

# Lecture 6: Network Security

- 6.1 Background: Network layers
- 6.2 Name resolution and attacks
- 6.3 Denial of Service (Dos) attacks
- 6.4 Useful network security tools
- 6.5 Protection: Securing the communication channel using cryptography
- 6.6 Protection: Firewall
- 6.7 Protection: Network security management

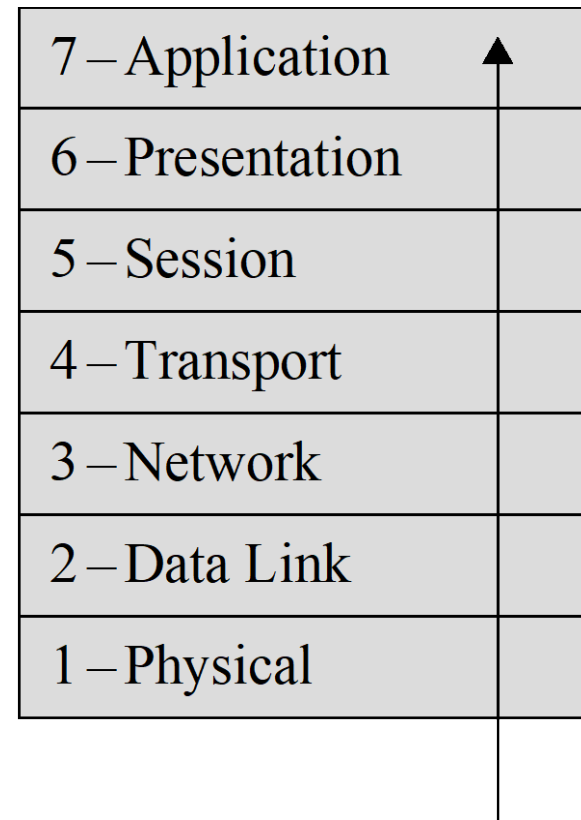
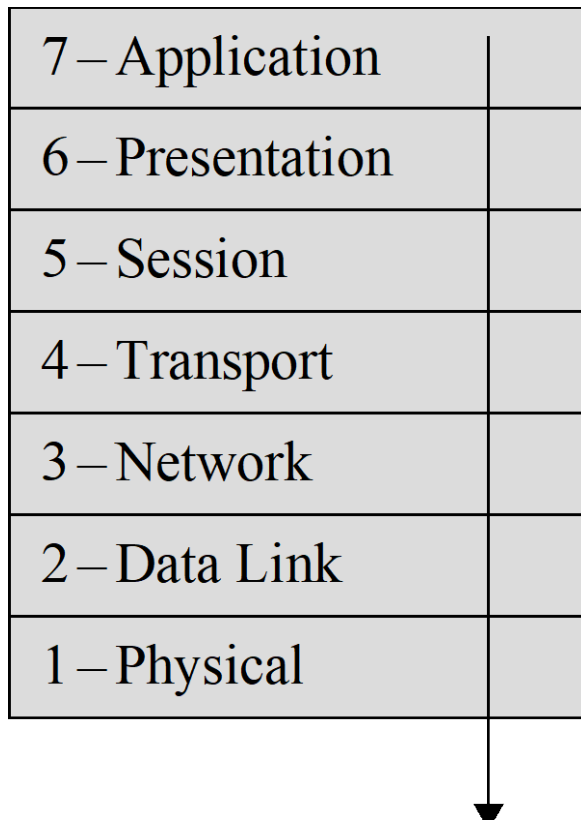
See: [PF6.1], [PF6.2],[PF6.4],[PF6.6], [PF6.9]

# **6.1 Background: Network Layers**

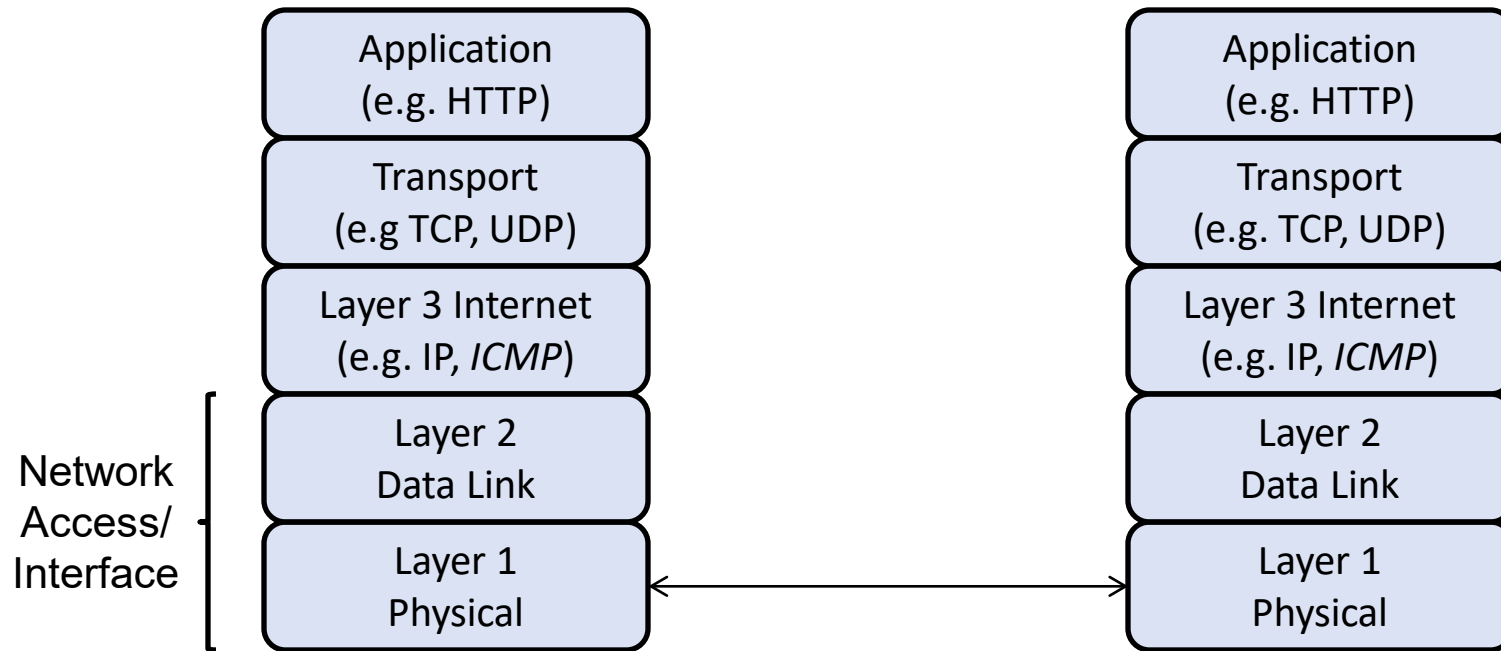
# The OSI Seven-Layer Network Model

The ***Open Systems Interconnection (OSI) model***:

a conceptual/reference model that standardizes the *communication functions* of a telecommunication/computing system

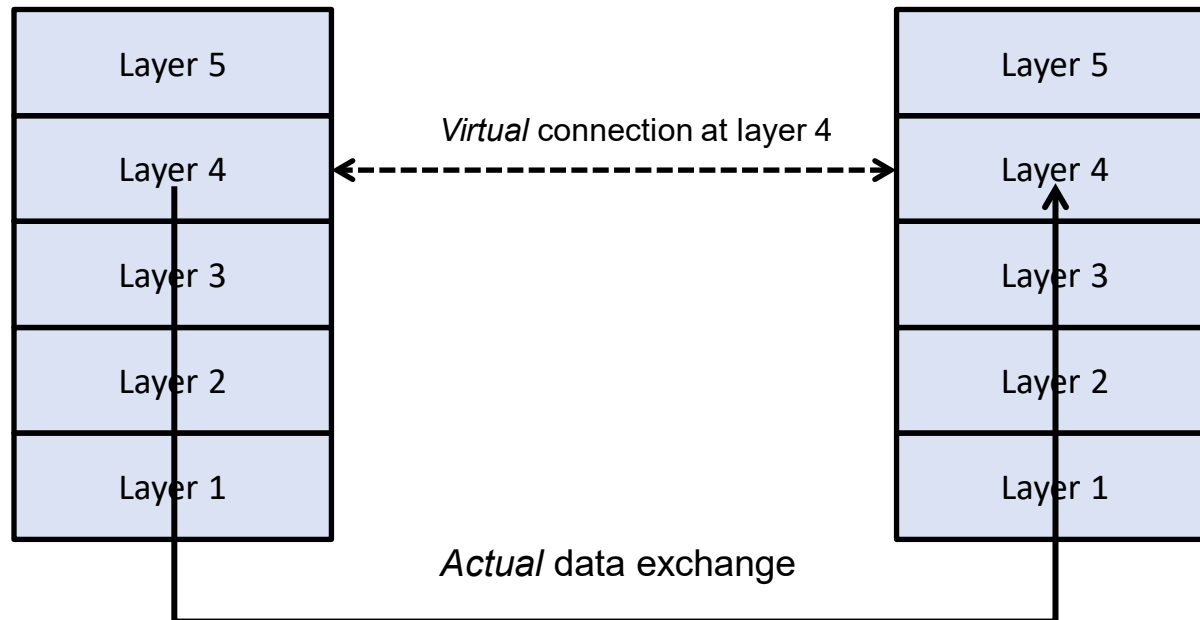


# The Internet (TCP/IP) Reference Model



# Why Network Layering?

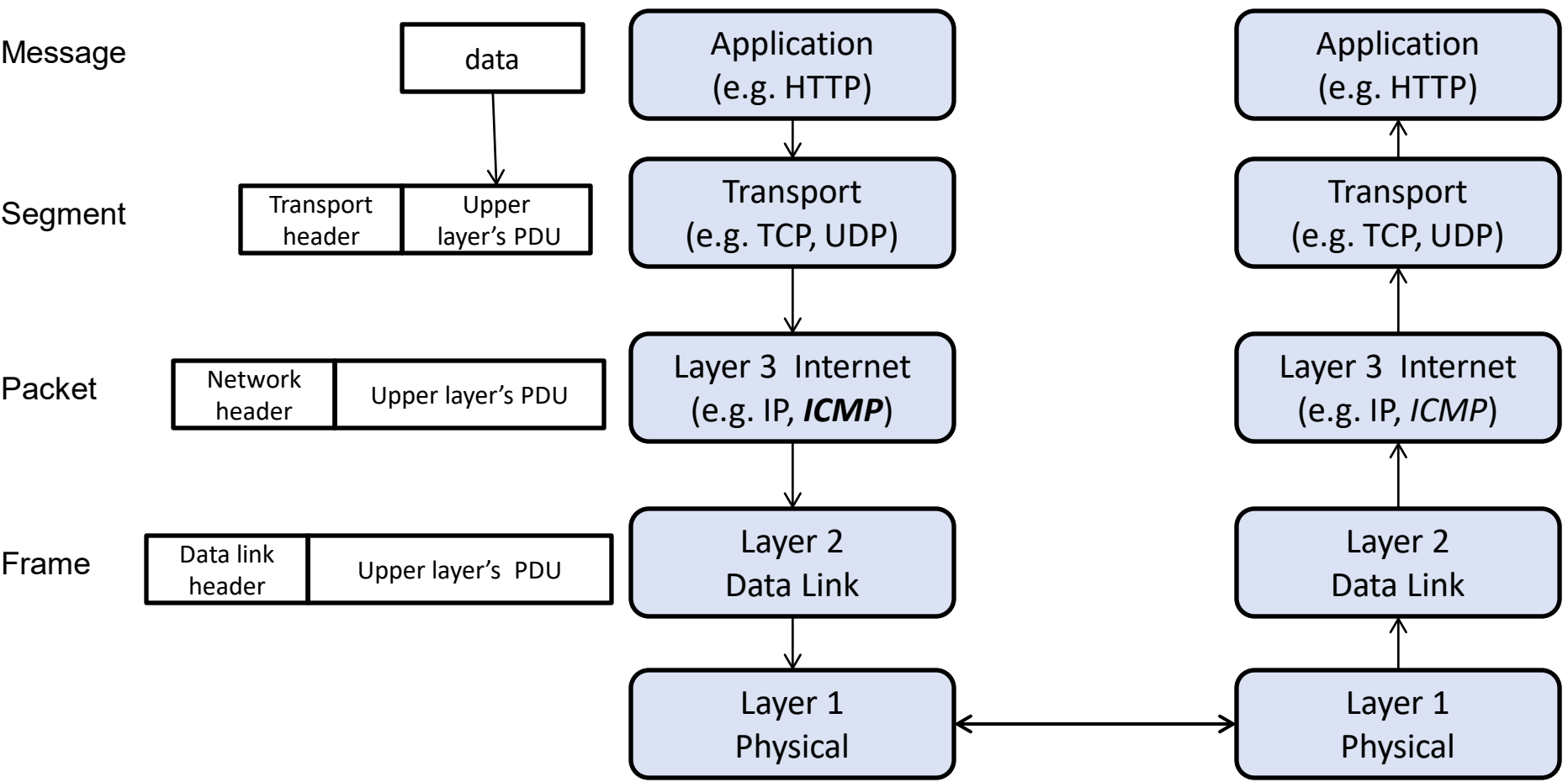
- It partitions a complex communication system into several **abstraction layers**
- The **peer entities** at the same layer  $N$  “conceptually” communicate with each other by executing a protocol *at that layer*



# Why Network Layering?

- The **layer- $N$**  protocol is built **on top of** a virtual connection at layer  **$N-1$**  below
- Example of a protocol at **layer 5** (authentication protocol):
  - (1)  $A \rightarrow B$ : “hello”
  - (2)  $A \leftarrow B$ : certificate of  $B$
- The “virtual connection” at layer 4 sends the message “hello” from  $A$  to  $B$  in Step (1), and sends  $B$ ’s certificate in Step (2)
- At **layer  $N$** :
  - A message to be sent is called:  
layer- $N$  ***protocol data unit*** (PDU)
  - **Encapsulation** of upper (i.e.  $N+1$ ) layer’s PDU

# Network Layers and Message Encapsulation



**General PDU format:**

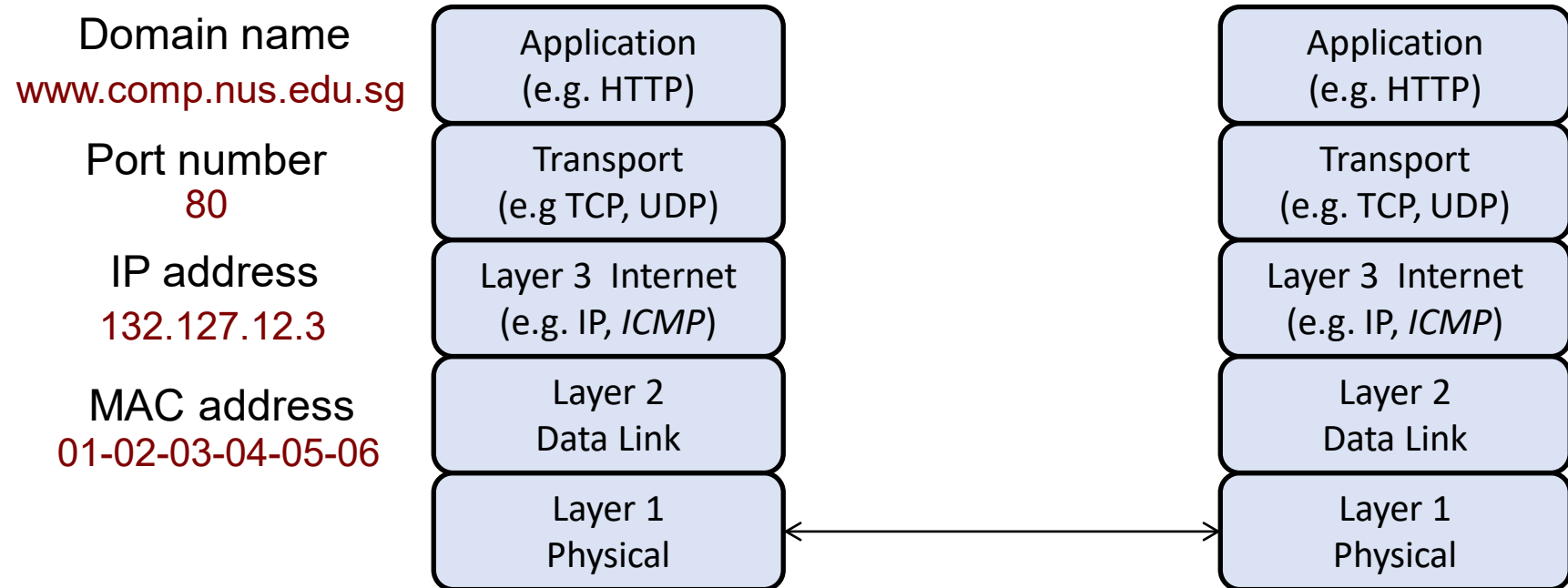


**Header:** meta data    **Payload:** the message/information intended to be sent

**Note:** An **ICMP** packet gets encapsulated by IP. Yet, ICMP is commonly considered as part of Layer 3, since it assists IP.

