Distributed Systems

Chun-Feng Liao

廖峻鋒

Department of Computer Science

National Chengchi University

Distributed Systems

Networking

Chun-Feng Liao

廖峻鋒

Dept. of Computer Science

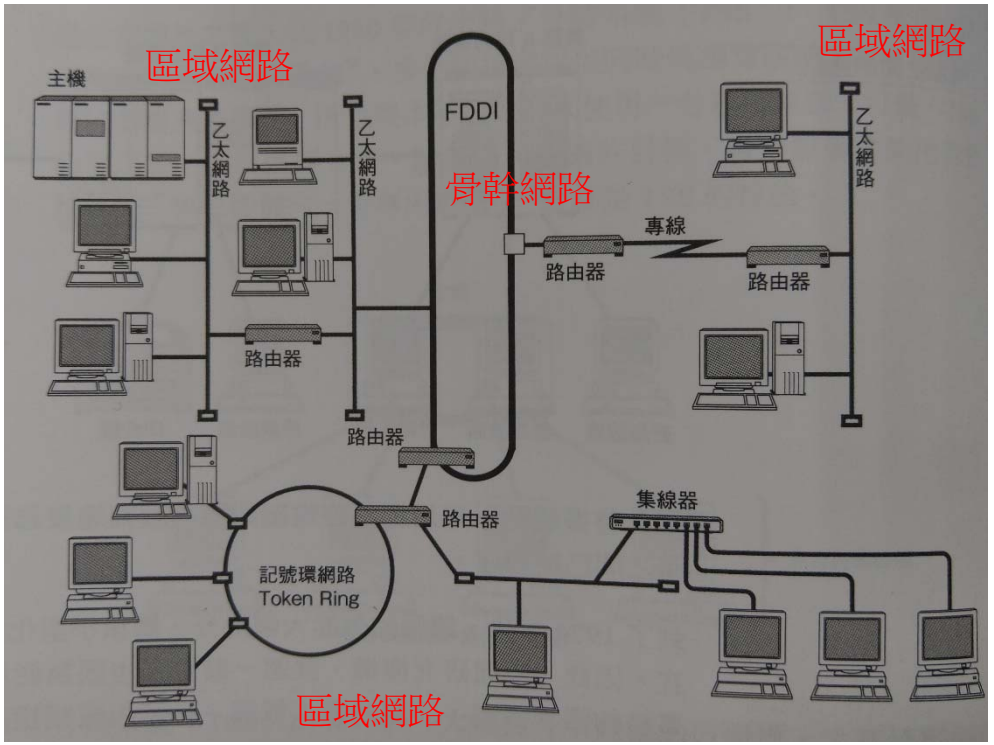
National Chengchi University

路由器管理區網

Introduction

router vs. switch => router 是傳封包， switch 是區網

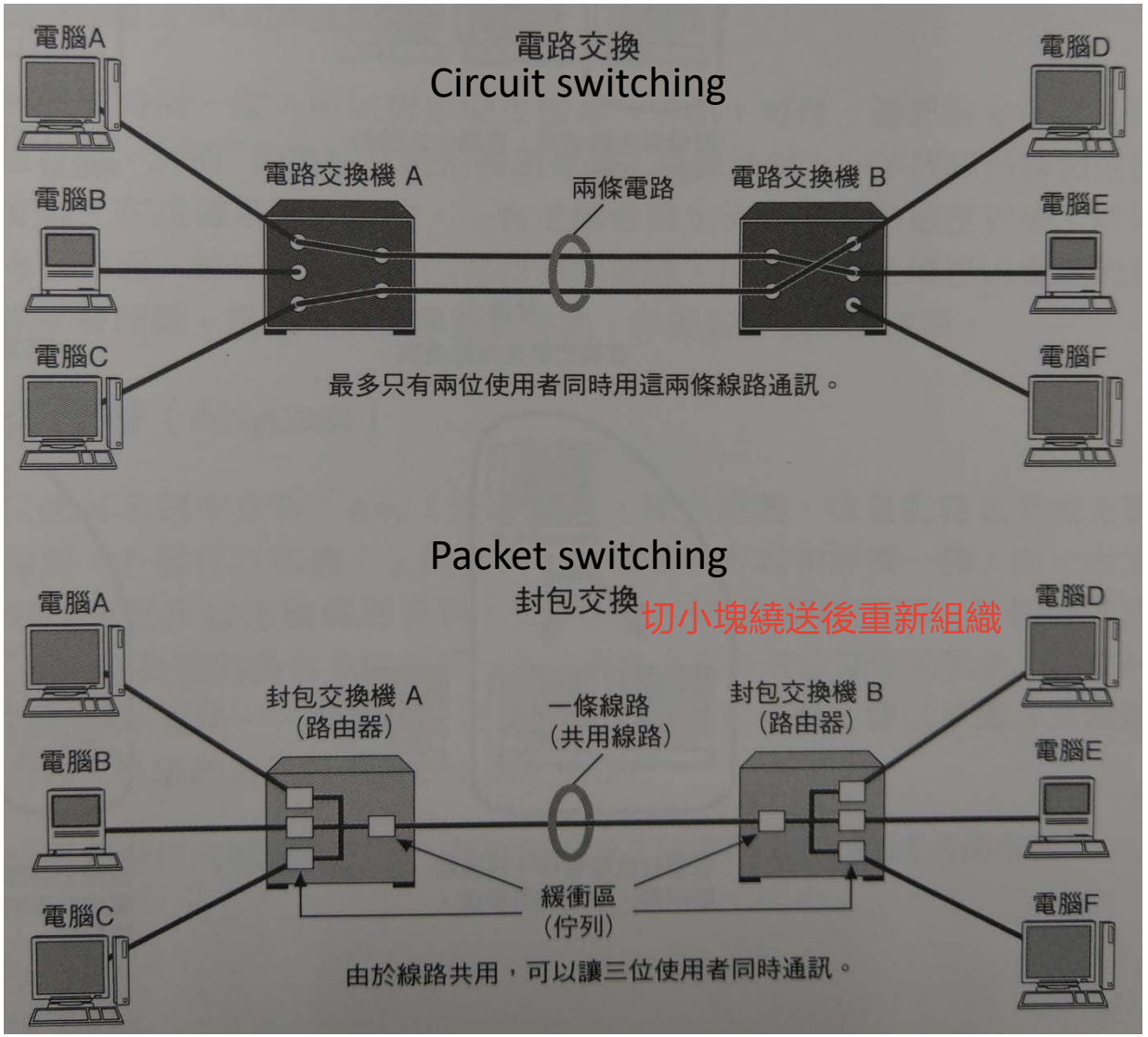
- Communication network of a distributed system
  - 實體節點 Device: router, switch, bridges, hubs, repeaters, 網卡
  - 節點間通訊連結 Connection: wire, cable, fiber, 無線電
  - 通訊控制 Software: protocol, drivers, API



Switching Network

- Switching
  - 判斷並執行「那個訊息往那裡送」的機制
- Circuit switching 專有網路，只能容納一人進來(電話)
  - Two nodes establish a dedicated communications channel (circuit)
  - 傳統電信網路(POTS, Plain old telephone system)\
- Packet switching 可容納多人(網路)
  - Packet: basic unit of transmission over network
    - 繞送資訊會放在封包表頭中
  - Packets are queued in a buffer
    - transmitted when the link is available
    - transmitted over a shared network channel

Circuit switching vs. Packet switching

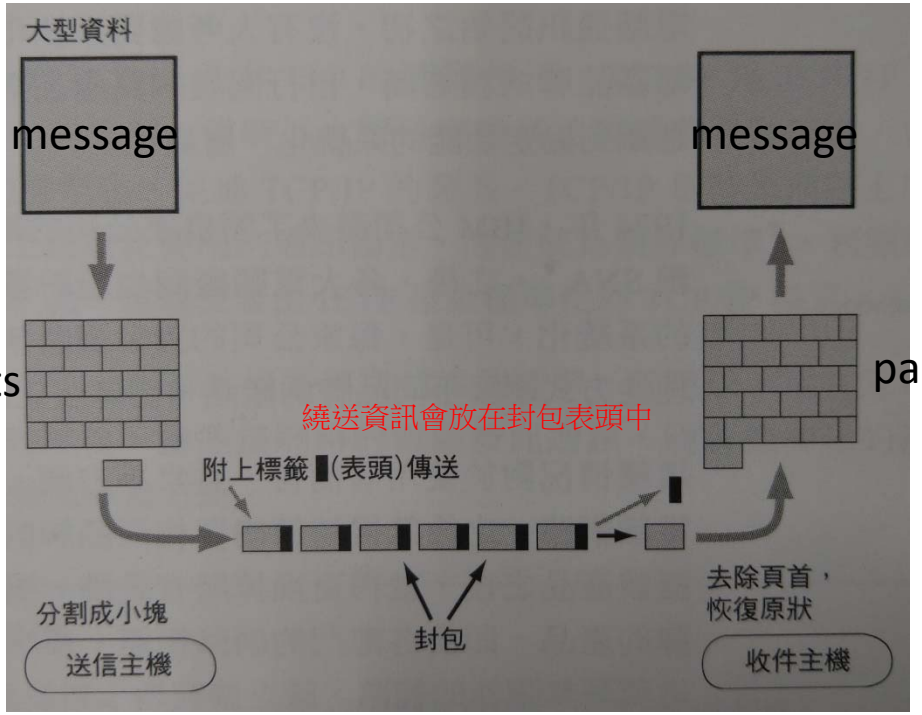


From message to packet

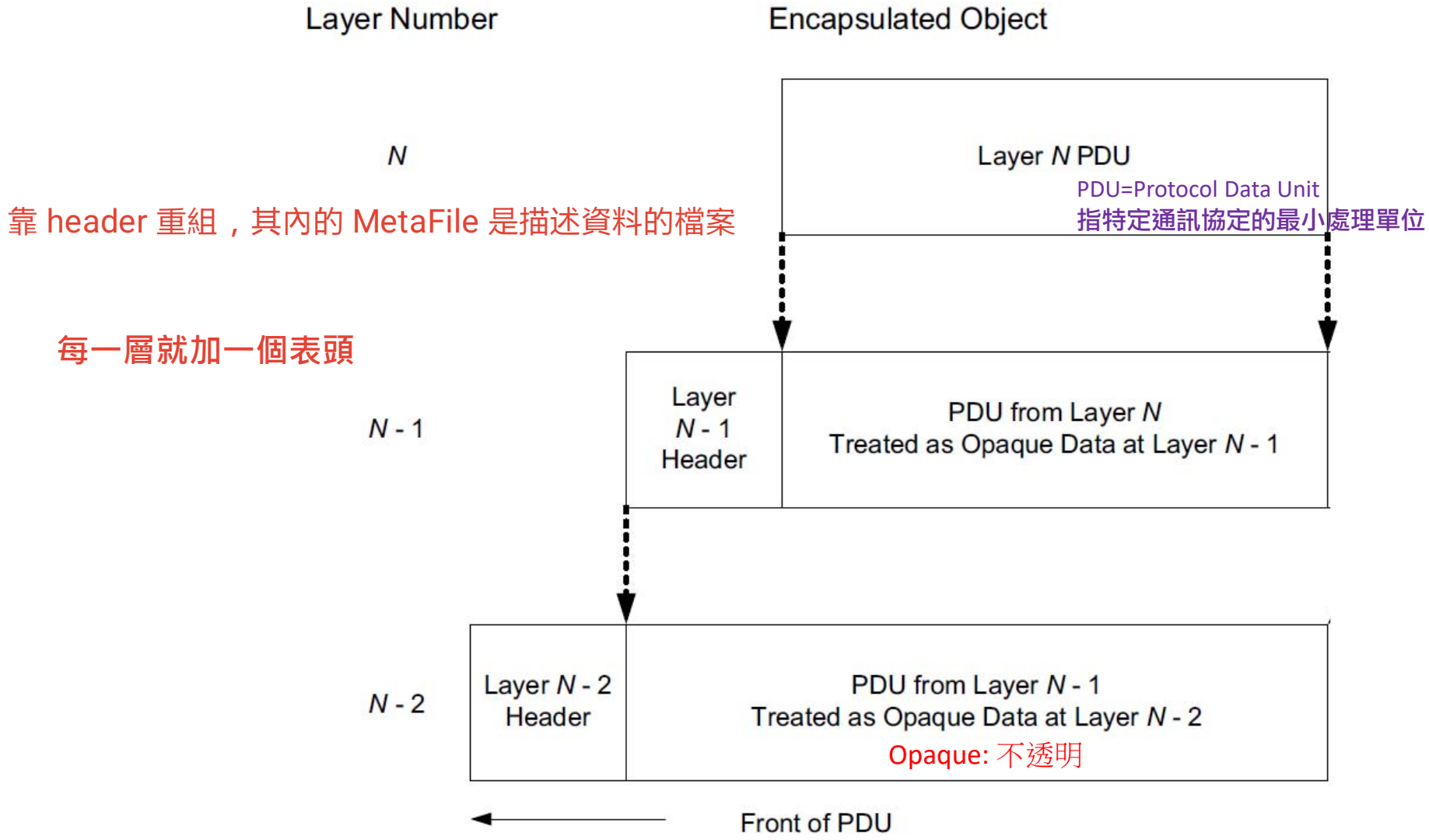
- Message

  軟體的訊息收發端稱為Endpoint

  – Logical data unit of communicating (endpoints' view point)
- Packet
  – The physical data unit of network transmission



封包分層處理概念



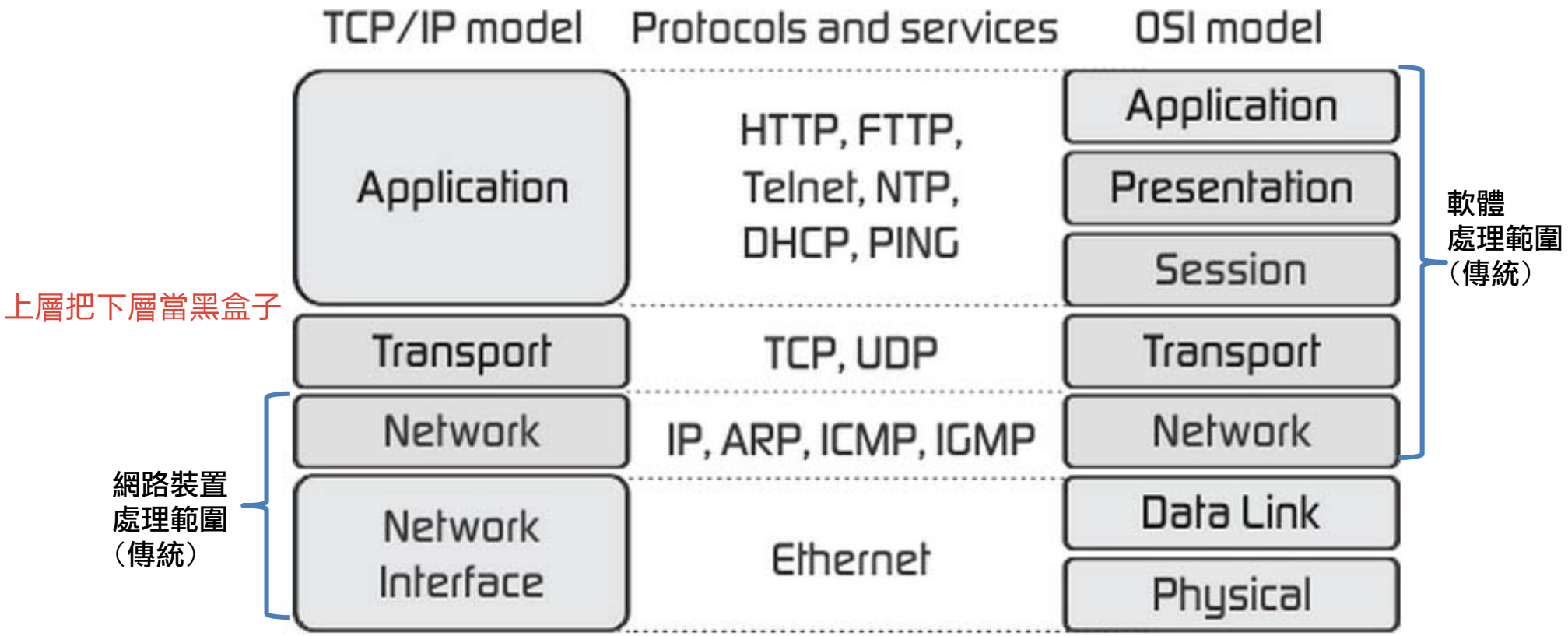
靠 header 重組，其內的 MetaFile 是描述資料的檔案

每一層就加一個表頭

網路分層概念

UDP 客製化

QUIC => 比 TCP 更精簡，且有 TCP 之功能



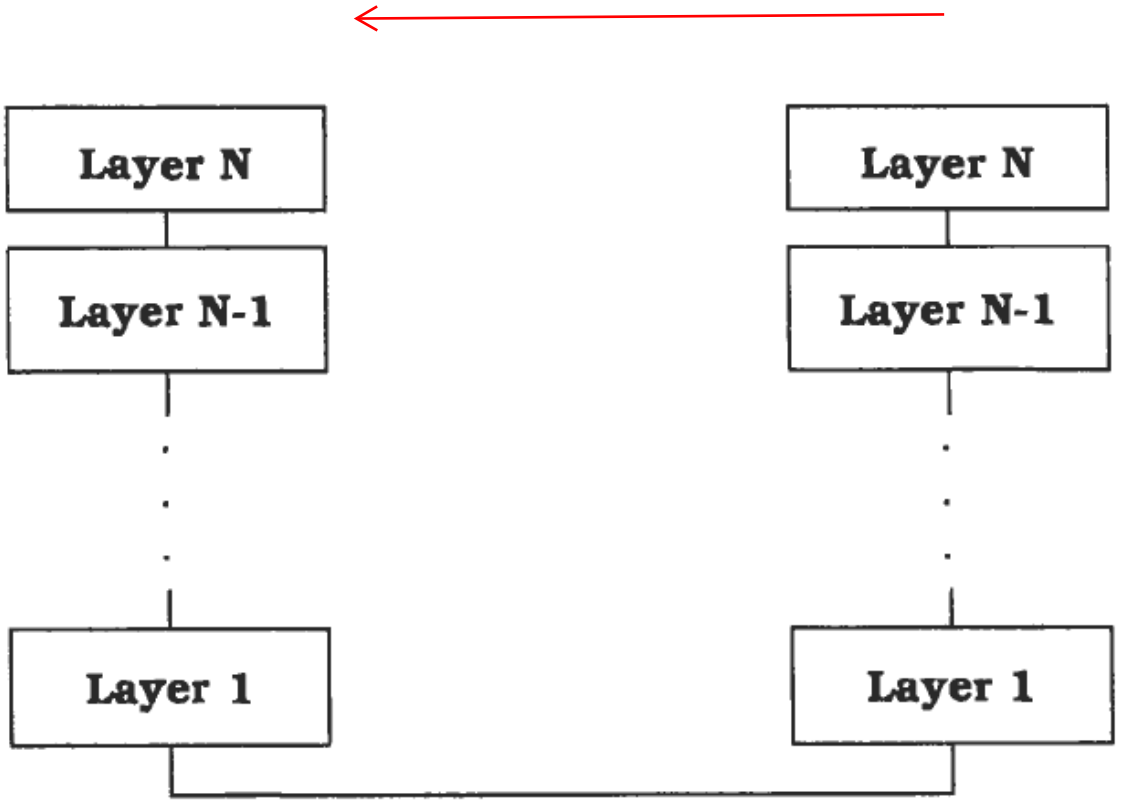
在SDN(Software Defined Network)的架構下，理論上所有層次均可用軟體處理

