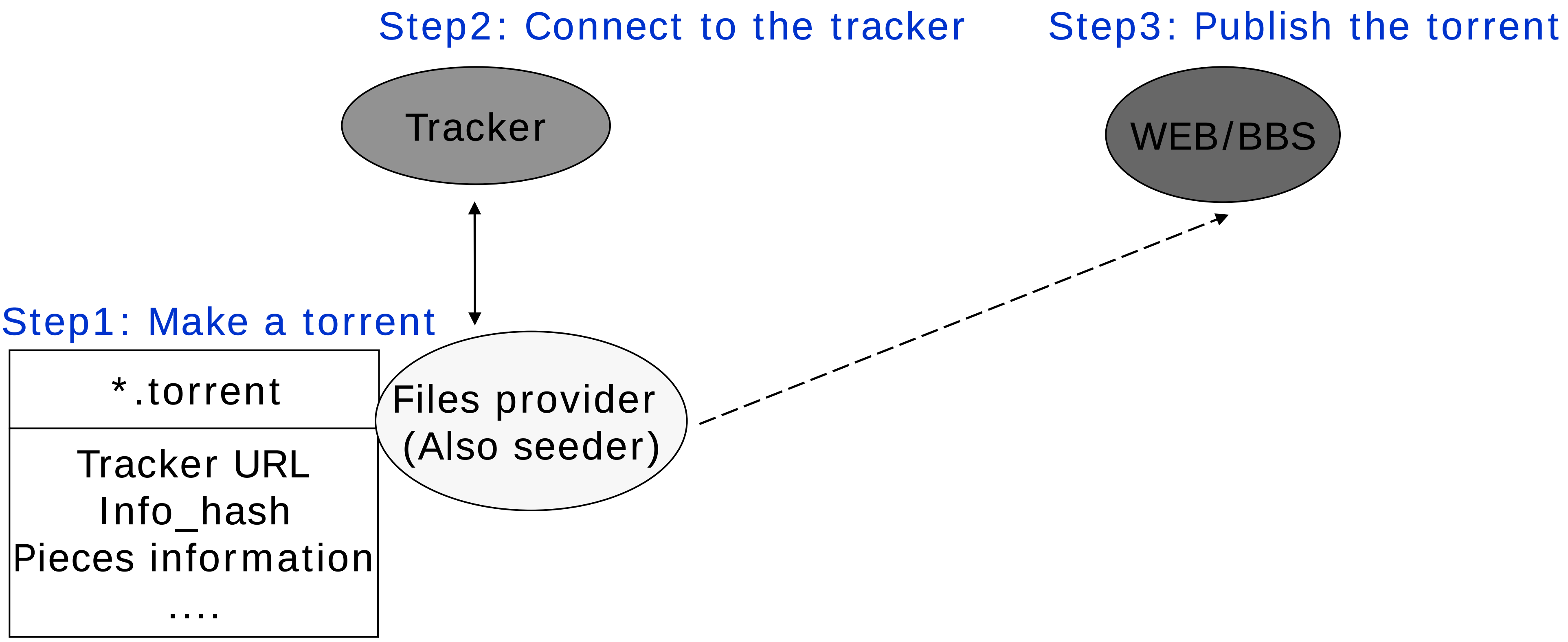
# Understanding BitTorrent

- Reference:
  - Incentives Build Robustness in BitTorrent
    - Bram Cohen, May 22, 2003
  - Understanding BitTorrent : An Experimental Perspective
    - Arnaud Legout, I.N.R.I.A., Guillaume Urvoy-Keller and Pietro Michiardi, Institut Eurecom Sophia Antipolis, France, Technical Report, November 2005
- Acknowledgement:
  - These slides were made by Ting-Liang Chou (周鼎量) & Che-Yi Lin (林哲毅)

## Outline

- BitTorrent
  - Publish File
  - Download Shared File
  - Upload Policies
  - Download Policies
  - Implementation
- Experiment
  - Methodology
  - Choking Algorithm
  - Protocol Overhead

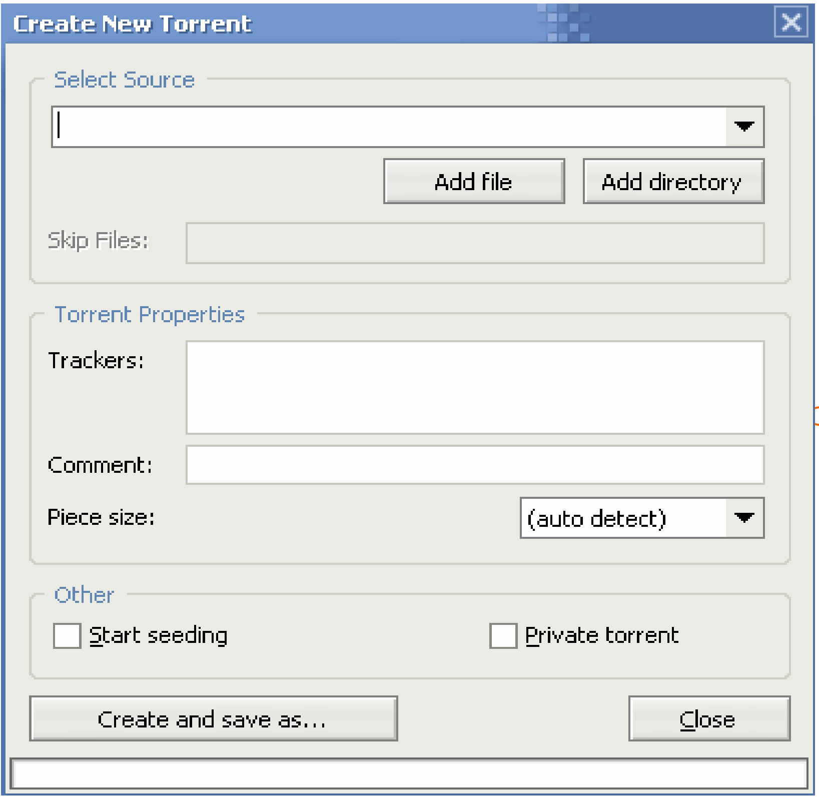
## Publish Files



# Make A Torrent

## Step1: Make a torrent

BT Application- uTorrent



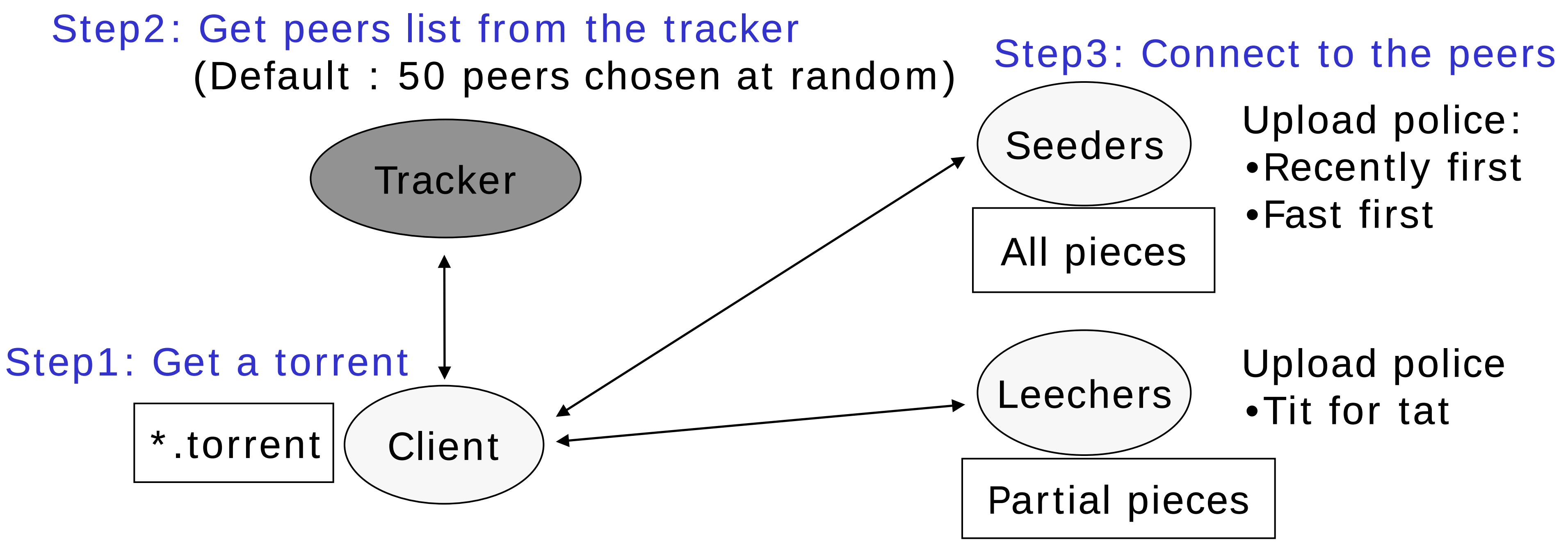
Tracker URL

```
dictionary
 [announce]    http://tpb.tracker.thepiratebay.org/announce
 [creation date]   1179784238

   [name]     Maroon 5
   [piece length]     262144
   [pieces]     347d2142a361fc9ddd22a289f56e07197143075770a.....

nodes:
trackers:
0: http://tpb.tracker.thepiratebay.org/announce
number of pieces: 331
piece length: 262144
info hash: 7b8dd66d3ddd01f62c166a4686cf0d1def2d06b5
created by:
files:
      6438912 Maroon 5/01  Maroon 5 - If I Never See Your Face Again.mp3
      6756352 Maroon 5/02  Maroon 5 - Makes Me Wonder.mp3
      4392960 Maroon 5/03  Maroon 5 - Little Of Your Time.mp3
      6447104 Maroon 5/04  Maroon 5 - Wake Up Call.mp3
```