# Internet Layers: Different Addressing Schemes

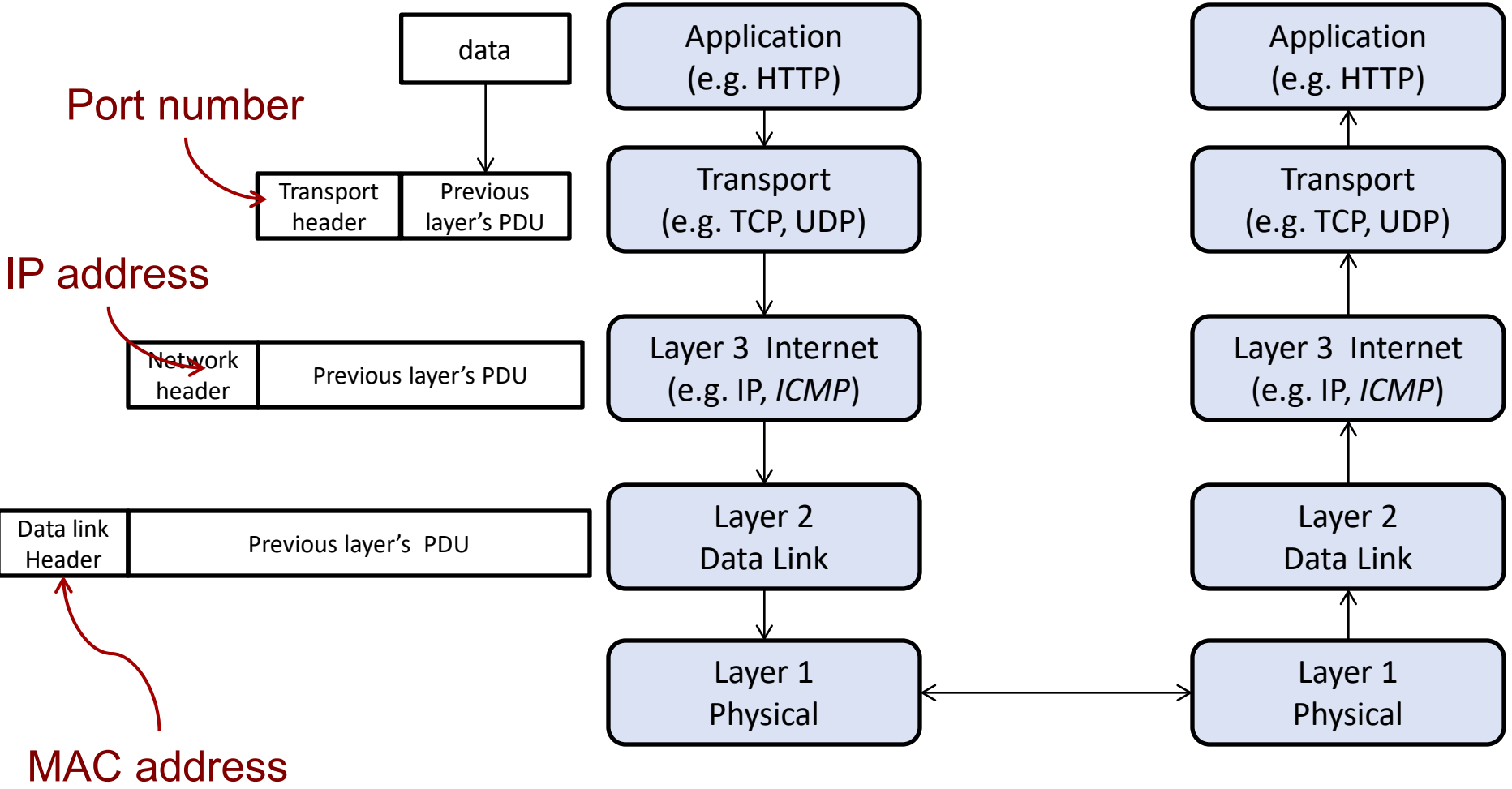
- Different *addressing schemes* at different layers



**Note:** MAC (medium access control) is *not* to be confused with crypto's MAC

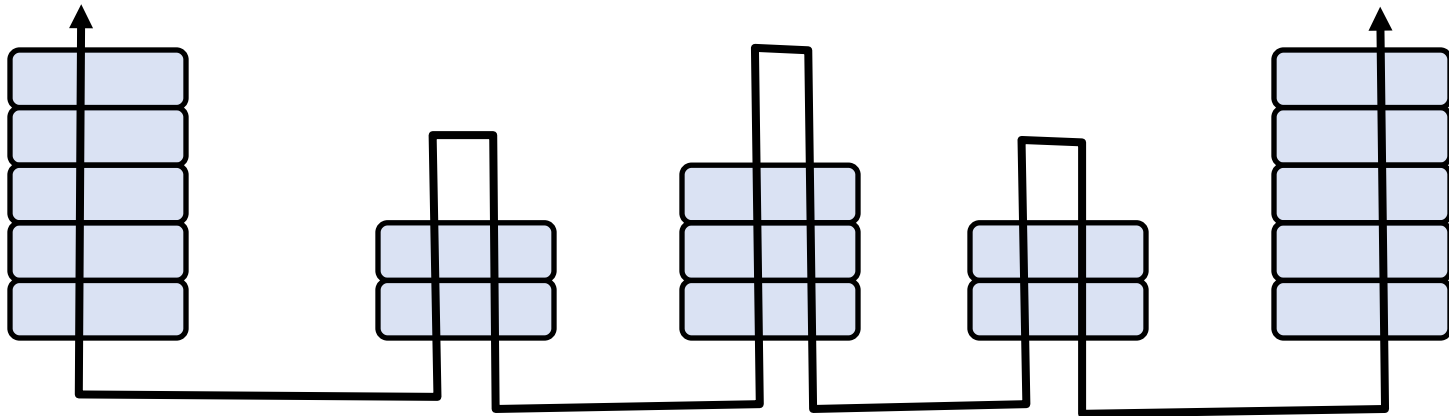


# Addressing at Various Layers



# Multiple Hops from Sender to Destination

- Note that data may go through **multiple hops**
- Can you guess each device type in the diagram below?
- Some networking devices: *router, switch, hub, repeater*



# Security Weakness

- The original Internet design **does *not*** take into account of *intentional* attacks
- Possible threats (from Lecture 1):
  - **Interception**: unauthorized viewing
  - **Modification**: unauthorized change
  - **Fabrication**: unauthorized creation
  - **Interruption**: preventing authorized access
- Attacker at any layer can modify the **data** *and* the **header**:
  - Consider the **source IP address** in the IP header, which indicates the sender address
  - Without any protection mechanisms, an attacker can easily send a packet with **spoofed** “source” IP address

## **6.2 Name Resolution and Attacks**

# Naming Schemes and Resolution

- Each peer entity has a **name**
- On a single node, at different layer, the name can be **different**

www.comp.nus.edu.sg

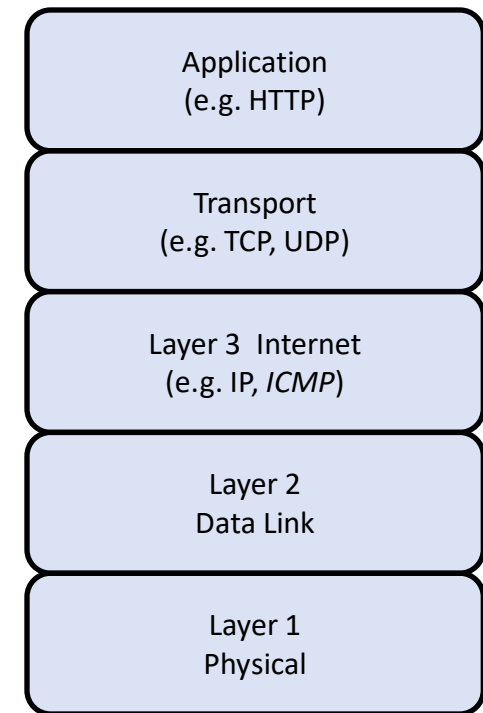
Domain name

132.127.12.3

IP address

01-02-03-04-05-06

MAC address



# Naming Schemes and Resolution

- When a peer entity uses the virtual connection in the layer below, it needs to find out the corresponding *name mapping*
- Example: finding the IP address of a domain name
- **Protocols** that perform name mappings are known as “*resolution*” protocols
- Many initial design of resolution protocols **didn't** take security into account, and thus *easy* for attackers to **manipulate the outcome**

# Resolution Protocols

- Domain Name System (DNS):
  - Maps **domain name** to **IP address**
  - A hierarchical decentralized naming system
  - An attacker can target the association of domain name with IP address

In this module, we only consider a *basic* type of DNS attack

- Address Resolution Protocol (ARP):
  - Associate/map **IP address** (logical address) with/to **MAC address** (physical address)
  - Use a *broadcast mechanism* on a local network
  - An attacker **on the local network** can target the association

# DNS (Domain Name System)

- Given a domain name (e.g. [www.comp.nus.edu.sg](http://www.comp.nus.edu.sg)), its IP address can be found by either looking up a locally stored **host table**, or by **querying a DNS server**. The process is known as ***name resolution***.
- The entity (a.k.a client) that initiates the query is called the ***resolver***
- If the address is found, we say that the domain name is ***resolved***

```
$ nslookup www.comp.nus.edu.sg
Server:      192.168.1.1
Address:     192.168.1.1#53

Non-authoritative answer:
www.comp.nus.edu.sg  canonical name =
www0.comp.nus.edu.sg.
Name:   www0.comp.nus.edu.sg
Address: 137.132.80.57

$
```

The domain name to look up

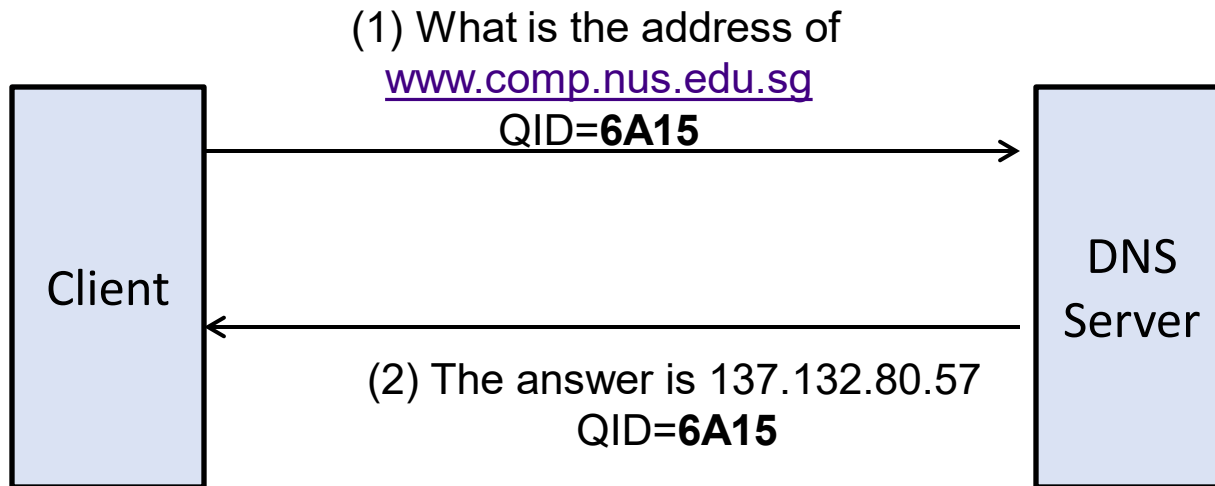
Address of  
the DNS server

Result of  
the query



# (Lightweight) Authentication of DNS Query

- Each query contains a 16-bit number, known as **Query ID (QID)**
- The response from the name server must also contains a QID
- If the QID in the response doesn't match the QID in the query, the client rejects the answer
- Note that **no** encryption/MAC is involved



**Remark:** In the original design consideration, the QID is probably *not* meant for authentication, but as an efficient way to match multiple queries

# Local DNS Attack Scenario

## Alice:

- is using a café's free/open (without protection) WiFi to surf the web
  - wants to visit the webpage [www.comp.nus.edu.sg](http://www.comp.nus.edu.sg)
  - types the domain name into the browser's address bar
- 
- **Alice's browser:**
    - makes a query to a DNS server to determine the IP address
    - then connects to the IP address (after the browser obtains the IP address)

# Local DNS Attack Scenario

## Active Attacker (Mallory):

- We consider an attacker at the *physical layer*:  
for example, he/she can be another person in the café
- Since the WiFi is not protected, the attacker can:
  - Sniff data from the communication channel
  - Spoof data into the communication channel
- Attacker, however, can't remove/modify data already sent by Alice
- Attacker also owns a **Web server** (e.g. with IP address 100.100.100.3), which is a spoofed SoC website

# The Attack (See [PF] page 409)

(1) Alice asks for the address

(2) Mallory sniffs and knows about it.

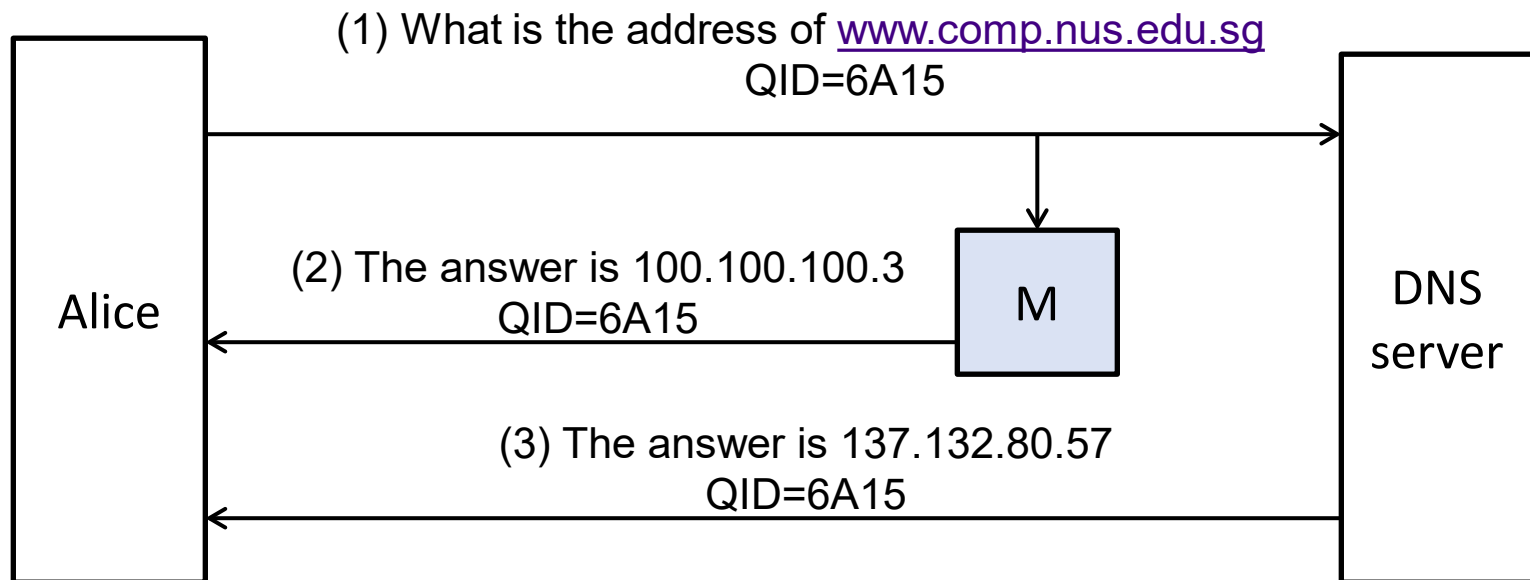
She quickly spoofs a reply with the same QID.

