

acpang@csie.ntu.edu.tw

<http://www.csie.ntu.edu.tw/~acpang>

Prof. Ai-Chun Pang

Graduate Institute of Networking & Multimedia,  
Dept. of Computer Science and Engineering  
National Taiwan University

# LAB1

## Trace route concept and implementation

# Outline

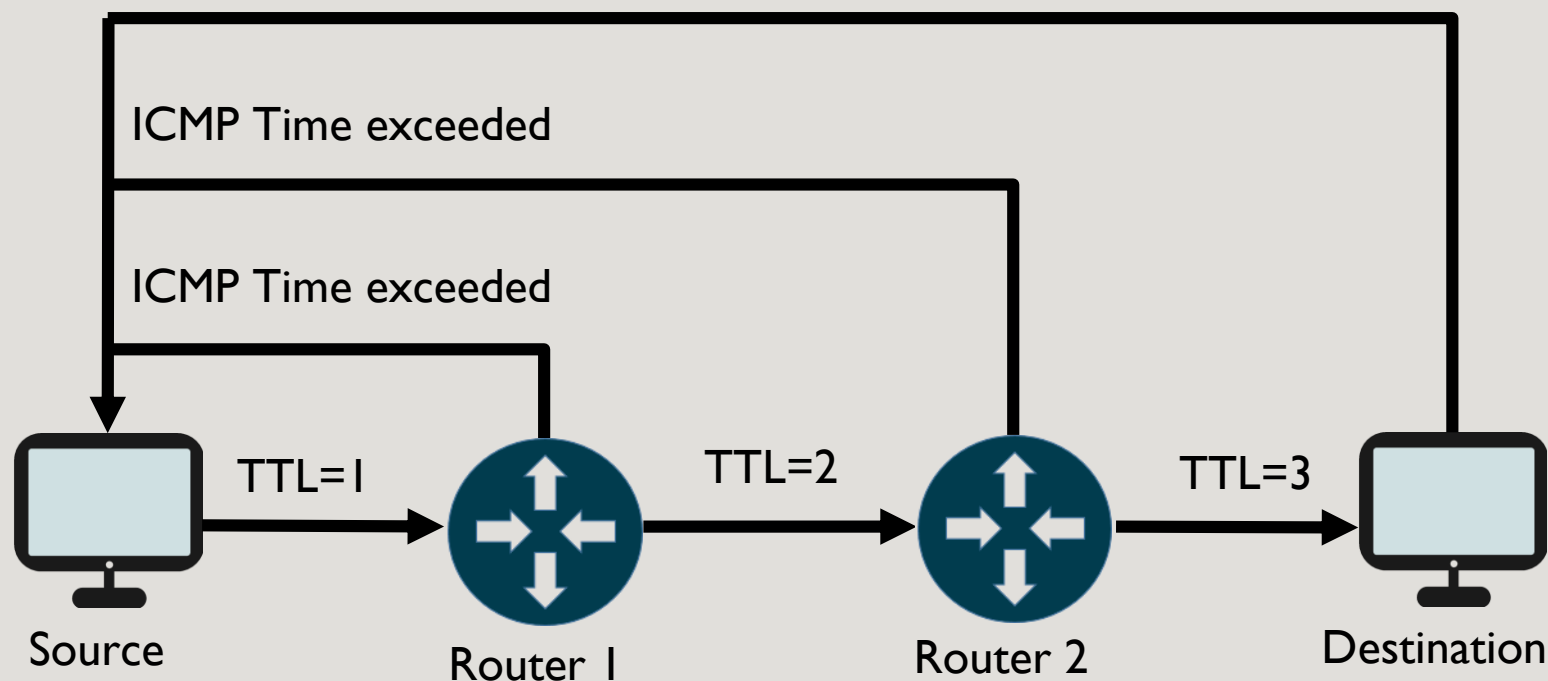
- Introduction
- Environment
- Procedure
- Grading Policy
- Regulation
- Deadline

# Introduction

- By setting the TTL number in the IP header, we can “trace” the routes from source to destination with RTT.
- Try to send and receive ICMP packets.
- Besides, you should analyze why traceroute cannot show the path of the full route and explain how to detect and defend traceroute.