Info_hash

Pieces information

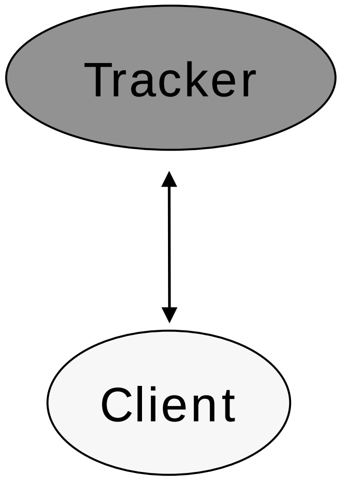
# Download Shared Files



Piece size：256KB
Block size:16KB

## Tracker

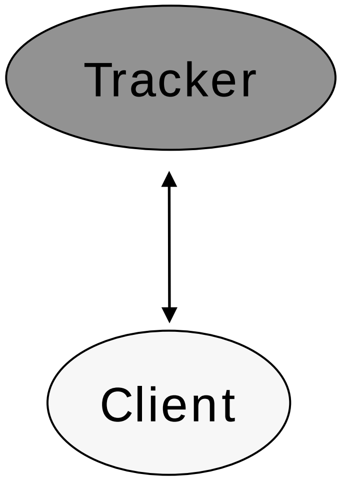
Step2: Get peers list from the tracker



- HTTP protocol
  - Client to tracker (periodically)
    - GET
      - [ID, File name, IP, Port]
  - Tracker to client
    - Check request
    - Return random list of peers
      - Involve with the same torrent file

# Tracker - Connection Behavior

Step2: Get peers list from the tracker

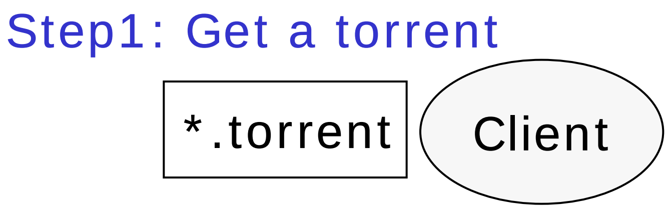


- Period
  - Timeout
    - uTorrent default update rate : 1 hour
  - Getmore

```
if self.ever_got_incoming(): //host probably is in the public network
    getmore = self.howmany() <= self.minpeers / 3
else:                        //host probably is behind the NAT
    getmore = self.howmany() < self.minpeers

if getmore or time() - self.last_time > self.announce_interval:
    self.announce()
```

## Download Shared Files
## Step 1: Get A Torrent

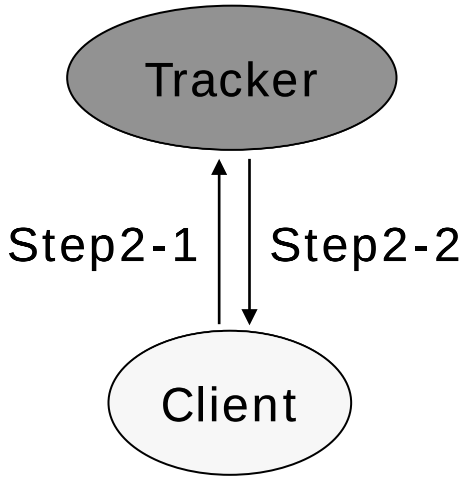
- Get *.torrent from internet



# Download Shared Files

## Step2: Get Peer List From The Tracker

Step2: Get peers list from the tracker



```
2 Tracker -> Client
  HTTP/1.0 200 OK
  d
  5:files
    d
    20:;lqt%
      d
      8:complete i35 e
      10:downloaded i3884 e
      10:incomplete i2 e
      e
    e
    5:flags
    d
    20:min_request_interval i3600 e
    e
  e
```

Seeders number:35

Leechers number: 2

Request interval: 1hr

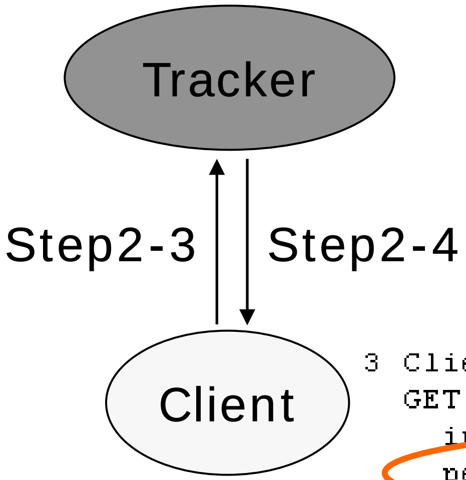
Info_hash

```
1 Client -> Tracker
  GET /scrape?info_hash=%dc%87%9a%3bl%ca%06q%e4%e6%ac%9d%b2t%8f%ee%f4%25%a4%b8 HTTP/1.1
  Host: www.torrent-downloads.to:2710
  User-Agent: uTorrent/1720
  Accept-Encoding: gzip
```

# Download Shared Files

## Step2: Get Peer List From The Tracker

Step2: Get peers list from the tracker



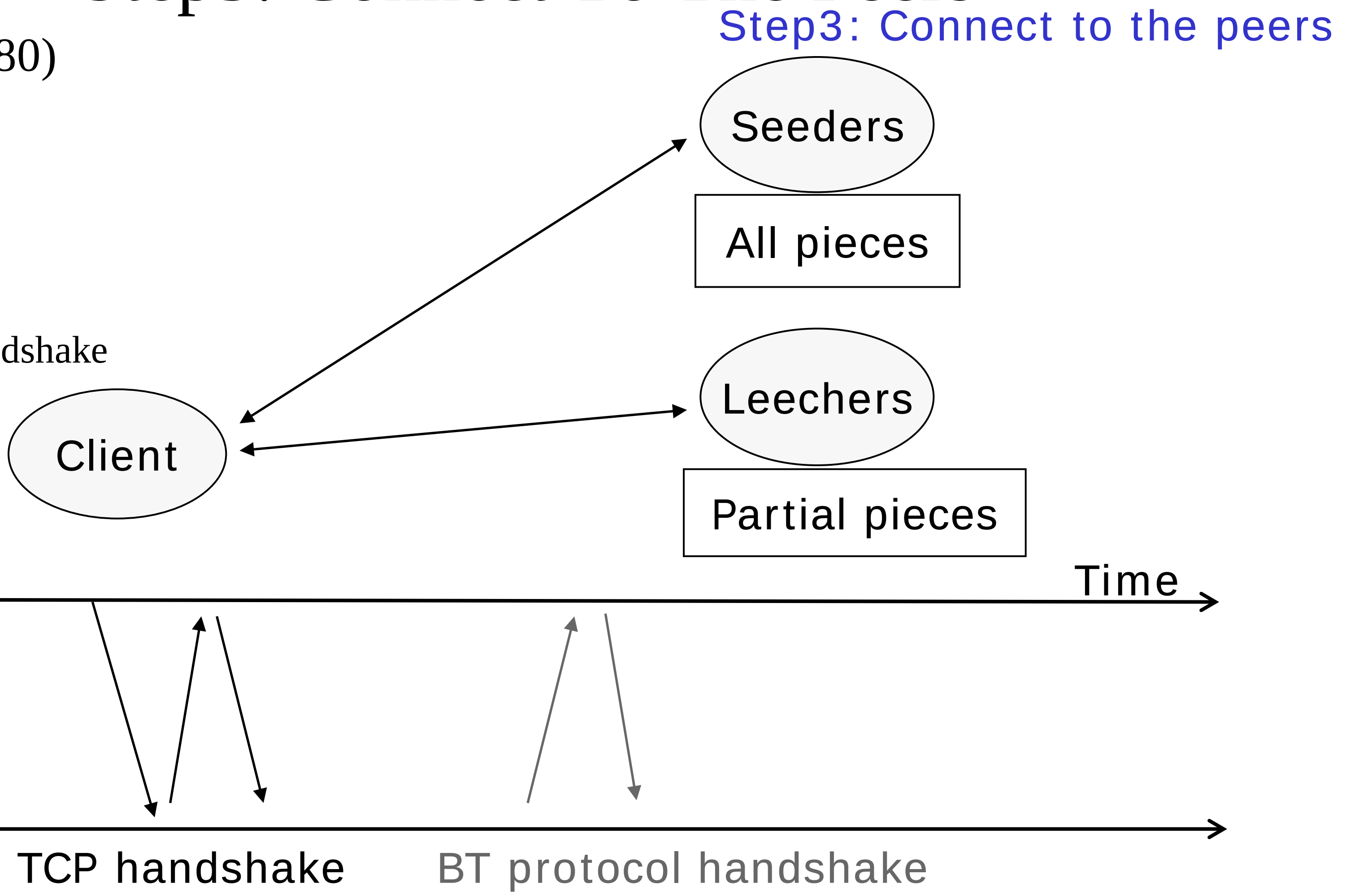
```
4 Tracker -> Client
  HTTP/1.0 200 OK
  d
  8:complete i35 e
  10:incomplete i3 e
  8:interval i3600 e
  12:min interval i3600 e
  5:peers
    228:qc%G|jdCYxqXjEFADC{[njCb!,FWM1B9=K&HA"T
  e
```

