
# Run namespace h2 in your right pane
$ ./scripts/main.sh run h2
```

Task 4. Run sender and receiver

- Run receiver.py first

```
# Switch between two panes
Ctrl-b
Arrow-right key
# Run receiver.py
h2> python receiver.py
```

- Run sender.py

```
# Switch between two panes
Ctrl-b
Arrow-left key
# Run sender.py
h1> python sender.py
```

Task 4. Run sender and receiver

- After finishing the sending and receiving process.

```
# Switch between two panes
Ctrl-b
Arrow-right key
# Stop receiving process
Ctrl-c
# exit h2
h2> exit
```

```
# Switch between two panes
Ctrl-b
Arrow-left key
# exit h1
h1> exit
```

- You will get lab1_yourID.pcap after receiving all packets in ./src/out/

Task 5. Push your files to git remote

- List files in ./src/out/ folder to check whether lab1_yourID.pcap is successfully generated

```
# In Lab1_Packet_Manipulation/src/
# List files in ./out/
$ ls ./out/
lab1_yourID.pcap
```

- Push your files to GitHub
  - In Lab1_Packet_Manipulation/

```
# Add your files into staging area
$ git add .
# Commit your files
$ git commit –m "Commit lab1 in class"
# Push your files to remote repository
$ git push origin master
```

Task 6. Load PCAP via Wireshark

- To get the files in Docker container to your own computer
  - Download your code from GitHub

```
$ git clone
https://github.com/nctucn/lab1-<YOUR_GITHUB_ID>.git
```

Or

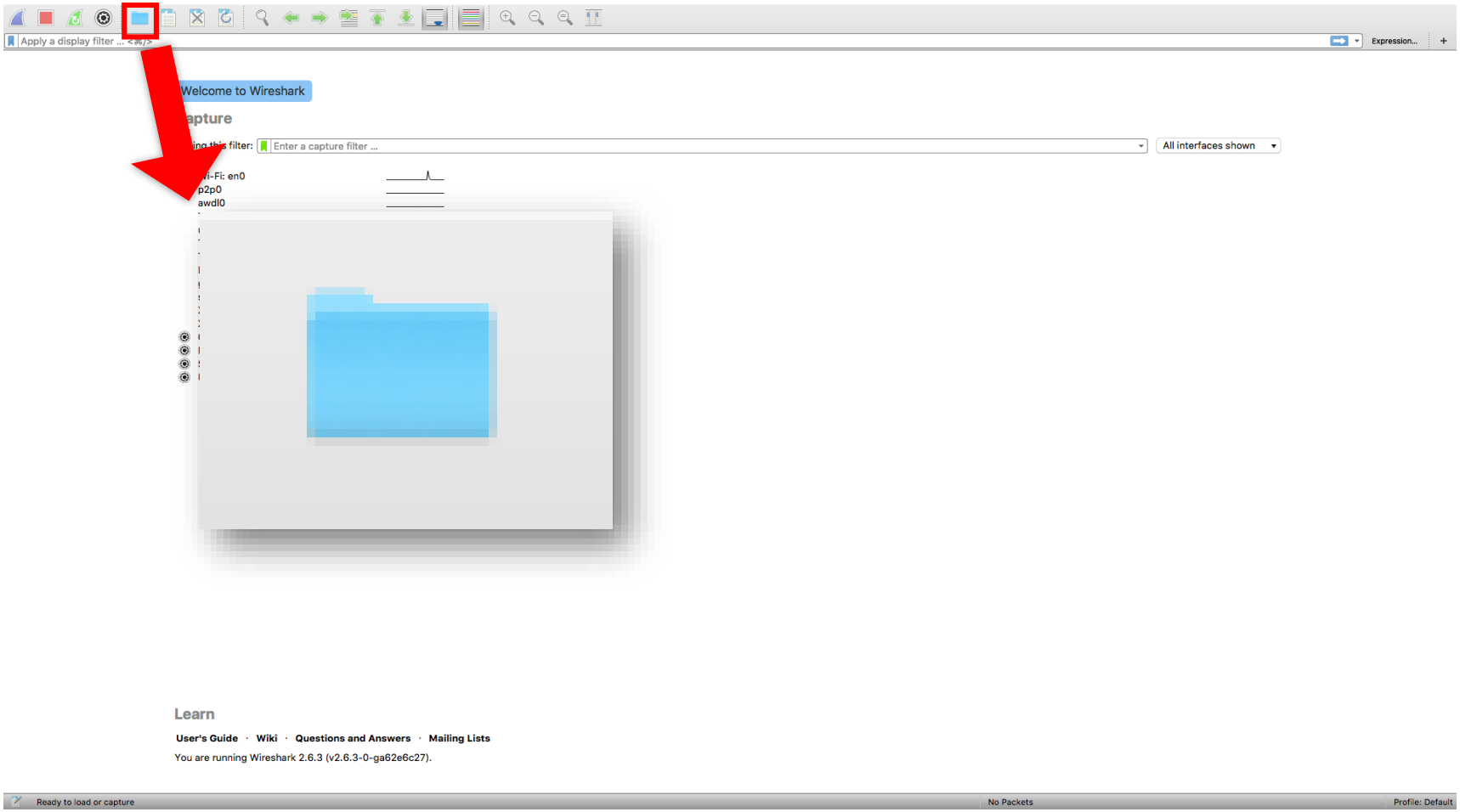
  - Use scp command

