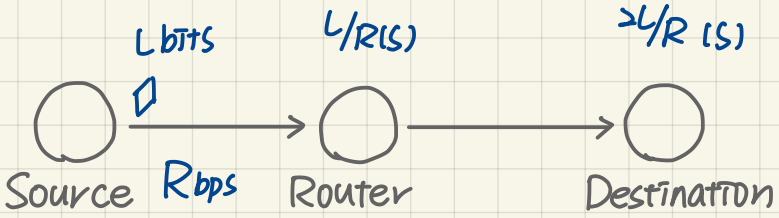
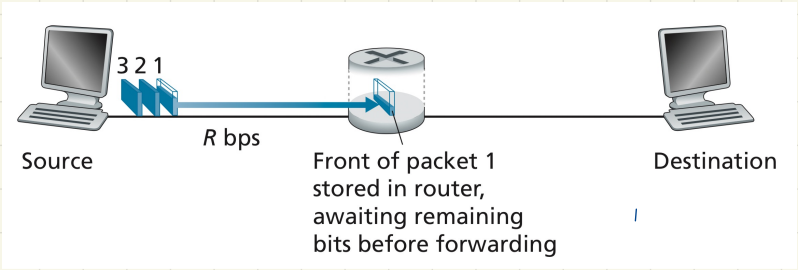
• Store-and-Forward Transmission

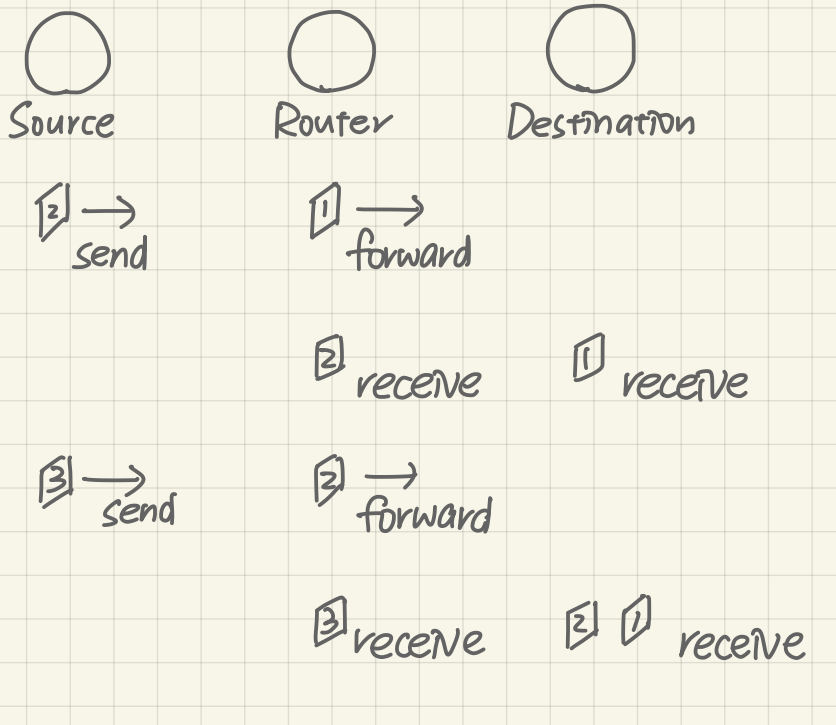
- L bits / packet



total delay = 2L/R (s)



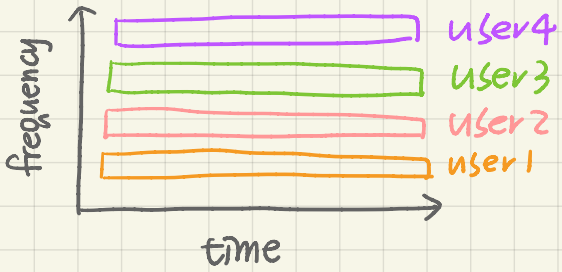
Router: recieve → store → process → forward