(3) DNS server also sends a reply.

Since Mallory is closer to Alice,

Mallory's reply is likely to reach Alice first.

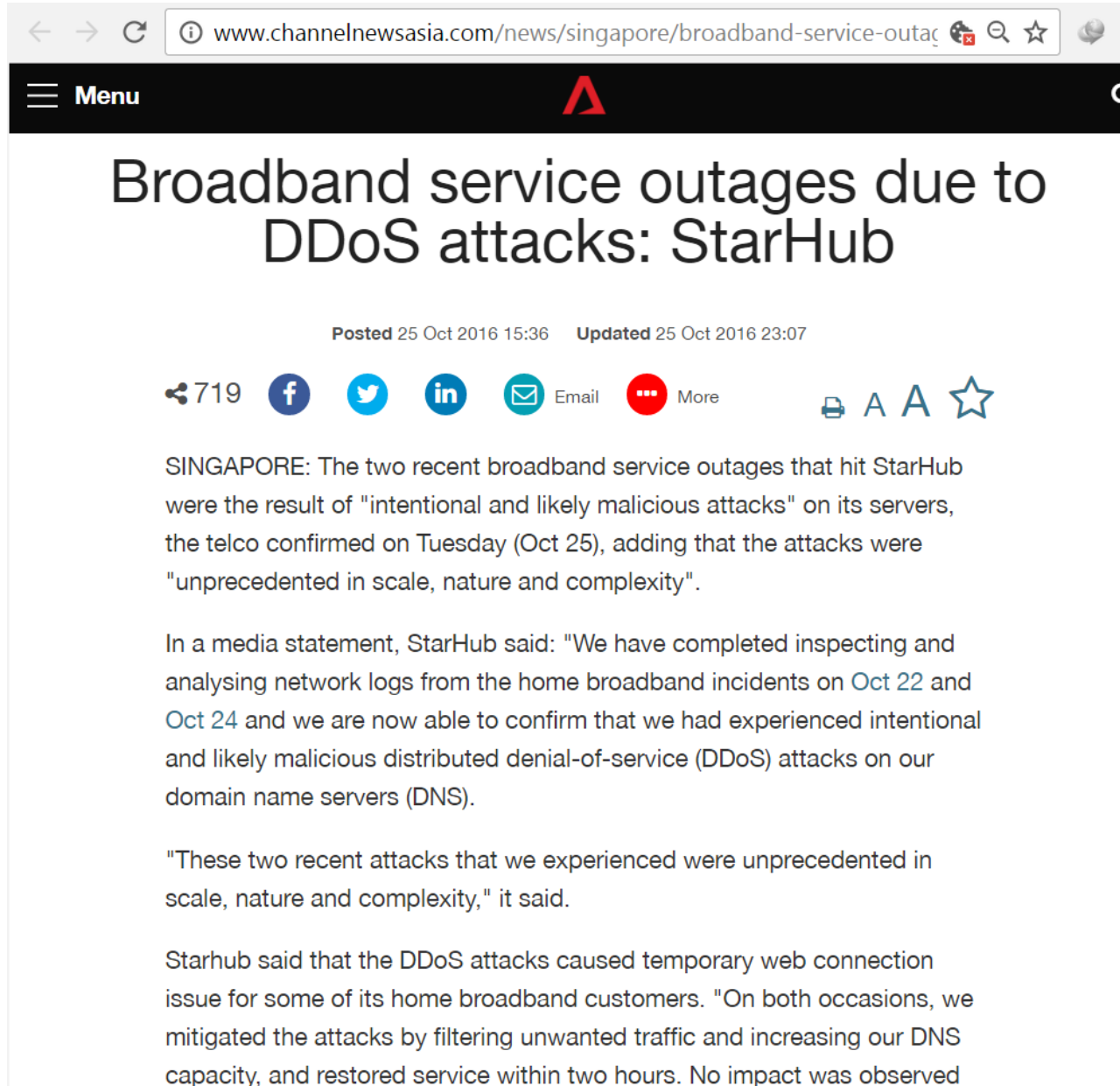
Alice takes the *first* reply as answer, and connects to 100.100.100.3.



# Some Remarks

- DNS operates at the **application layer**.
- Although the attacker is at the physical layer, for ease of analysis, we can assume that the attacker is **just below the application layer**: That is, there exists some virtual connection that can send the message across.  
Hence, the previous slide doesn't mention about the MAC and IP addresses, etc., of the DNS server.
- The DNS is an important component as it resolves the domain name. Hence, an DNS server can be the “**single-point-of-failure**” for the network.
- A DoS attacks, instead of attacking a Web server, could attack the **DNS server** instead.  
E.g. see attack on WikiLeaks (See [PF6.5] pg. 414, [PF] pg. 485),  
StarHub attack 2016 (next slide).

# Recent DNS Attacks



The screenshot shows a web browser displaying a news article. The address bar shows the URL: [www.channelnewsasia.com/news/singapore/broadband-service-outage](http://www.channelnewsasia.com/news/singapore/broadband-service-outage). The page has a black header with a 'Menu' button and a red logo. The main headline is 'Broadband service outages due to DDoS attacks: StarHub'. Below the headline, it says 'Posted 25 Oct 2016 15:36' and 'Updated 25 Oct 2016 23:07'. There are social media sharing icons for Facebook, Twitter, LinkedIn, Email, and a 'More' button. The article text states: 'SINGAPORE: The two recent broadband service outages that hit StarHub were the result of "intentional and likely malicious attacks" on its servers, the telco confirmed on Tuesday (Oct 25), adding that the attacks were "unprecedented in scale, nature and complexity".' It continues: 'In a media statement, StarHub said: "We have completed inspecting and analysing network logs from the home broadband incidents on Oct 22 and Oct 24 and we are now able to confirm that we had experienced intentional and likely malicious distributed denial-of-service (DDoS) attacks on our domain name servers (DNS)."'. The final paragraph says: 'Starhub said that the DDoS attacks caused temporary web connection issue for some of its home broadband customers. "On both occasions, we mitigated the attacks by filtering unwanted traffic and increasing our DNS capacity, and restored service within two hours. No impact was observed'.

www.channelnewsasia.com/news/singapore/broadband-service-outage

Menu

## Broadband service outages due to DDoS attacks: StarHub

Posted 25 Oct 2016 15:36 Updated 25 Oct 2016 23:07

719 f t in Email More

SINGAPORE: The two recent broadband service outages that hit StarHub were the result of "intentional and likely malicious attacks" on its servers, the telco confirmed on Tuesday (Oct 25), adding that the attacks were "unprecedented in scale, nature and complexity".

In a media statement, StarHub said: "We have completed inspecting and analysing network logs from the home broadband incidents on Oct 22 and Oct 24 and we are now able to confirm that we had experienced intentional and likely malicious distributed denial-of-service (DDoS) attacks on our domain name servers (DNS).

"These two recent attacks that we experienced were unprecedented in scale, nature and complexity," it said.

Starhub said that the DDoS attacks caused temporary web connection issue for some of its home broadband customers. "On both occasions, we mitigated the attacks by filtering unwanted traffic and increasing our DNS capacity, and restored service within two hours. No impact was observed

Channel News Asia,  
25 Oct 2016

## **6.3 Denial of Service Attacks**

# DOS Attacks

- **Availability:** the property of being accessible and usable upon demand by an authorized entity
- **Denial of service (DoS):**
  - The prevention of authorized access to resources or the delaying of time-critical operations
  - An attack on availability
- Types of **DoS attacks:**

|               | Stopping Service  | Exhausting Resources  |
|---------------|---|---|
| Local Attack  | <ul style="list-style-type: none"><li>• Process killing</li><li>• Process crashing</li><li>• System reconfiguring</li></ul> | <ul style="list-style-type: none"><li>• Spawning processes</li><li>• Filling up file system</li></ul> |
| Remote Attack | Sending malformed packet attacks  | <i>Packet flooding</i>  |



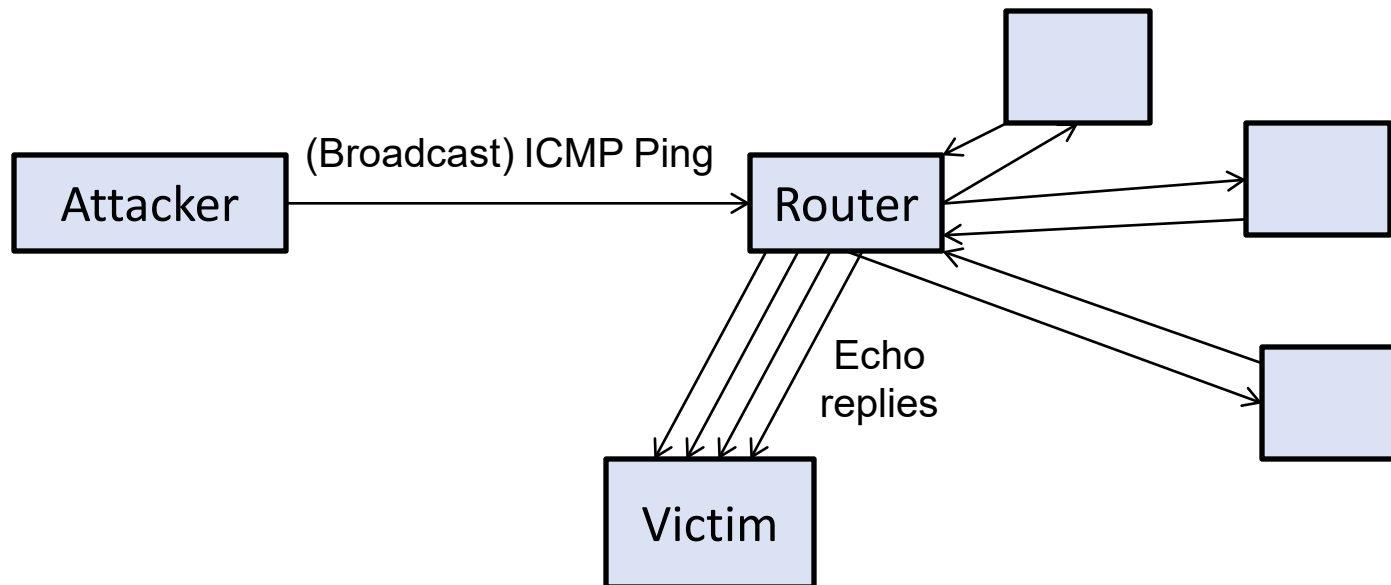
# DoS Attack Types

- **Local attacks** can be more easily tracked
- Sending **malformed attack** remotely does not usually work on updated OSes
- **Packet flooding** attacks:
  - Many effective DOS attacks simply remotely **flood** the victims with overwhelming requests/data
  - The attacker can ***amplify*** small traffic to obtain large traffic, typically by using available **public servers** (Internet infrastructure), such as DNS, NTP, CharGen

# ICMP/Smurf Flood Attack [PF] page 404

- (1) An attacker sends the “**ICMP PING**” request to a router, instructing the router to **broadcast** this request to all local nodes. The request’ source IP address is spoofed with the victim IP address.
- (2) The router broadcasts this **Echo request**.
- (3) Each entity who has received this request, replies to it by sending an “**Echo reply**” to the source, which is the victim

The victim is thus overwhelmed with “**Echo reply**” from the entire network. The attacker takes advantage of the ***amplification*** effect.



# ICMP/Smurf Flood Attack: Preventive Measures

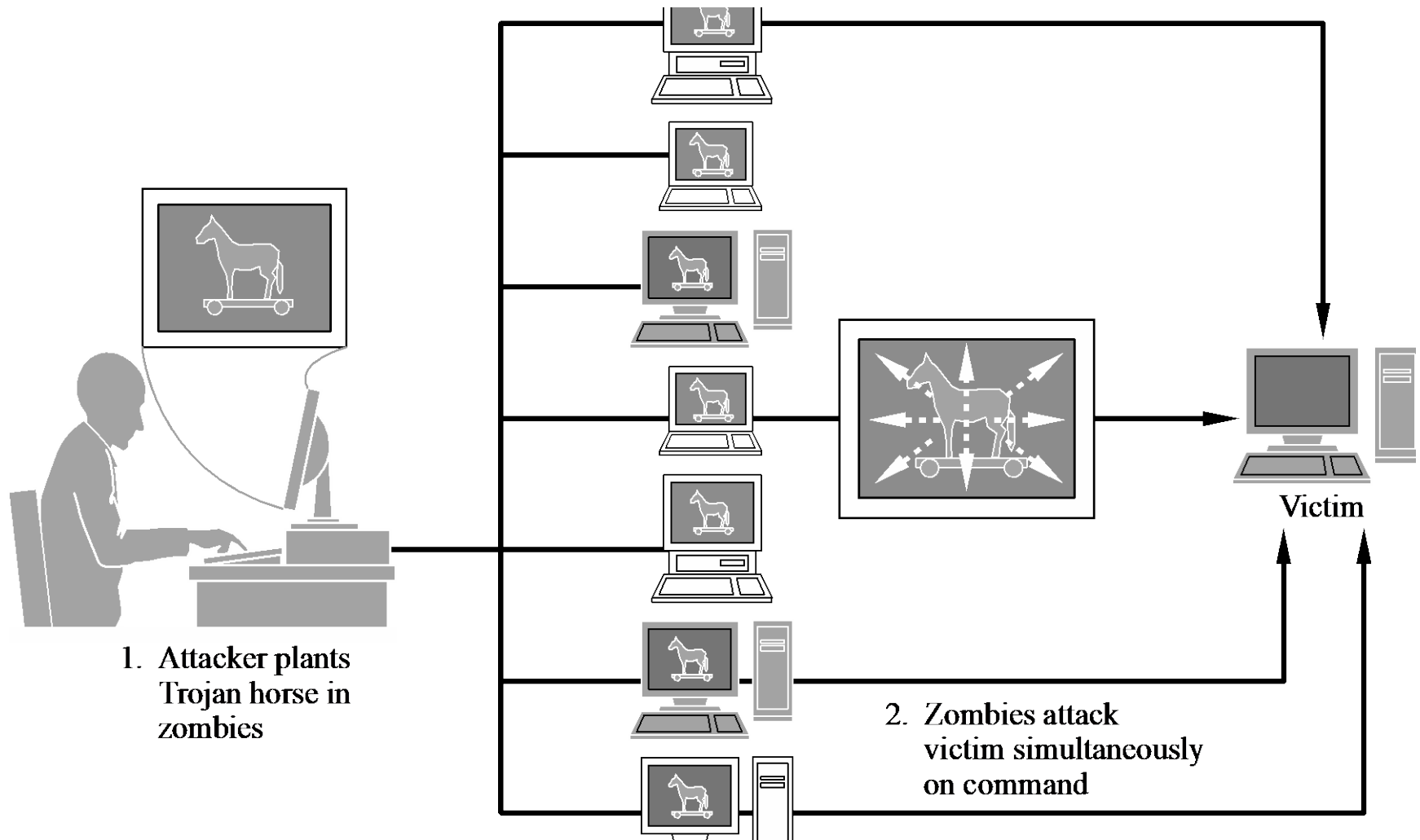
- Is this attack technique still effective?
- *No!*
- Why not?
- Most routers are now configured *not* to broadcast the requests
- To prevent the attack, this measure simply *disables* a feature that was previously thought to be useful

See: IP broadcasting, [http://en.wikipedia.org/wiki/Broadcast\\_address](http://en.wikipedia.org/wiki/Broadcast_address)

# Example of Application-Layer DoS Attack (HTTP Get)

- Simply flood a **Web server** with **HTTP requests**
- Example: **MyDoom worm**, which targeted SCO's website. Attacks started on Feb 12, 2004.  
This is after the SCO's legal actions and public statements against Linux.
- For this attack to be effective, **a large number** of attackers are required.  
(Since each attacker can send requests at a low rate only).
- When DoS is carried out by large number of attackers, this is called **Distributed Denial of Service (DDoS)**.