```
# scp to download the folder from Docker container to
# your own computer.
$ scp -P xxxxx -r
root@140.113.195.69:~/Lab1_Packet_Manipulation ./
```

Note: These commands should be typed in your local machine, not the remote docker server

Task 6. Load PCAP via Wireshark

- **Open the PCAP file using Wireshark** (./src/out/lab1_yourID.pcap)



Task 7. Filter the target packet

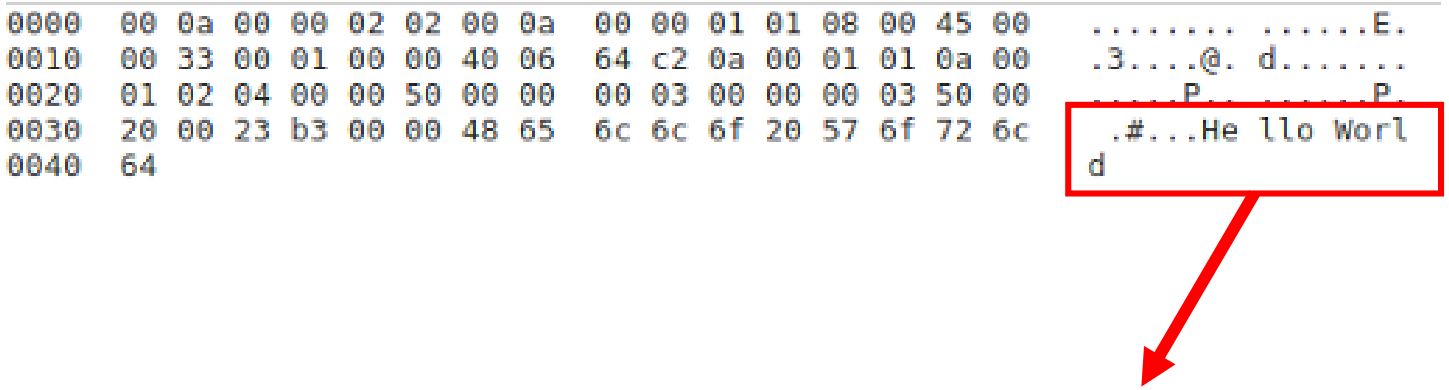
- **Filter the packets of our defined protocol**
  - Enter your filter command on DisplayFilters
  - Hint: use header info., e.g., IP address or/and TCP seq

```
>> tcp.port eq 25 or icmp  # Example command
```

  - **[TODO]** Save the screenshot of your filtering result to answer the question in report.
- **Filter the packets with the payload**
  - Enter your filter command on DisplayFilters
  - Find out the message payload in these three packets
  - **[TODO]** Save the screenshot of your filtering result to answer the question in report.

Task 7. Filter the target packet

- Example of getting payload in packet



The message payload is "Hello World"

Task 8. Report

- **Describe each step in this lab in detail**
  - e.g. explain how the code works, where you have modified, the procedure you sent the packets …
- **Answer the following questions in short**
  1. What is your command to filter the packet with customized header on Wireshark?
  2. Show the screenshot of filtering the packet with customized header.
  3. What is your command to filter the packet with payload on Wireshark?
  4. Show the screenshot of filtering the packet with payload .
  5. Show the screenshot of three messages in packets' payload .

Hint

- **Vim** has been installed in your Docker container. If you are not familiar with it, you can edit the code by any text editors you prefer, and transfer the files to your container.

- How to copy files and folders from/to your Docker container ? (scp cheatsheet)

```
$ scp [-P <port>][-r] <SOURCE_PATH> <TARGET_PATH>

# For example, if you edit ‘sender.py’ on your computer,
# and you want to upload ‘sender.py’ to the container.
# $ scp –P xxxxx ./sender.py
root@140.113.195.69:~/Lab1_Packet_Manipulation/src
```

Submission

- **Push your works to GitHub repository (nctucn)**
  - **Trace files** (Lab1_Packet_Manipulation/src/out/)
    - PCAP file (lab1_ID.pcap)
  - **Python code** (Lab1_Packet_Manipulation/src/)
    - sender.py
    - receiver.py
    - Protocol.py
  - **Report** (Lab1_Packet_Manipulation/)
    - Report.pdf

- **No need to submit to new E3**

Grading Policy

- Deadline - Oct. 25, 2019. 23:59
- Code correctness – 40 %
- Report – 60 %
  - Description of working steps -30 %
  - Answer the questions -30 %

Grading Policy (cont.)

- **Late Policy** (follow syllabus)

$$(\textbf{Your score}) \times \mathbf{0.8}^{\boldsymbol{D}},$$

where D is the number of days over due

- **Cheating Policy** (follow syllabus)
  - Academic integrity
  - Homework must be your own – cheaters share the score
  - Both the cheaters and the students who aided the cheater will be held responsible for the cheating