Encrypted [ID, IP, Port] list

```
3 Client -> Tracker
  GET /announce?
    info_hash=%dc%87%9a%3b1%ca%06q%e4%e6%ac%9d%b2t%8f%ee%f4%25%a4%b8 &
    peer_id=-UT1720-%82%8d%3a8%8aa%ee%7b%e3%e7%0b%1c &
    port=44131 &
    uploaded=0 &
    downloaded=0 &
    left=86213643 &
    key=D14E7DDB &
    event=started &
    numwant=200 &
    compact=1 &
    no_peer_id=1 HTTP/1.1
  Host: www.torrent-downloads.to:2710
  User-Agent: uTorrent/1720
  Accept-Encoding: gzip
```

# Download Shared Files Step3: Connect To The Peers

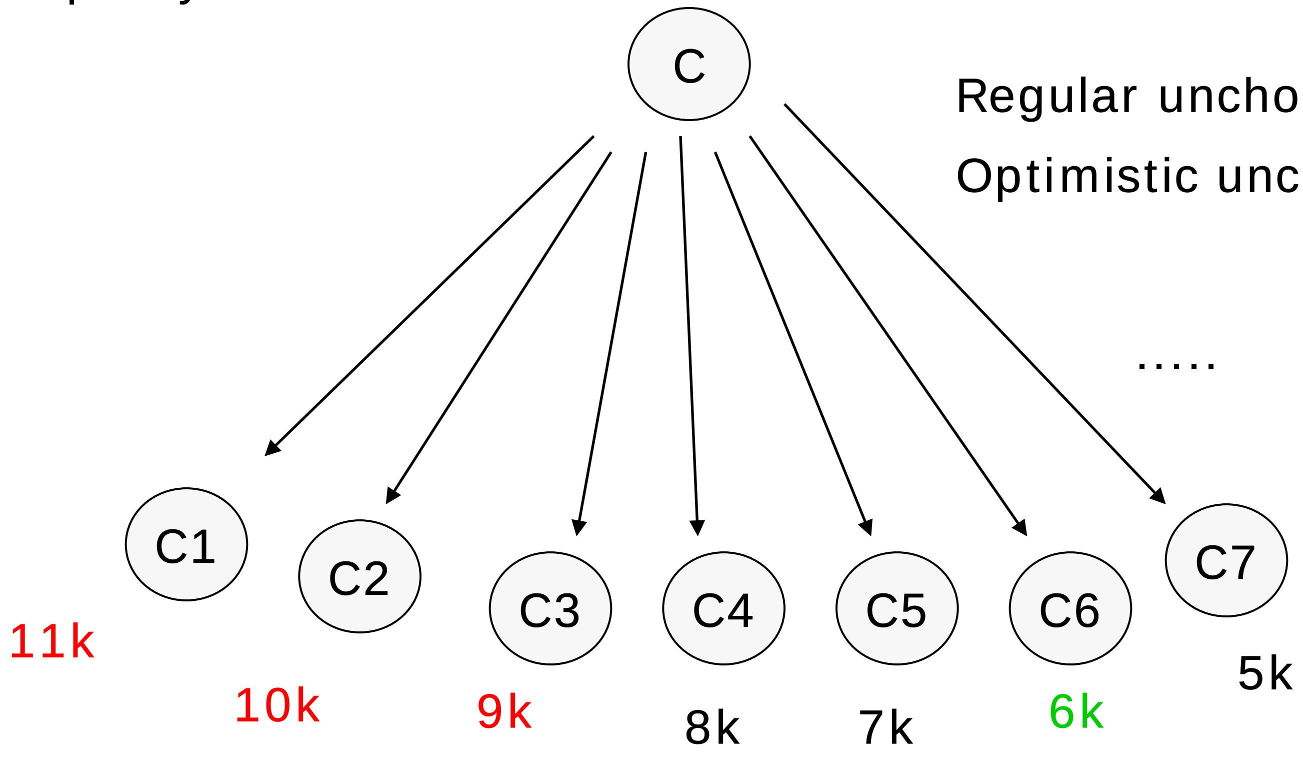
- Connection (ex: 80)
  - Active (ex: 40)
  - Passive (ex: 40)
- Handshake
  - TCP handshake
  - BT protocol handshake



# Upload Policies - Choking Algorithms

## Leecher state

Upload policy: Tit for tat



Regular unchoking : [C1,C2,C3]

Optimistic unchoking : [C6]

Average download rate in the last 30 seconds

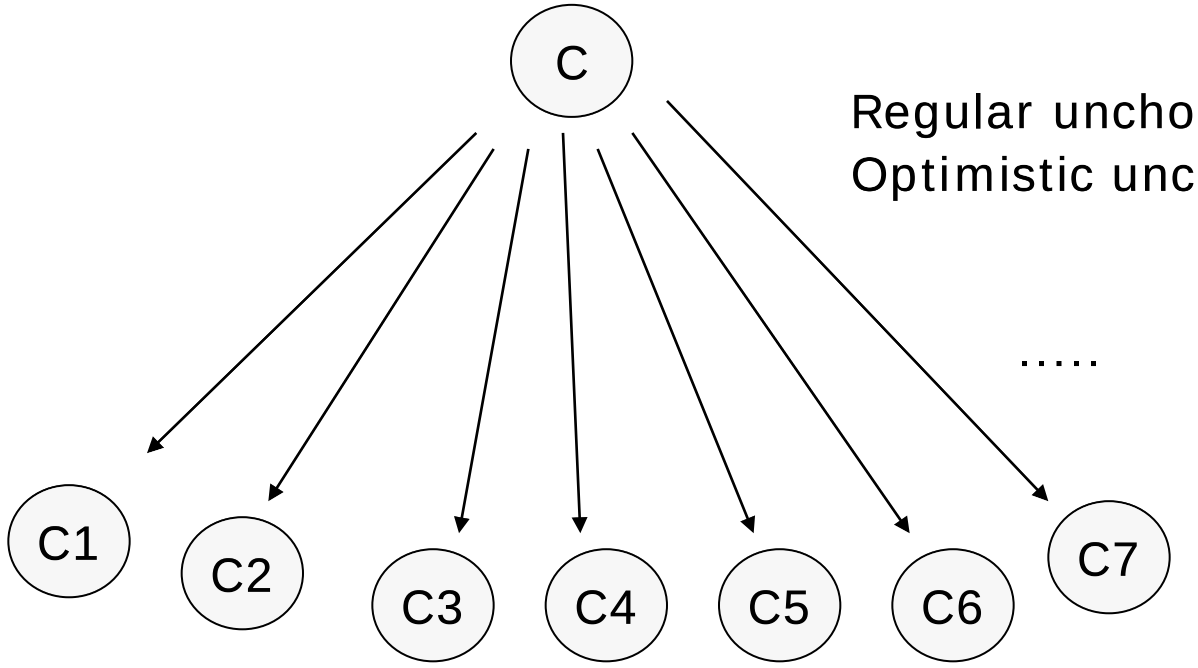
## Upload Policies - Choking Algorithms

Seeder state

Upload policy:

1.The time they were last unchoked (recently first)

2.Upload rate



Regular unchoking : [C1,C2,C3]

Optimistic unchoking : [C5]

Drawback :

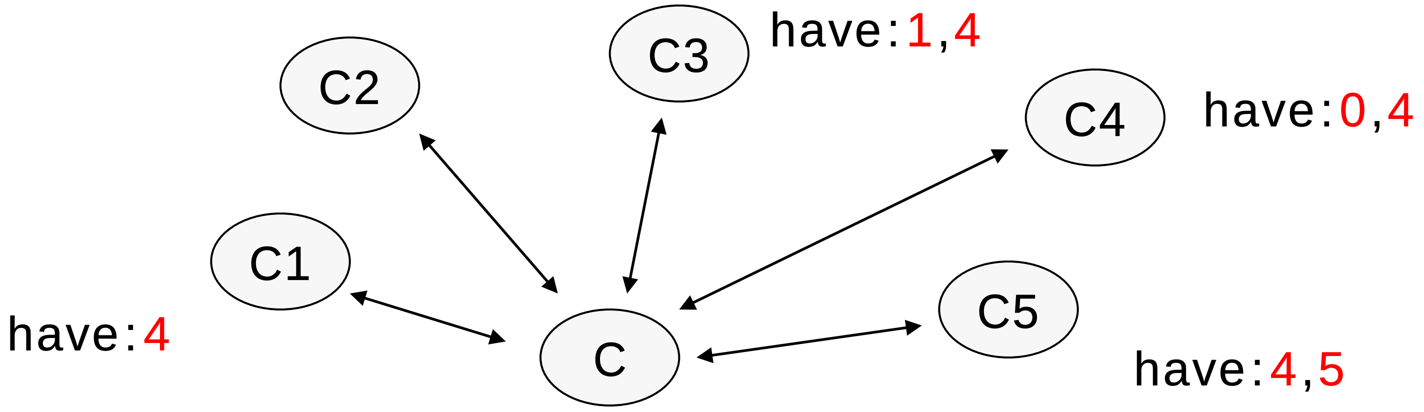
A malicious free-rider can get a high download rate without contributing anything