# Distributed Denial of Service (DDoS)



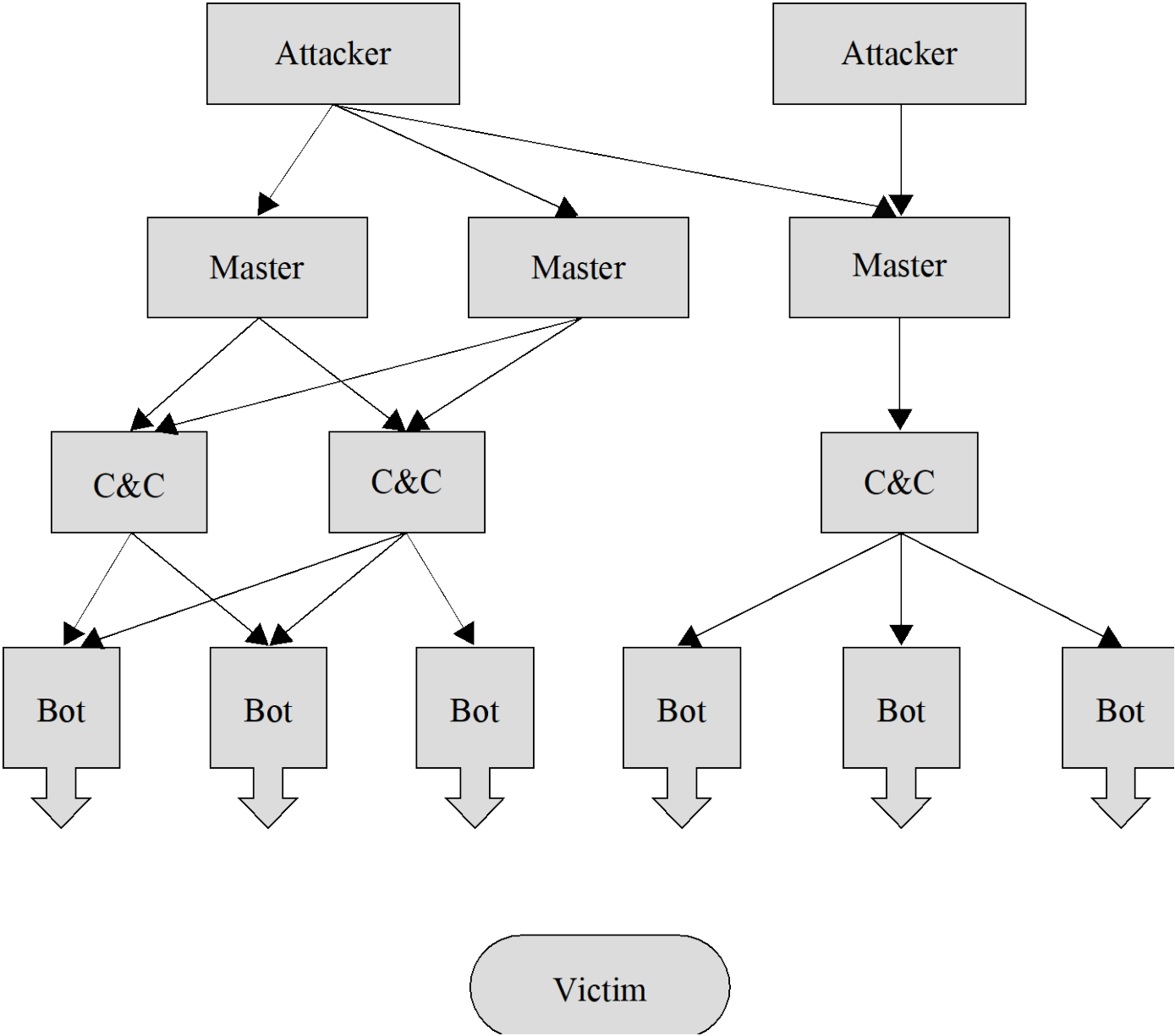
From *Security in Computing, Fifth Edition*, by Charles P. Pfleeger, et al. (ISBN: 9780134085043). Copyright 2015 by Pearson Education, Inc. All rights reserved.

# Botnet

- A **bot** (aka **zombie**) is a compromised machine
- A **botnet** (aka **zombie army**) is a large collection of connected bots, communicating via covert channels
- A botnet has a **command-and-control** mechanism, and thus can be control by an individual to carry out DDOS
- Possible **usages** of a botnet:
  - DDoS flooding, vulnerability scanning, anonymizing HTTP proxy, email address harvesting, cipher breaking!
- See Wiki for the size of known botnets:  
<http://en.wikipedia.org/wiki/Botnet>

**Question:** Why **covert** channels are used by a botnet?

# Botnet



# IoT Devices: a New Trend



TECHNOLOGY

M1 launches commercial IoT network in boost to Singapore's Smart Nation drive



By Kevin Kwang  
@KevinKwangCNA

07 Aug 2017 06:20PM

(Updated: 07 Aug 2017 09:28PM)



Smart rubbish bins enabled with sensors will trigger an alert to cleaners to clear them after a certain level is met. (Photo: M1)

Channel News Asia,  
7 Aug 2017



## 6.4 Useful Tools

# Wireshark (a Packets Analyzer)

- **Wireshark:** a popular free open-source network packet analyzer, <https://www.wireshark.org/>.
- What does Wireshark exactly capture?

Generally performs capturing at the **link layer**.

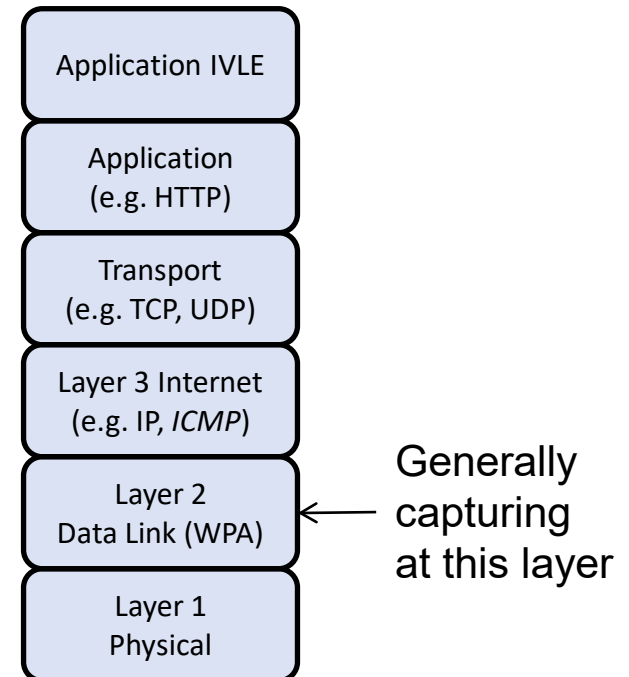
This depends on OS and hardware.

Essentially captures “interactions” between the OS and the network card driver.

See the following FAQ for more info:

<https://ask.wireshark.org/questions/22956/where-exactly-wireshark-does-captures-packets>

(**Demo:** Wireshark)



# Wireshark (a Packets Analyzer)

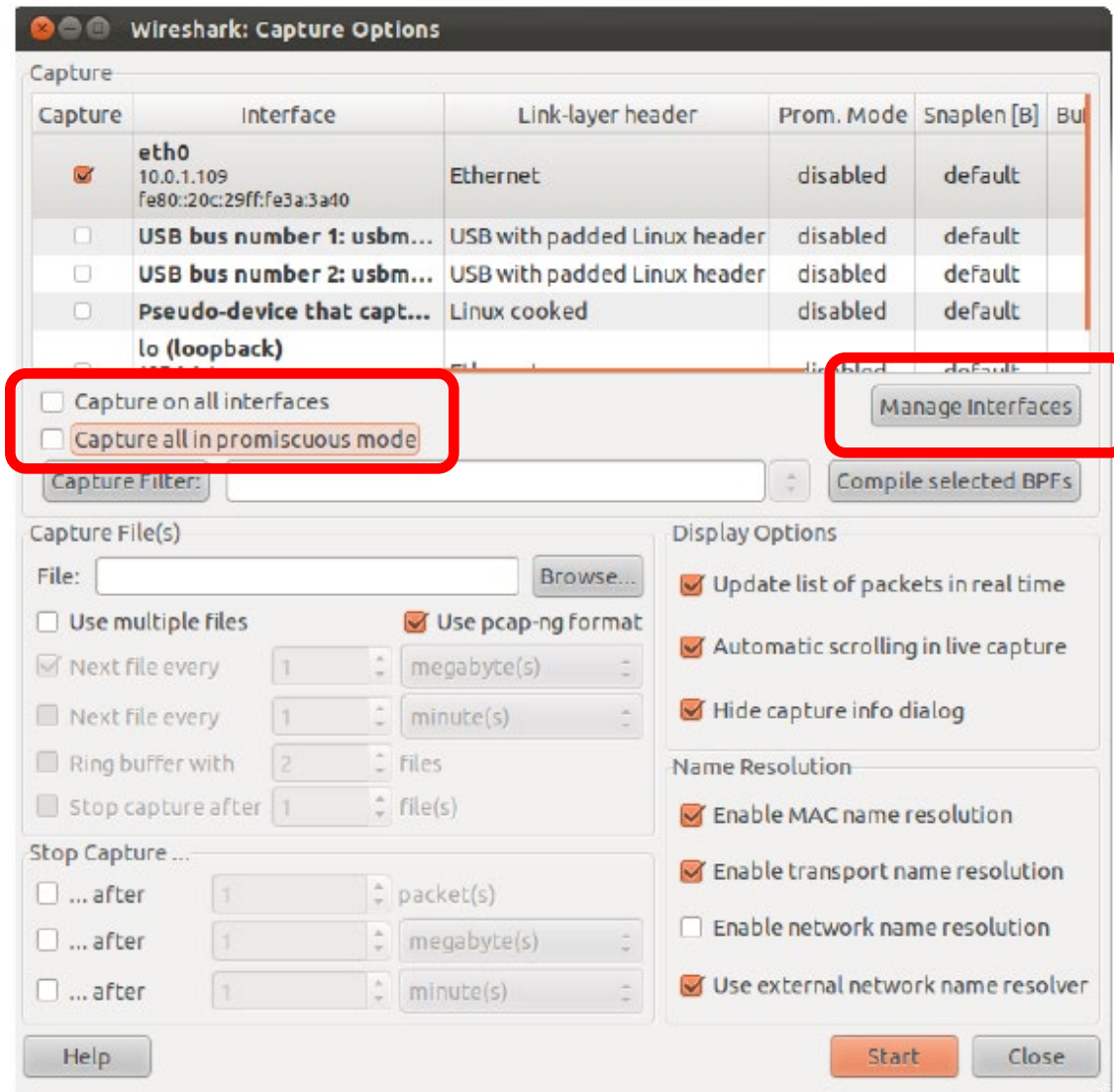
The screenshot displays the Wireshark network protocol analyzer interface. The top menu bar includes File, Edit, View, Go, Capture, Analyze, Statistics, Telephony, Wireless, Tools, and Help. Below the menu is a toolbar with various icons for file operations, capture control, and analysis. A display filter bar shows 'Apply a display filter ... <Ctrl-/>'. The main pane is divided into three sections:

- Packet list pane:** A table of captured packets. The selected packet is 349, a DNS Standard query response from 192.168.0.1 to 192.168.0.21.
- Packet details pane:** A hierarchical view of the selected packet's structure. It shows Ethernet II, Internet Protocol Version 4, User Datagram Protocol, and Domain Name System (response). The DNS section is expanded, showing transaction ID 0x2188, flags, questions, answer RRs, and queries.
- Packet bytes pane:** A hex dump of the packet data, showing the raw bytes of the DNS response.

Red text labels are overlaid on the image to identify these panes: 'Packet list pane' points to the packet list, 'Packet details pane' points to the packet details, and 'Packet bytes pane' points to the packet bytes.

For Wireshark usage, see: Wireshark User's Guide,  
[https://www.wireshark.org/docs/wsug\\_html/](https://www.wireshark.org/docs/wsug_html/)

# Sniffing using Wireshark: Capture Options



## Sniffing using Wireshark: Popup Menu

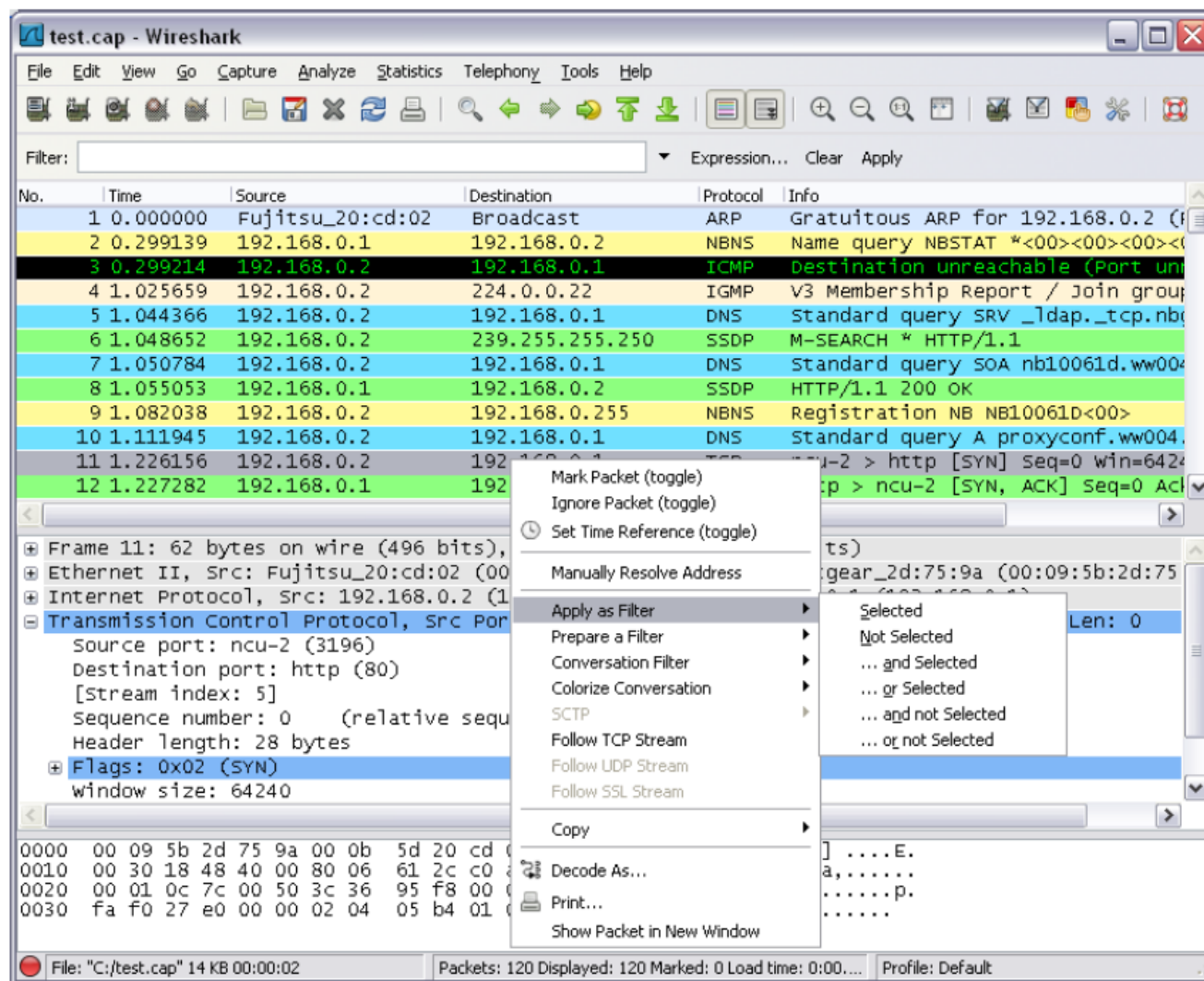
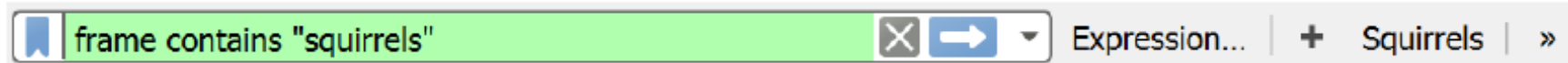