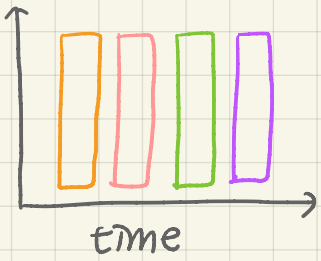
end-to-end = $N\frac{L}{R}$

- path consist N links each rate R
  - i.e. (N-1) routers between source & destination

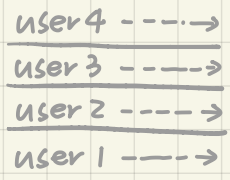
FDM



TDM



- FDM (Frequency-Devision Multiplexing)
  → each circuit continuously gets a fraction of bandwidth
  → 依照頻率切割
  → 每個 user 同時在不同頻率上傳輸
    ↳ 設置 f 間距



[stimutanously]

i.e. → 1條線路在一段時間內能提供多個 user 傳送訊號，
每個user各自使用1個頻道 ex. 有線電視

- TDM (Time-Devision Multiplexing)
  → each circuit gets all of the bandwidth periodically
  → 依照時間切割
  → 每個 user 各自使用一個時間區段 (time shot)
    ↳ 是週期性的

ex. 每1小時就有4人要搭公車，每台公車只能載4人
→ 現有8人要搭公車 → 後4人要等前4人離开，公車每小時一班

i.e. → 劃分時間，讓多個user共享一個信道，
每個 user 佔用的時隙相同

· Packet Switching

不会保留一個特定的Path

要進去Buffer排隊
↳Packet太多的時候

不会保留頻寬，
只要有空位來就可以坐

Circuit Switching

會保留一個特定的Path

進去之前先預定好Path了

最多只能容納10個user同時使用

∵會保留頻寬給user

∴user還沒進行坐輸時就會一直保留，
導致後面進來的user只能卡在buffer裡)

User share 1 Mbps link
constant rate 100 kbps
active 10% of the time

circuit-switching TDM 100ms

user support: $\frac{1M}{100k} = 10$ user at the same time

packet-switching

active : 0.1

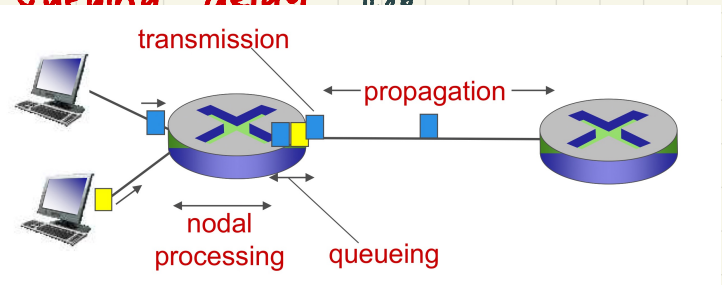
propagation delay p.37

- dtrans
- dend-to-end
- dprop

definition

packet switching
circuit switching

Queuing delay p.36



$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$

· Transmission Delay

- packet's length : L bits
- transmission rate of the link R bits/sec

→ transmission Delay : $\frac{L}{R}$

↳ the amount of time required to push (transmit) all the packet's bit into the link.

· Propagation Delay

- distance between router A and router B : d
- propagation speed of the link : s

→ propagation Delay : $\frac{d}{s}$

↳ time required to propagate from the beginning of the link to router B (from one router to the next)

· Tproc : nodal processing
→ check bit errors
→ determine output link
→ typically < msec

dqueue
→ time waiting at output link for transmission
→ depends on congestion level of router

caravan analogy

Queueing delay $\frac{La}{R}$ ~ 0 / → 1 / > 1

packet loss

✓ Throughput

Rs < Rc
Rs > Rc

Internet Scenario

roadmap
layering

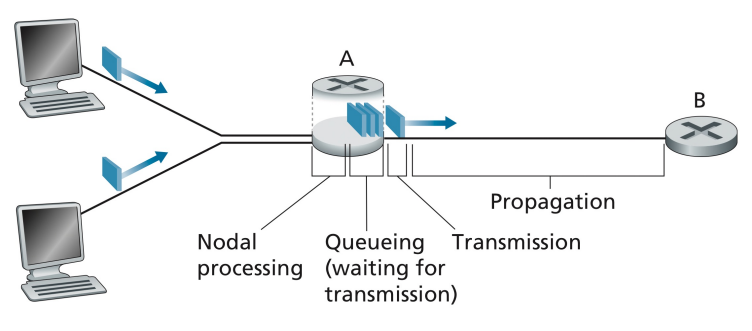
Internet protocol stack
↳ application
transport
network
link
physical

ISO/OSI ↳ presentation / session

Network Security

DoS.