## Download Policies

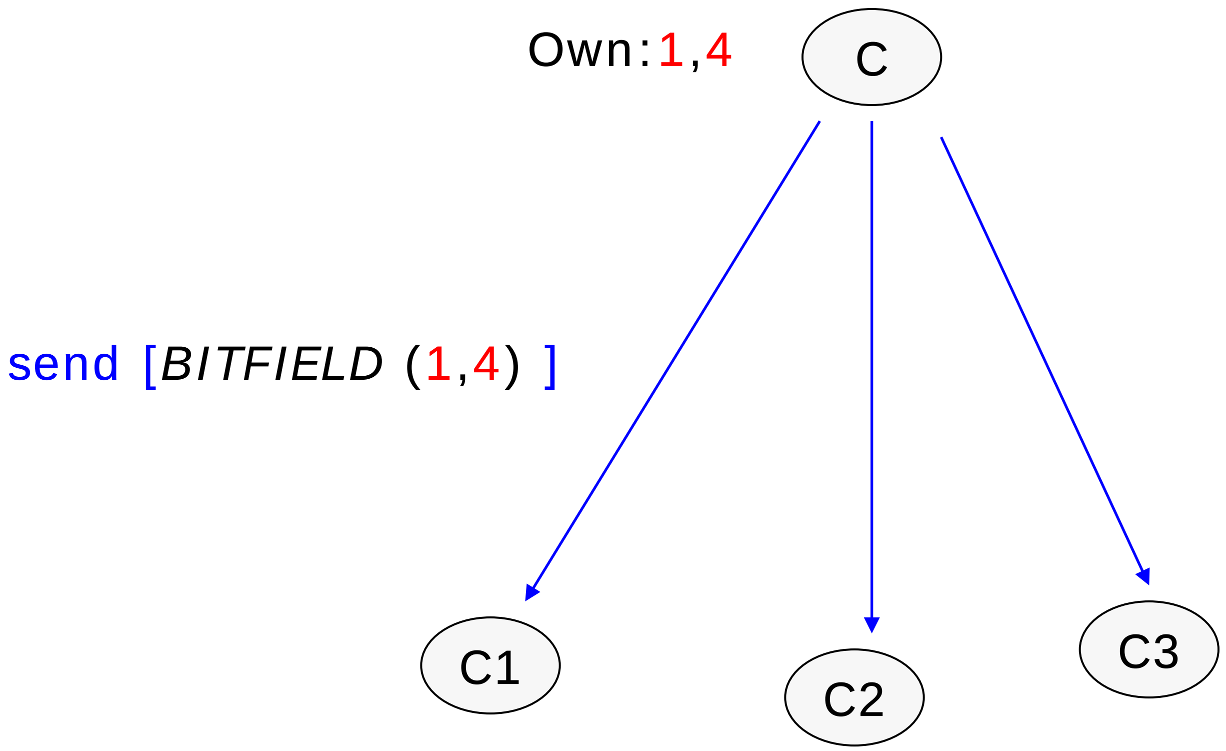
- Piece selection
  - Random first piece
  - Strict priority
  - Rarest First
    - *Have*
  - Endgame Mode

# Download Policies - Rarest First

```
total number :    0          1        2    3     4
interests = [[2, 3], [5, 0, 1], [], [], [4]]
```