Figure 59. Pop-up menu of the “Packet List” pane

# Sniffing using Wireshark: Filter

You need to specify a good (while-viewing) filter:



## Filter comparison operators

*Table 20. Display Filter comparison operators*

| English     | C-like | Description and example  |
|-------------|--------|--|
| eq          | ==     | Equal. <code>ip.src==10.0.0.5</code>   |
| ne          | !=     | Not equal. <code>ip.src!=10.0.0.5</code>   |
| gt          | >      | Greater than. <code>frame.len &gt; 10</code>   |
| lt          | <      | Less than. <code>frame.len &lt; 128</code>   |
| ge          | >=     | Greater than or equal to. <code>frame.len ge 0x100</code>  |
| le          | <=     | Less than or equal to. <code>frame.len &lt;= 0x20</code>   |
| contains    |        | Protocol, field or slice contains a value. <code>sip.To contains "a1762"</code>                            |
| matches     | ~      | Protocol or text field match Perl regular expression. <code>http.host matches "acme\.(org com net)"</code> |
| bitwise_and | &      | Compare bit field value. <code>tcp.flags &amp; 0x02</code>   |

# Sniffing using Wireshark: Follow TCP Stream

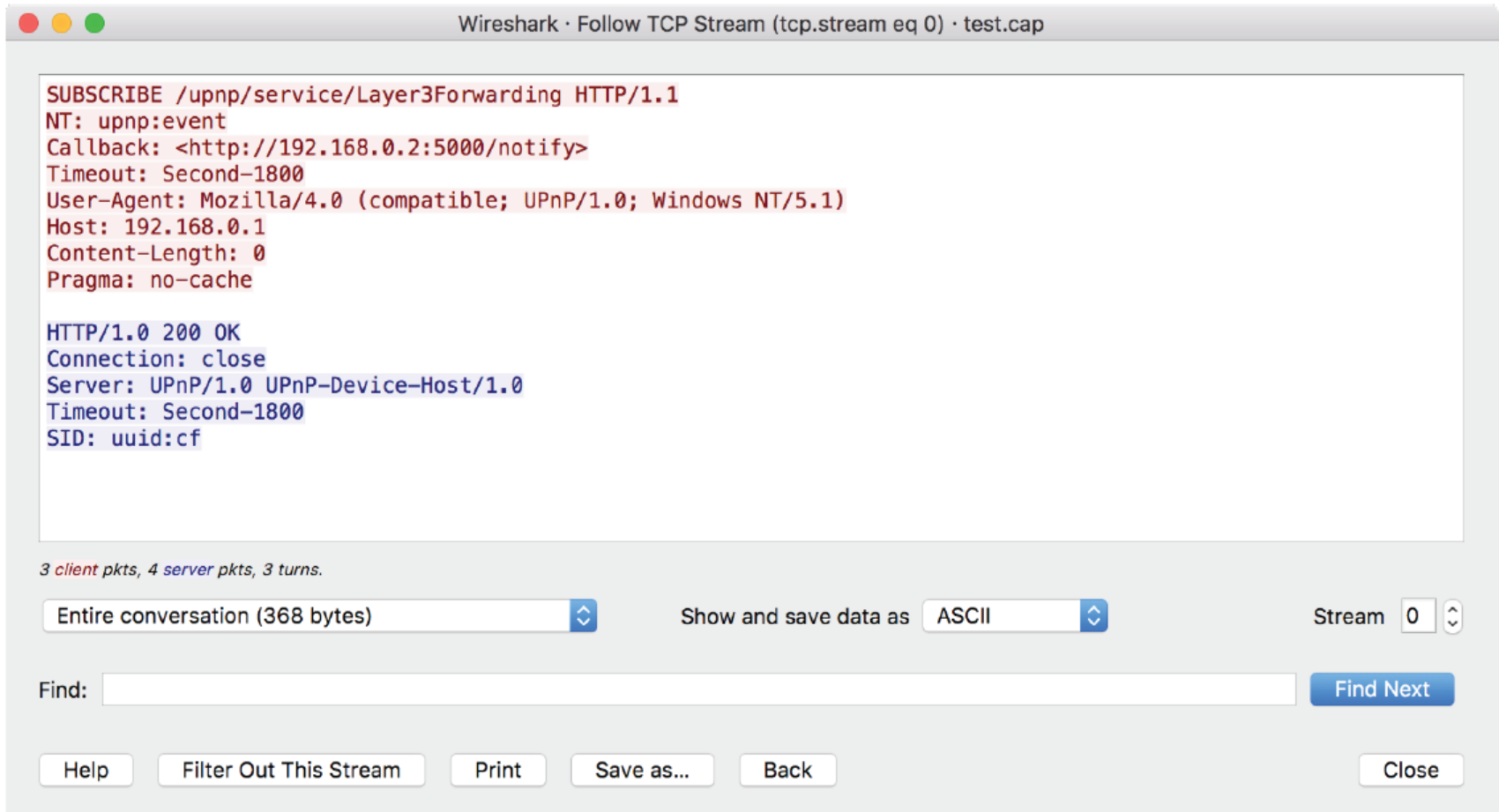
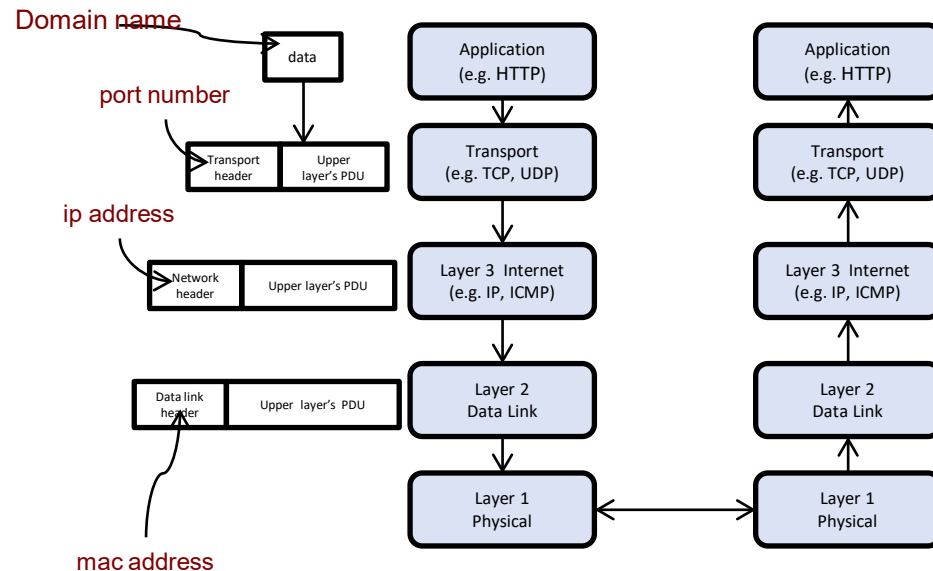


Figure 67. The “Follow TCP Stream” dialog box

# Nmap (Port Scanner)

- What is a **port**?



- When a server receives an incoming packet, it will decide **which application process** to handle that packet based on the port no
- By saying that a process/service is “**listening**” to a particular port, we mean that the process **is running and ready to process** packets with that particular port no



# Nmap (Port Scanner)

- When a port is “***open***”, there exist such a process running in the server
- See **well-known port numbers**:  
[https://en.wikipedia.org/wiki/List\\_of\\_TCP\\_and\\_UDP\\_port\\_numbers#Well-known\\_ports](https://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers#Well-known_ports).
- ***Port scanning***: the process of determining which ports are open on hosts in a network
- Ports are “doors” into each machine, hence port scanning is like **knocking at the doors**
- ***Port scanner***:
  - A tool for performing port scanning
  - Is useful for both attacker and network administrator to scan for vulnerabilities
  - E.g. **Nmap** (very popular!)

# Nmap (Port Scanner)

- **Nmap** is a full featured port-scanning tool:
  - Command-line tool, with GUI frontend
  - Installation:  
`sudo apt-get install nmap, zenmap`
  - Usage: `nmap [Scan Type(s)] [Options]`  
`{target specification}`
  - Examples:  
TCP ACK scan (a stealthier scan): `nmap -sA`  
OS fingerprinting: `nmap -O`  
Service/version detection: `nmap -sV`

**(Demo: Nmap)**

# Nmap: Sample Output

---

```
Nmap scan report
192.168.1.1 / somehost.com (online) ping results
address: 192.168.1.1 (ipv4)
hostnames: somehost.com (user)
The 83 ports scanned but not shown below are in state: closed
```

| Port | State | Service  | Reason  | Product     | Version       | Extra info     |
|------|-------|----------|---------|-------------|---------------|----------------|
| 21   | tcp   | open     | ftp     | syn-ack     | ProFTPD       | 1.3.1          |
| 22   | tcp   | filtered | ssh     | no-response |               |                |
| 25   | tcp   | filtered | smtp    | no-response |               |                |
| 80   | tcp   | open     | http    | syn-ack     | Apache        | 2.2.3 (CentOS) |
| 106  | tcp   | open     | pop3pw  | syn-ack     | poppassd      |                |
| 110  | tcp   | open     | pop3    | syn-ack     | Courier pop3d |                |
| 111  | tcp   | filtered | rpcbind | no-response |               |                |
| 113  | tcp   | filtered | auth    | no-response |               |                |
| 143  | tcp   | open     | imap    | syn-ack     | Courier Imapd | released       |
| 2004 |       |          |         |             |               |                |
| 443  | tcp   | open     | http    | syn-ack     | Apache        | 2.2.3 (CentOS) |
| 465  | tcp   | open     | unknown | syn-ack     |               |                |
| 646  | tcp   | filtered | ldp     | no-response |               |                |
| 993  | tcp   | open     | imap    | syn-ack     | Courier Imapd | released       |
| 2004 |       |          |         |             |               |                |
| 995  | tcp   | open     |         | syn-ack     |               |                |
| 2049 | tcp   | filtered | nfs     | no-response |               |                |
| 3306 | tcp   | open     | mysql   | syn-ack     | MySQL         | 5.0.45         |
| 8443 | tcp   | open     | unknown | syn-ack     |               |                |

```
34 sec. scanned
1 host(s) scanned
1 host(s) online
0 host(s) offline
```

---

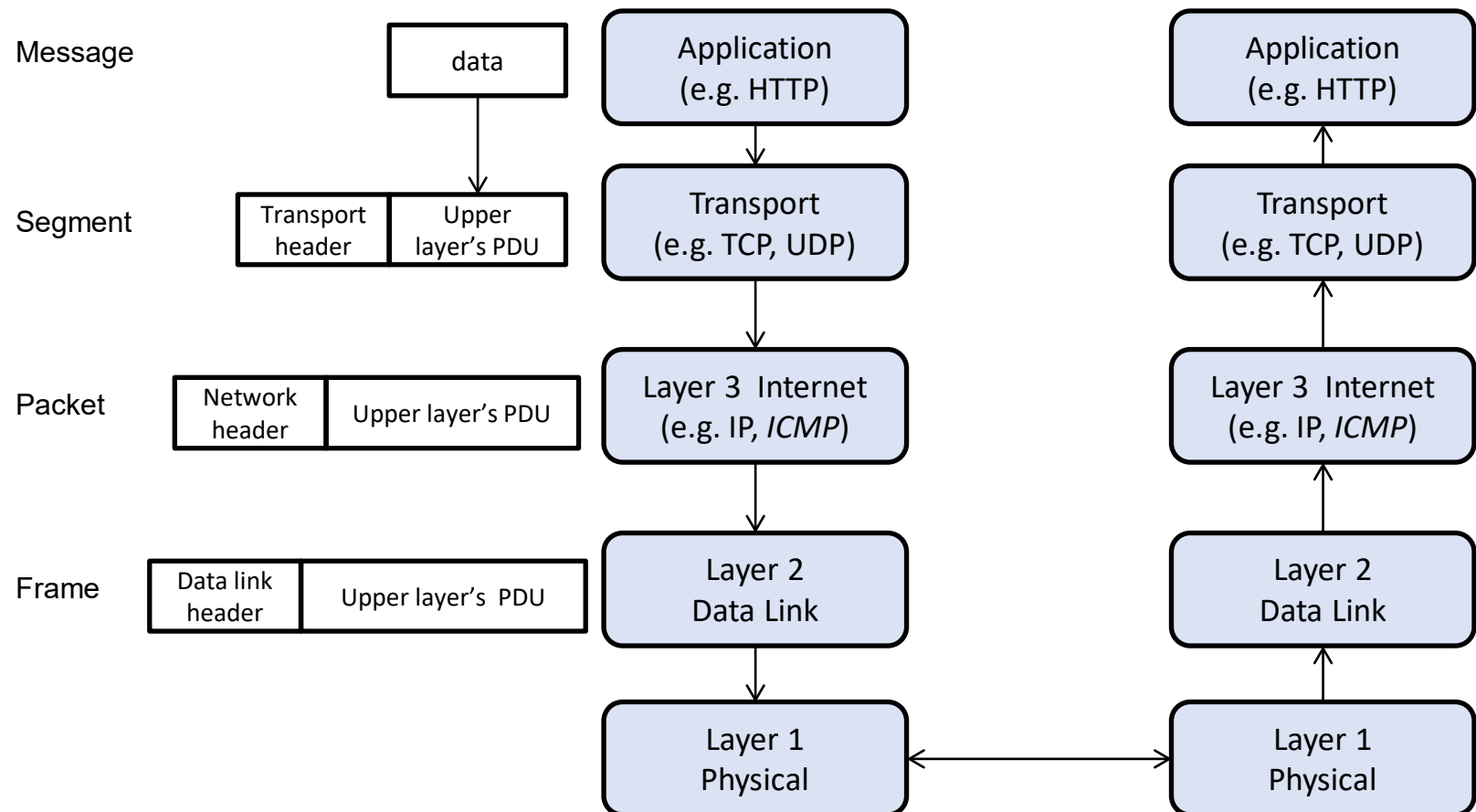
## **6.5 Protection: Securing the Communication Channel using Cryptography**

# Cryptography for Securing Network Communication

- Several cryptographic techniques to achieve **confidentiality** (encryption) and **authenticity** (MAC, PKI, strong authentication) over a public communication channel, even if the adversary can sniff & spoof data
- There are **many** security protocols that essentially achieve that, but operates at **different “layers”**
- Prominent **protocols**:
  - TLS/SSL
  - Wi-Fi Protected Access II (WPA2)
  - Internet Protocol Security (IPsec)

# Security Protocols at Different Layers

- Recall the network layering shown previously
- TLS/SSL, WPA, IPSEC protect **different** layers



# Remarks on Security Protocols and Network Layering

- Very often, when referring to a **security protocol**, we indicate the “**layer**” that the protocol **targets to protect**
- **Complication:** some protections *span across* multiple layers, or do *not* provide full protection of the targeted layer
- When analyzing an **attack**, it is also insightful to figure out at **what layer the attacker resides**
- **Complication:** likewise, some attacks *span across* multiple layers. In such situations, trying hard to pinpoint the layer could sometimes be very confusing