在雲端可以放機器 => 什麼事都使用軟體

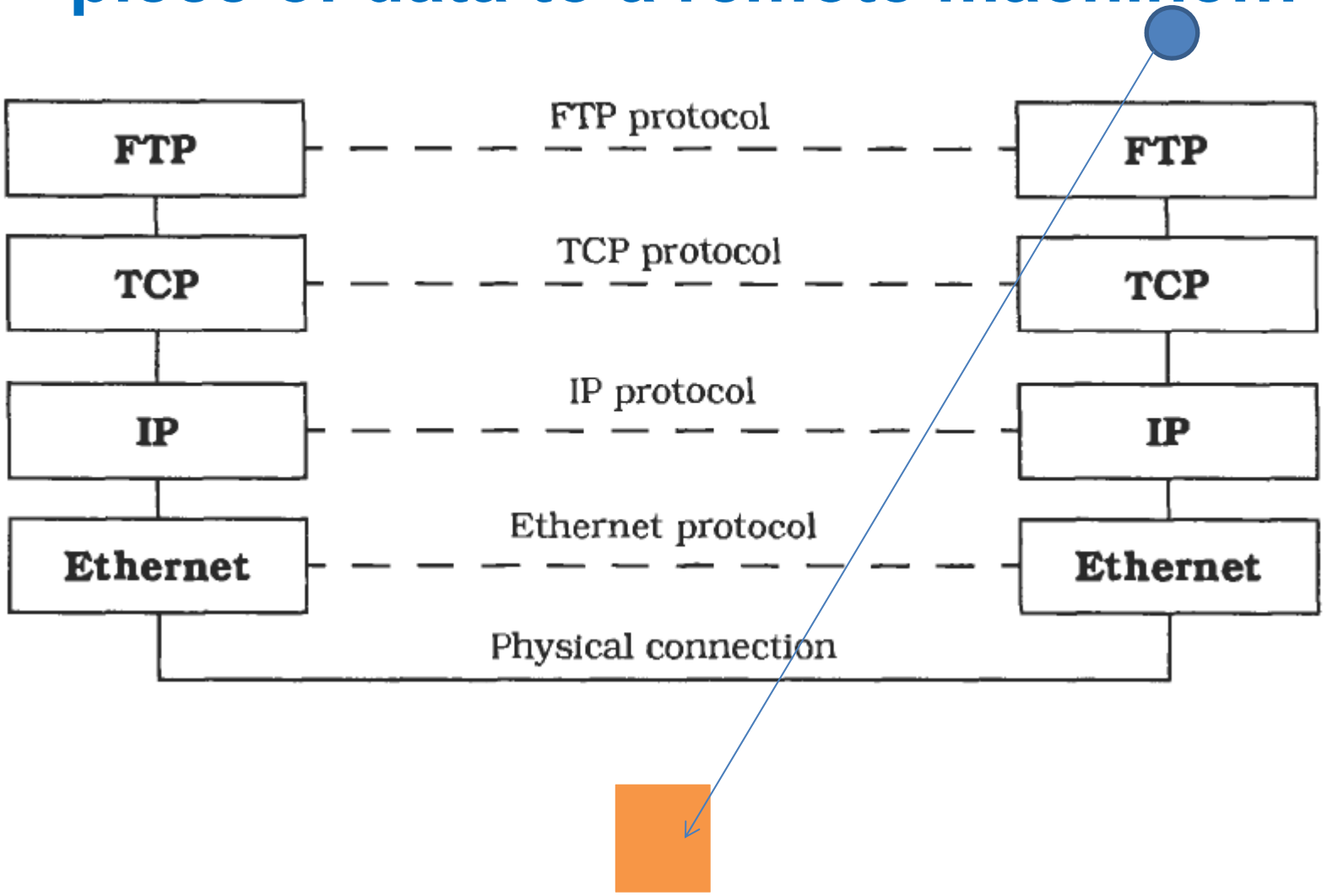
Layer

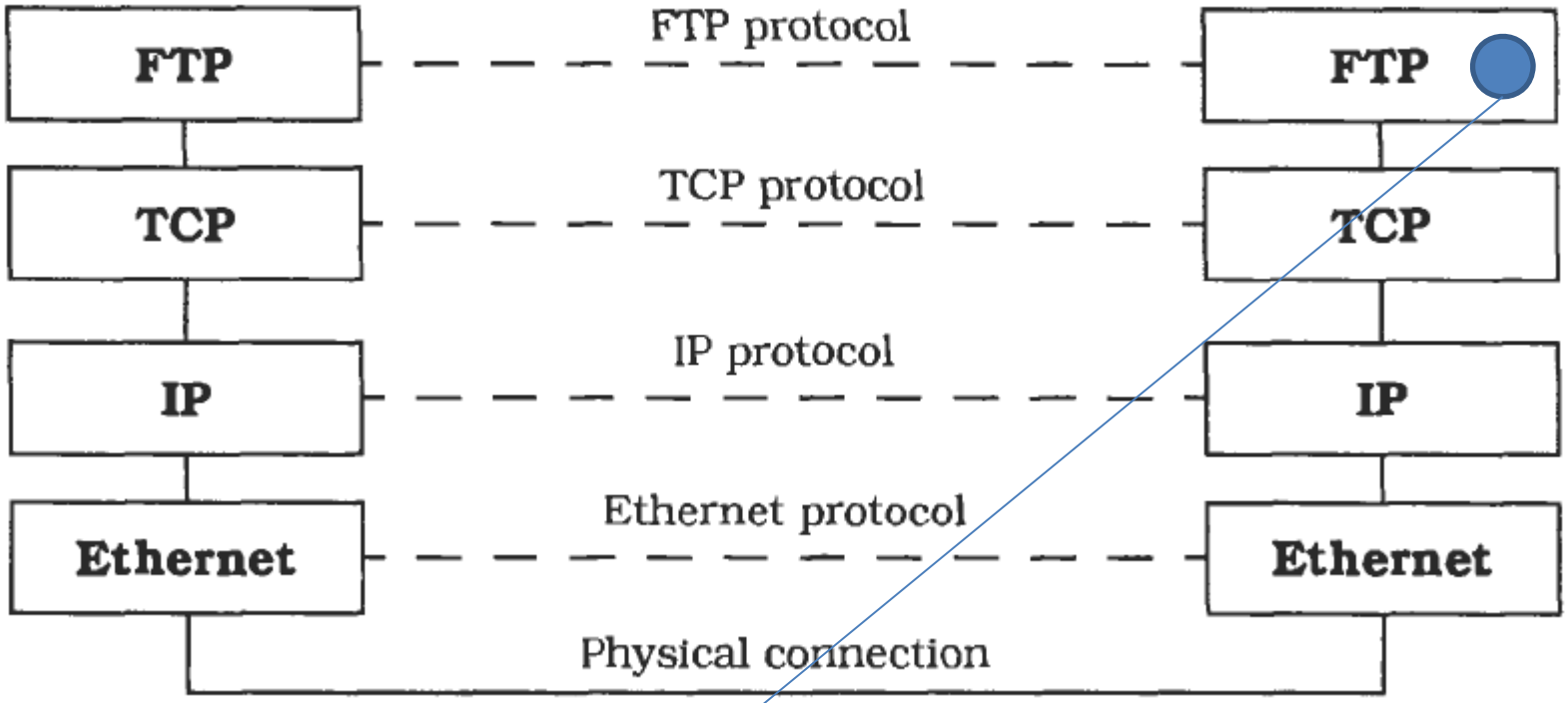
- Two-way Communication
  - Example: TCP/IP

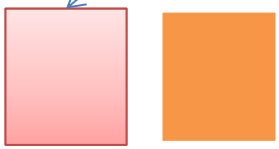
右往左傳封包



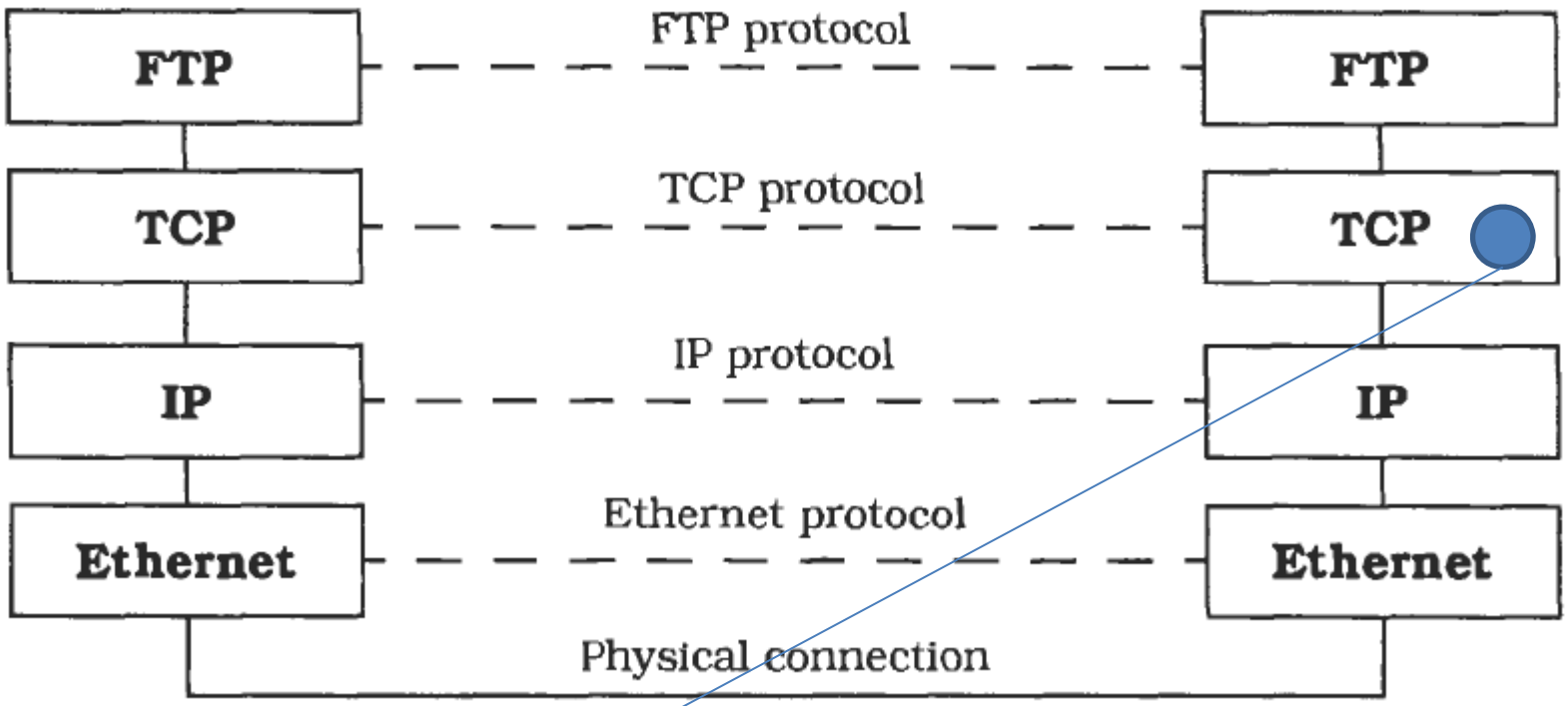
When an application wants to send a piece of data to a remote machine…