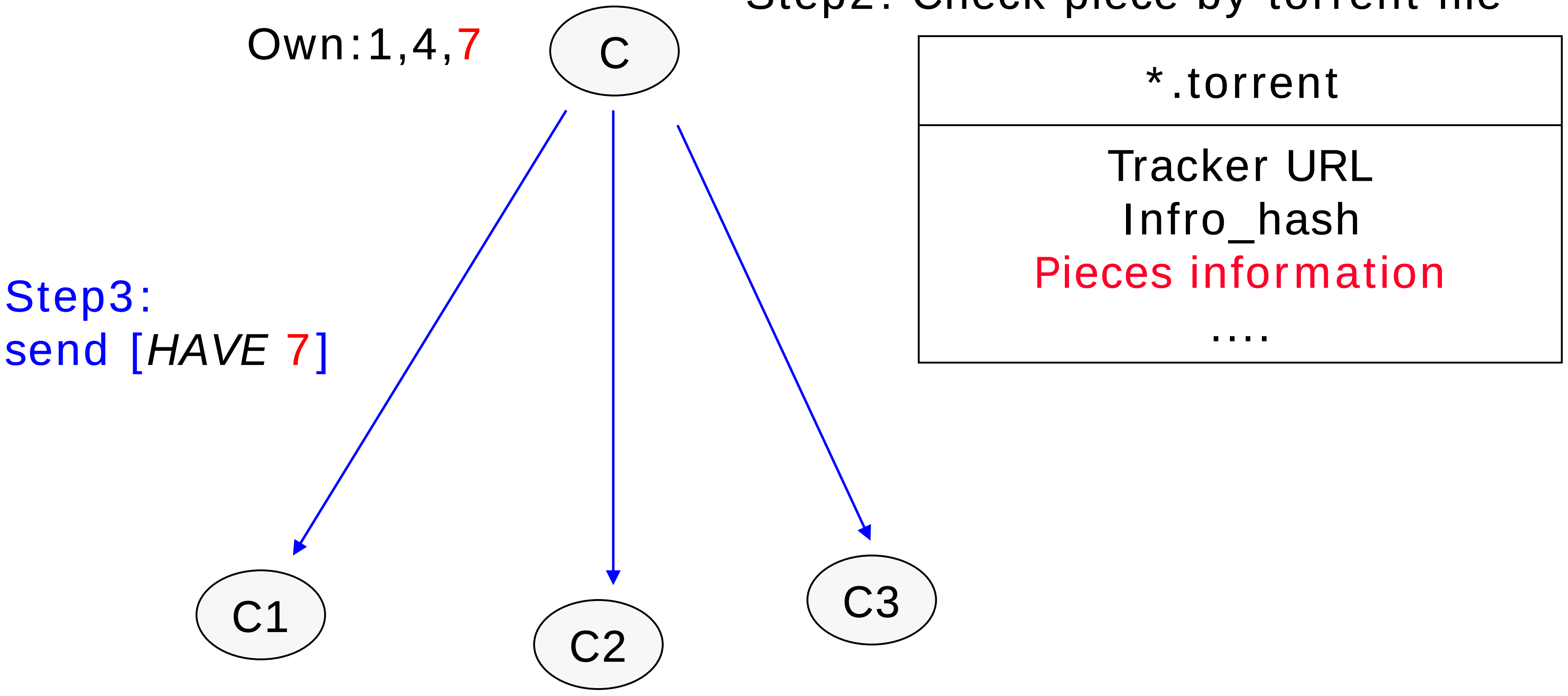


## Implementation - Announce Pieces

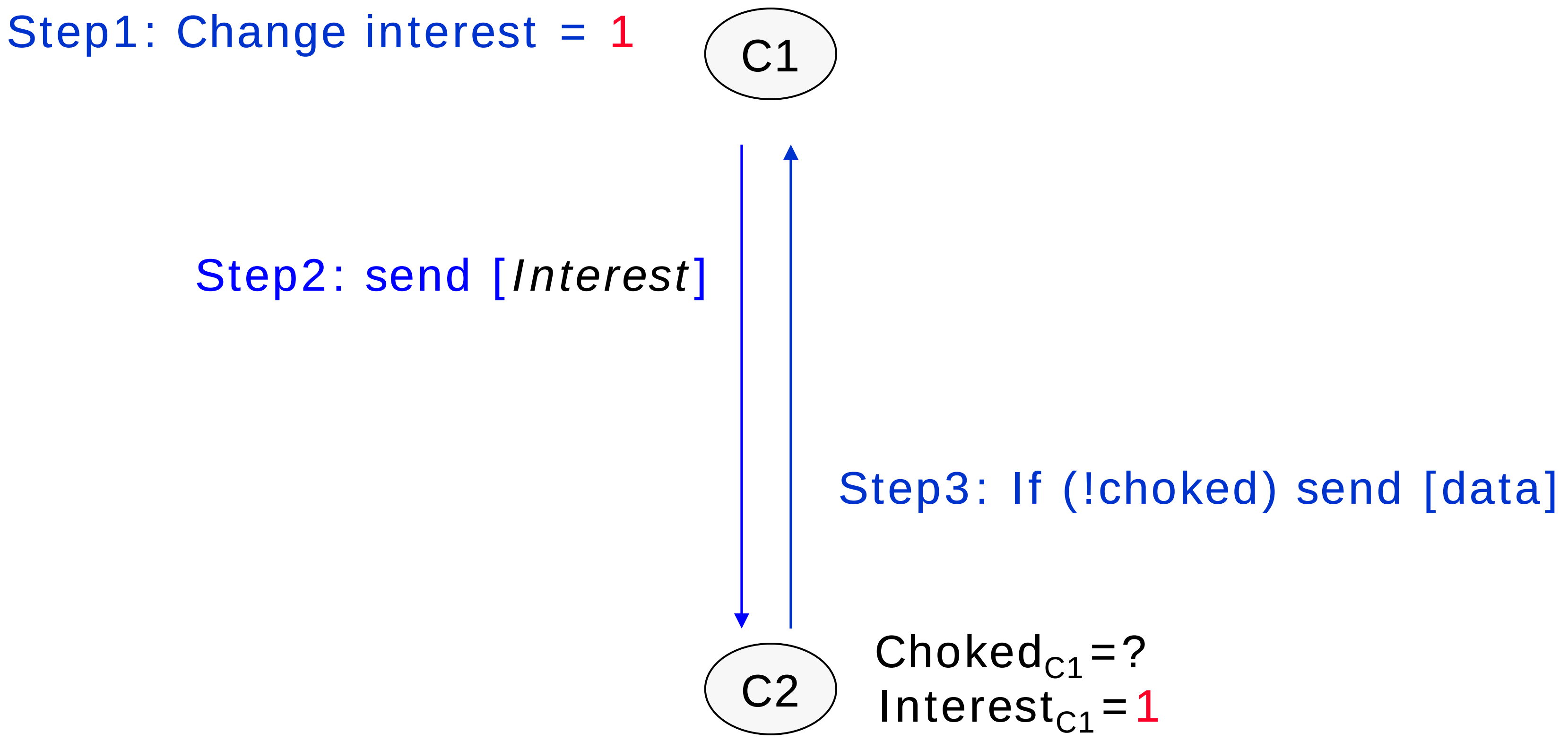


# Implementation - Announce Pieces

Step1: Get a whole piece
Step2: Check piece by torrent file



## Implementation － Interest



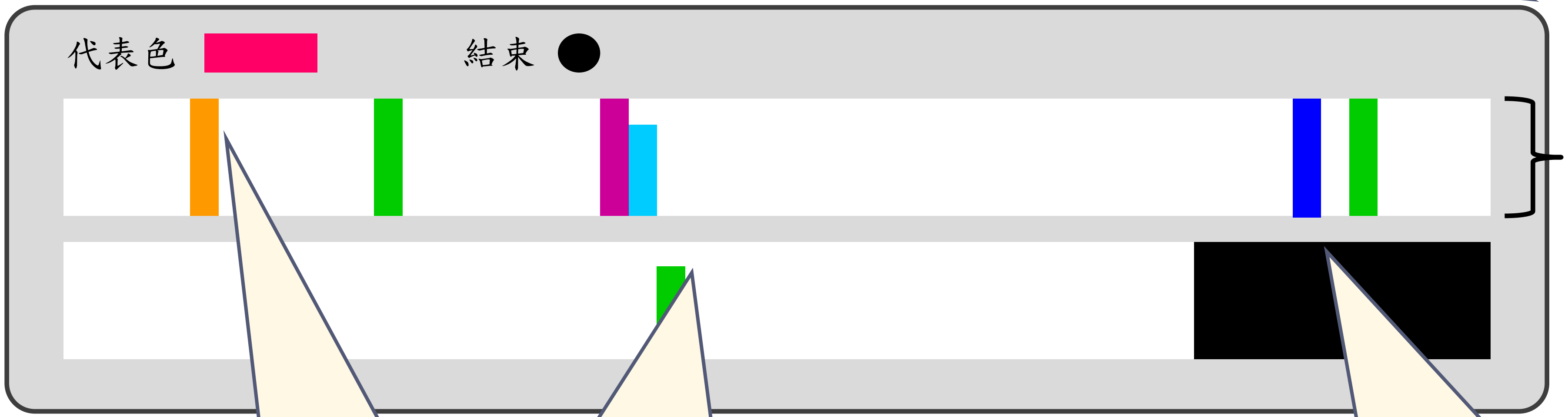
# 實驗說明 1

- 程式畫面說明

此 Client 代表的顏色

下載完成時會變白色

每一行表示 200 個片段 Piece



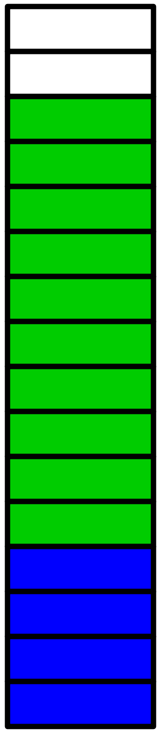
表示一個已完成下載的片段 Piece，來源皆為代表橘色的 Seeder

表示一個未完成下載的片段 Piece，來自兩個不同的 Clients

檔案結尾，表示此展示檔案未滿 200 * 2 個片段，未滿部分以黑色表示

# 實驗說明 2

- 片段 Piece 說明
  - 每個片段大小為 256 KB，分為 16 個 Blocks
  - Block 為真正的傳輸單位
- 以右圖為例 (等同程式畫面)
  - 圖中表示，16 個 Blocks 中：
    - 白色部分，尚未下載到
    - 綠色部分，從代表綠色的 Client 下載
    - 藍色部分，從代表藍色的 Client 下載



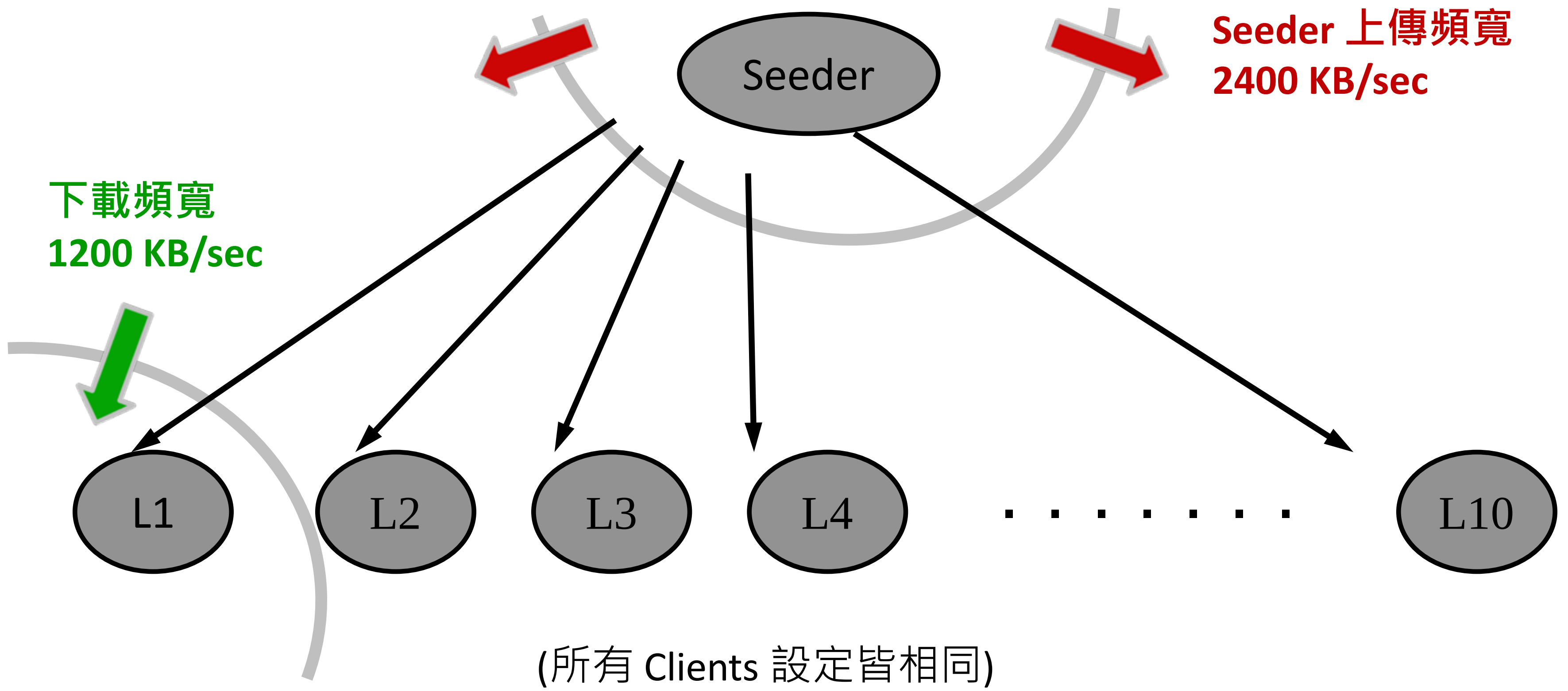
## 實驗內容

- 實驗內容說明
  - 傳輸的 File 大小為 93 MB

  - 觀察 Seeder 與 Clients 之間的傳輸現象，並模擬：
    1. 傳統 Clients / Server 多對一架構
    2. P2P 架構 - 全體互相連線 - 相同頻寬
    3. P2P 架構 - 部分互相連線 - 相同頻寬
    4. P2P 架構 - 全體互相連線 - 不同頻寬

## 實驗一

- 模擬傳統 Client / Server 多對一架構



(所有 Clients 設定皆相同)