# Remark on Security Protocols and Network Layering

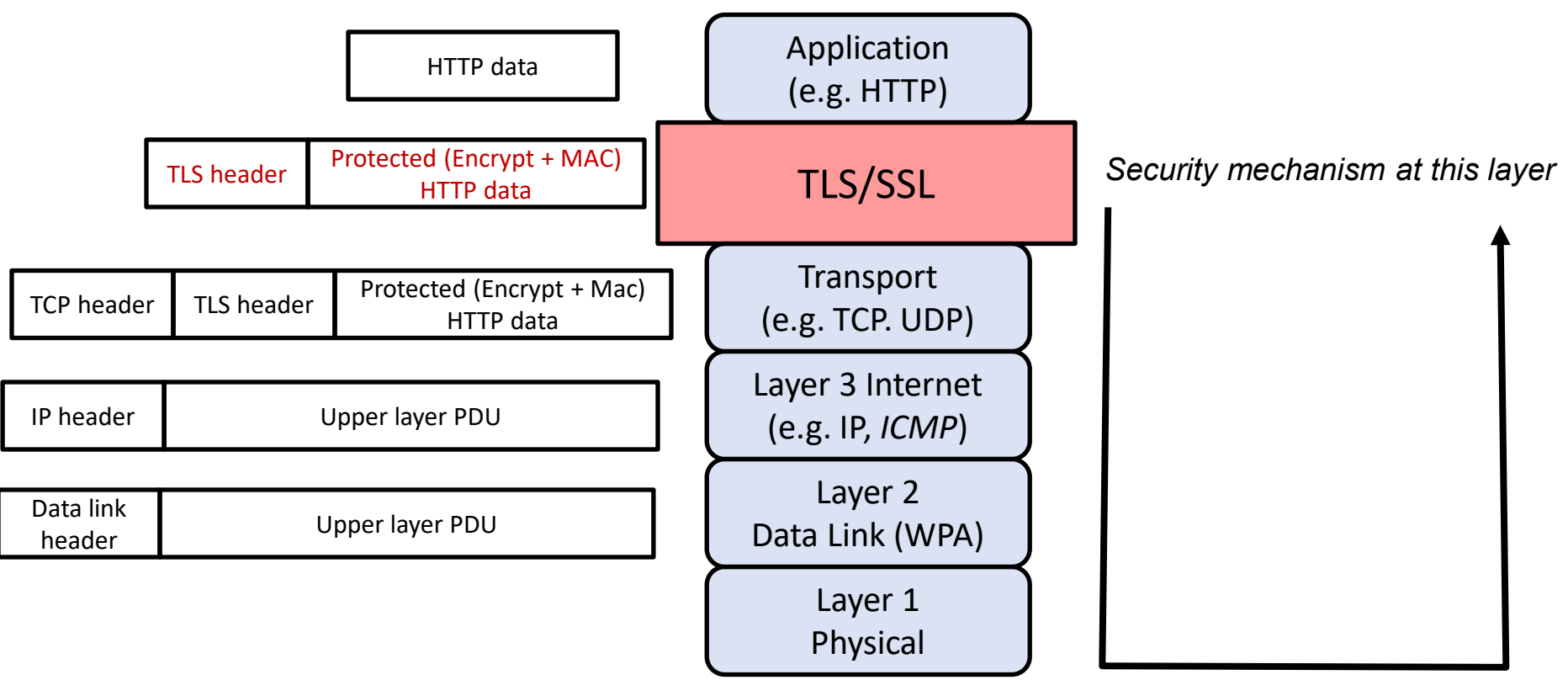
- Below are the **general guideline** and example
- A security protocol that **protects layer  $k$**  would protect information from **that layer and above** *against* an attacker sitting at **layer  $k-1$  and below**
- **Example:** what happens if an attacker resides at layer 1, and there is a security protocol that protects layer 3?
- What is **protected** by the security protocol:  
the information generated in layer 3 and above
- What is ***not* protected**:  
the information generated in layer 2



# 1. SSL/TLS

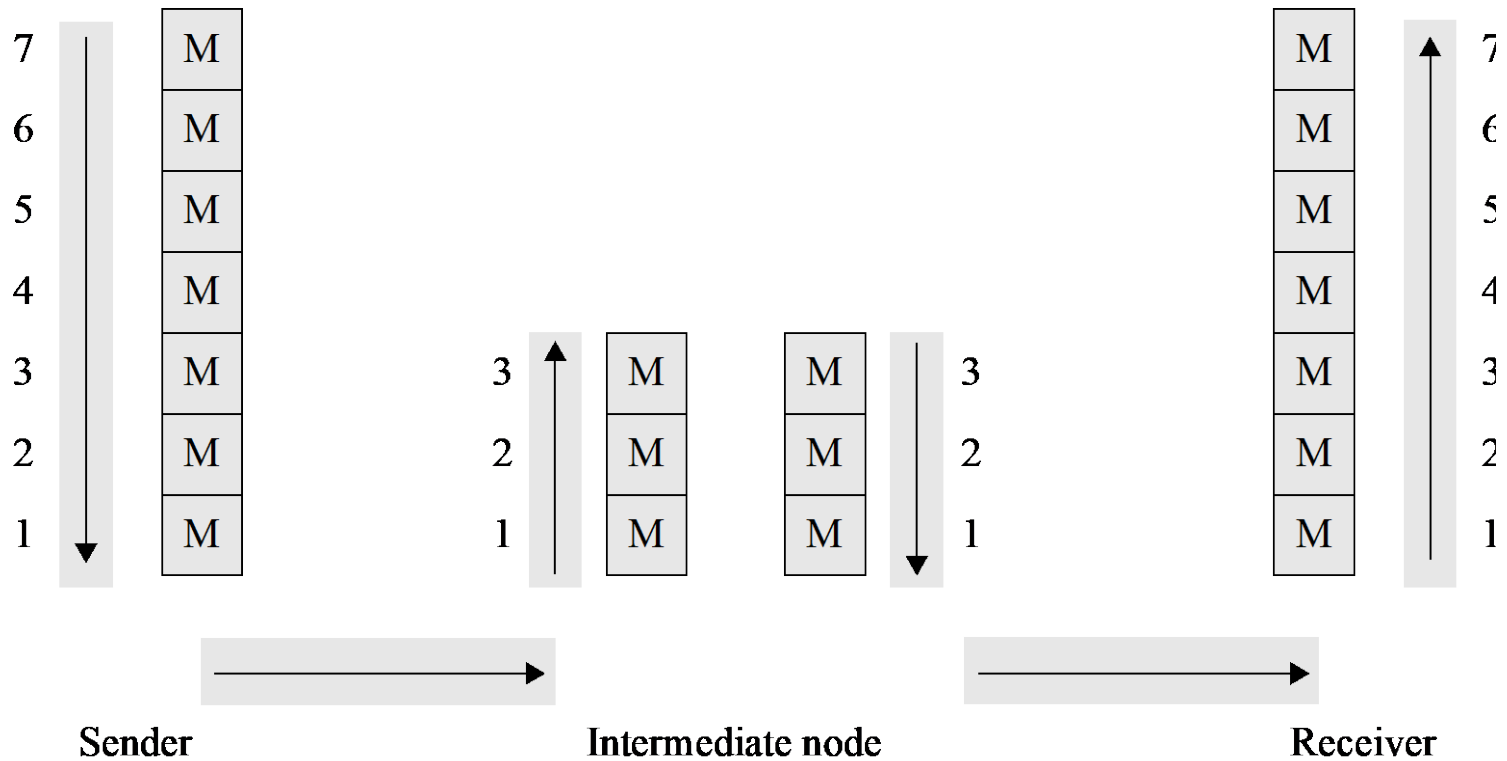
- The SSL/TLS sit on top of **transport layer**
- We can imagine that, when an application (e.g. browser or email agent) wants to send data to the other end point, it first pass the data and the destination IP address to SSL/TLS
- Next, SSL/TLS first “protects” the data using encryption (for confidentiality) and MAC (for authenticity), and then instructs the transport layer to send the protected data
- An **end-to-end encryption** is performed

# SSL/TLS Location



The receiver end-point **decrypts** the received data at the corresponding layer

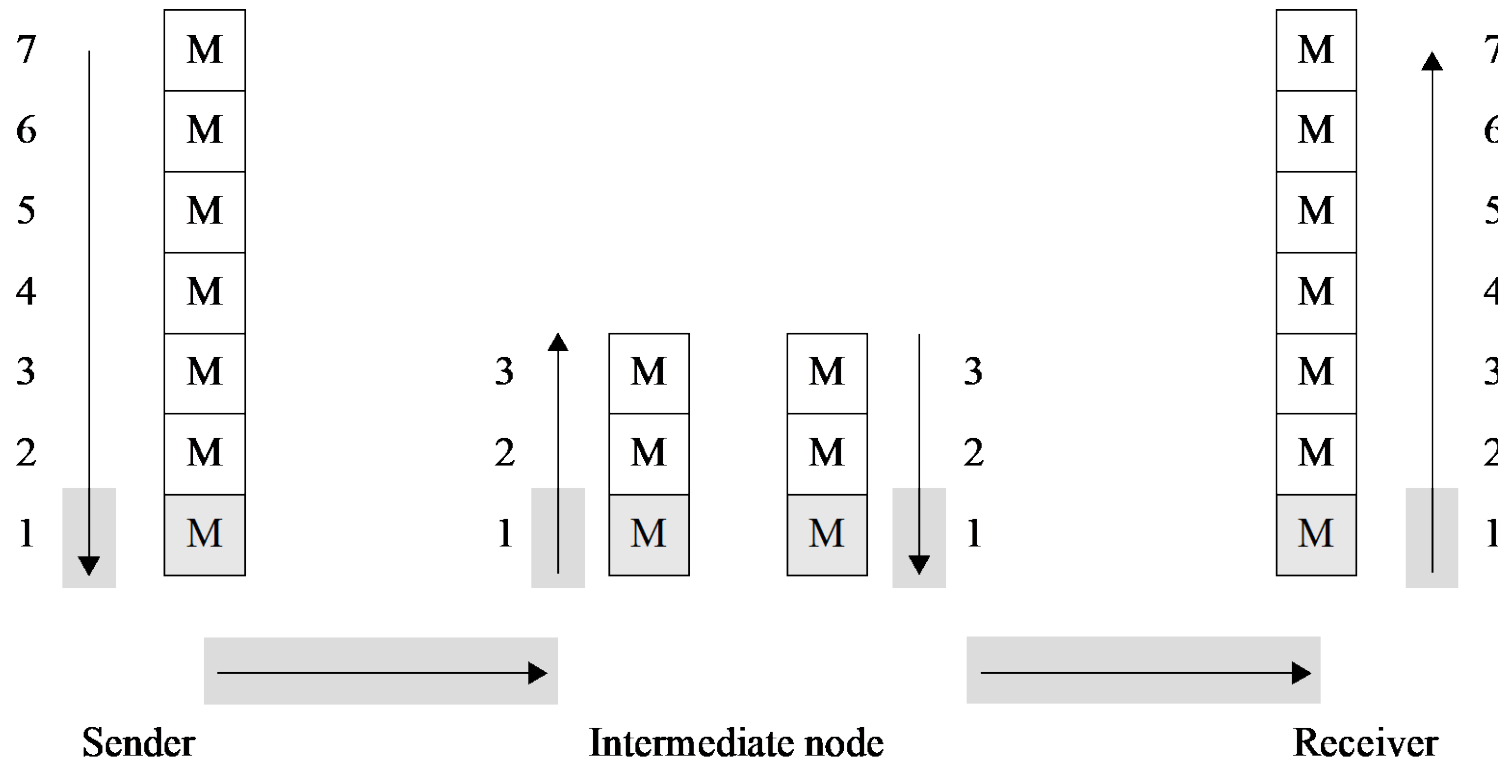
# Terminology: End-to-End Encryption



**M** Encrypted

**M** Plaintext

# Terminology: Link (Hop-by-Hop) Encryption



**M** Encrypted

**M** Plaintext

## Sample Usage Scenario

- Alice accesses LumiNUS web application to upload her report a .pdf to the LumiNUS server
- Note that LumiNUS uses HTTPS, which in turn employs SSL/TLS

Alice's machine carries the following:

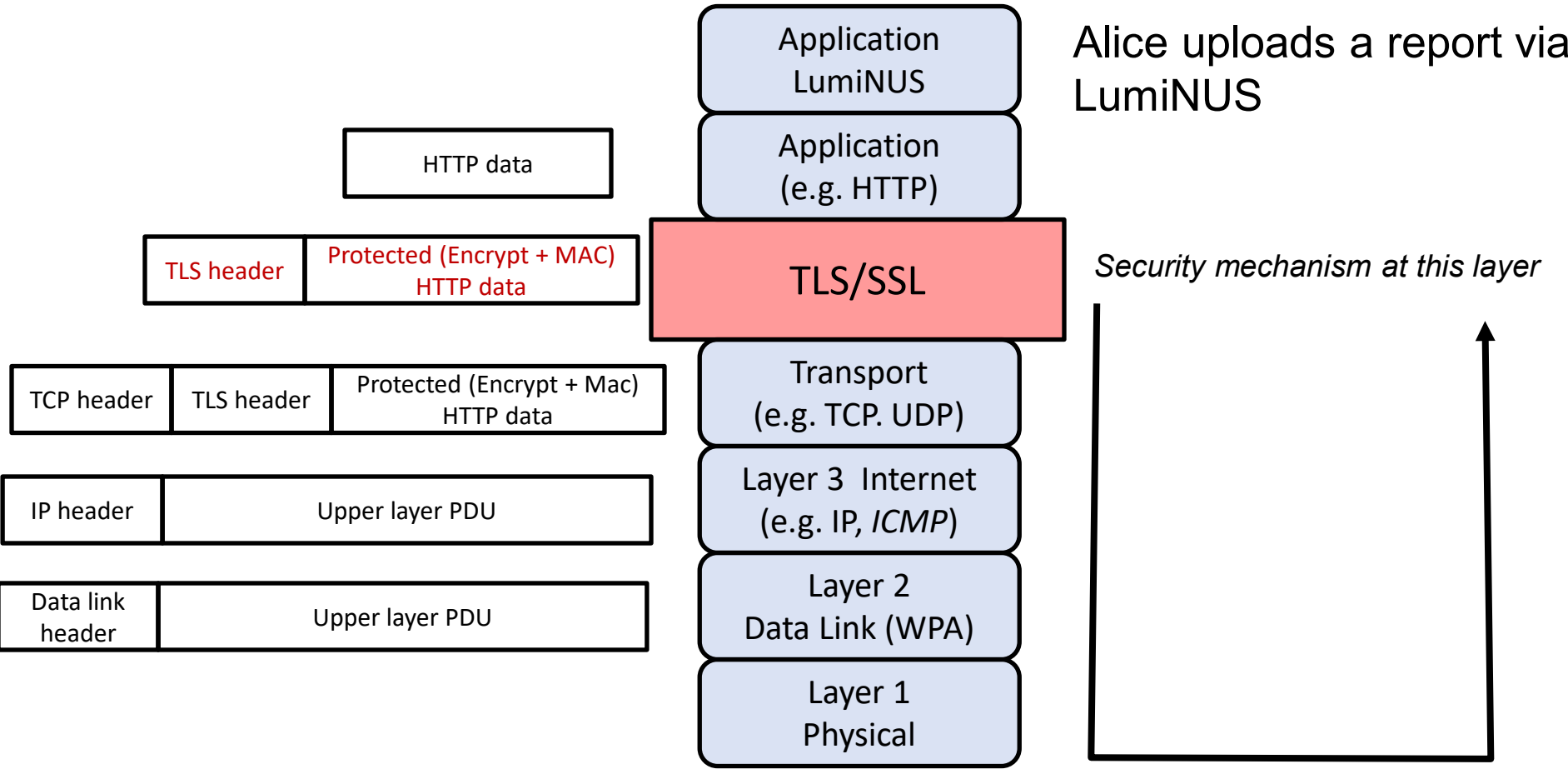
1. The “LumiNUS client” passes the file a .pdf to HTTPS, and then to TLS
2. TLS protects the data by encryption and MAC
3. TLS passes the protected data to the transport layer

## Sample Usage Scenario

The LumiNUS server carries out the following:

1. The transport layer passes the protected data to TLS
  2. TLS decrypt the data and verify the MAC for integrity
  3. TLS passes the decrypted data to LumiNUS application
- ***Remark:*** Many details are omitted in the description. For instance, the “handshaking”, whereby the two parties establishing the session keys.