Delay



- **Processing Delay**
  - time to examine the packet's header and determine where to direct the packet.

- **Queuing Delay**
  - time to wait to be transmitted onto the link
  - i.e. queue is empty, no other packet is currently being transmitted → $T_{queue} = 0$
  - i.e. traffic is heavy, many packets are waiting → $T_{queue} >> 0$

- **Transmission Delay**
  - packet's length : L bits
  - transmission rate of the link : R bits/sec
  - → $T_{trans} = L/R$
  - the time required to push (transmit) all of the packet's bits into the link.

- **Propagation Delay**
  - distance between router A and router B : d
  - propagation speed of link : s
  - → $T_{prop} = d/s$
  - time require to take a bit to propagate from one router to the next.

- $d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$

Traceroute

- output → 6 columns
  - 1st column : n value described (the number of the router along the route)
  - 2nd column : name of router
  - 3rd column : router's address
  - last 3 columns : round-trip delays for 3 experiments
  - example:

```
1  gw-vlan-2451.cs.umass.edu (128.119.245.1)  1.899 ms 3.266 ms  3.280 ms
2  j-cs-gw-int-10-240.cs.umass.edu (10.119.240.254) 1.296 ms 1.276 ms
   1.245 ms
3  n5-rt-1-1-xe-2-1-0.gw.umass.edu (128.119.3.33) 2.237 ms  2.217 ms
   2.187 ms
4  core1-rt-et-5-2-0.gw.umass.edu (128.119.0.9) 0.351 ms 0.392 ms 0.380 ms
5  border1-rt-et-5-0-0.gw.umass.edu (192.80.83.102) 0.345 ms 0.345 ms
   0.344 ms
6  nox300gw1-umass-re.nox.org (192.5.89.101) 3.260 ms  0.416 ms 3.127 ms
7  nox300gw1-umass-re.nox.org (192.5.89.101) 3.165 ms 7.326 ms  7.311 ms
8  198.71.45.237 (198.71.45.237) 77.826 ms 77.246 ms 77.744 ms
9  renater-1b1-gw.mx1.par.fr.geant.net (62.40.124.70) 79.357 ms 77.729
   79.152 ms
10 193.51.180.109 (193.51.180.109) 78.379 ms  79.936 80.042 ms
11 * 193.51.180.109 (193.51.180.109) 80.640 ms *
12 * 195.221.127.182 (195.221.127.182) 78.408 ms *
13 195.221.127.182 (195.221.127.182) 80.686 ms 80.796 ms 78.434 ms
14 r-upmc1.reseau.jussieu.fr (134.157.254.10) 78.399 ms * 81.353 ms
```

name / address / round-trip delays

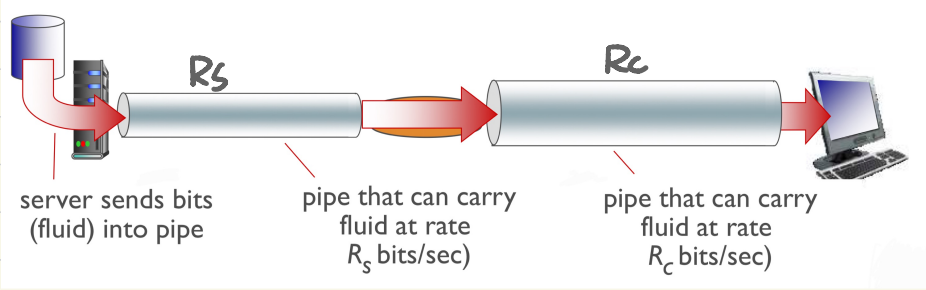
(14 routers)