# Description

ICMP Destination unreachable(UPD) / ICMP Echo reply(ICMP)



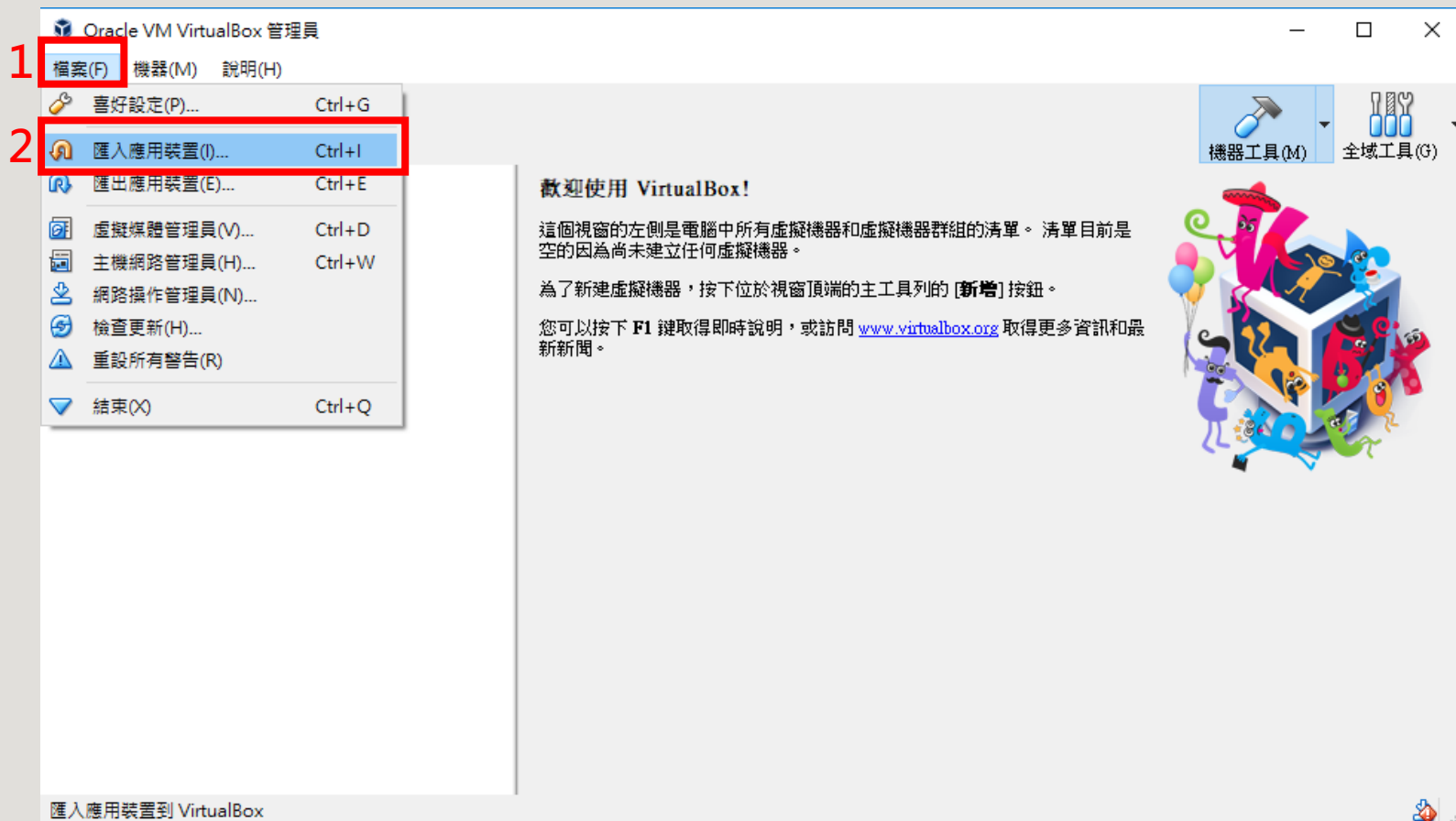
# Description

```
~/Desktop/lab1 (-zsh) 100% (15:00)
[ OSX ~ lijunyu IP 10.5.4.238 ✓ ]
tracert -I ox.ac.uk
tracert: Warning: ox.ac.uk has multiple addresses; using 151.101.66.216
tracert to ox.ac.uk (151.101.66.216), 64 hops max, 72 byte packets
 1  10.5.7.253 (10.5.7.253)  8.292 ms  5.237 ms  57.710 ms
 2  172.17.0.2 (172.17.0.2)  1.865 ms  1.374 ms  7.654 ms
 3  140.112.16.190 (140.112.16.190)  8.879 ms  8.325 ms  6.257 ms
 4  140.112.149.121 (140.112.149.121)  2.837 ms  3.560 ms  5.005 ms
 5  140.112.0.174 (140.112.0.174)  4.458 ms  1.766 ms  5.559 ms
 6  140.112.0.206 (140.112.0.206)  15.865 ms  5.367 ms  2.446 ms
 7  203.160.226.233 (203.160.226.233)  8.759 ms  4.315 ms  4.993 ms
 8  181-61-41-175.twgate-ip.twgate.net (175.41.61.181)  3.126 ms  6.481 ms  3.174 ms
 9  218-60-41-175.twgate-ip.twgate.net (175.41.60.218)  27.585 ms  34.835 ms  25.869 ms
10  54113.hkg.equinix.com (36.255.56.96)  25.987 ms  30.466 ms  35.795 ms
11  151.101.66.216 (151.101.66.216)  28.688 ms  26.265 ms  36.899 ms
[ OSX ~ lijunyu IP 10.5.4.238 ✓ ]
```