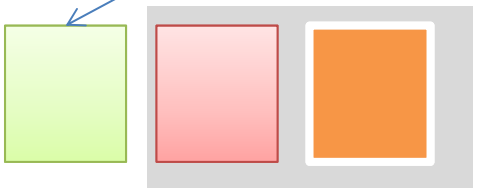
要怎麼「處理」這個內容  傳送的內容

Add header of FTP



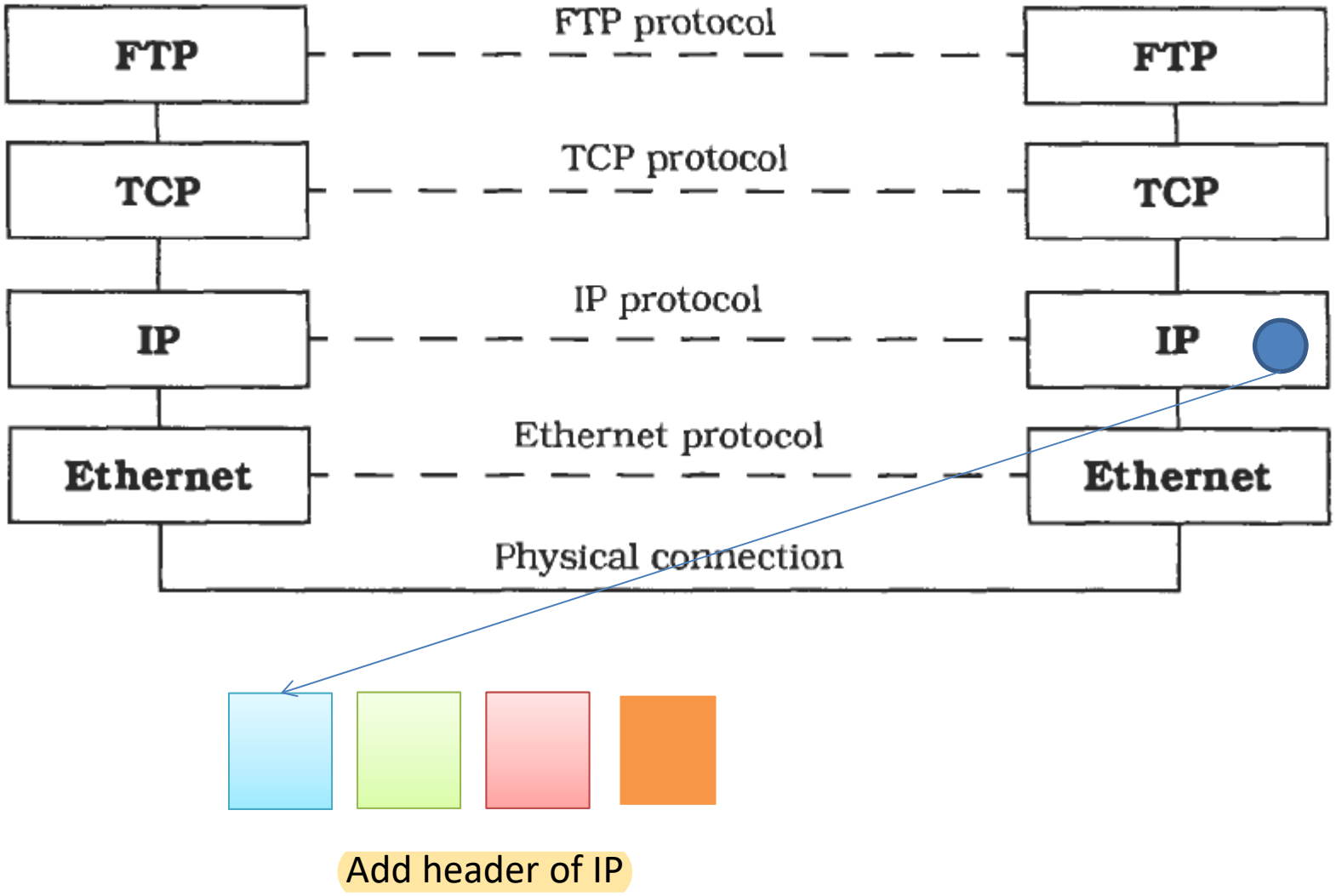
黑盒子

要怎麼「處理」這個內容

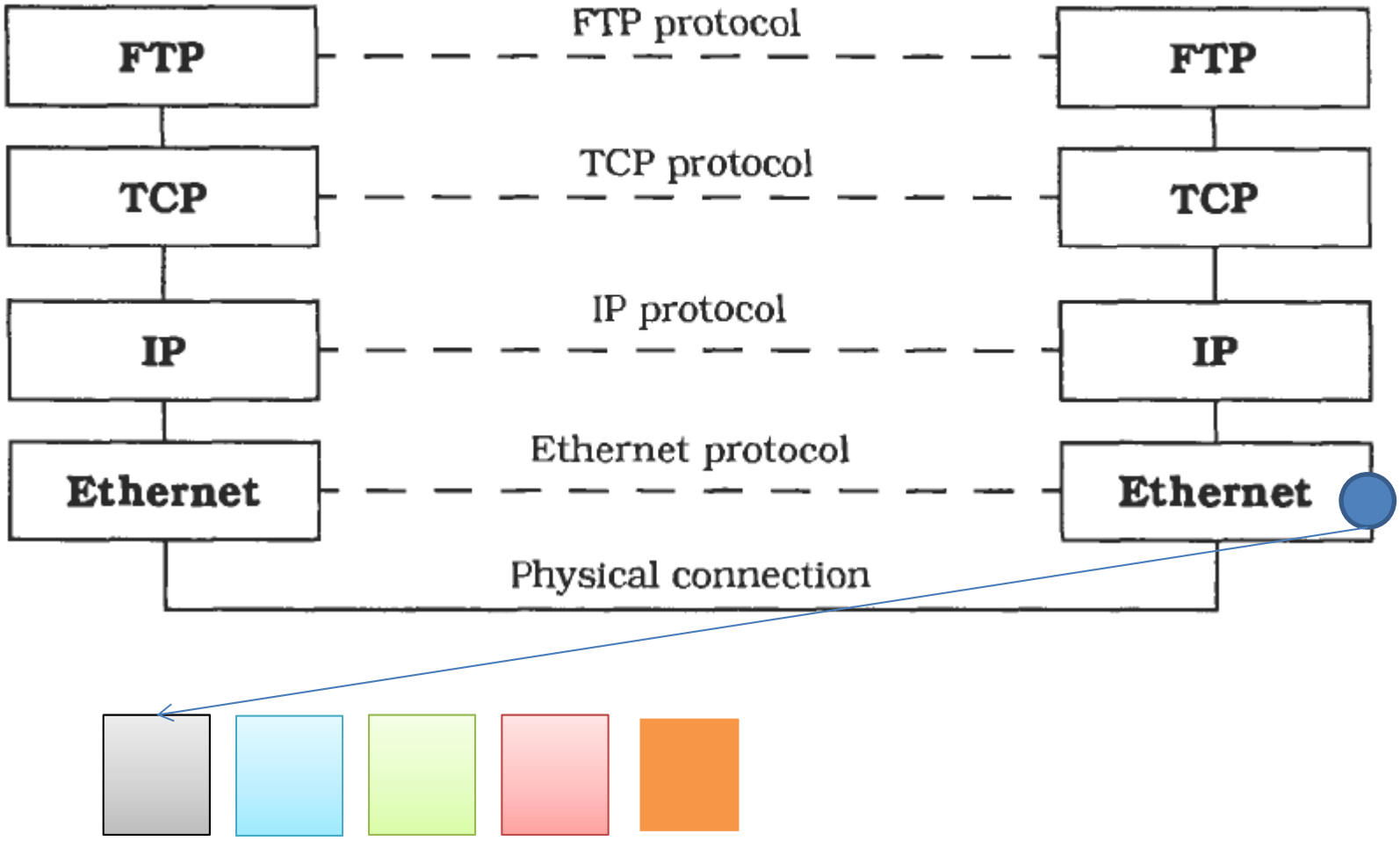


傳送的內容

Add header of TCP

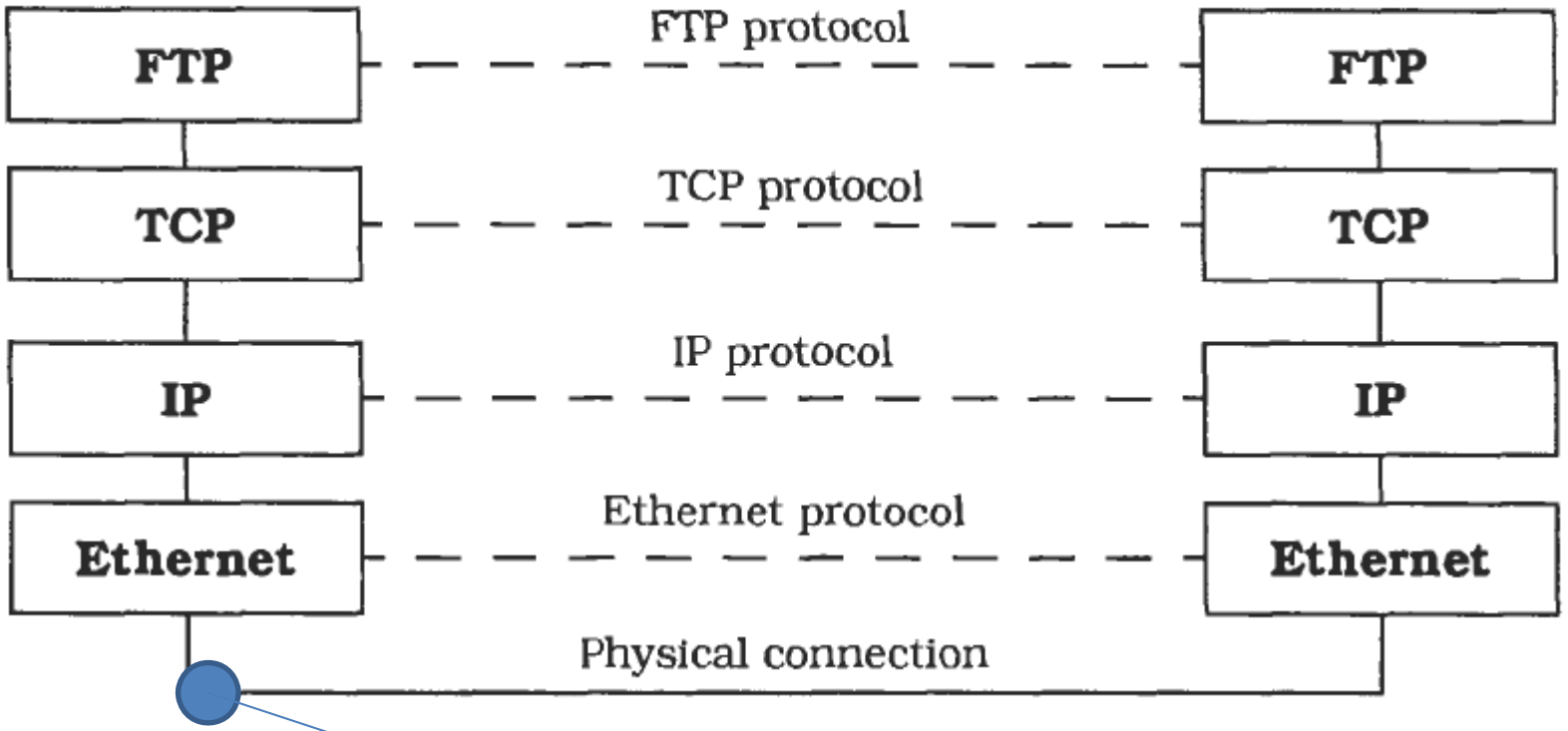

FTP
FTP protocol
FTP
TCP
TCP protocol
TCP
IP
IP protocol
IP
Ethernet
Ethernet protocol
Ethernet
Physical connection
Add header of IP