*Traceroute

- Packet 從你的電腦到網路另一端電腦所走的路徑
- 測量時間：送出封包→抵達目的地→回來
- 1條路徑測3次

Throughput (吞吐量)

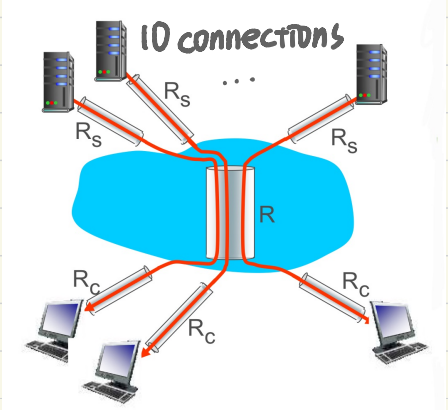
- rate (bits / time) that bits transferred between sender / receiver
- instantaneous : at given point in time
- average : over long period of time



*Throughout

- 以一個時間區間為單位
- 單位時間內可執行「幾次」操作 / 運算的「次數」

example : file consist F bits, transfer takes T sec ⇒ average throughput : $\frac{F}{T}$ (bits/sec)



per-connection end-end throughput : $\min(R_c, R_s, \frac{R}{10})$

Protocal layers (協定層級)

- layers: each one implements a service
  - rely on service provided by layer below

*Layer
- 個別層級有自己的服務，個別執行後再向下層玩源
- 下層TCP賴上一層的服務

Internet protocol stack

第1層
1. application → 分佈於 end system 上 (end system 則將 protocol 拿去換 packet)
2. transport → 加上 header information ex. TCP, UDP
3. network → routing from source to destination
   - 負責 datagrams
4. link → data transfer between neighboring network elements
   - 在node中玩輸 packet / datagram
   (在 link-layer 的 packet 又稱 frames)
5. physical → bits on the wire (實體線路)

ISO/OSI reference model

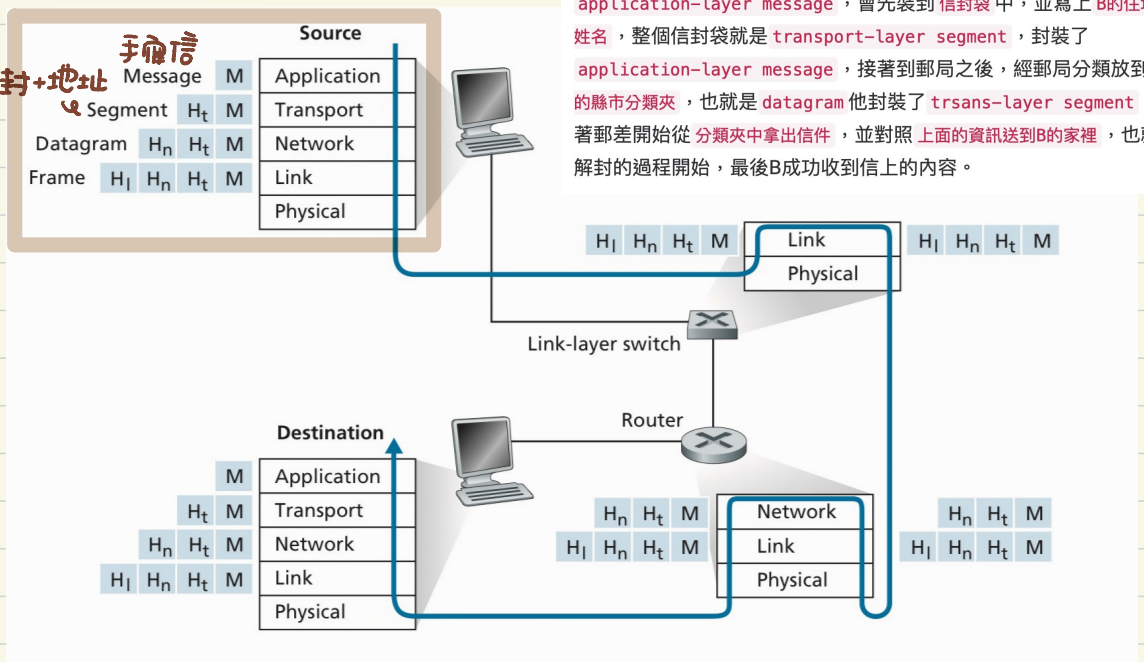
7. application
6. presentation → allow applications to interpret meaning of data
5. session → 設定和電腦的通訊連接
4. transport
3. network
2. link
1. physical