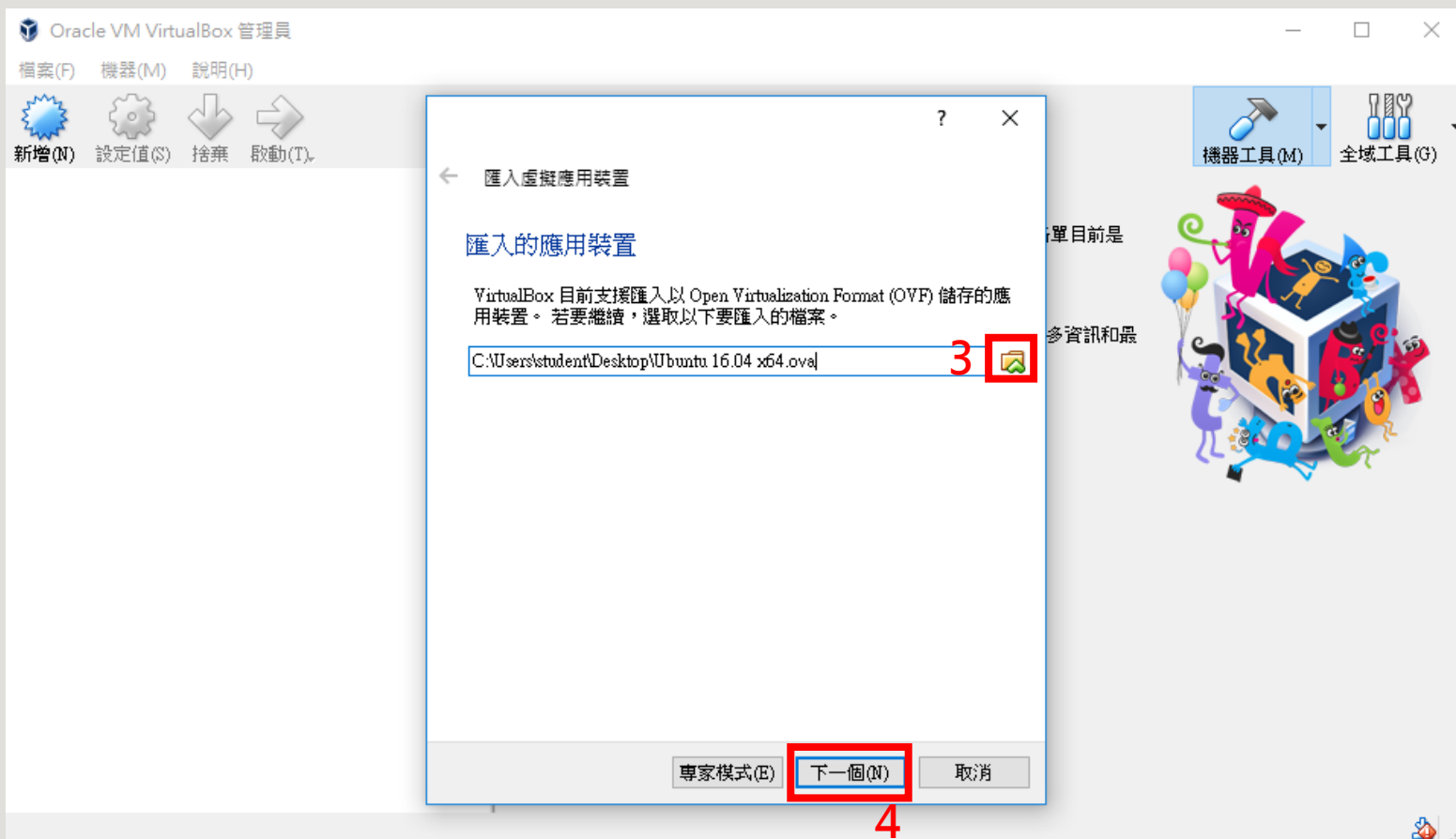
# Environment

- Language: C / C++
- OS: ubuntu(recommended)
- Virtual box ova file, password: cnlab2021
  - Dropbox: <https://reurl.cc/Kx011q>
  - Google drive: <https://reurl.cc/Q7XqA9>
  - USB
- You can start with our sample code in NTU COOL.