只要專心處理 header，其它當黑盒就可

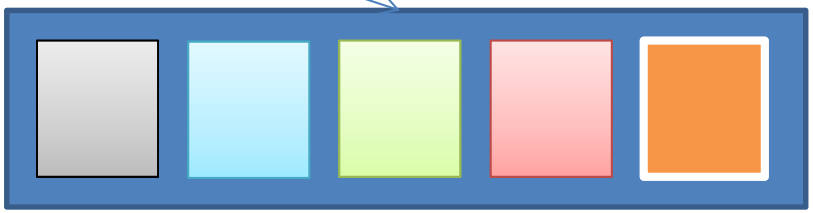


Add header Ethernet

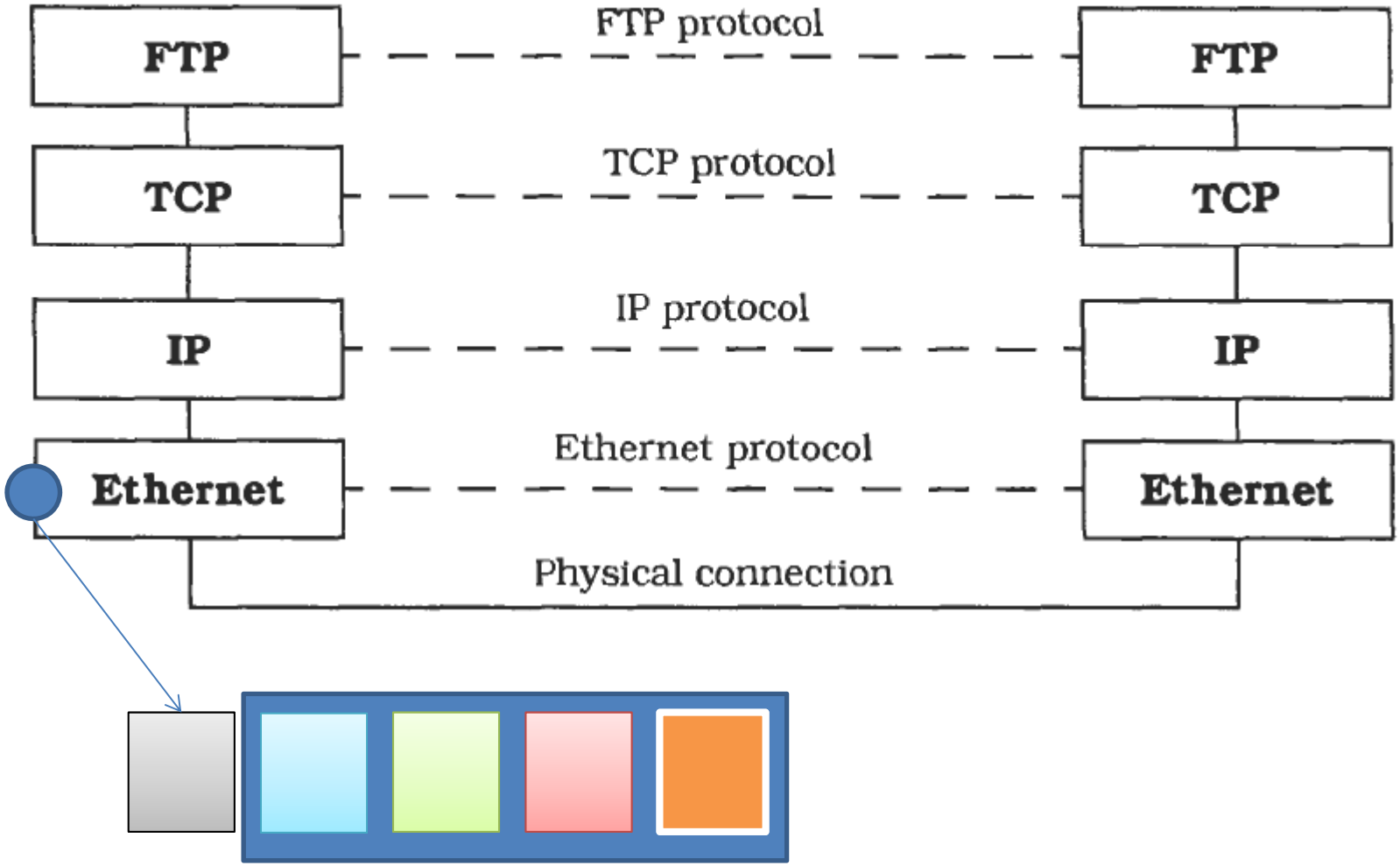
When a machine receives a packet…



開始解碼

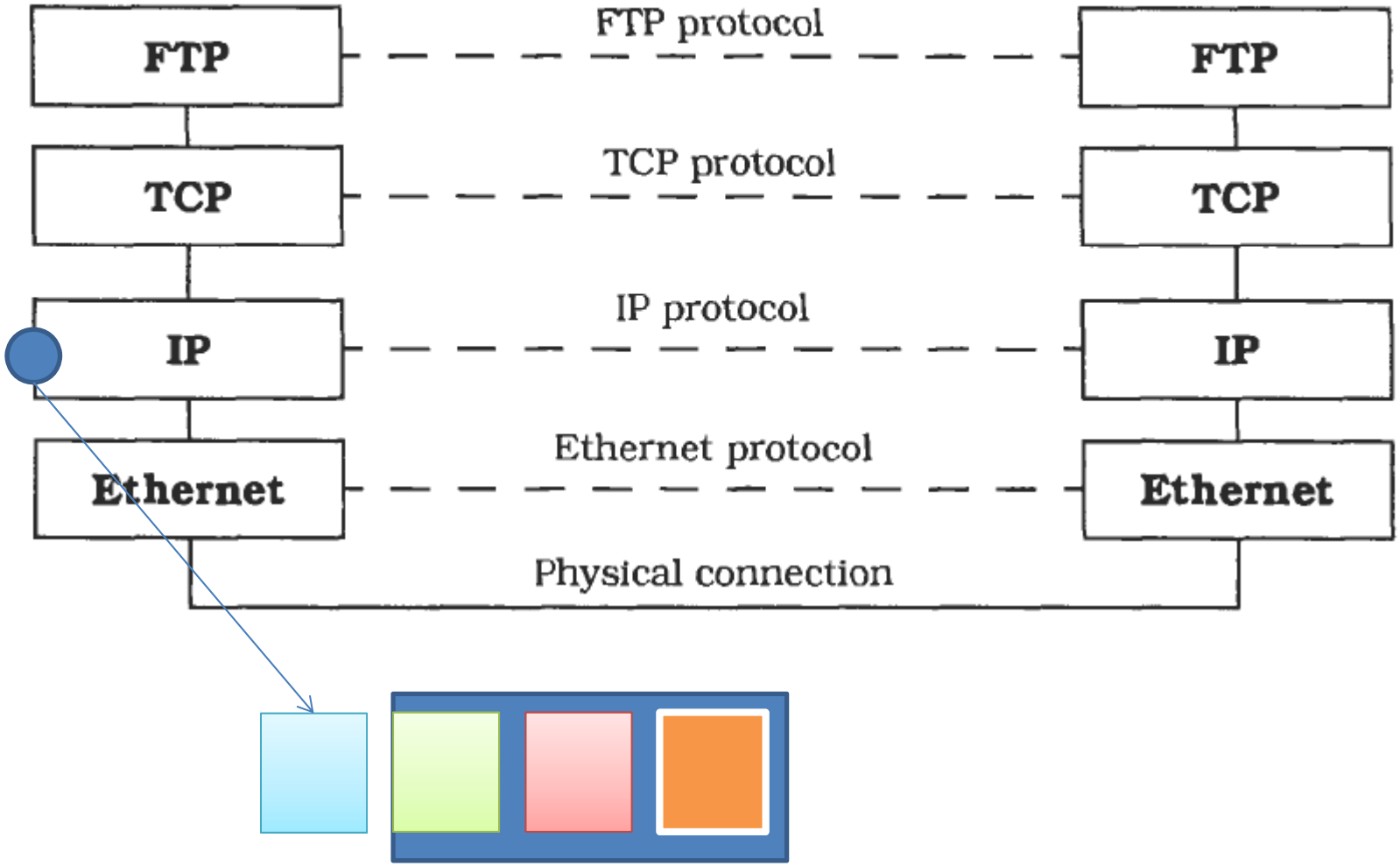


A full packet



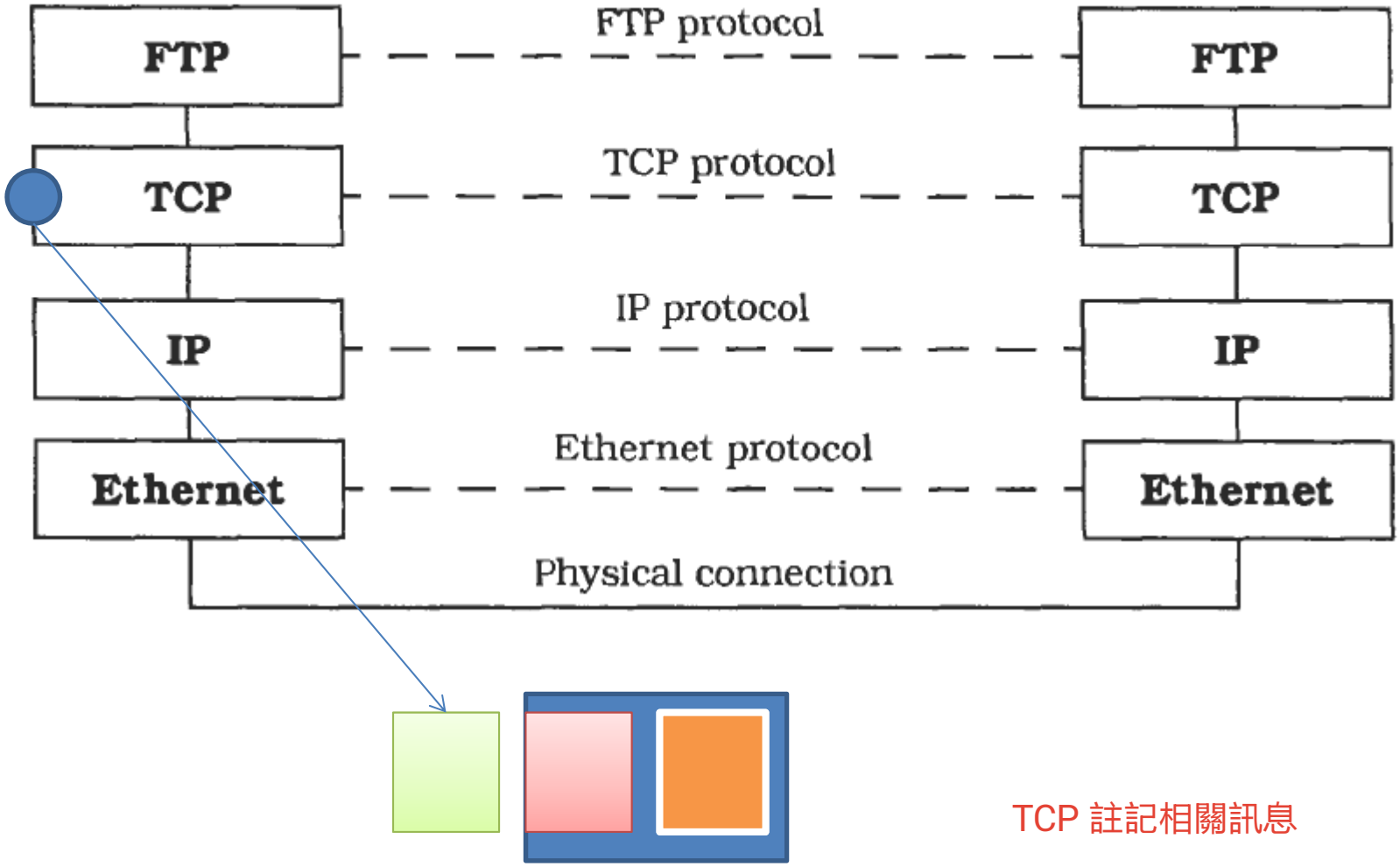
Processing header of Ethernet

依照header中的指示處理「內容」



Processing header of IP

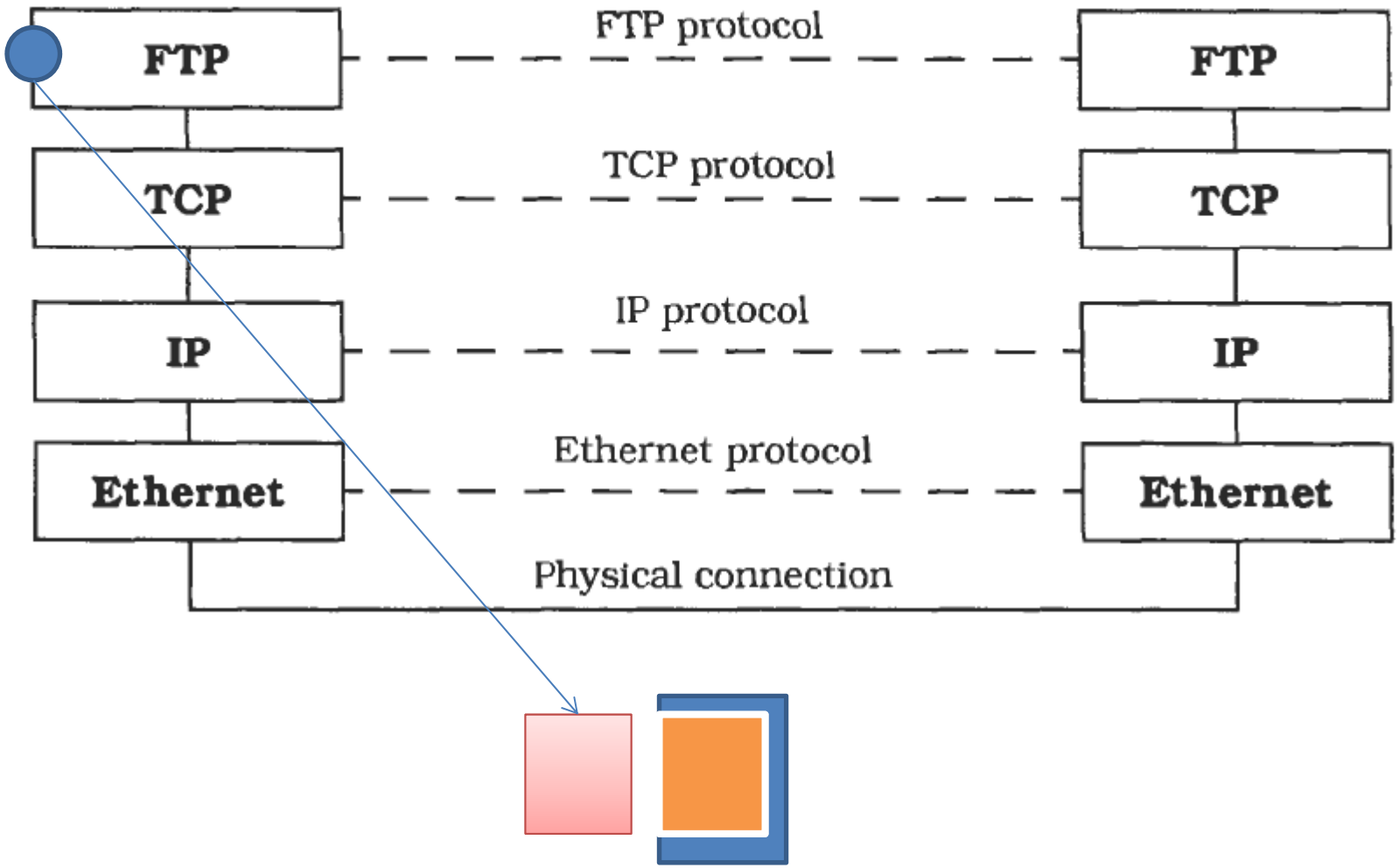
依照header中的指示處理「內容」



Processing header of TCP

依照header中的指示處理「內容」

TCP 註記相關訊息

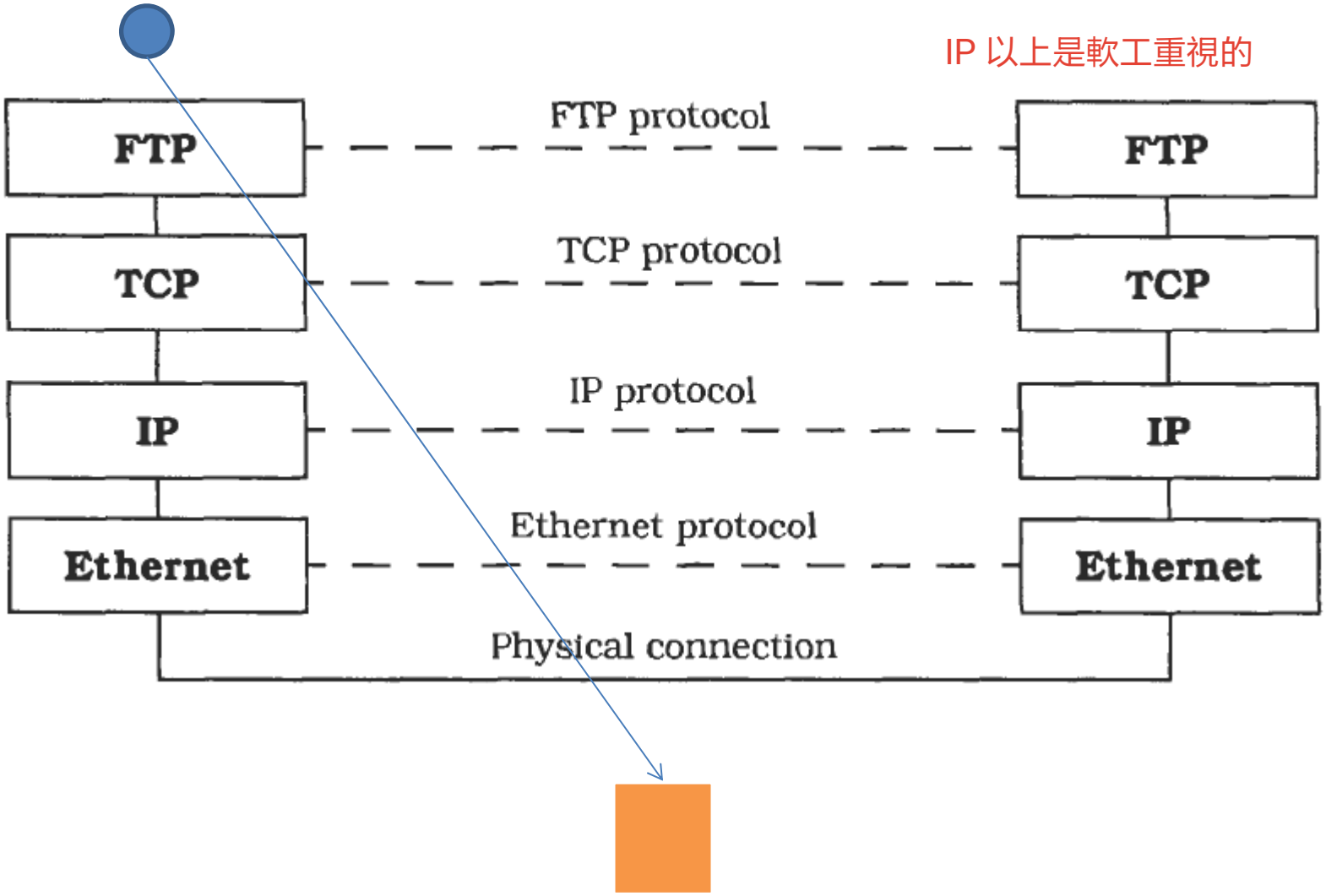


Processing header of FTP

依照header中的指示處理「內容」

加入FTP內容

IP 以上是軟工重視的

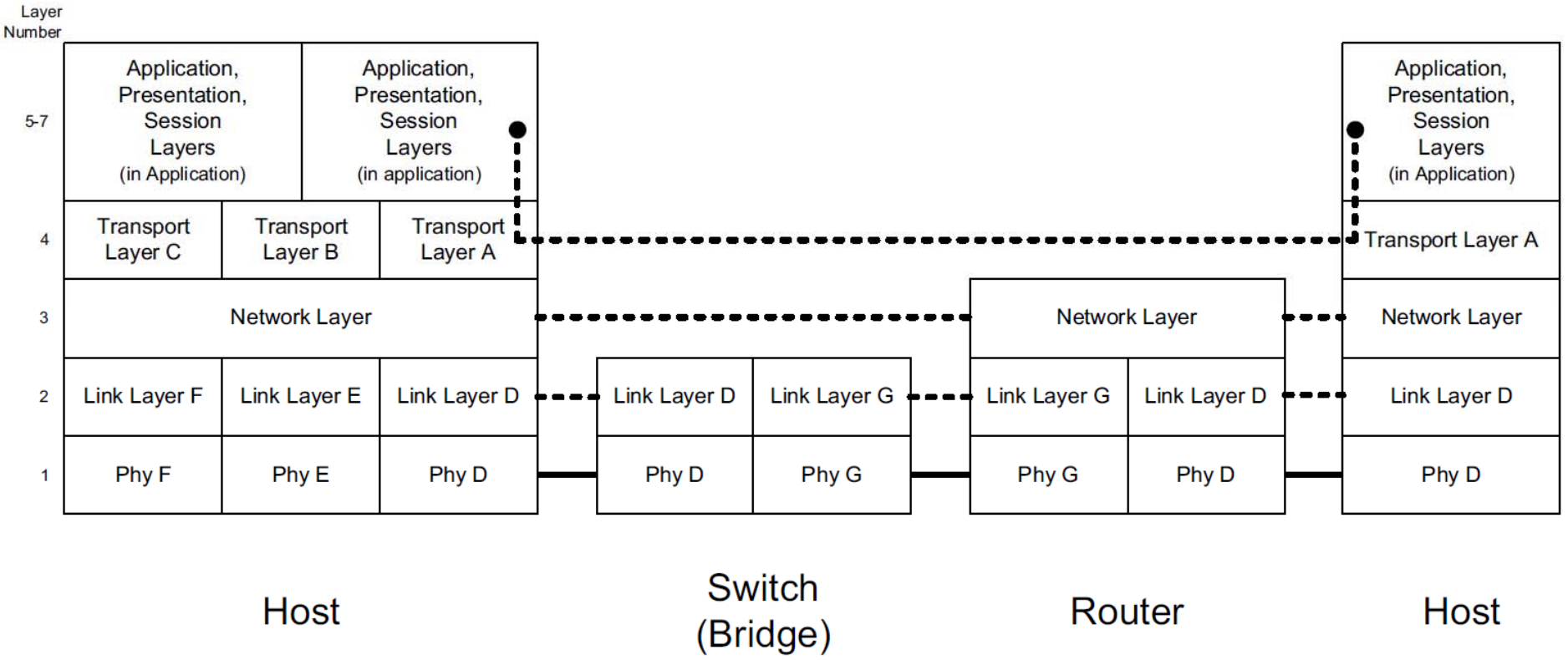


Finally, the application receives the data

網路分層概念

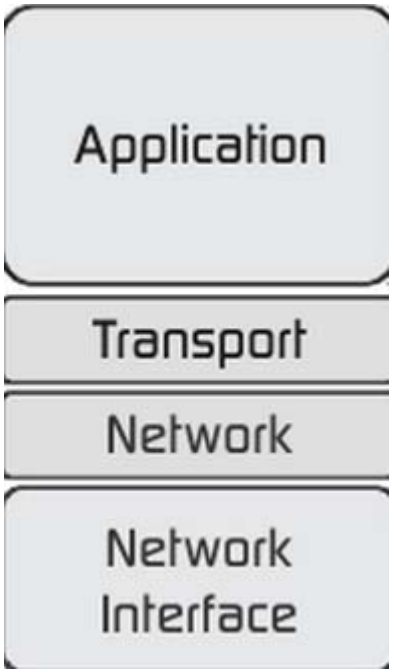
不一定每個端點都要解到AP層，有的網路設備只會解到「剛好需要」的層次，如switch等網路設備

- 例如一般 switch產品為layer 2 switch
- 處理到愈多的layers功能愈強，但價格愈貴，對效能影響也愈大



PDU

- 各式PDU (不同層次的資料單位)
  - Physical = Frame
  - Ethernet = Segment
  - TCP/UDP over IP = Datagram/Packet
  - HTTP = Message



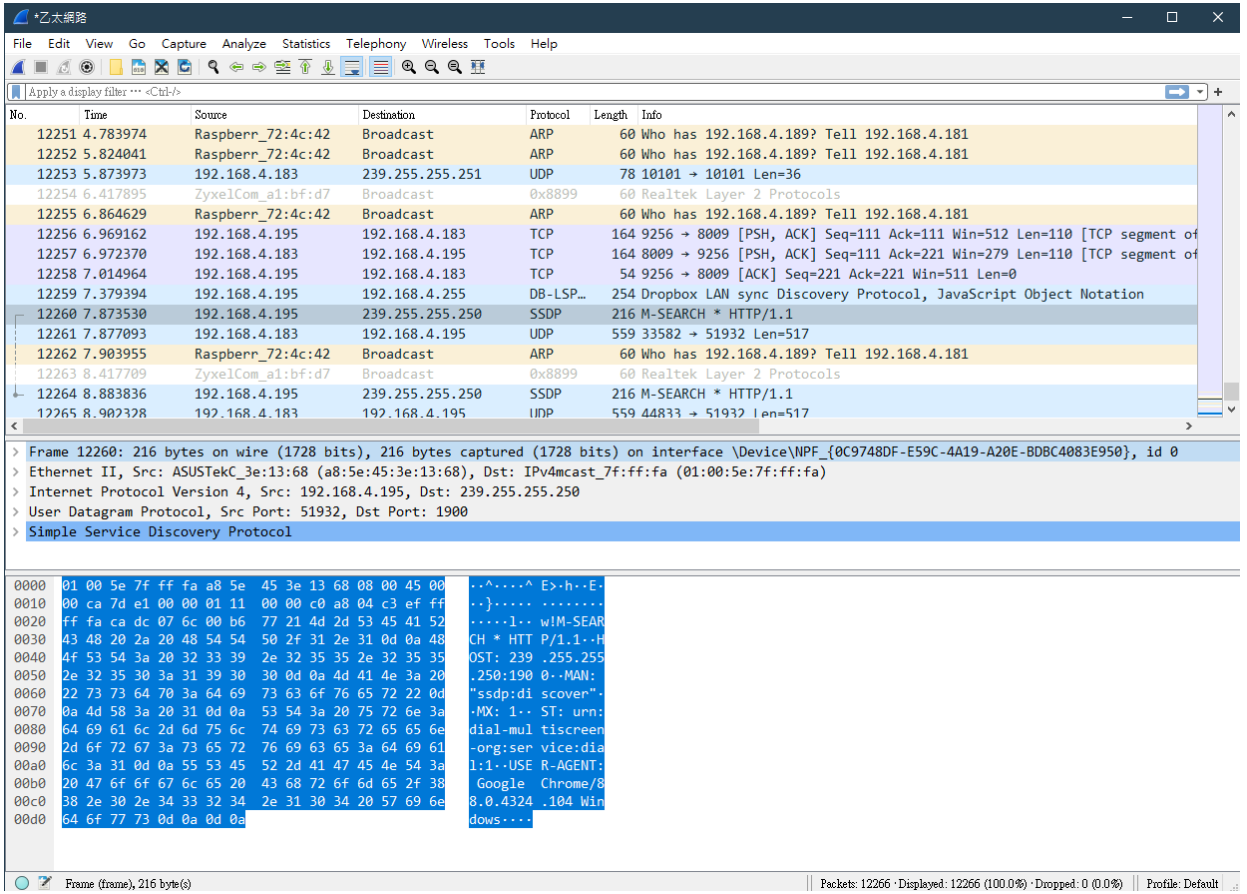
PDU
Protocol Data Unit

MTU 該層最多幾 Bytes
Maximum Transfer Unit
Ex:
MTU for Ethernet = 1500 bytes
MTU for IP = 64K bytes

Demo

- Wireshark
  - https://www.wireshark.org/
  - 練習觀察
    - Frame
    - Segment
    - Datagram
    - Message
    - 各層次的headers