*TCP - messages的玩法
- packet 切成 segments
- message flow control
- packet 塞車處理机制

UDP - 提供無網路狀態時 web app 的 service

Encapsulation

信+信封+地址 (手寫信)



以一個假設來說，今天A想要寄信到B，首先A手寫的信，也就是 application-layer message，會先裝到信封袋中，並寫上B的住址以及姓名，整個信封袋就是 transport-layer segment，封裝了 application-layer message，接著到郵局之後，經郵局分類放到對應的縣市分類夾，也就是 datagram 他封裝了 trsans-layer segment，接著郵差開始從分類夾中拿出信件，並對照上面的資訊送到B的家裡，也就是解封的過程開始，最後B成功收到信上的內容。

Bad Guys

- **Malware**（惡意軟件）
  - virus：can modify or delete data
  - worm（蠕蟲）：DOESN'T modify the program
    replicates itself more and more to slow down the computer system

- **Spyware malware**（間諜程式）
  - record keystrokes（鍵盤側錄）, web sites visited ...

- **Botnet**（殭屍網路）
  - 隨著 email，通訊軟体侵入电腦，再躲到別的程式，
    遇到有漏洞的电腦主机就自行 attack

- **DoS（Denial of Service）**（阻斷服務攻擊）
  - select target
  - break into hosts around the network
  - send packets to target from compromised hosts
  - 藉由不當方式佔用系統分享資源，干擾系統运行
  - 不一定要取得系統使用权
    ↳ ex. message flood，發送大量無意義訊息，佔用網路頻寬

- **Packet sniffer**（封包監聽工具）
  - a passive receiver that records a copy of every packet that flies by

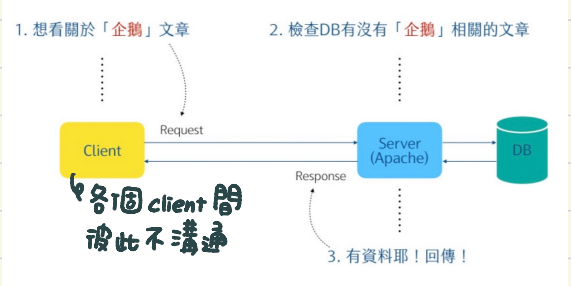
  ex. wireshark

- **IP spoofing**
  - send packet with false source adress

Application Architectures

- **Client-server**
  - always-on host
  - permanent IP adress
  - serve：提供 web 服務相關的伺服器
  - client：呈現 sever 端所玩來的資訊

  ex.



↳ 各個 client 間彼此不溝通

↳ 透过 HTTP 協定溝通

- **P2P**
  - no always-on server
  - arbitrary end systems directly communicate
  - peers request service from other peers,
    provide service to other peers
  - self scalability
  - peers are intermittently connected and change IP

·Process

- program running within a host

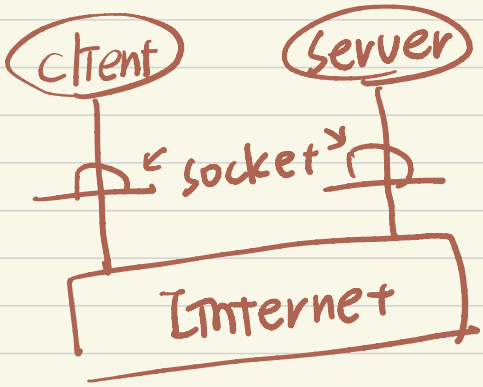
same host, 2 processes → inter-process

exchangeng message

Socket → 網路上的通訊端點.