# Environment

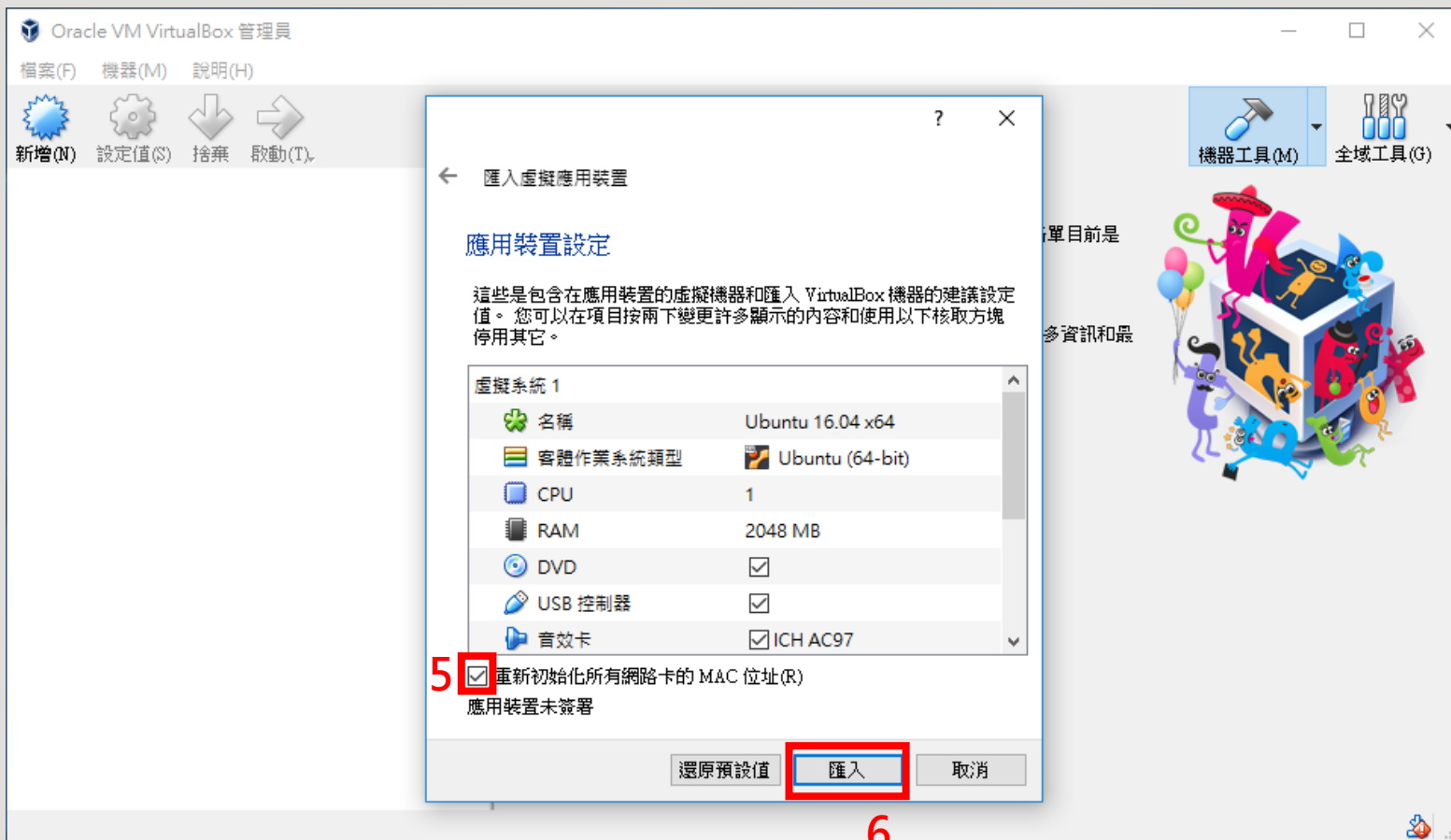


# Environment





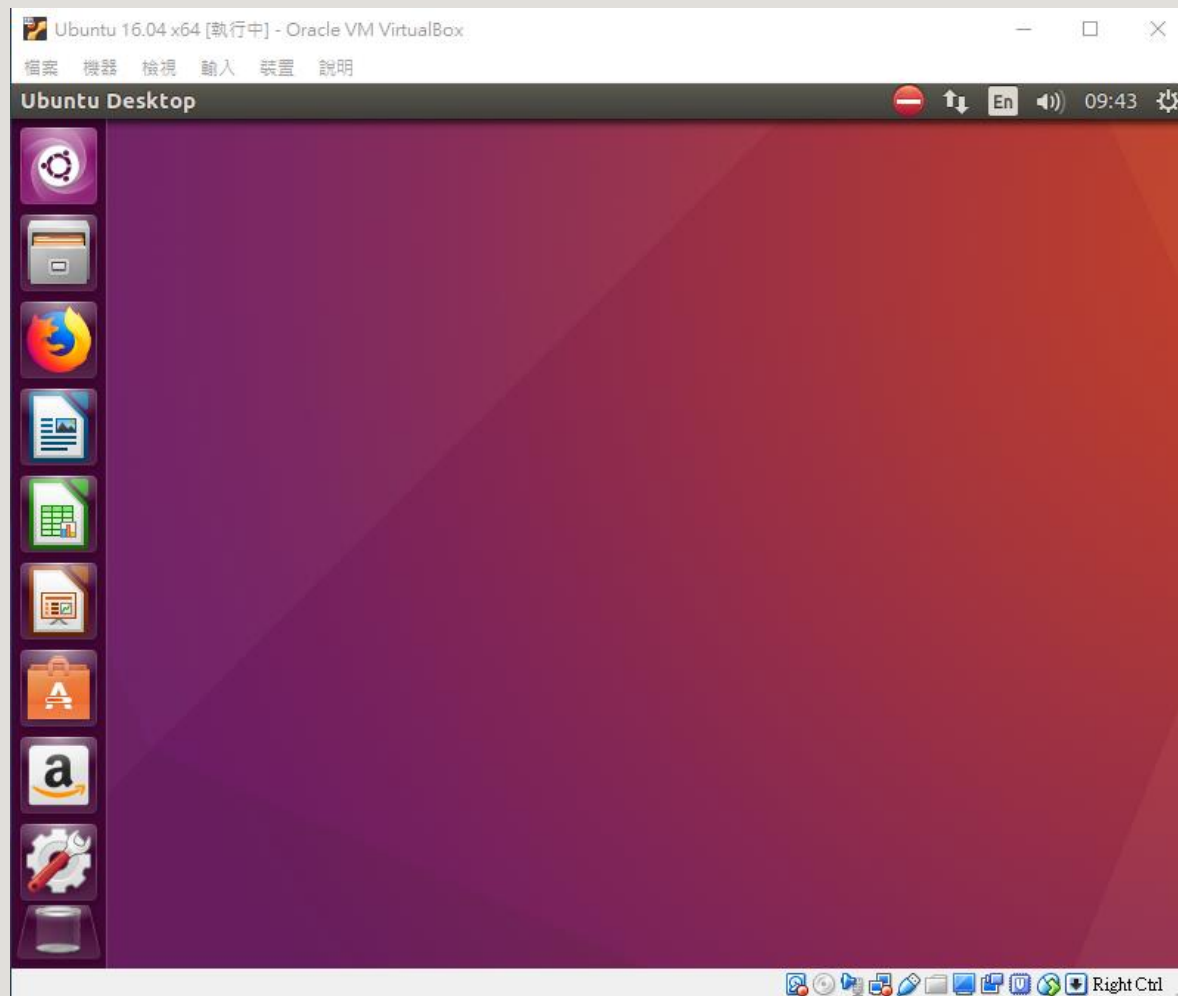
# Environment



# Environment



# Environment



# Step

- Open a socket
- Create an ICMP packet
- Send and receive
- Extract the packet
- Print the result

# Step

- Open a socket
- Create an ICMP packet
- Send and receive
- Extract the packet
- Print the result

## Step - Open a socket

- `int socket(int domain, int type, int protocol)`
  - domain : IPv4 (Inter Protocol Version 4)
  - type : The communication semantics, we used raw type here
  - protocol : ICMP
- `int setsockopt(int socket, int level, int option_name, const void *option_value, socklen_t option_len);`
  - Setting Time To Live value

# Step

- Open a socket
- Create an ICMP packet
- Send and receive
- Extract the packet
- Print the result

## Step - Create an ICMP packet

Type	Code	Checksum
Identifier		Sequence Number
Option Data		

- Type :
  - 0 : Echo reply
  - 8 : Echo request
- Code :
  - ICMP subtype (e.g. Error code)
  - Filled with 0 in this case



## Step - Create an ICMP packet

Type	Code(0)	Checksum
Identifier		Sequence Number
Option Data		

- Identifier : 2 Bytes
- Sequence Number : 2 Bytes
- Identifier and Sequence Number can be used by the client to match the timestamp reply with the timestamp request

## Step - Create an ICMP packet

Type	Code(0)	Checksum
Identifier		Sequence Number
Option Data		

- Optional Data : can be used to identify whether Echo-Reply and Echo-Request is a couple
- In this experience, the Optional Data field can be empty.