每一層的參數代表的意思與意義
過濾到剩下你感興趣的，可以確認封包是否有異狀



IP 位址

- IPv4
  - 由四個連續0-255的數字構成，中間以「.」區隔
    - Ex: 140.119.1.110
    - 總共32 bits=4 bytes

      10001100 01110111 00000001 01101110

      8 bit => 0-255

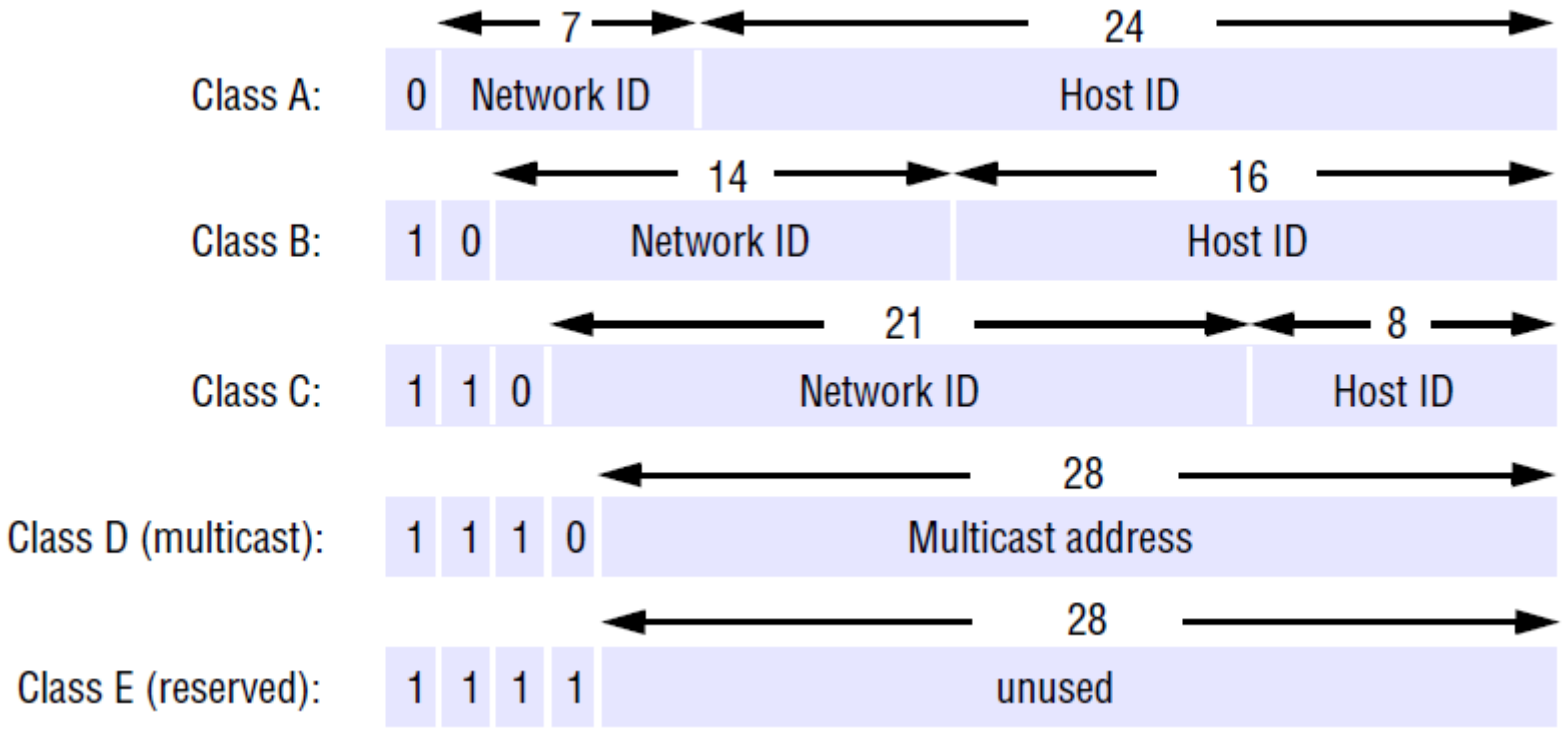
      遮前 24 個

      前 24 個為網段，最後 8 個是主機段
  - 分為網段與主機段: 140.119.163.122 / 24 ←用來代表網段的長度

    mask => 遮罩以分段，只保留特定地方的數字
    - 140.119.163 代表 “政大資科系1”
      - 10001100 01110111 10100011 (共24 bits)
    - 122代表某台主機
      - 01111010

IP 位址 (早期分配)

漸漸網段不夠用
已漸漸被淘汰



140.119.1.110 = 10001100 01110111 00000001 01101110
由上可知，政大網段原來是分配到Class B; 後16 bits其中8bits用來分割為子網段
Ex: 140.119.1.110, 140.119.163.122

switch 遇到特殊 IP，就會特殊處理
路由器遇到群播就會群播
遇到私有位址就不會往外送

特殊網段

現代仍保留的早期網段

Prefix	Special Use	Reference
0.0.0.0/8	Hosts on the local network. May be used only as a source IP address.	[RFC1122]
10.0.0.0/8	Address for private networks (intranets). Such addresses never appear on the public Internet.	[RFC1918]
127.0.0.0/8	Internet host loopback addresses (same computer). Typically only 127.0.0.1 is used.	[RFC1122]
169.254.0.0/16	"Link-local" addresses—used only on a single link and generally assigned automatically. See Chapter 6.	[RFC3927]
172.16.0.0/12	Address for private networks (intranets). Such addresses never appear on the public Internet.	[RFC1918]
192.0.0.0/24	IETF protocol assignments (IANA reserved).	[RFC5736]
192.0.2.0/24	TEST-NET-1 addresses approved for use in documentation. Such addresses never appear on the public Internet.	[RFC5737]
192.88.99.0/24	Used for 6to4 relays (anycast addresses).	[RFC3068]
192.168.0.0/16	Address for private networks (intranets). Such addresses never appear on the public Internet.	[RFC1918]
198.18.0.0/15	Used for benchmarks and performance testing.	[RFC2544]
198.51.100.0/24	TEST-NET-2. Approved for use in documentation.	[RFC5737]
203.0.113.0/24	TEST-NET-3. Approved for use in documentation.	[RFC5737]
224.0.0.0/4	IPv4 multicast addresses (formerly class D); used only as destination addresses. 224.0.0.0–239.255.255.255	[RFC5771]
240.0.0.0/4	Reserved space (formerly class E), except 255.255.255.255.	[RFC1112]
255.255.255.255/32	Local network (limited) broadcast address.	[RFC0919] [RFC0922]

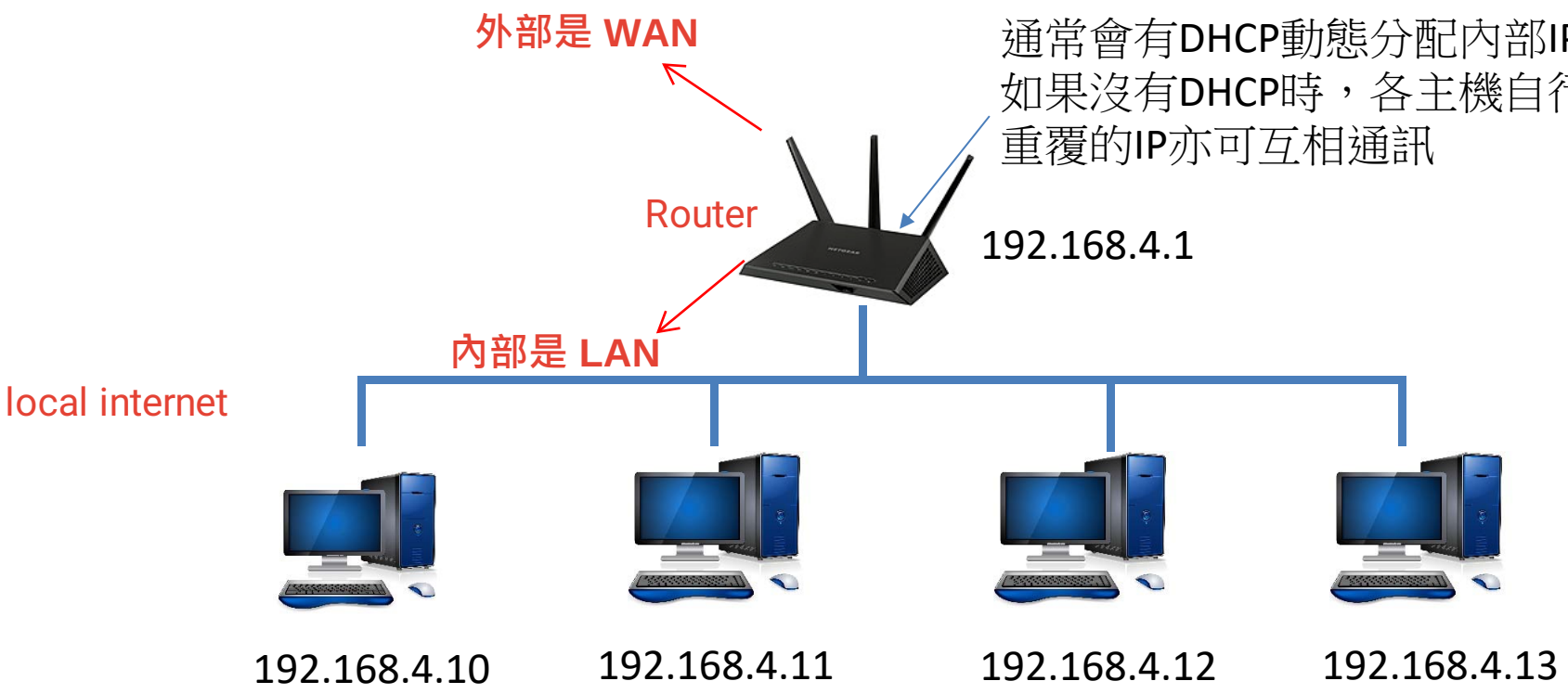
localhost

虛擬位址
可以自己分配

群播 => 於區網內

LAN

Local Area Network



高級一點的 Router 會支援 DHCP => 電腦一加入區網，就會動態分配虛擬位址

做實驗或Demo時常用配置，即使沒連到Internet，網內的各台主機還是可以互通

沒有支援 DHCP 的 Router => 可以自己寫死不重覆的 IP，仍可互相通訊

外面的固定 IP 進來後
內部會自動轉成內部虛擬 IP

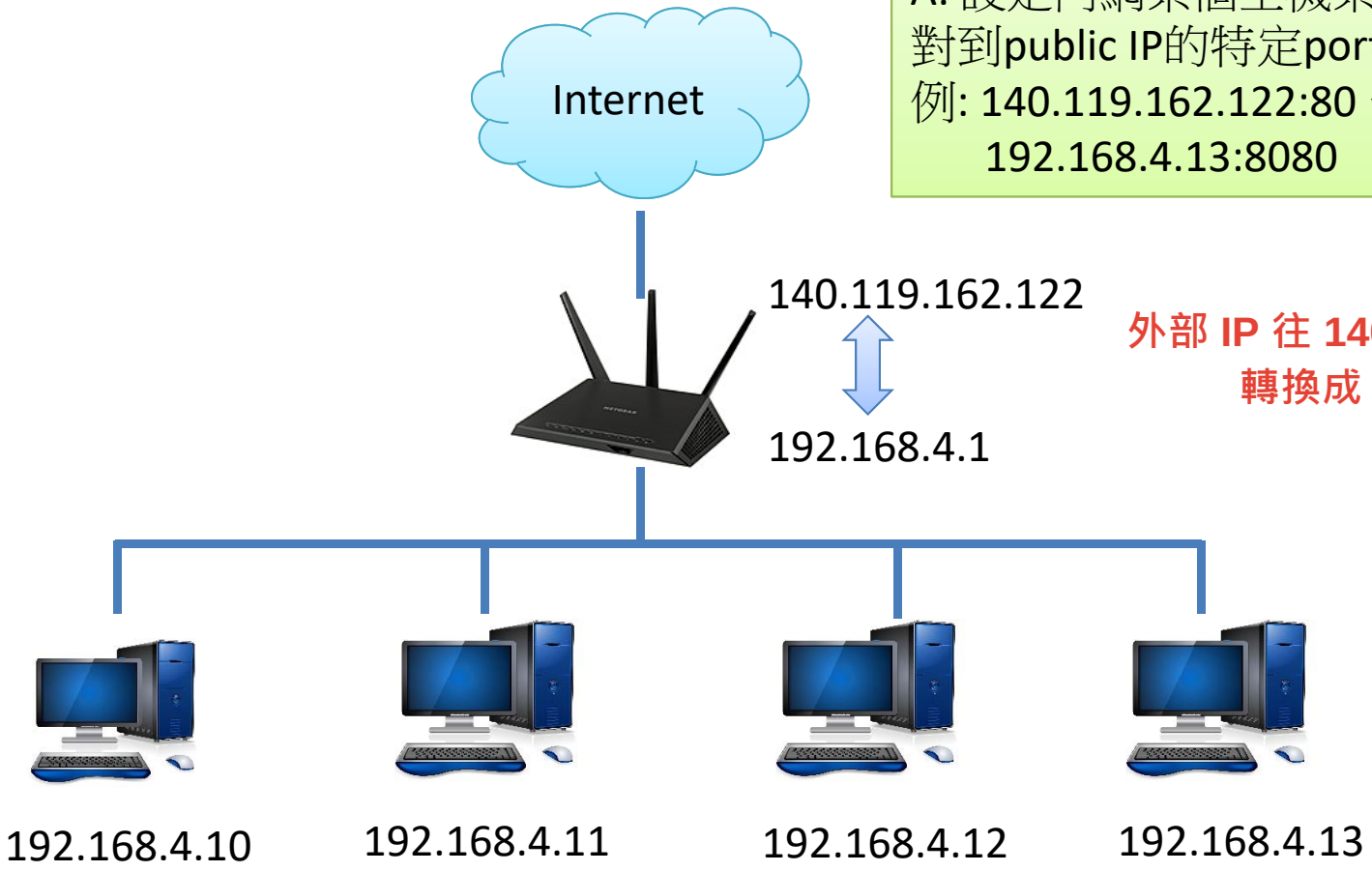
裡往外通訊
但有時外也需要和裡面通訊

NAT

Q: 如何在網內架設public 服務?

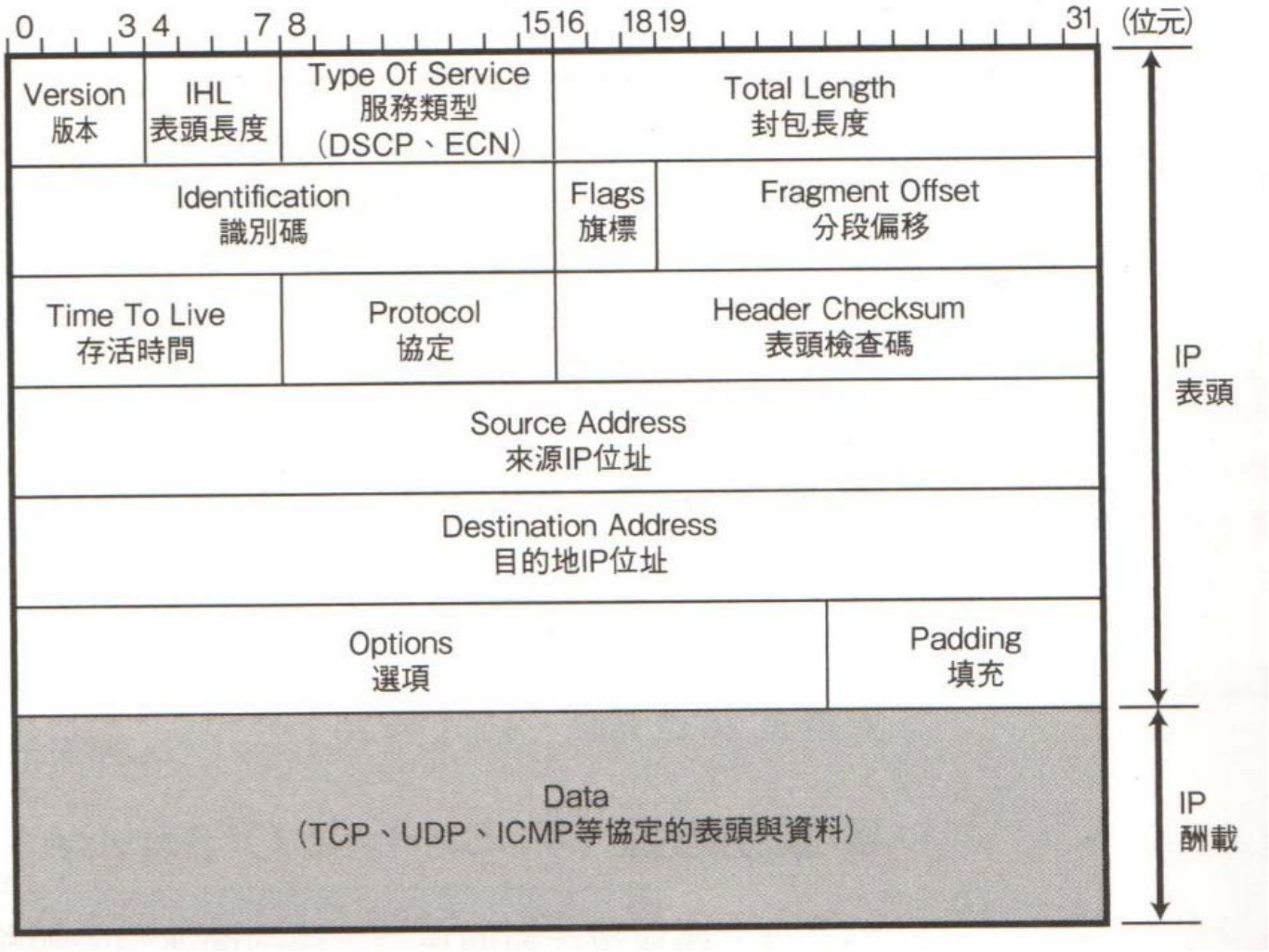
A: 設定內網某個主機某個port
對到public IP的特定port
例: 140.119.162.122:80 ←→
192.168.4.13:8080

port 對接



外部 IP 往 140.119.162.122 通訊
轉換成 192.168.4.13

IP 封包格式


0
3
4
7
8
15
16
18
19
31
(位元)
Version
版本
IHL
表頭長度
Type Of Service
服務類型
(DSCP、ECN)
Total Length
封包長度
Identification
識別碼
Flags
旗標
Fragment Offset
分段偏移
Time To Live
存活時間
Protocol
協定
Header Checksum
表頭檢查碼
Source Address
來源IP位址
Destination Address
目的地IP位址
Options
選項
Padding
填充
Data
(TCP、UDP、ICMP等協定的表頭與資料)
IP
表頭
IP
酬載