# Sample Usage Scenario



The receiver end-point **decrypts** the received data at the corresponding layer

# Attack Scenario 1: Attacker at Physical Layer

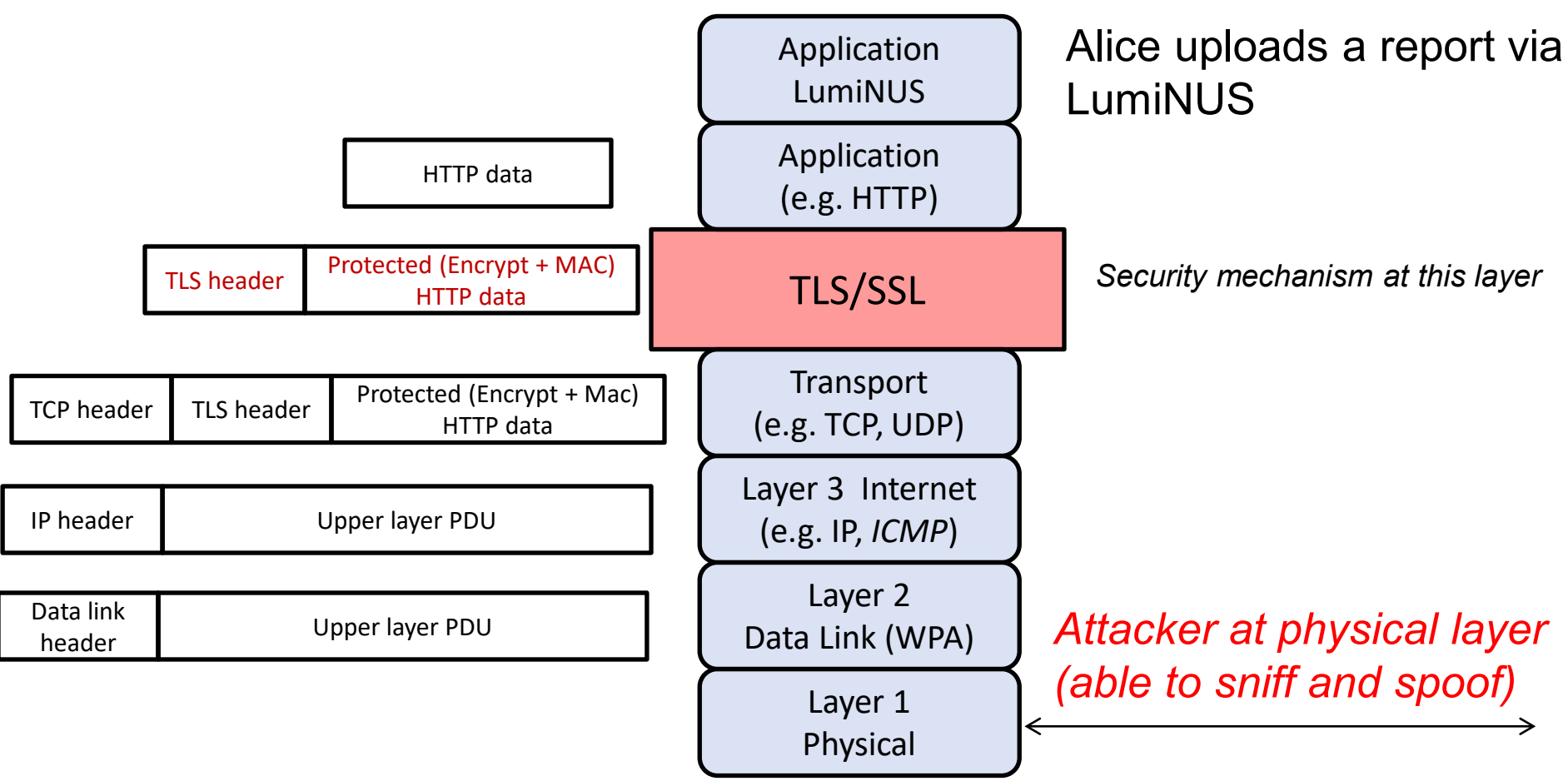
- Suppose that there is an attacker **at the physical layer**, who can sniff and spoof message at that layer
- **For example**, Alice uploads her report in a cafe using a free/open WiFi (without WPA protection).  
Hence, anyone in the café has access to the physical layer, and thus can sniff and spoof messages in that layer.

**Question:** Can the attacker learn:

1. Alice's uploaded report?
2. The fact that Alice is visiting LumiNUS website (i.e. can the attacker learn the website's IP address)?



# Attack Scenario 1: Attacker at Physical Layer



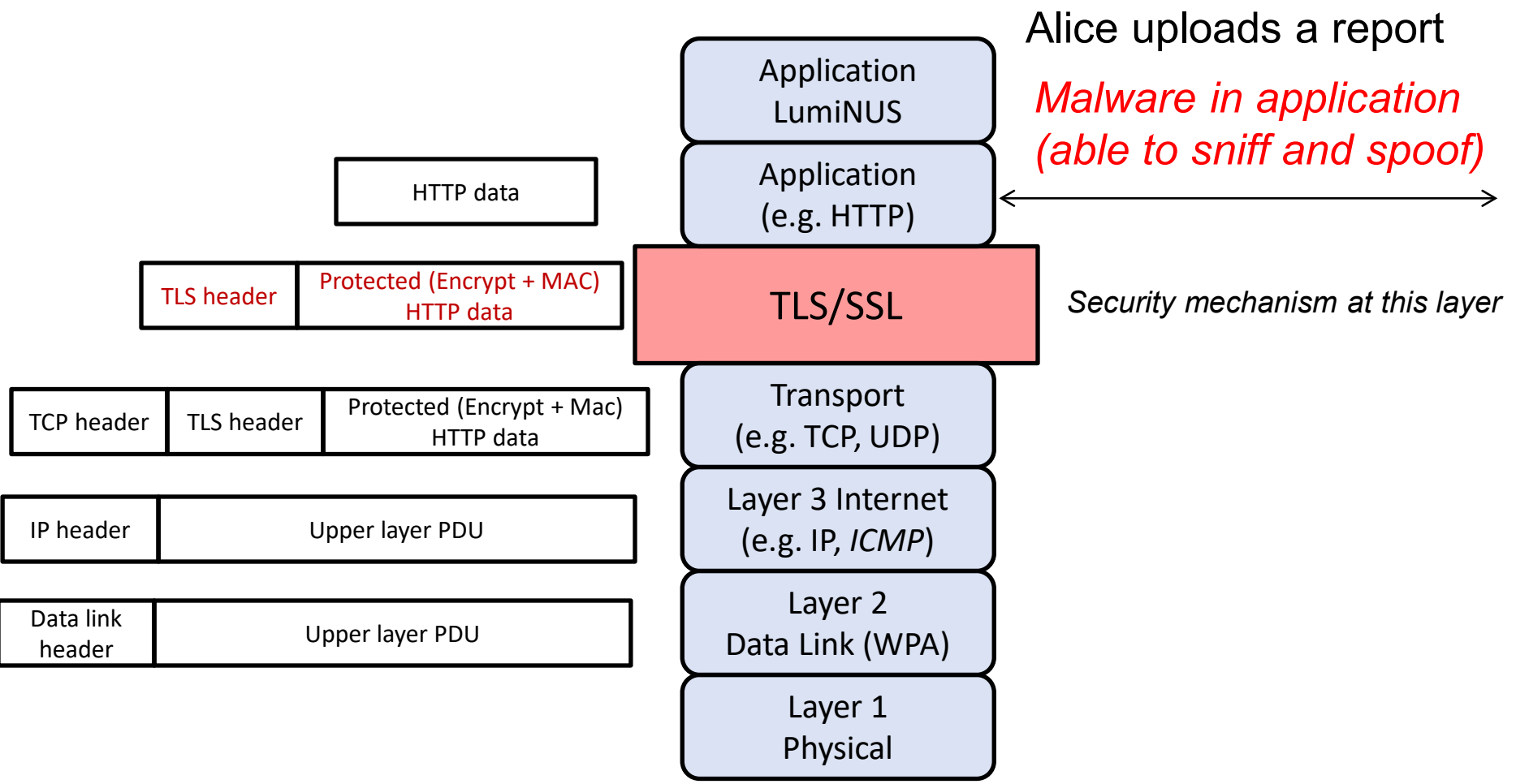
## Attack Scenario 2: Attacker at Application Layer

- Suppose that there is an adversary at the **application layer**
- For example, a **malicious JavaScript** is injected into LumiNUS and being executed by Alice's browser

**Question:** Can the malicious script learn:

1. Alice's report?
2. *Alice's MAC address?* (an interesting issue)

# Attack Scenario 2: Attacker at Application Layer



## 2. WPA2

### ***WiFi Protected Access II (WPA2):***

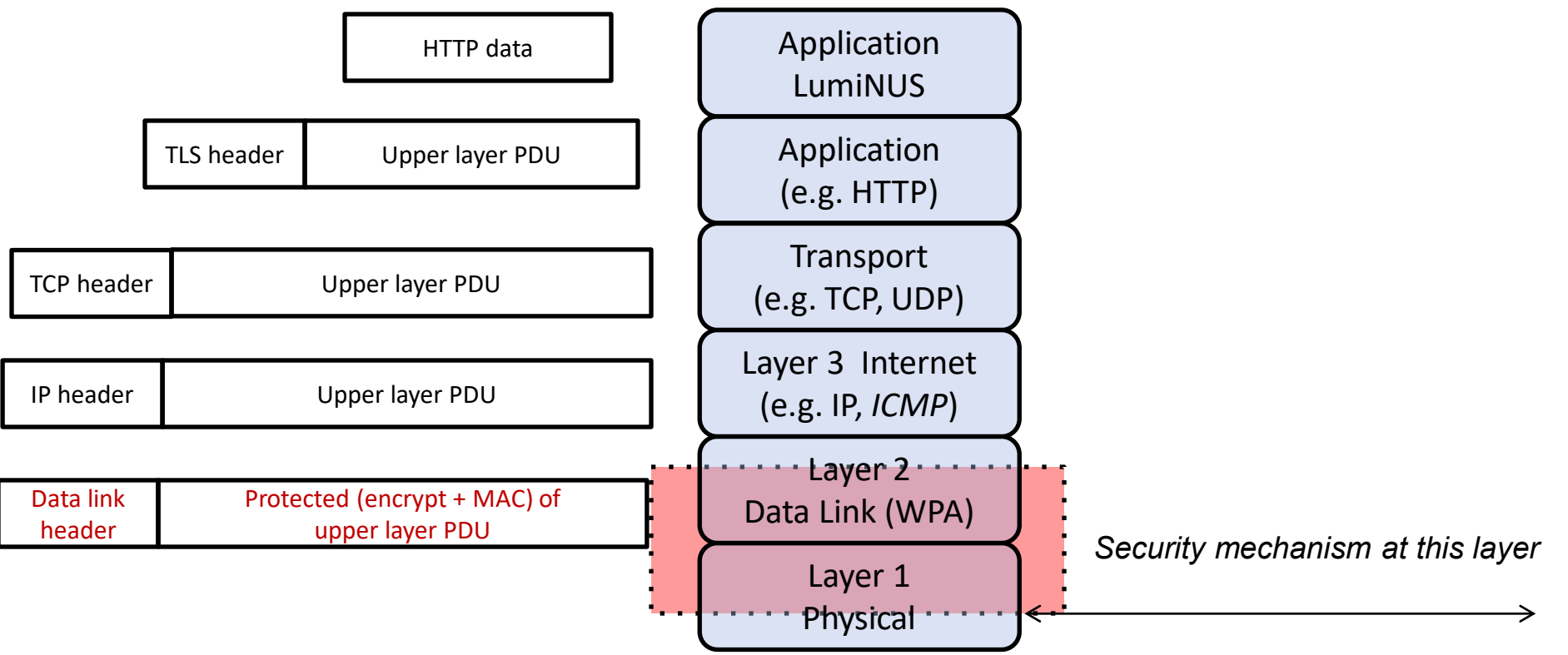
- A popular protocol employed in home WiFi access point
- More secure than WEP (broken), WPA

The protections provided:

- WPA2 provides protection at **layer 2 (link)**  
**and layer 1 (physical)**
- Note: *not* all information in layer 2 are protected

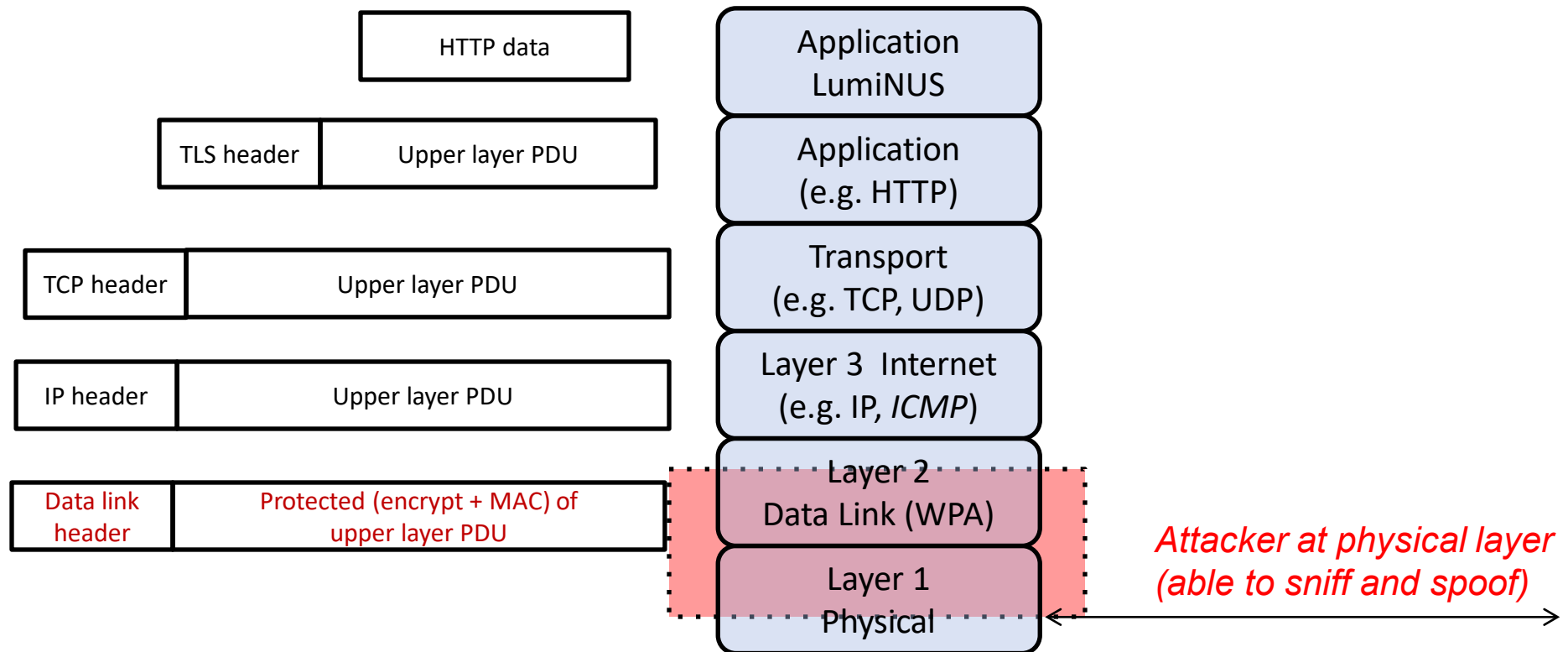
# WPA2 and Network Layers

Alice uploads a report



# Attacker at Physical Layer

Alice uploads a report

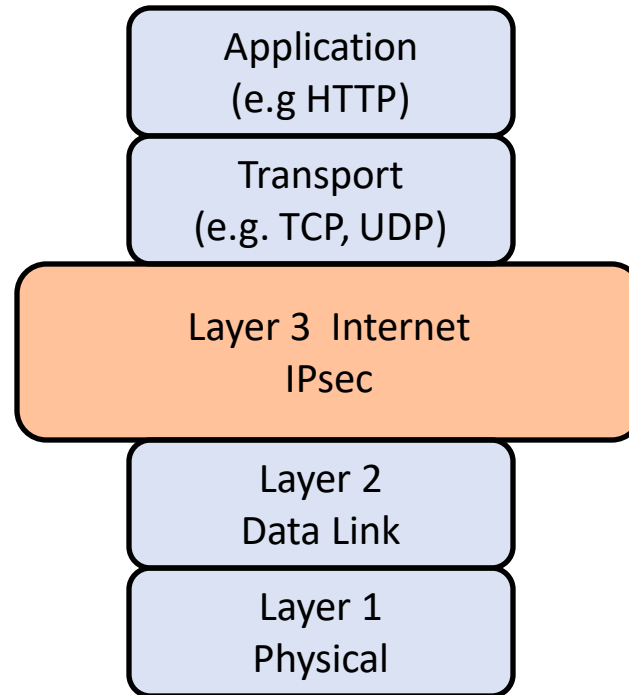


*Question: Can the attacker learn:*

- 1. Alice's report?*
- 2. The fact that Alice is visiting LumiNUS website?*
- 3. The MAC address (link layer)?* **not clear**

### 3. IPsec

- IPsec provides **integrity/authenticity** protection of IP addresses, but ***not* their confidentiality**:
  - Hence, attackers are unable to “spoof” the source IP address
  - But they can still learn the source and destination IP addresses of the sniffed packets



# Remarks on IPsec

IPsec is a mechanism whose goal is *to protect the IP layer*

Its description:

- “**Internet Protocol Security (IPsec)** is a [protocol suite](#) for securing [Internet Protocol \(IP\)](#) communications by [authenticating](#) and [encrypting](#) each [IP packet](#) of a communication session. IPsec includes protocols for establishing [mutual authentication](#) between agents at the beginning of the session and negotiation of [cryptographic keys](#) to be used during the session. IPsec can be used in protecting data flows between a pair of hosts (*host-to-host*), between a pair of security gateways (*network-to-network*), or between a security gateway and a host (*network-to-host*).<sup>[1]</sup>
- Internet Protocol security (IPsec) uses cryptographic security services to protect communications over Internet Protocol (IP) networks. IPsec supports network-level peer authentication, data origin authentication, data integrity, data confidentiality (encryption), and replay protection.
- IPsec is an end-to-end security scheme operating in the [Internet Layer](#) of the [Internet Protocol Suite](#), while some other Internet security systems in widespread use, such as [Transport Layer Security \(TLS\)](#) and [Secure Shell \(SSH\)](#), operate in the [upper layers](#) at Application layer. Hence, only IPsec protects any application traffic over an IP network. Applications can be automatically secured by IPsec at the IP layer.”