## Step - Create an ICMP packet

Type	Code(0)	Checksum
Identifier		Sequence Number
Option Data		

- Type : unsigned 8 bits
- Code : unsigned 8 bits
- Checksum : unsigned 16 bits
- Identifier : unsigned 16 bits
- Sequence Number : signed 16 bits

# Step - Create an ICMP packet

- Wireshark

```
> Frame 1347: 106 bytes on wire (848 bits), 106 bytes captured (848 bits) on interface 0
> Ethernet II, Src: Azurewav_a7:97:e7 (6c:71:d9:a7:97:e7), Dst: Cisco_ff:fc:a8 (00:08:e3:ff:fc:a8)
> Internet Protocol Version 4, Src: 10.5.1.176, Dst: 124.108.103.104
v Internet Control Message Protocol
  Type: 8 (Echo (ping) request)
  Code: 0
  Checksum: 0xf7f7 [correct]
  [Checksum Status: Good]
  Identifier (BE): 1 (0x0001)
  Identifier (LE): 256 (0x0100)
  Sequence number (BE): 7 (0x0007)
  Sequence number (LE): 1792 (0x0700)
> [No response seen]
> Data (64 bytes)
```

0000	00 08 e3 ff fc a8 6c 71 d9 a7 97 e7 08 00 45 00	.....lq .....E.
0010	00 5c 31 9f 00 00 03 01 96 79 0a 05 01 b0 7c 6c	·\1.....·y..... l
0020	67 68 08 00 f7 f7 00 01 00 07 00 00 00 00 00 00	gh.....
0030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
0060	00 00 00 00 00 00 00 00 00 00	.....

# Step - Checksum

0000	00	08	e3	ff	fc	a8	6c	71	d9	a7	97	e7	08	00	45	00	.....lq.....E.
0010	00	5c	31	9f	00	00	03	01	96	79	0a	05	01	b0	7c	6c	·\1.....·y..... 1
0020	67	68	08	00	f7	f7	00	01	00	07	00	00	00	00	00	00	gh.....
0030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0060	00	00	00	00	00	00	00	00	00	00							.....

- Sum up each half-word (excluding checksum)
  - $0800 + 0001 + 0007 + 0000 + \dots + 0000 = 0000\ 0808$
- Sum of the higher bit and lower bit
  - $0000\ 0808 \Rightarrow 0000 + 0808 = 0808$
- Ones complement of the sum (1 to 0 , 0 to 1)
  - $0808 \Rightarrow F7F7$

# Step

- Open a socket
- Create an ICMP packet
- Send and receive
- Extract the packet
- Print the result

## Step - Send and receive

- `ssize_t sendto(int sockfd, const void *buf, size_t len, int flags, const struct sockaddr *dest_addr, socklen_t addrlen);`
- `ssize_t recvfrom(int sockfd, void *buf, size_t len, int flags, struct sockaddr *src_addr, socklen_t *addrlen);`
- Calculate the response time
  - `int gettimeofday(struct timeval *tv, struct timezone *tz);`
  - Timeout handling

# Step

- Open a socket
- Create an ICMP packet
- Send and receive
- Extract the packet
- Print the result