IP 封包格式

參考就好

TOS目前實務上很少使用
目前亦有人用來做DSCP (Differentiated Service Codepoint, DiffServ)與 ECN(Explicit Congestion Notification)

此分段位於原資料的第幾個位置
(單位=8bytes)

4: IP
6: IPv6

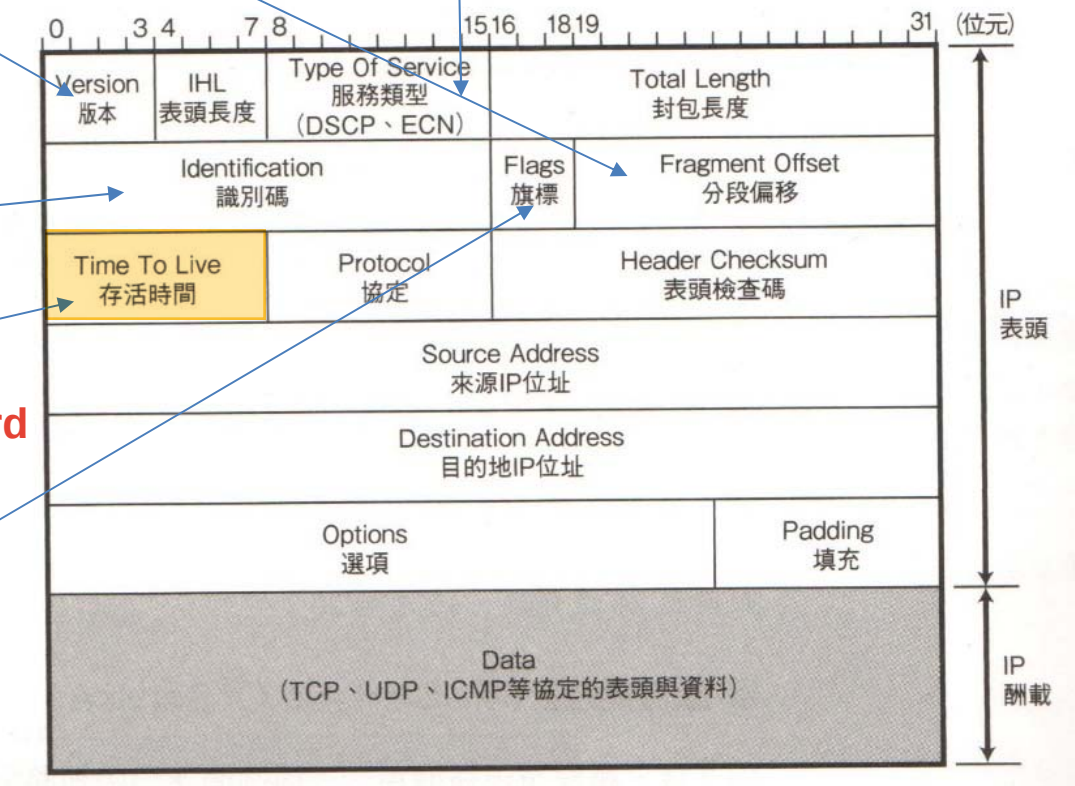
參考就好

分段，重組fragment用

可跨過多少中繼路由器

僅是參考值，也要看對方路由器是否會 forward

0: 一定要設成0
1: 是否分割
2: 是否為最後的分割



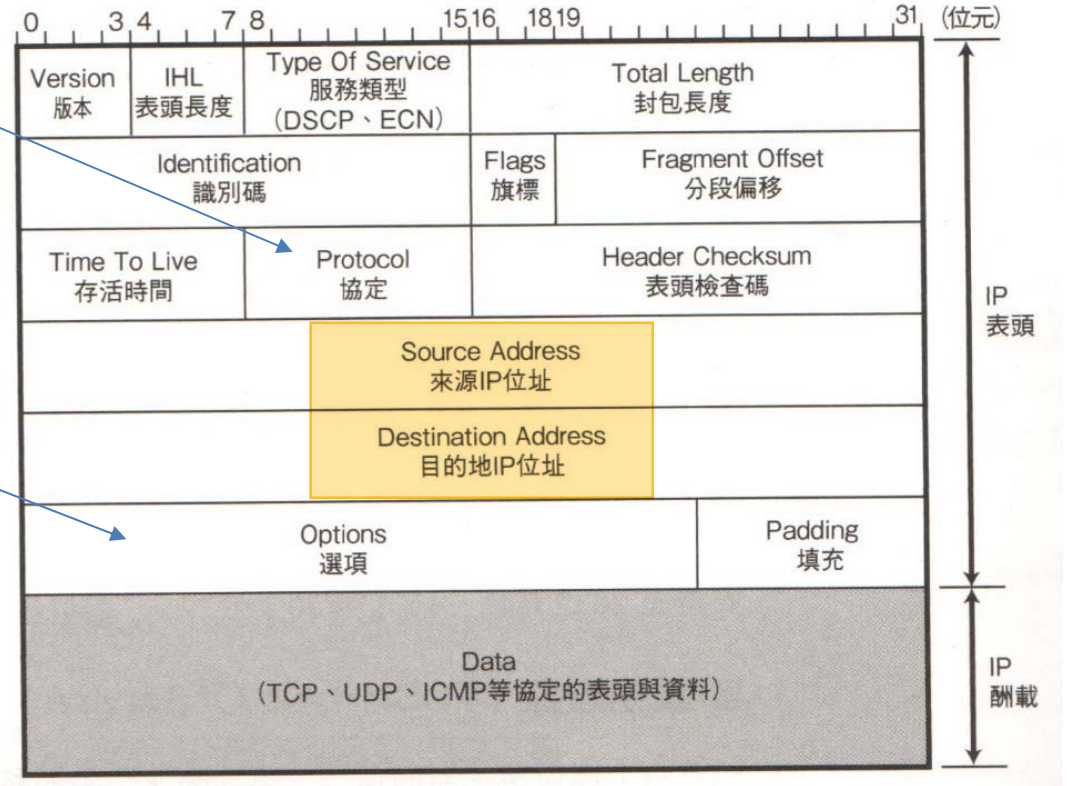
IP 封包格式

一些黑盒資訊提前洩露
不用解到黑盒子就知道要不要繼續往下解
可提升效能

在payload中的協定代碼
TCP(6)
UDP(17)

一般不會使用，主要用於偵錯、測試

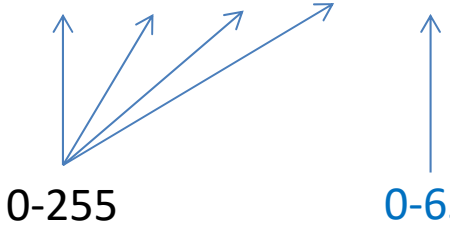
黑盒子



網路通訊

A 送到 B
A 有 IP 和 Port
B 亦有 IP 和 Port
A 的 Port 是 OS 會隨機指派閒置的
通常只需注意 A 的 IP、B 的 IP、B 的 Port

- 網路通訊程式
  - 位於二台不同電腦的程式互相傳送訊息
- Internet上每台電腦(的網卡)至少有一個固定位址
  - IP + Port
  - 例如
    - 140.119.168.10: 80



http://www.nccu.edu.tw

browser 私下會做

網路名稱轉譯: nslookup
www.nccu.edu.tw →140.119.168.10
http:// → 80

https:// -> 443

每個IP都有65535個port(埠)

網路通訊

- 不同Port代表不同服務
- Well-known port
  - 80: HTTP
  - 443: HTTPS
  - 21: FTP
  - 22: SSH
  - 23: Telnet

UDP length = UDP payload + UDP header
UDP header 為 8 bytes

User Datagram Protocol (UDP)

公板沒有特定需求 => TCP
公板有特定需求 => UDP

- 目的
  - 原意: 可按使用者(開發人員)需求客製化的協定
  - 快速、短、簡單的訊息傳送
  - 有基本的checksum機制(可得知封包是否損壞)

可能會掉包、被竄改
缺少連線功能

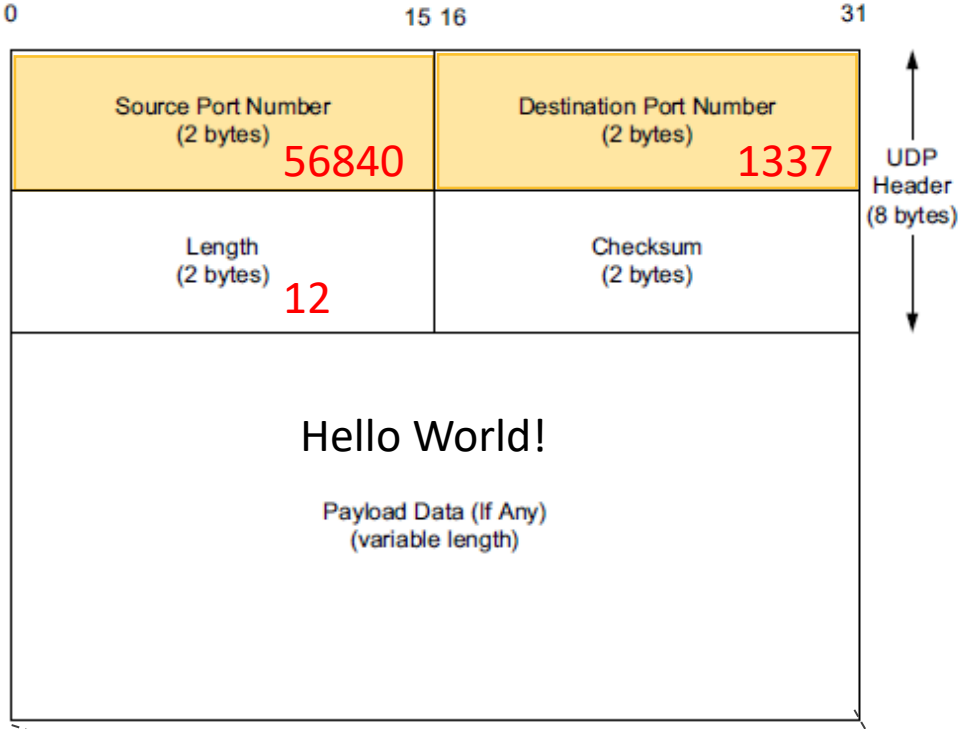
- 限制 (缺乏TCP的主要功能)
  - No session
  - No duplicate protection
  - No delivery guarantee
  - No order guarantee
  - No traffic control

可能的適用場合
Heartbeat
Service discovery
LLN (Low power and lossy network 低功耗網路)
Idempotent operations: 媒體群播、DNS查詢

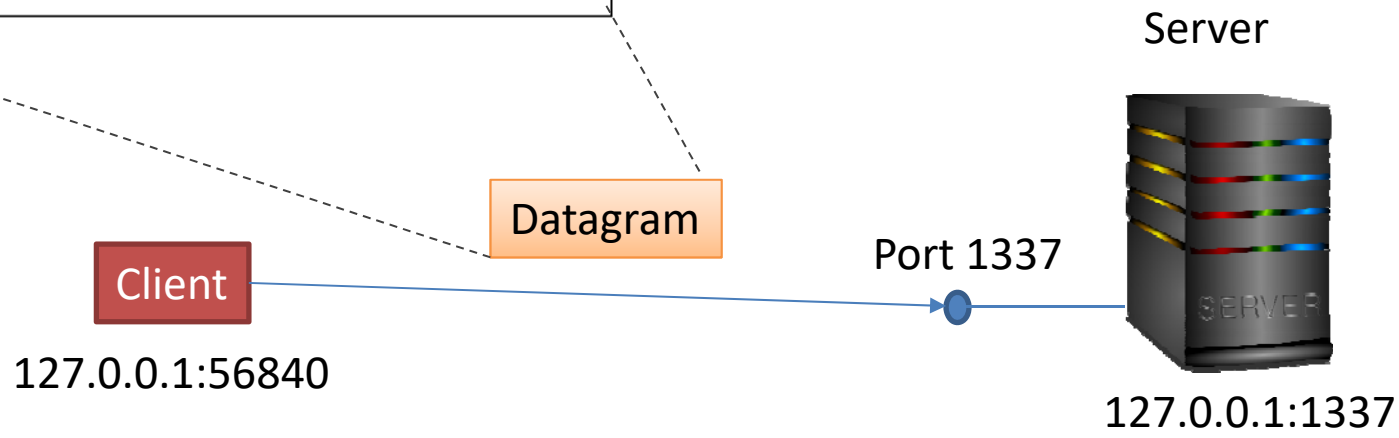
User Datagram Protocol (UDP)

8 bytes header

2 bytes = 16 bits
2^16 = 65535



重點來源和目的的 IP 和 Port



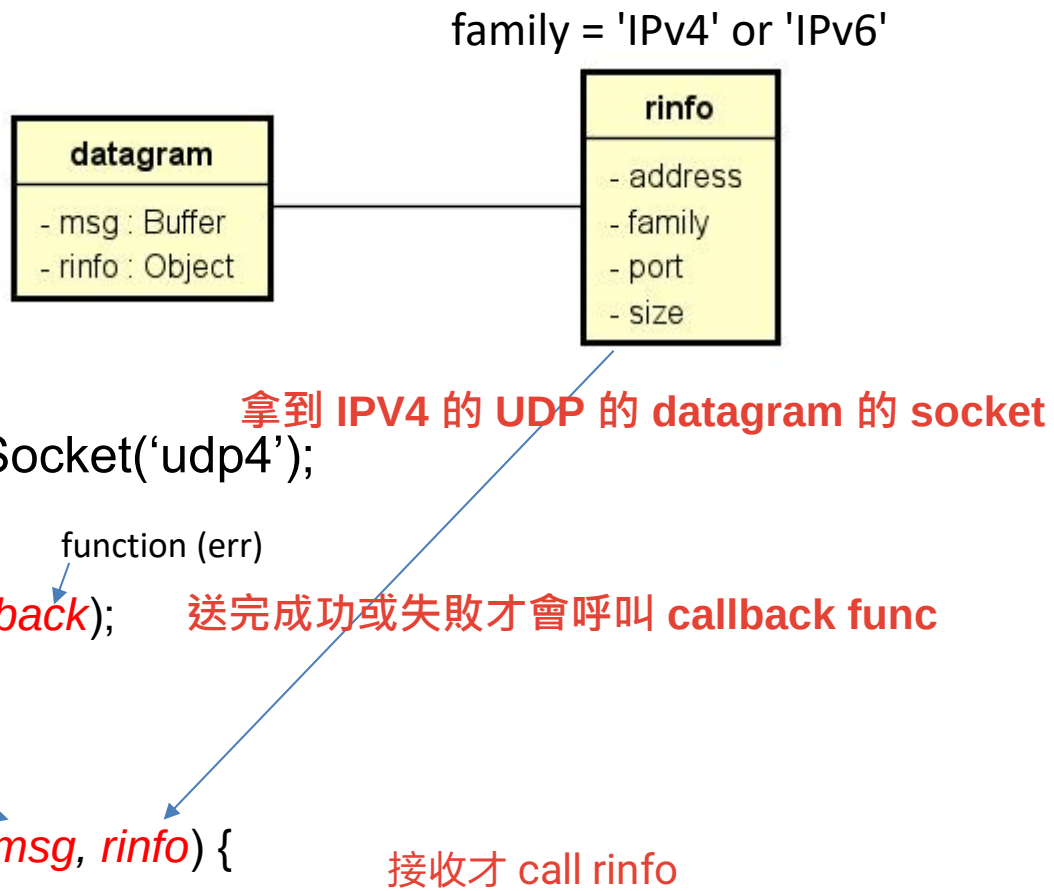
Node.js 的UDP API

- Module name
  - UDP/datagram
- Usage
  - const dgram = require(‘dgram’);
  - const udpSocket = dgram.createSocket(‘udp4’);
  - **傳送訊息**

```
udpSocket.send(msg, port, ‘IP’, callback);
```

  - **接收訊息**

```
udpSocket.on(‘message’, function (msg, rinfo) {
   …
});
```



https://nodejs.org/dist/latest-v16.x/docs/api/dgram.html

Buffer的處理

- 資料轉換為Buffer
  - 資料要在網上傳送前，要先轉為Buffer
  - Buffer.from(x)
    - x建議為string
      - x若為物件，先用JSON.stringify(x)轉為string

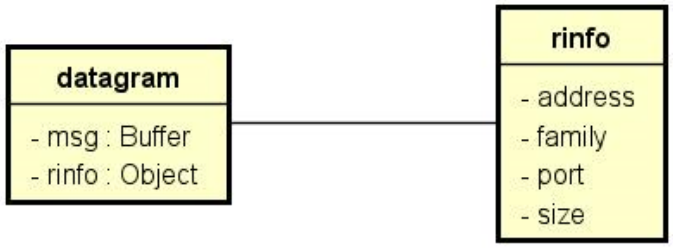
資料轉 buffer，送出收到後， buffer 再轉文字

- Buffer轉為資料
  - 使用buffer.toString()　還原成字串
  - 還原為物件: JSON.parse()