使用者 or app 只要連接到 socket 便可以和網路上任一通訊端點連線 (類似建立通道的概念)

→ sockets' communication is similar to process



TCP → reliable
→ in-order

UDP → connectionless
→ unreliable data transfer
→ x flow control
→ x congestion control

Transmission Control Protocol

↳ TCP → 每個封包分配唯一一個識別碼 & 序號

↳ ∴ 接收端識別封包之完整性 & 順序 (一次一個封包)

→ 收到封包後, 若順序正確 → 回傳確認信號 (Acknowledgement)

錯誤 → 再傳送一次封包

↳ ex. email
web visiting
檔案傳輸

User Datagram Protocol

→ 耗時, 但正確性高

↳ UDP → 啥也不管, 就一直丟封包

→ 串流方式傳送 ↳ 即時服務

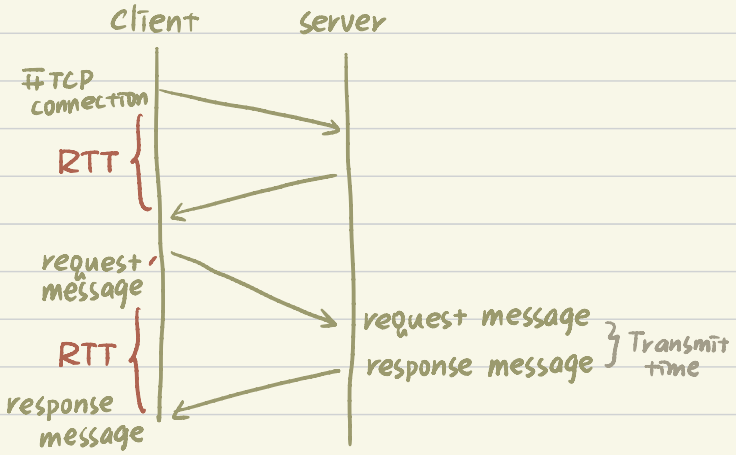
↳ ex. online game / phone call
stream video 串流媒體

Non-Persistent

ex. 有10張圖要玩

1. HTTP client process 開起 11條 TCP connection
2. client 透過 socket 開始玩送 message
3. sever 透過 socket 接收 request message
   回玩 封裝好的 response message
4. HTTP server process 告訴關閉 TCP connection (還沒關)
5. client 接收到 response, TCP connection 關閉, 開始檢查回玩的 HTML, 找圖片

repeat 10 times



RTT: packet 從 client 到 server 來回所需時間

T_{tot} = 2RTT + transmit time

Persistent HTTP

→和 non-persistent 相比,
  但 connection 不關閉
  且可連續發出 request

TLS是啥

URL

DNS

SMTP

CDN

host 的 { client / server

handshaking → prepare for data transfer ahead of time

flow control → sender wont overwhelm receiver

congestion control → senders "slow down sending rate" when network congested

Coaxial cable	Fiber
→ 双向的	→ 高速
→ 寬頻的	→ 不易錯

TCP & UDP

L4

HTTP < non-persistent / persistent

Multiplexing → 多工.

→ add transport header

→ 從不同socket收集資料片段,
再將它們封裝給網路層

demul → 分工 IPadress & port numbers

→ use header to deliver segment
to correct socket

→ 把運輸層區段中的資料
送給正確的socket工作

UDP checksum → add 2 16-bit integers

→ carryout 要加回去

→ 取 1's comp 得 checksum

SMTP → phases

① handshaking

② transfer of message

③ closure

→ commands : 4 bit ASCI

→ response: status code & phrase

→ persistent connections

Intra-AS Routing (IGP)

- RIP: Routing Information Protocol
  - → Bellman-Ford
- OSPF: Open Shortest Path First
  - → Dijsktra
- IGRP: Interior Gateway Routing Protocol
  - → Bellman