# 實驗一

- 數據分析

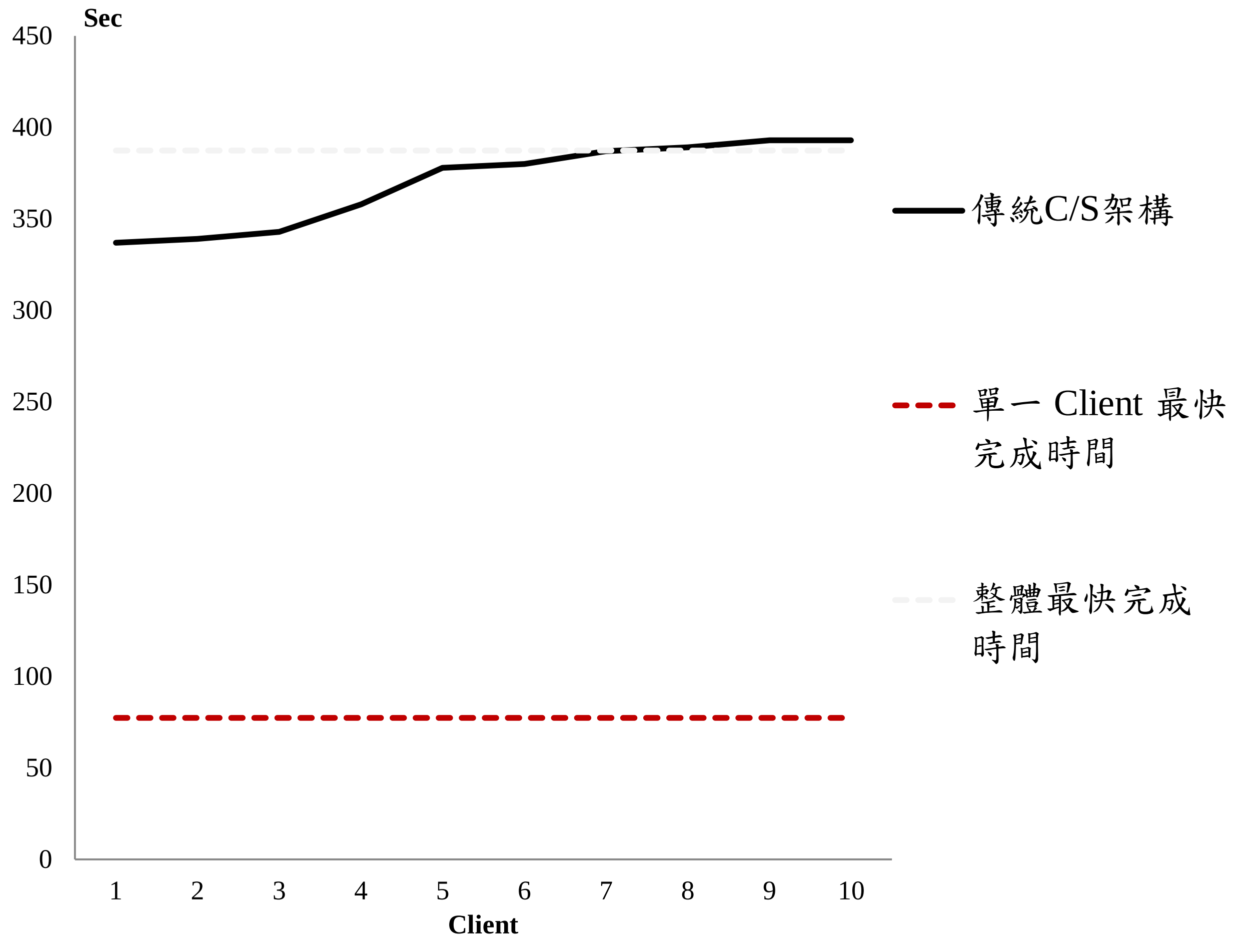
理論值：

- 單一 Client 最快完成時間

$$\frac{93\text{MB}}{1200\text{ KB/sec}} = 77.5\text{ sec}$$

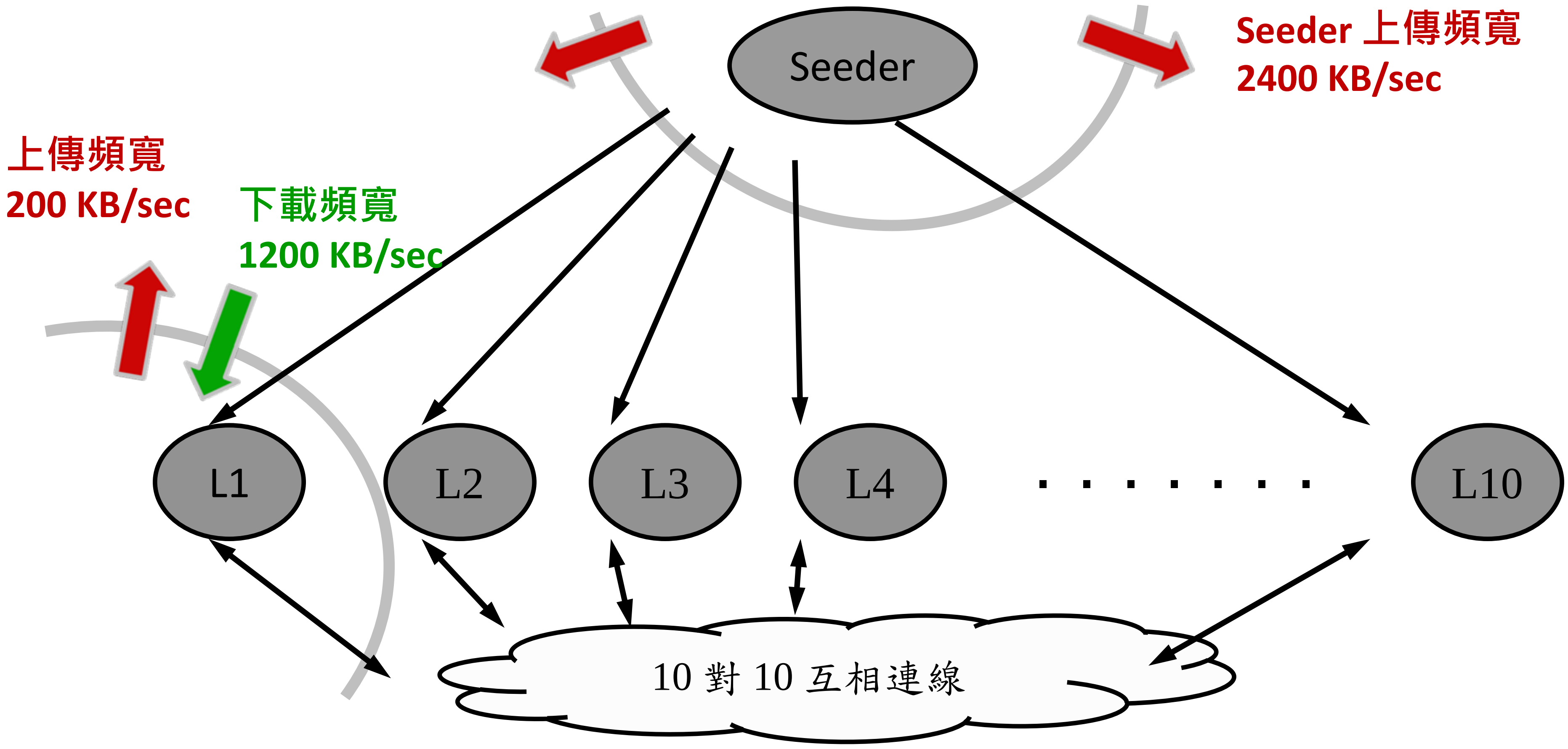
- 整體最快完成時間

$$\frac{93\text{MB} \times 10}{2400\text{ KB/sec}} = 387.5\text{ sec}$$



# 實驗二

- P2P 架構



(所有 Clients 設定皆相同)