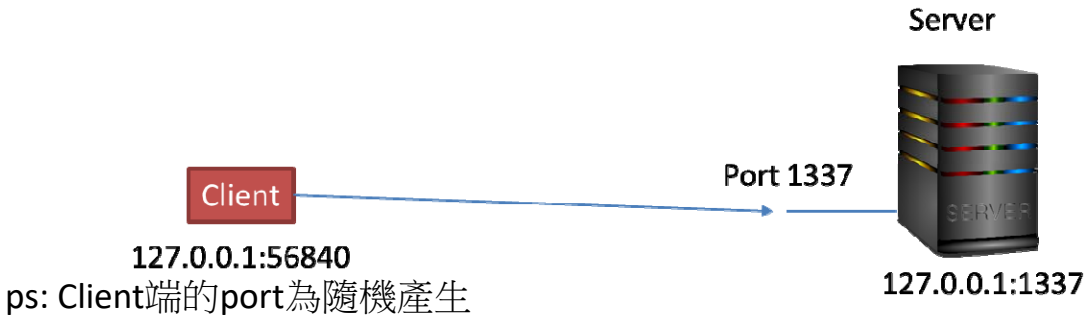
```
let obj = {
   key0: 'value0',
   key1: 'value1',
   key2: 'value2',
   key3: 'value3',
   key4: 'value4'
};

let buf = Buffer.from(JSON.stringify(obj));
console.log(buf);

let obj1 = JSON.parse(buf.toString());
console.log(obj1);
```

Node.js UDP Server

```
const dgram = require('dgram');
const server = dgram.createSocket('udp4');

server.on('message', (msg, rinfo) => {
    ....處理與回應訊息...
});

server.on('error', (err) => {
    ...處理錯誤...          error 時要幹嘛
});

server.bind(1337);
```



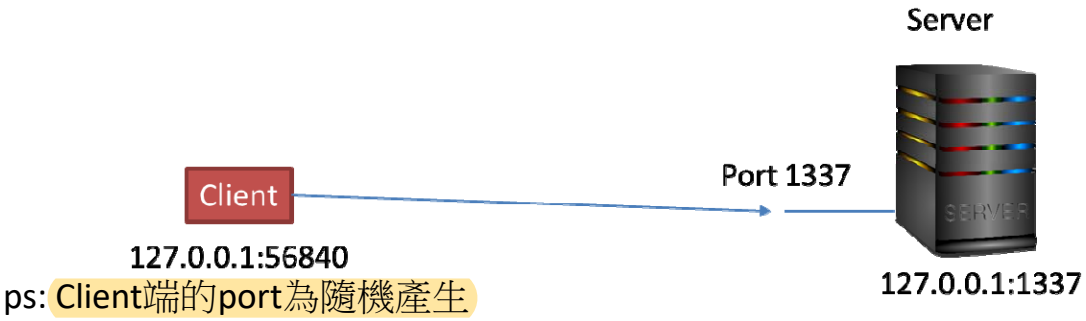
msg是Buffer型別，
所以若傳送進來的訊息是JSON物件，
可使用
let obj = JSON.parse(msg.toString());
來取得物件



Node.js UDP Client

```
const dgram = require('dgram');
const client = dgram.createSocket('udp4');

const message = Buffer.from('Hello World!');

client.send(message, 1337, 'localhost', (err) =>
{
   client.close();
});
```

作業 Client 程式碼 可用於作業



Python UDP Server

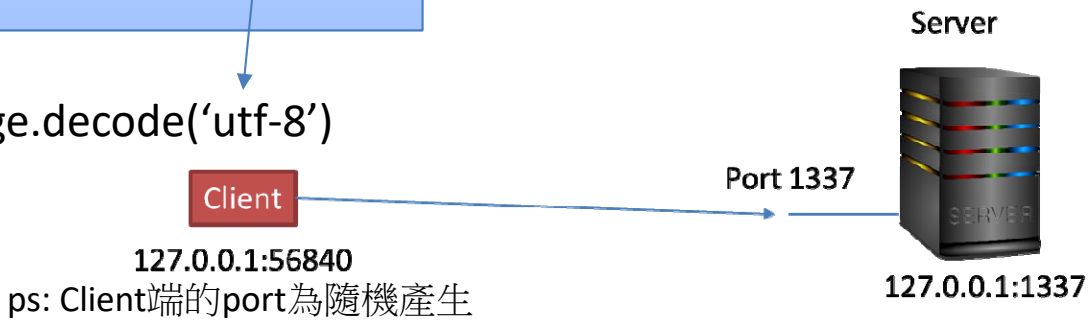
```
import socket

ss = socket.socket(family=socket.AF_INET,
                   type=socket.SOCK_DGRAM)
ss.bind(("127.0.0.1", 1337))

while True:
    message, address = ss.recvfrom(1024)
    clientMsg = f"Message from Client:{message}"
    clientIP = f"Client IP Address:{address}"
    print(clientMsg)
    print(clientIP)
```

指定使用IPv4

指定使用UDP

最大buffer size =1024 每次收 1024

Binary to text的方法: message.decode('utf-8')

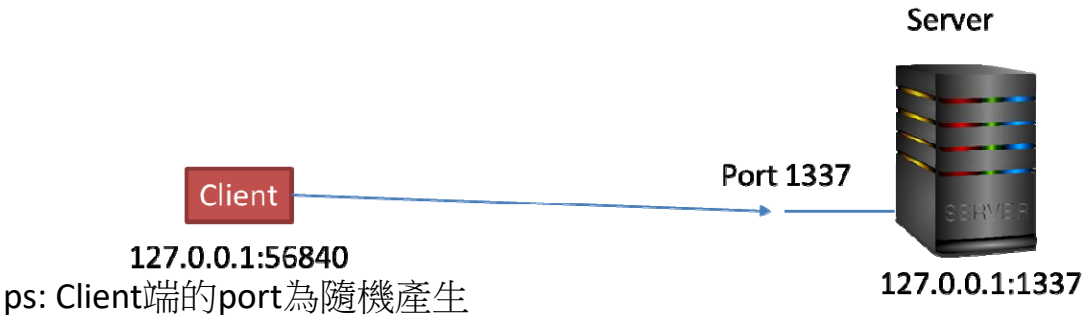


Python UDP Client

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s.connect(("127.0.0.1", 1337))
msg = b"Hello UDP Server"
s.send(msg)
s.close()
```

加個 b 就能轉了

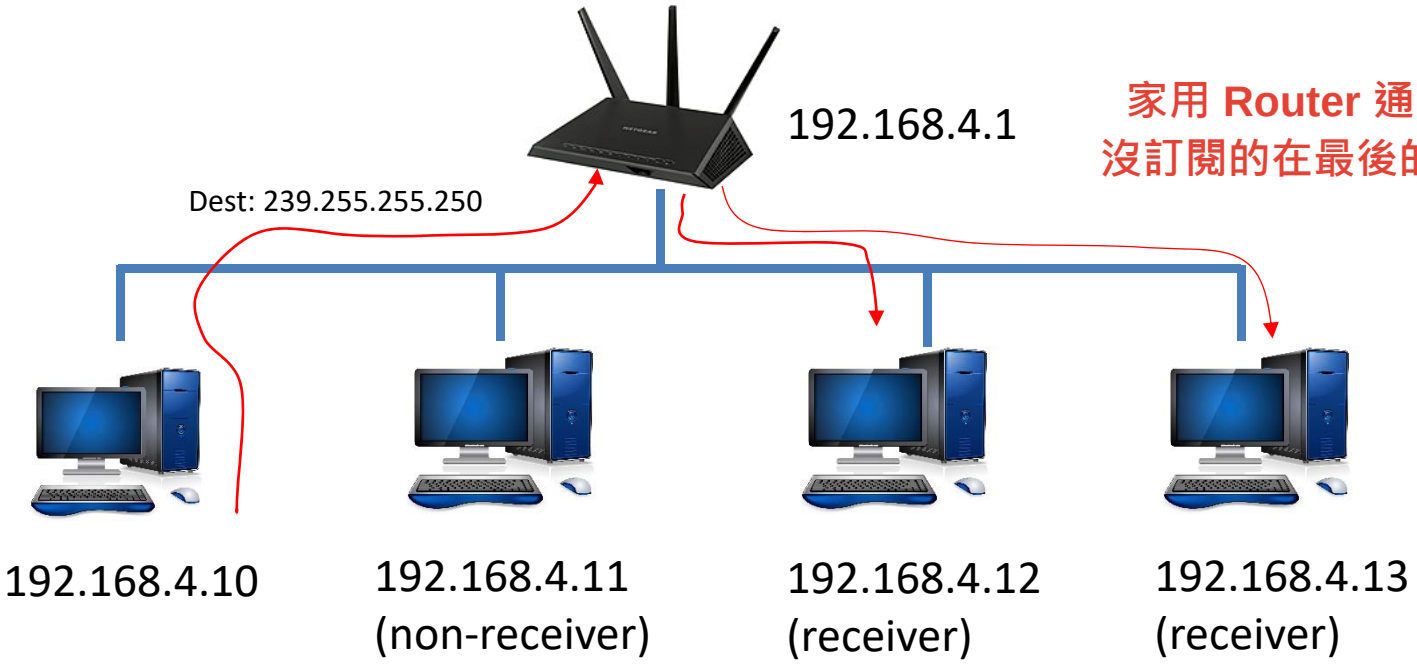
要先轉成byte才能送出



UDP Multicast

進行IP Multicast
Sender會送訊息到某個class D的IP (224.0.0.0–239.255.255.255)**目的位址在這裡面，路由器會當作群播要求**
Receiver向switch(router)訂閱同一個multicast address (add membership)

Router二種機制:
With IGMP:
　Router在接收到Sender送到該multicast address的封包後派送到有訂閱的receiver
Without IGMP:
　廣播封包到所有主機，由該主機網路卡過濾是否接收封包



Multicast Sender

同台機器當作 **sender and receiver**

同一台機器，如果需要多個receiver的話，要多個IP 要bind到239.255.255.250

```
let dgram = require('dgram');
let server = dgram.createSocket({type: 'udp4', reuseAddr: true});

server.bind(2391, () => {
    setInterval(() => {    每5秒送一次訊息
        server.send('Test', 2390, "239.255.255.250");
    }, 5000);
});          ms

server.on('message', function (msg, rinfo) {
    ...
});
```

回應時的處理方式

receiver 回覆給 sender

Multicast Receiver

```
let dgram = require('dgram');
let server = dgram.createSocket({type: 'udp4', reuseAddr: true});

server.bind(2390, () => {
    server.addMembership('239.255.255.250');
});

server.on('message', function (message, remote) {
    …
});
```

訂閱群播訊息

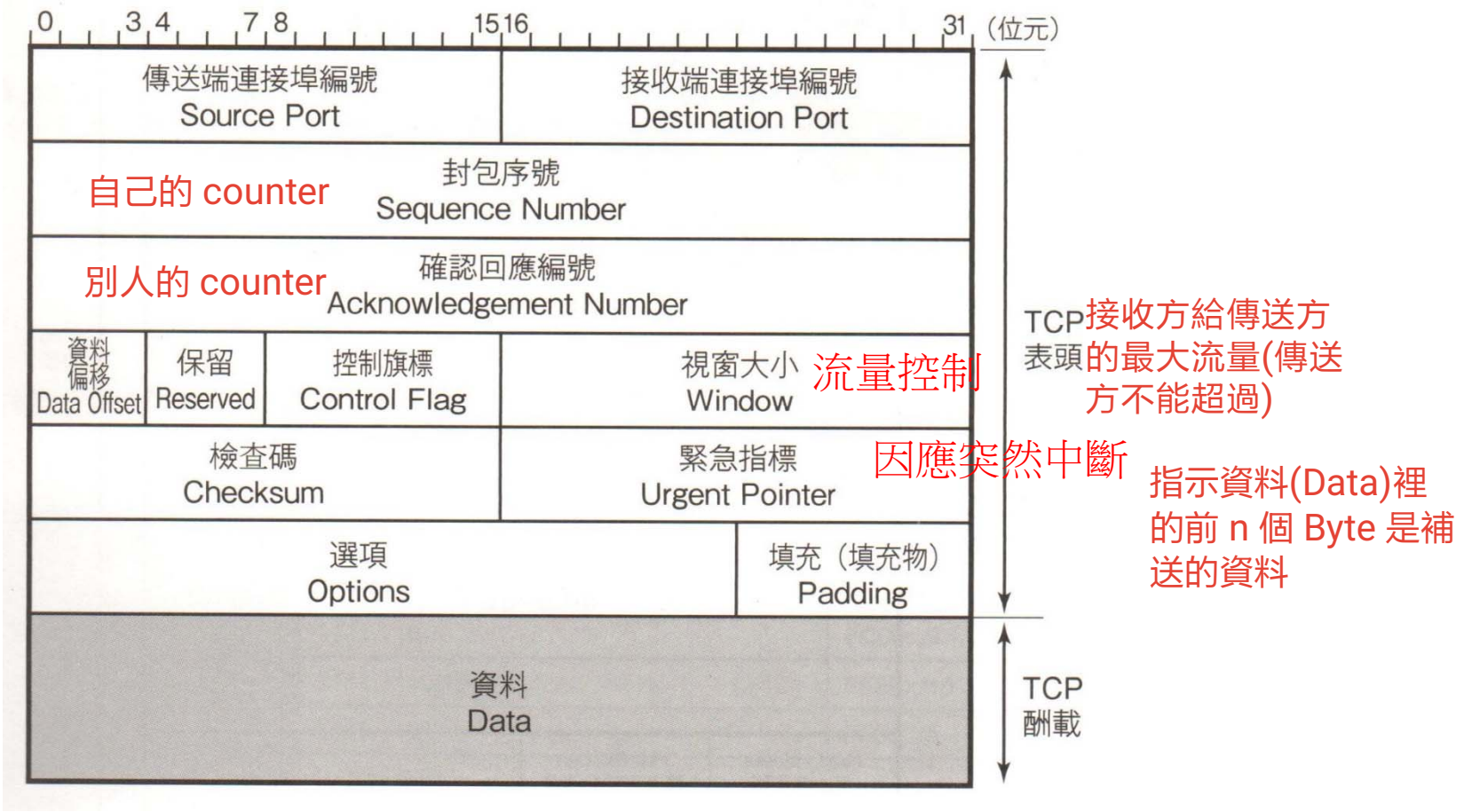
收到群播訊息時的處理方式

Lab説明

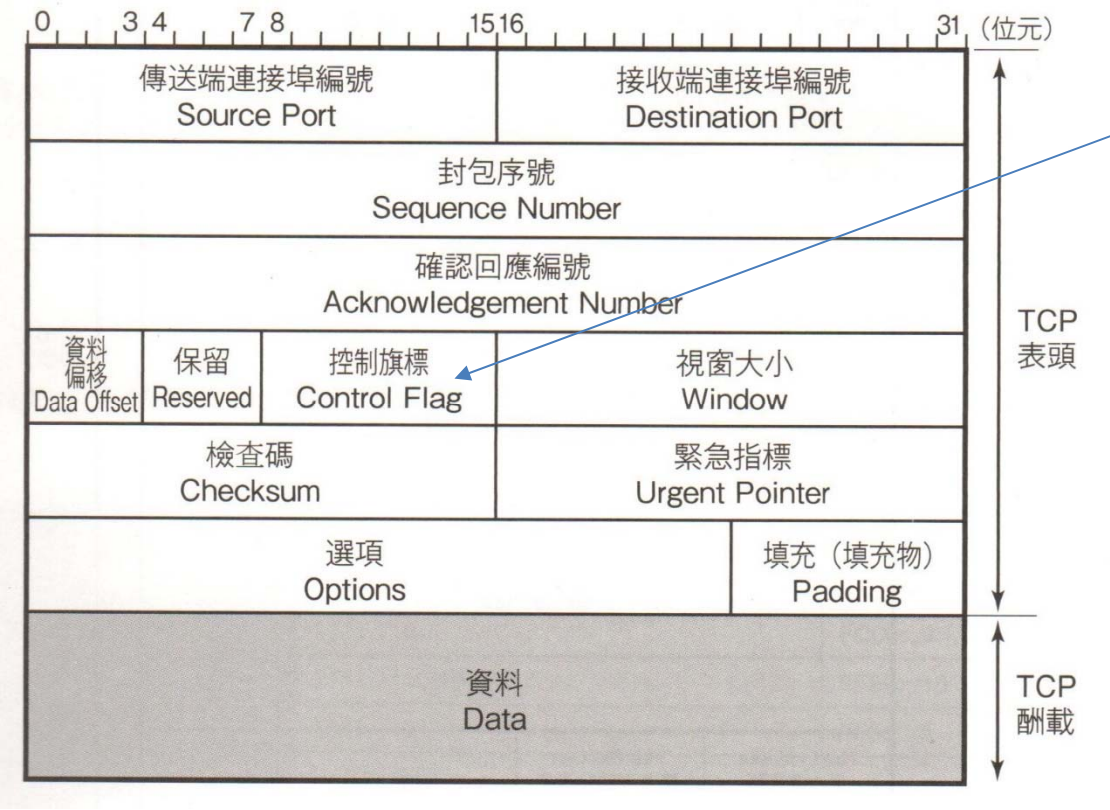
TCP 封包格式

可確保交易依次序到達

IP 接在上面



TCP 封包格式

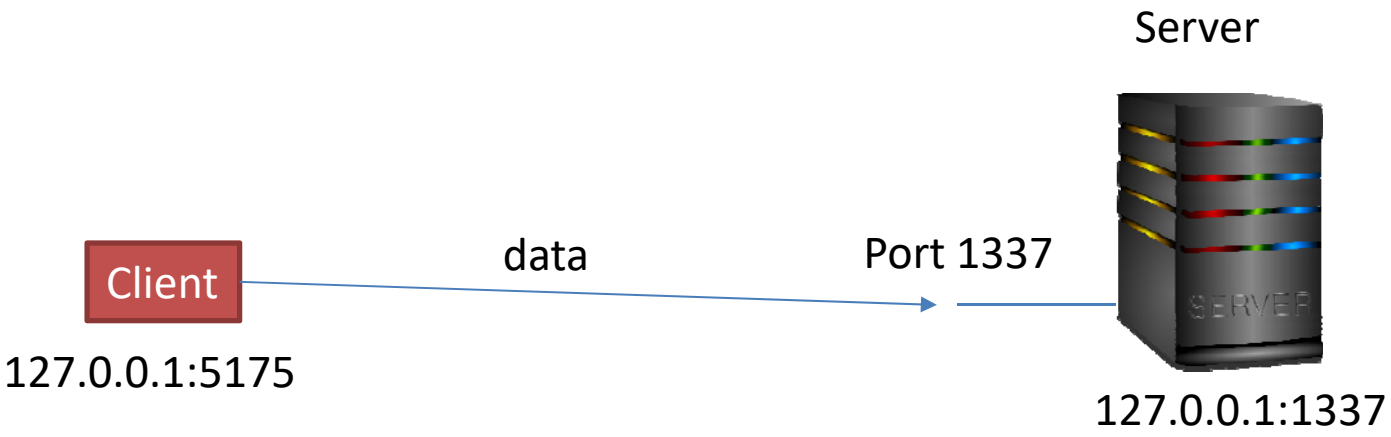


Flags : 控制資料的傳輸與連結

URG : 有使用緊急指標
ACK : 封包是否有ACK
PSH : push function
RST : 重設連結
SYN : 建立順序號碼
FIN : 傳送資料到此為止

例:
SYN, ACK=010010
ACK=010000
PSH, ACK=011000
FIN, ACK=010001

傳送範例


Server
Client
data
Port 1337
SERVER
127.0.0.1:5175
127.0.0.1:1337

TCP Server

- 伺服器