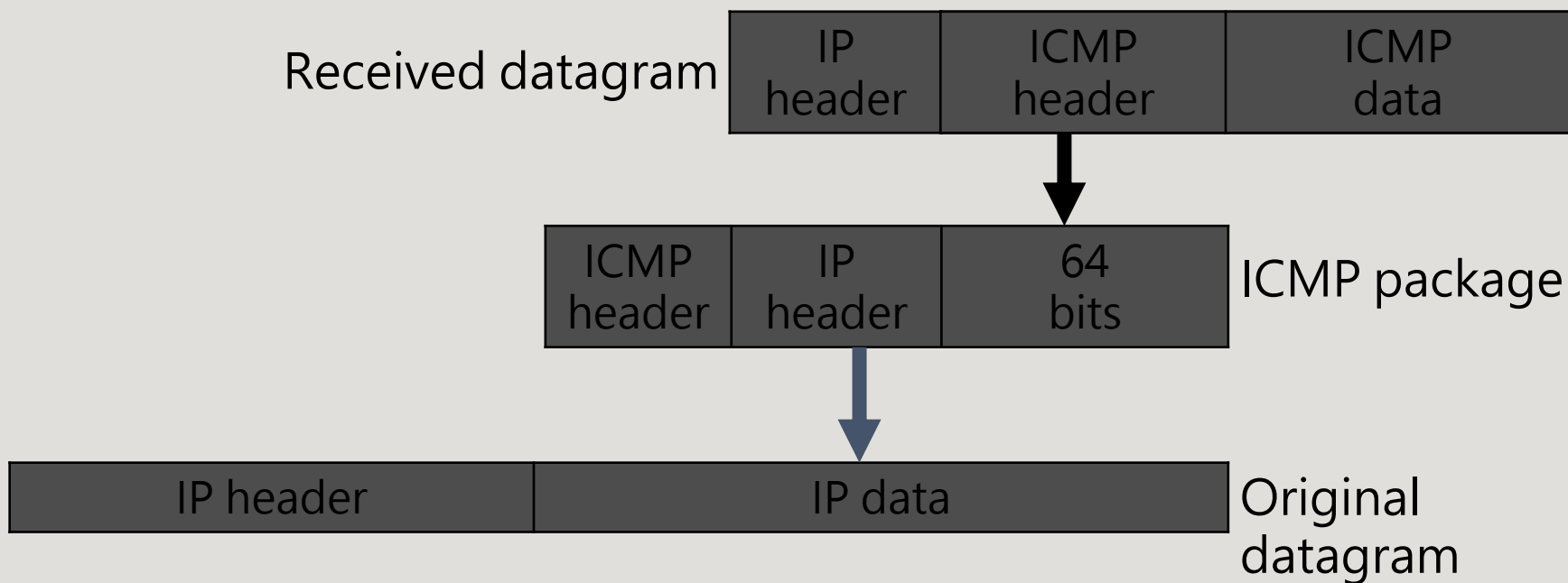


## Step - Extract the packet

Type	Code	Checksum
Undefined or other (32 bits)		
Data Section (IP header + 64 bits of original data)		

- Check ICMP type is either time exceeded or ICMP\_ECHOREPLY
- Extract the original ICMP packet in the received IP packet
- Check identifier and sequence number in the original

## Step - Extract the packet



# Step

- Open a socket
- Create an ICMP packet
- Send and receive
- Extract the packet
- Print the result

## Step - Print the result

- Linux traceroute format
  - <ttl, IP, RTT, RTT, RTT>
  - Timeout
  - No response / not reachable

# Grading Policy

- Experiment : (50%)
  - Localhost (10%)
  - Nearest router (10%)
  - No response (handling time out) (10%)
  - Handle DNS lookup (10%)
  - Using TCP and UDP to implement traceroute (by options) (10%)

# Grading Policy

- Report : (50%)
  - Environment (5%)
  - How to detect and defend traceroute (5%)
  - Why traceroute cannot show the full route (10%)
  - Why the result may not always be the same (10%)
  - Compare the results between local and foreign, and explain what causes the difference. (10%)
  - Explain the difference by using TCP, UDP, and ICMP. (10%)

# Demo

- Traceroute localhost(127.0.0.1)
- Traceroute wireless AP
- Traceroute by hostname
- Traceroute by TCP or UDP

# Regulation

- No cheating.
- You can only include libraries that appear in our sample code.
- List reference in your report.



# Deadline

- 3 / 18 Demo
- 3 / 25 23:59:00
  - Submit report and source code to NTU cool
  - Report has to be in pdf format