# 實驗二

## 數據分析

理論值：

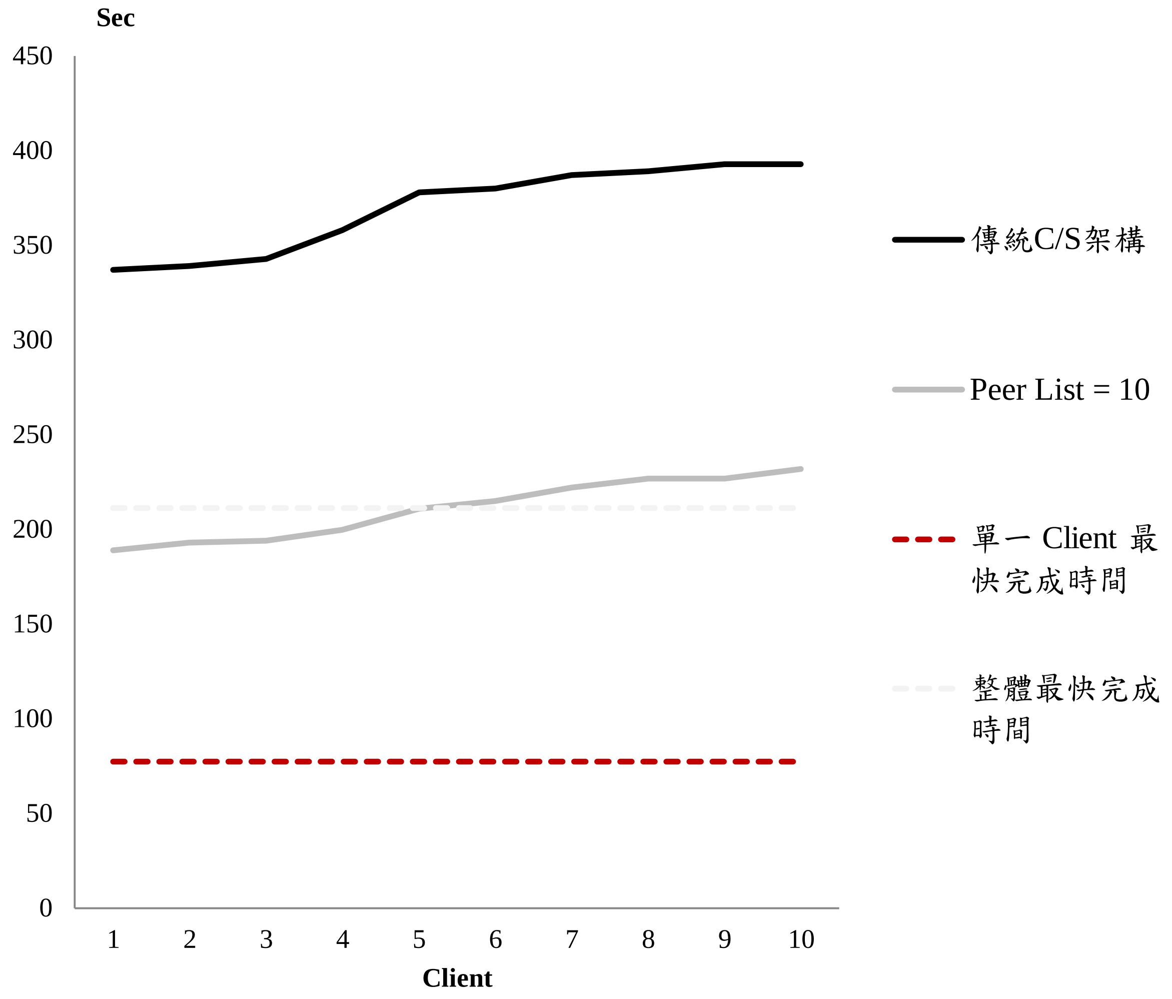
- 單一 Client 最快完成時間

$$\frac{93\text{MB}}{1200\ \text{KB/sec}} = 77.5\ \text{sec}$$

- 整體最快完成時間

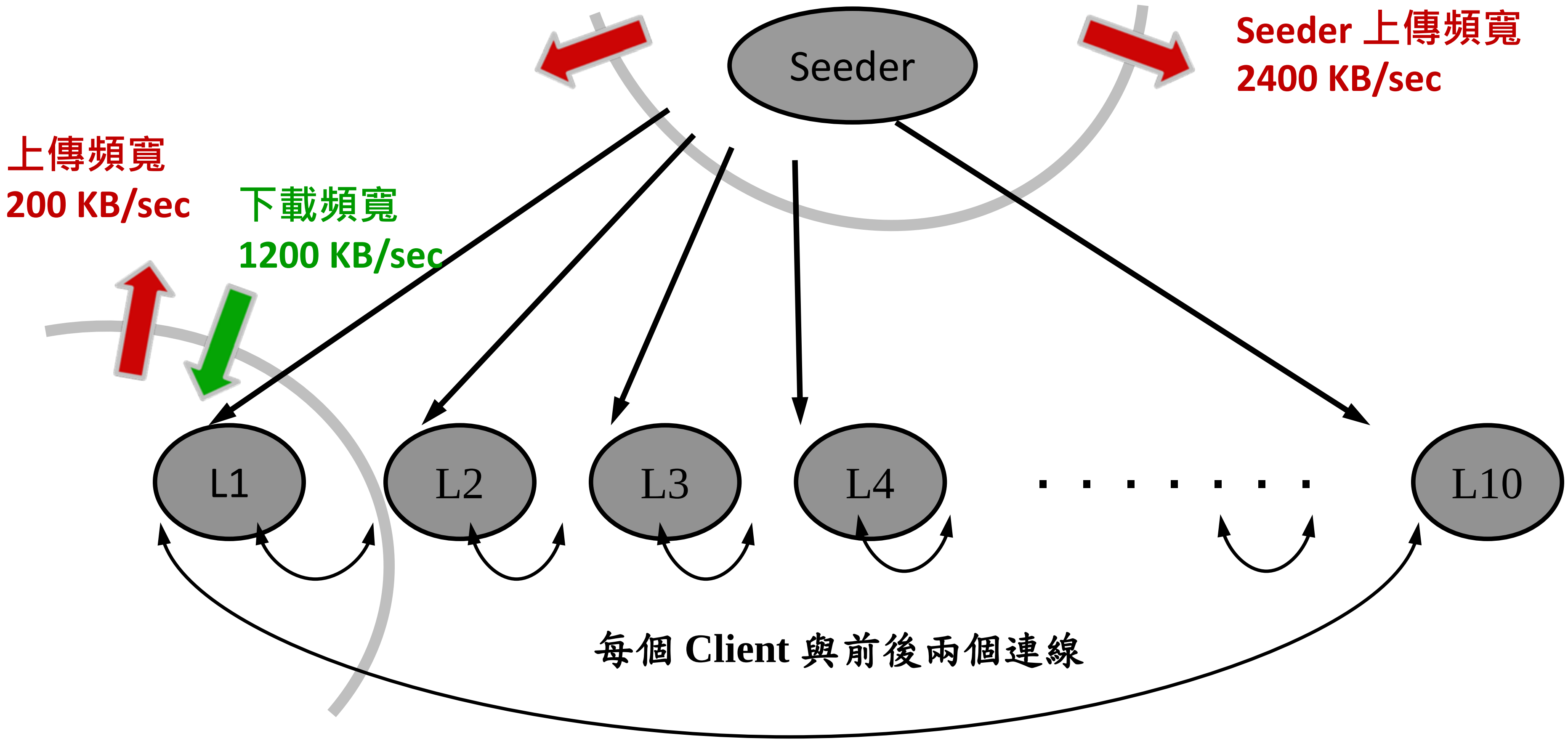
總下載量：
93 MB x 10
總上傳頻寬：
2400 KB/sec + 200 KB/sec x 10

$$\frac{93\text{MB}\ \times\ 10}{2400\ \text{KB/sec} + 200\ \text{KB/sec}\ \times\ 10} = 211.36\ \text{sec}$$



## 實驗三

- P2P 架構



(所有 Clients 設定皆相同)