```
let net = require('net');
let server = net.createServer(function(socket) {
    socket.on('data', function(data) {
        ……
    });
});
server.listen(1337);
```

net.Socket

Buffer or String

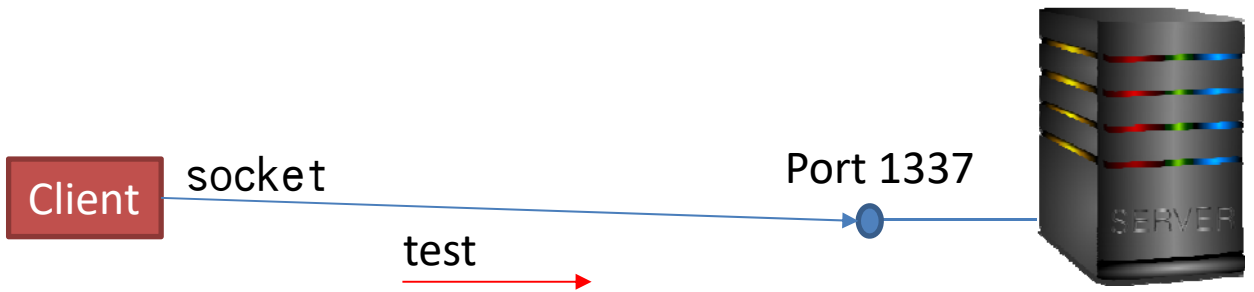
當收到資料時要做什麼?

Port 1337

練習查 API

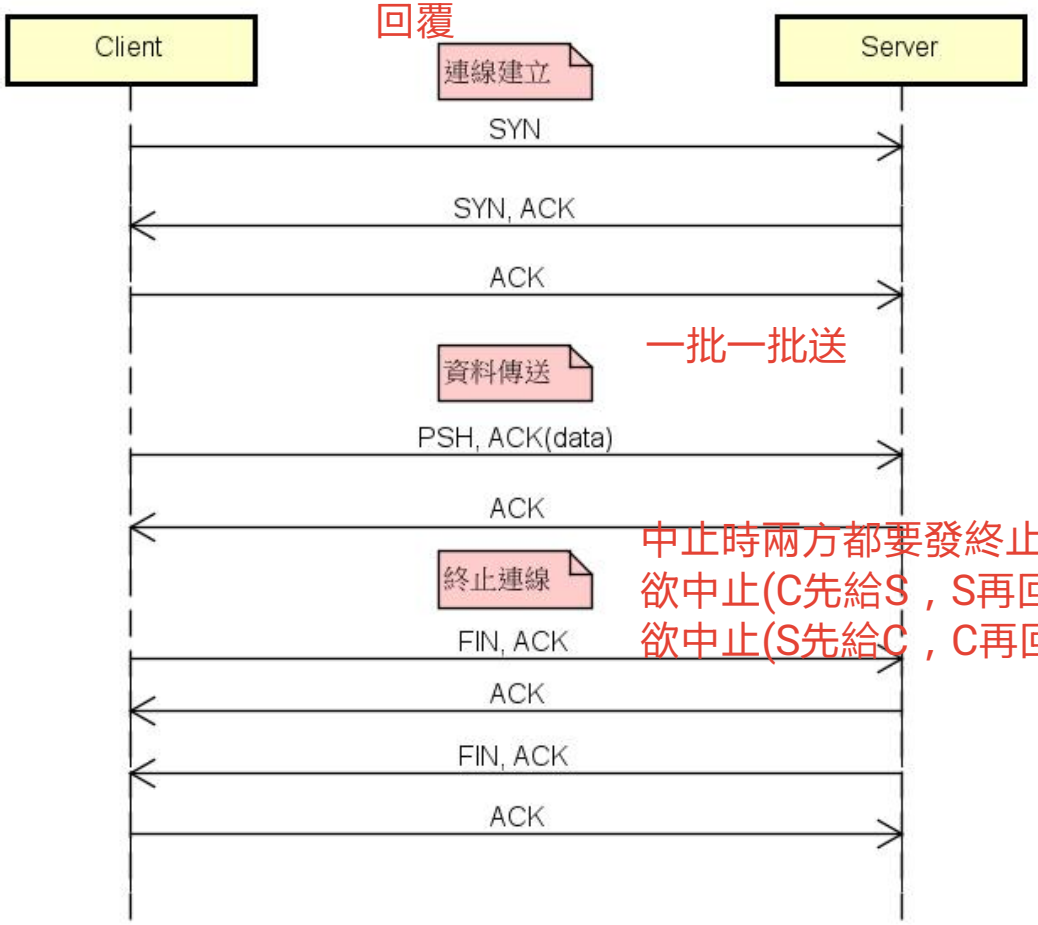
TCP Client

```
let net = require('net');

let client = new net.Socket();

client.connect(1337, '127.0.0.1', function () {
        console.log('connected');
        client.write('test', () => console.log('written'));
        client.end();
    }
);
```



TCP 3-way Handshaking

傳 SYN，Server 接受，
Client 傳收到 Server 的
回覆

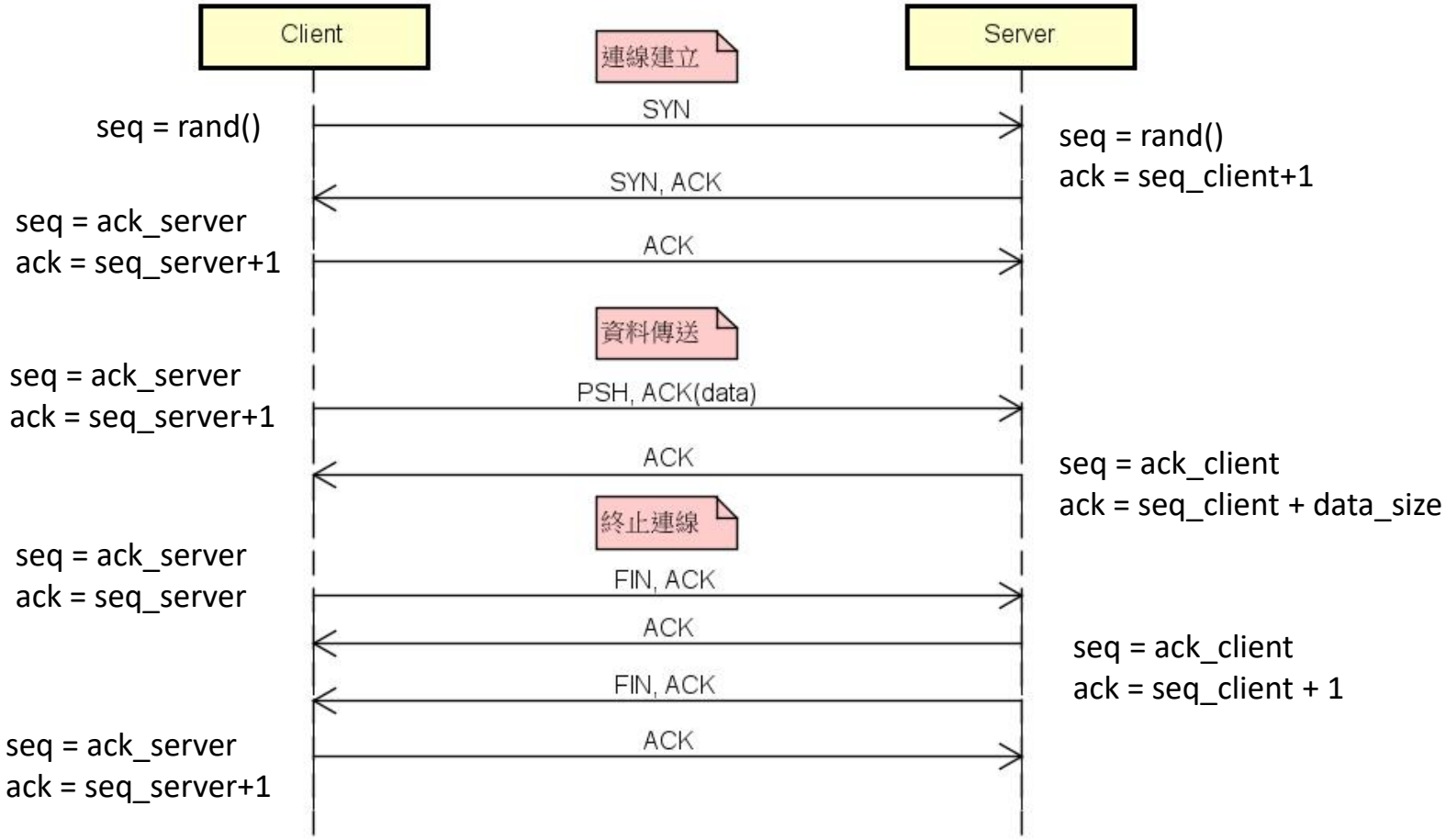


一批一批送

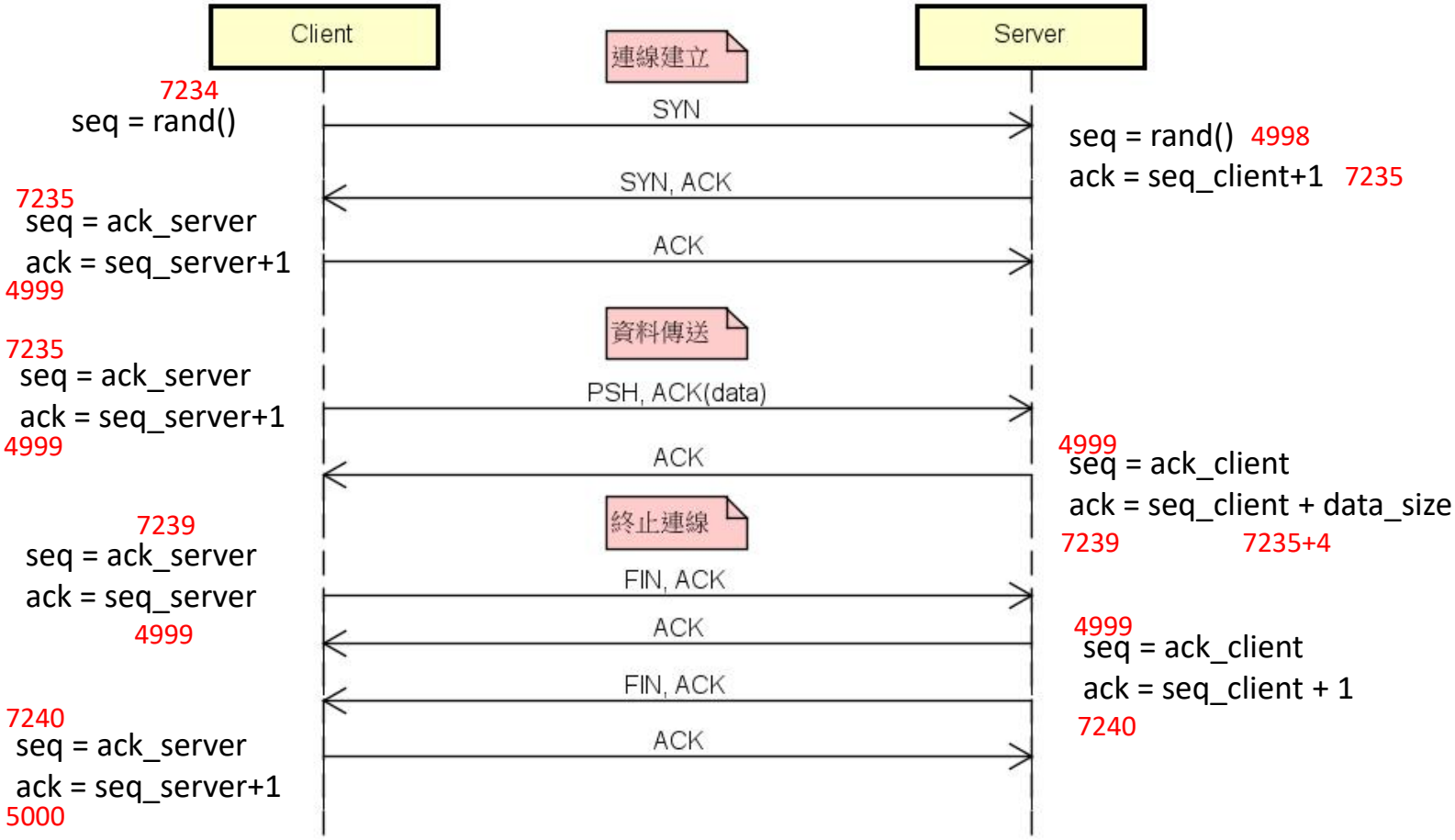
中止時兩方都要發終止請求
欲中止(C先給S，S再回傳)
欲中止(S先給C，C再回傳)

寫程式欲終止三方交握時
=> C和S都要寫 close()

TCP 3-way Handshaking
SEQ and ACK

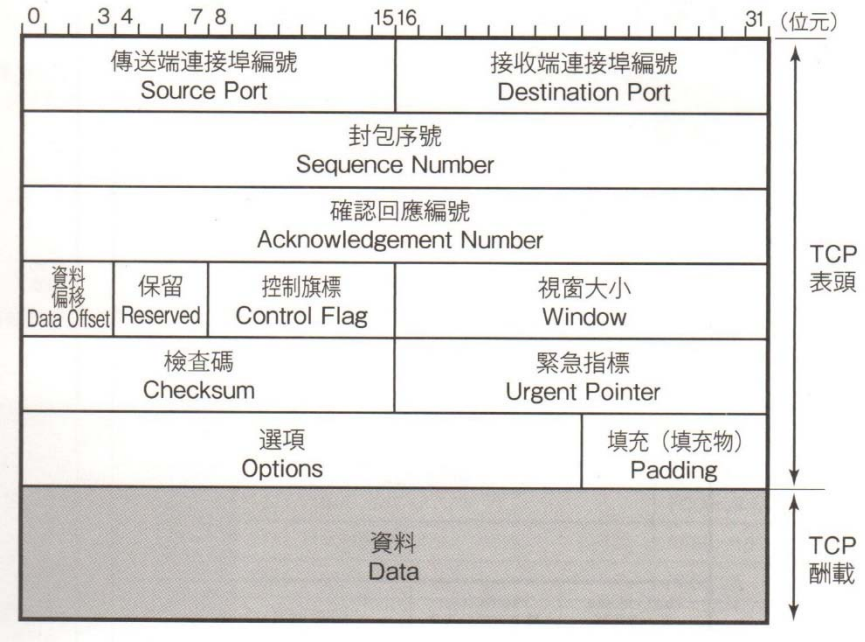


TCP 3-way Handshaking
SEQ and ACK Example



封包觀察

No.	Time	Source	Destination	Protocol	Length	Info
257	6.206781	127.0.0.1	127.0.0.1	TCP	56	5175 → 1337 [SYN] Seq=0
258	6.206891	127.0.0.1	127.0.0.1	TCP	56	1337 → 5175 [SYN, ACK]
259	6.206913	127.0.0.1	127.0.0.1	TCP	44	5175 → 1337 [ACK] Seq=1
260	6.211008	127.0.0.1	127.0.0.1	TCP	48	5175 → 1337 [PSH, ACK]
261	6.211094	127.0.0.1	127.0.0.1	TCP	44	1337 → 5175 [ACK] Seq=1
262	6.212223	127.0.0.1	127.0.0.1	TCP	44	5175 → 1337 [FIN, ACK]
263	6.212239	127.0.0.1	127.0.0.1	TCP	44	1337 → 5175 [ACK] Seq=1
264	6.215214	127.0.0.1	127.0.0.1	TCP	44	1337 → 5175 [FIN, ACK]
265	6.215235	127.0.0.1	127.0.0.1	TCP	44	5175 → 1337 [ACK] Seq=6



Q & A