-Wiki



# Question

## Question: Explain the underlined sentences

- “Internet Protocol Security (IPsec) is a protocol suite for securing Internet Protocol (IP) communications by authenticating and encrypting each IP packet of a communication session. IPsec includes protocols for establishing mutual authentication between agents at the beginning of the session and negotiation of cryptographic keys to be used during the session. IPsec can be used in protecting data flows between a pair of hosts (*host-to-host*), between a pair of security gateways (*network-to-network*), or between a security gateway and a host (*network-to-host*).<sup>[1]</sup>
- Internet Protocol security (IPsec) uses cryptographic security services to protect communications over Internet Protocol (IP) networks. IPsec supports network-level peer authentication, data origin authentication, data integrity, data confidentiality (encryption), and replay protection.
- IPsec is an end-to-end security scheme operating in the Internet Layer of the Internet Protocol Suite, while some other Internet security systems in widespread use, such as Transport Layer Security (TLS) and Secure Shell (SSH), operate in the upper layers at Application layer. Hence, only IPsec protects any application traffic over an IP network. Applications can be automatically secured by IPsec at the IP layer.”

-Wiki

## **6.6 Protection: Firewall**

# Motivation

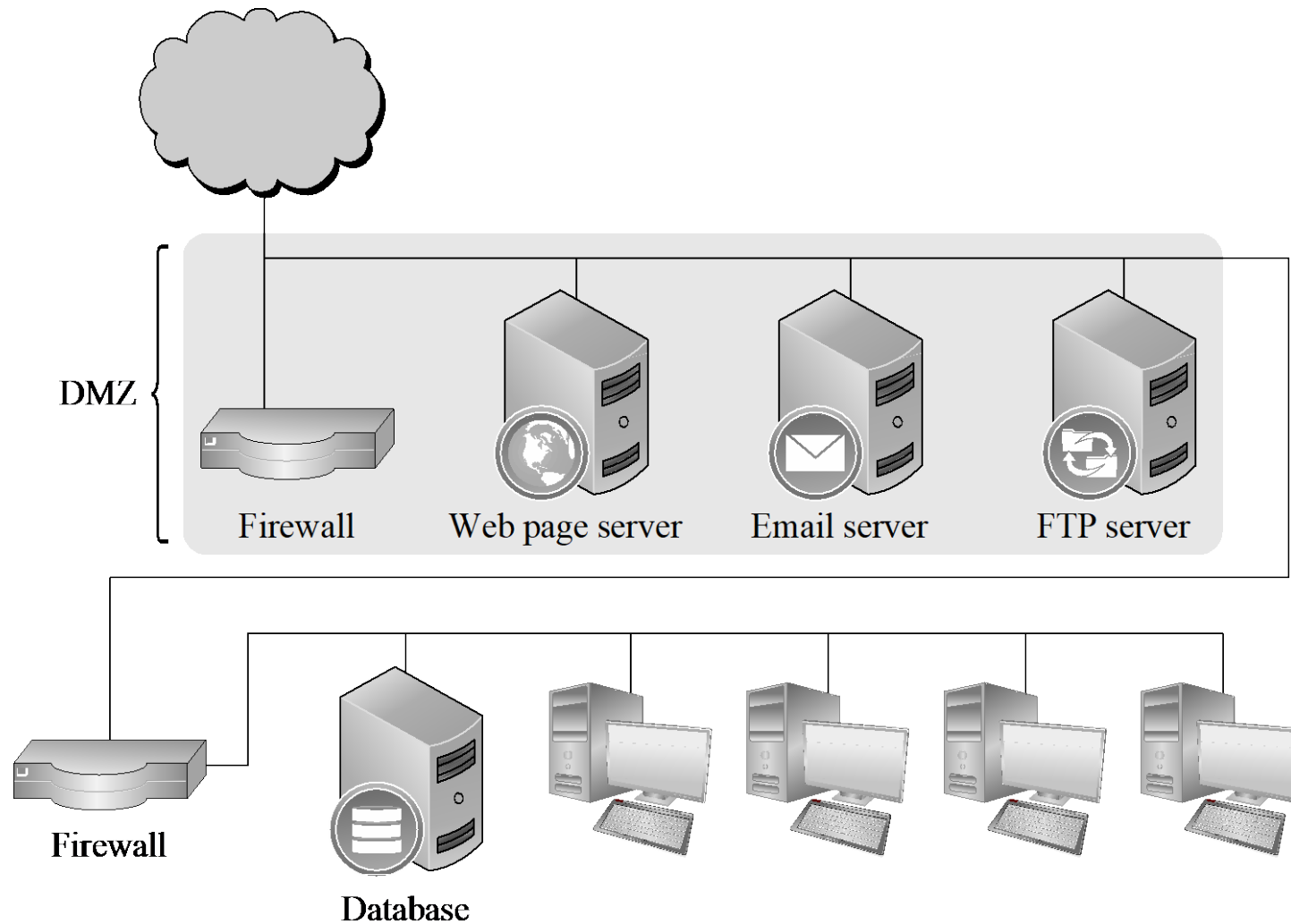
- Having SSL/TLS and WPA2 is still ***not*** sufficient to protect the network:
  - There are concerns of **DoS** that they can't prevent
  - Many services and applications **can't be protected** by SSL/TLS & WPA2: e.g. **DNS spoofing**  
(It is not practical, due to efficiency, to establish SSL/TLS to the DNS server for DNS query)
- There is a need to control **the flow of traffic** between networks, especially between the **untrusted** public network (Internet) and the **trusted** internal network

# Firewall

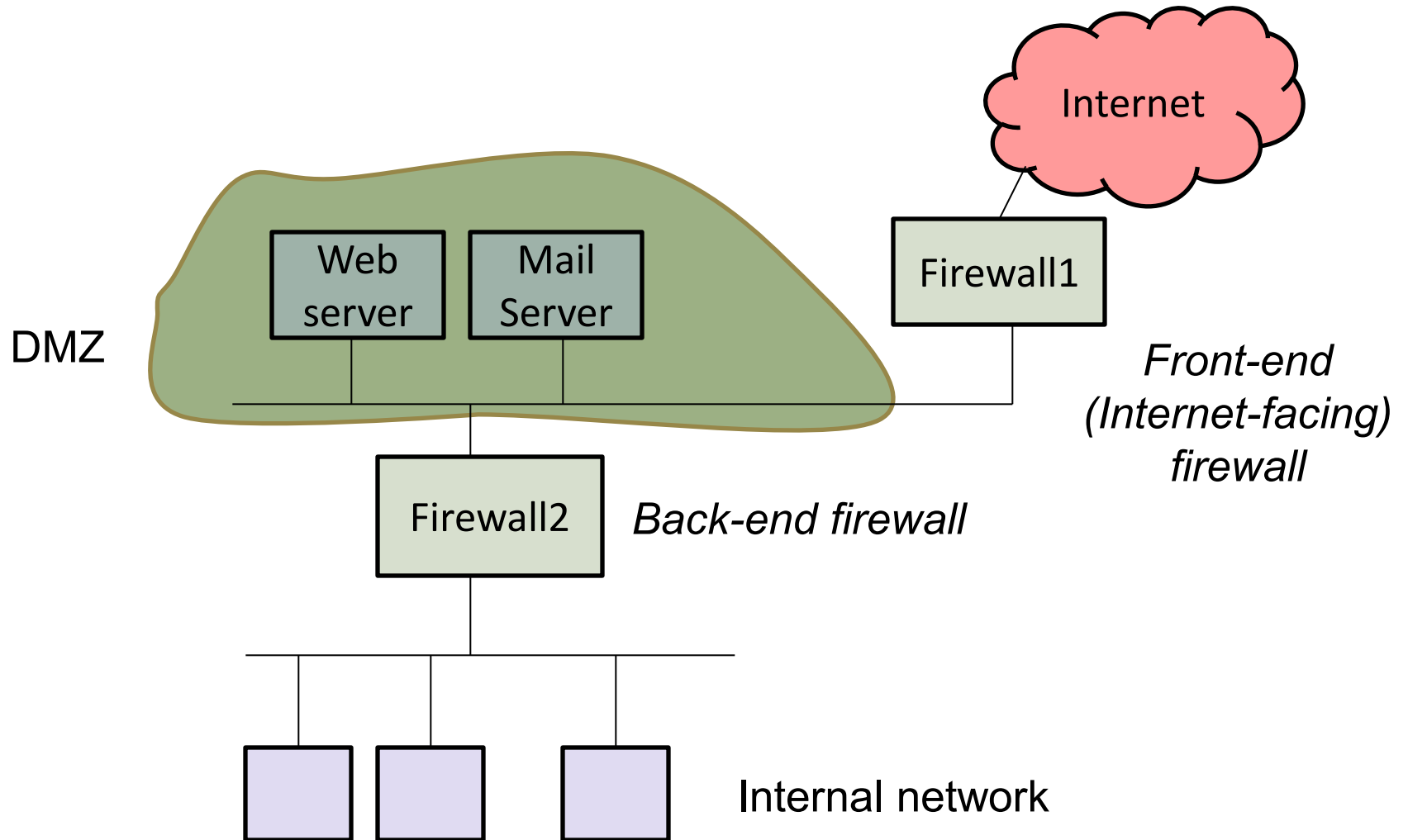
- **Firewall:**
  - Sits at border between networks
  - Looks at addresses, services, other characteristics of traffic
  - Controls what traffic is allowed to enter the network (ingress filtering), or leave the network (egress filtering)
- **Definition:** *“Firewall are devices or programs that control the flow of network traffic between networks or hosts that employ differing security postures.”*

(From “Guidelines on Firewalls and Firewall Policy”, NIST, special publication 800-41  
<http://csrc.nist.gov/publications/nistpubs/800-41-Rev1/sp800-41-rev1.pdf>.)
- **DMZ (Demilitarized Zone):**
  - A small **sub-network** that exposes the organization’s external service to the (untrusted) Internet
  - The original military term: an area between states in which military operations are not permitted

# Demilitarized Zone (DMZ)



# Typical 2-Firewall Setting



# Firewall Design (Read [PF] pg 453)

- A firewall enforces a **set of rules** provided by the network administrator
- Examples of rules for Firewall2 (back-end firewall):
  - Block HTTP
  - Allow from Internal network to Mail Server: SMTP, POP3
- Examples of rules for Firewall1 (front-end firewall):
  - Allow from anywhere to Mail Server: SMTP only
- How the rules are to be specified differs on different devices and software
- The next slide gives a typical design/configuration (from [PF] pg. 453)

# Sample Firewall Configuration

| Rule No  | Protocol Type | Source Address | Destination Address | Source Port | Designation Port | Action |
|----------|---------------|----------------|---------------------|-------------|------------------|--------|
| 1        | TCP           | *              | 192.168.1.*         | *           | 25               | Permit |
| 2        | TCP           | *              | 192.168.1.*         | *           | 69               | Permit |
| 3        | TCP           | 192.168.1.*    | *                   | *           | 80               | Permit |
| 4        | TCP           | *              | 192.168.1.18        | *           | 80               | Permit |
| 5        | TCP           | *              | 192.168.1.*         | *           | *                | Deny   |
| 6        | UDP           | *              | 192.168.1.*         | *           | *                | Deny   |
|          |               |                |                     |             |                  |        |
| <i>n</i> | *             | *              | *                   | *           | *                | Deny   |

\* (means “any”), which matches **any** values

The table is processed in a **top-down manner**

The **first matching rule** determines the **action taken**

Hence: put your most specific rule *first*, and put your most general rule *last*



# Types of Firewall

The textbook [PF] lists **6** types of firewalls

The literature, including NIST's document (NIST 800-41), usually groups firewalls into **3 types**:

## **1. (Traditional) packet filters:**

- Filters packets based on information in *packet headers*

## **2. Stateful-inspection (packet filters):**

- Maintains a *state table* of all active connections
- Filters packets based on active connection states

## **3. Application proxy:**

- Understands application logic
- Acts as a relay of application-level traffic

(Details are not required for this module)

# Comparison of Firewall Types [PF]

Optional

| <b>Packet Filter</b>                                   | <b>Stateful Inspection</b>   | <b>Application Proxy</b>  | <b>Circuit Gateway</b>   | <b>Guard</b>   | <b>Personal Firewall</b>  |
|--|--|---|--|--|---|
| Simplest decision-making rules, packet by packet       | Correlates data across packets   | Simulates effect of an application program  | Joins two subnetworks  | Implements any conditions that can be programmed                               | Similar to packet filter, but getting more complex  |
| Sees only addresses and service protocol type          | Can see addresses and data   | Sees and analyzes full data portion of pack   | Sees addresses and data  | Sees and analyzes full content of data   | Can see full data portion   |
| Auditing limited because of speed limitations          | Auditing possible  | Auditing likely   | Auditing likely  | Auditing likely  | Auditing likely   |
| Screens based on connection rules                      | Screens based on information across multiple packets—in either headers or data | Screens based on behavior of application  | Screens based on address   | Screens based on interpretation of content                                     | Typically, screens based on content of each packet individually, based on address or content            |
| Complex addressing rules can make configuration tricky | Usually preconfigured to detect certain attack signatures                      | Simple proxies can substitute for complex decision rules, but proxies must be aware of application's behavior | Relatively simple addressing rules; make configuration straightforward | Complex guard functionality; can be difficult to define and program accurately | Usually starts in mode to deny all inbound traffic; adds addresses and functions to trust as they arise |

From *Security in Computing, Fifth Edition*, by Charles P. Pfleeger, et al. (ISBN: 9780134085043). Copyright 2015 by Pearson Education, Inc. All rights reserved.

## **6.7 Protection: Network Security Management**

# Network Security Management

There is a need to **continuously** monitor and adjust network characteristics

(Details on this are omitted. See [PF6.9])

Some terms:

- **Security Operations Center (SOC):**
  - A centralized unit in an organization that monitors the IT systems and deals with security issues
- **Security Information and Event Management (SIEM):**
  - Pronounced as “SIM”
  - Provides **real-time analysis** of security alerts generated by network hardware and applications
  - May include the following **capabilities**:  
data aggregation & correlation, event alerting,  
compliance report generation, forensic analysis

# Security Information and Event Management (SIEM)