## 實驗三

- 數據分析

理論值：

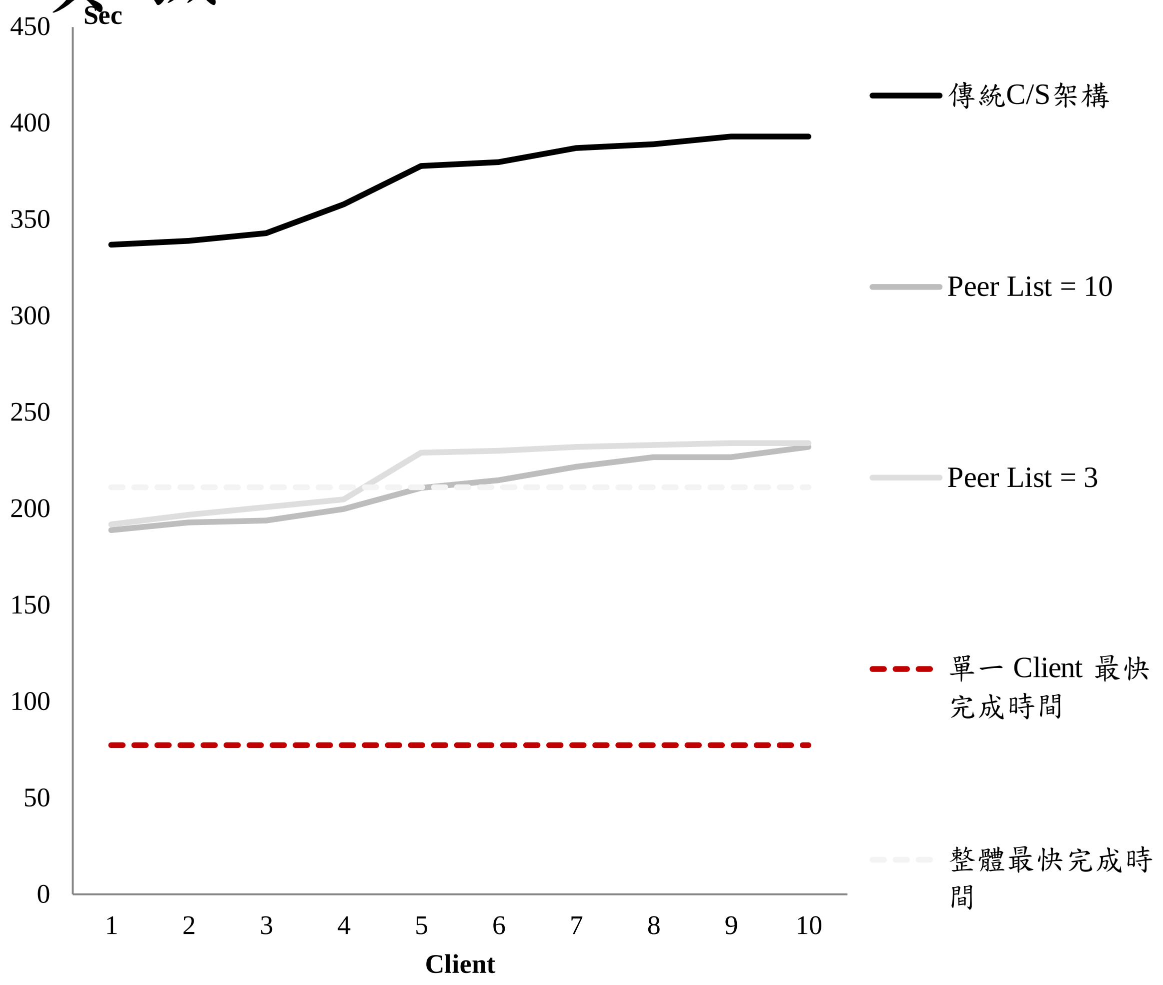
• 單一 Client 最快完成時間

$$\frac{93\text{MB}}{1200\ \text{KB/sec}} = 77.5\ \text{sec}$$

• 整體最快完成時間

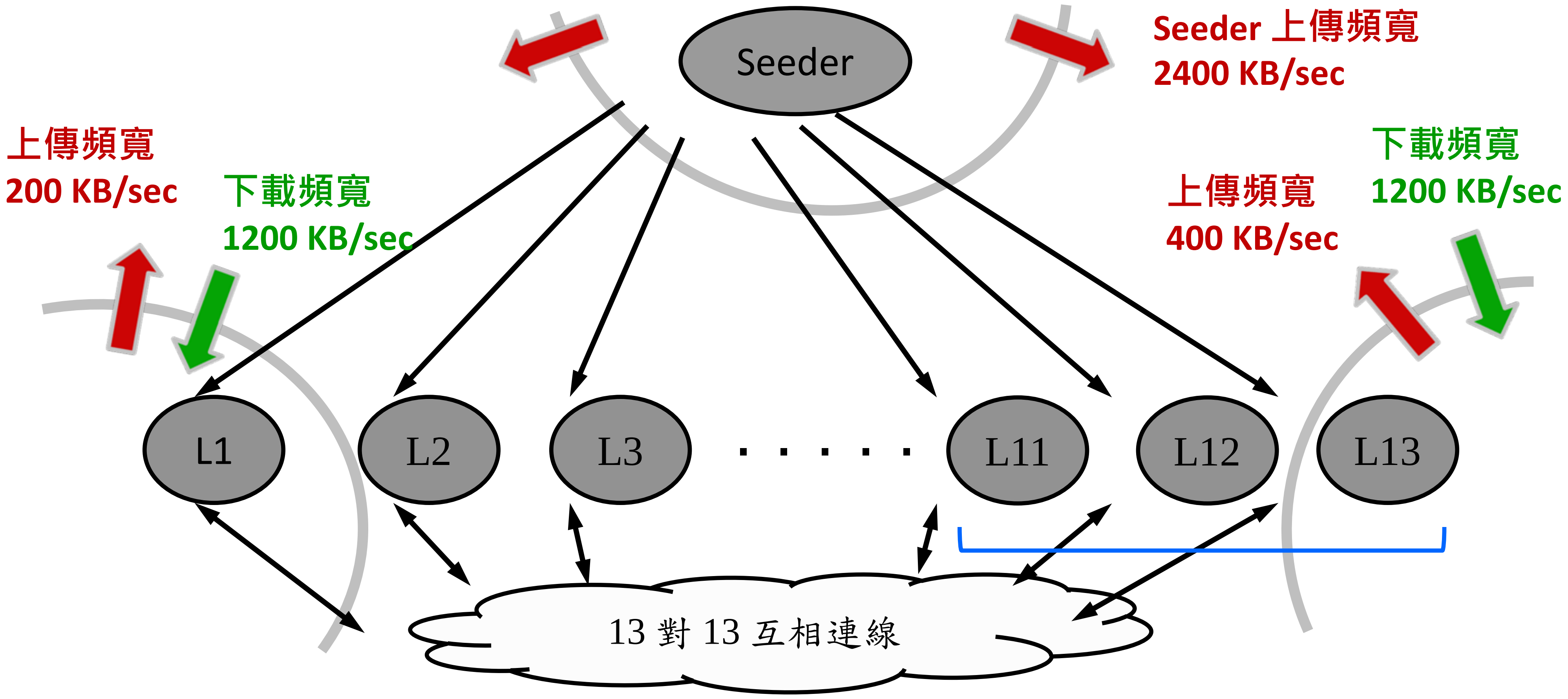
總下載量：
93 MB x 10
總上傳頻寬：
2400 KB/sec + 200 KB/sec x 10

$$\frac{93\text{MB} \times 10}{2400\ \text{KB/sec} + 200\ \text{KB/sec} \times 10} = 211.36\ \text{sec}$$



# 實驗四

- P2P 架構



(Clients 1 到 10 設定相同)

**(Clients 11 到 13 設定相同)**

## 實驗四

- 數據分析

理論值：

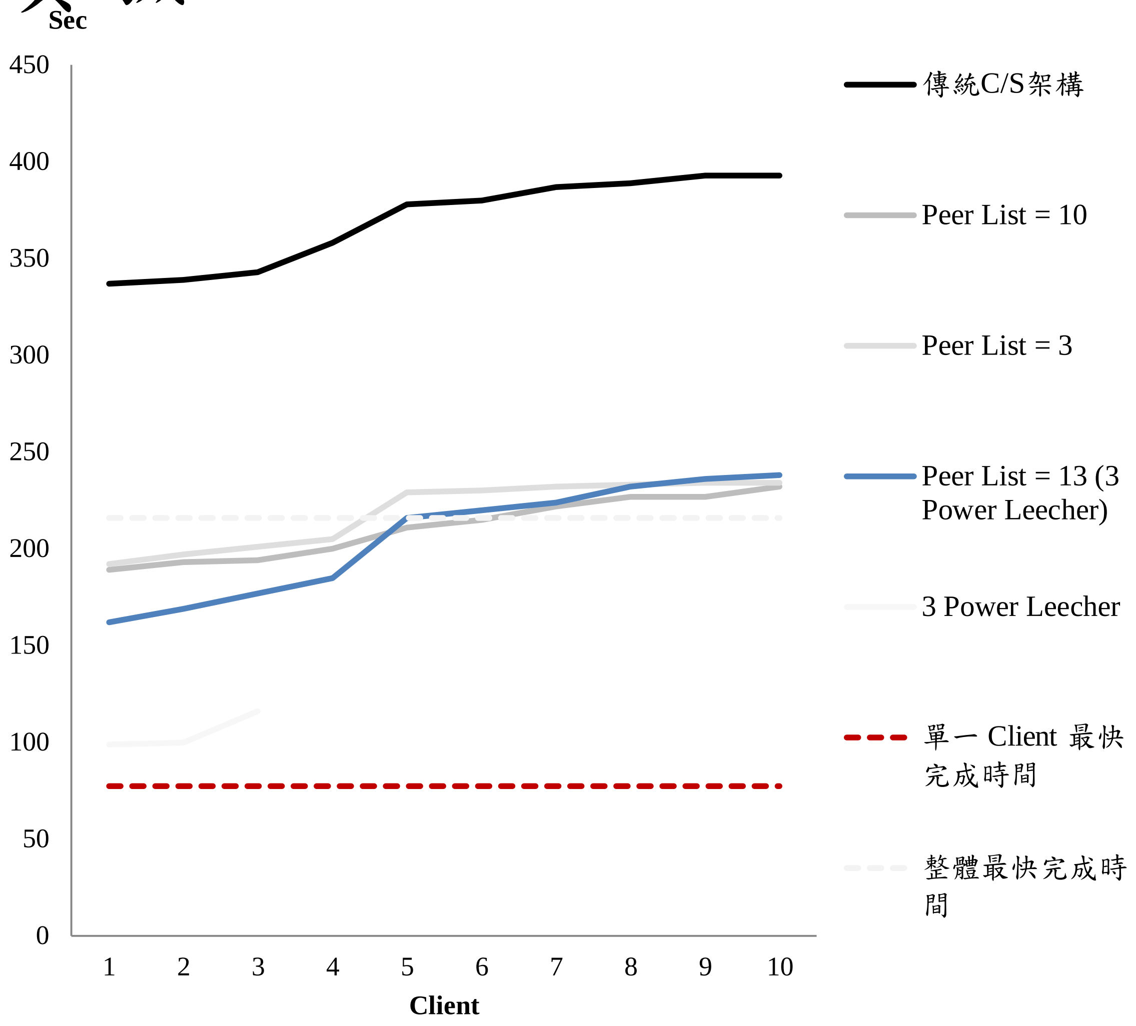
- 單一 Client 最快完成時間

$$\frac{93\text{MB}}{1200\ \text{KB/sec}} = 77.5\ \text{sec}$$

- 整體最快完成時間

$$\frac{93\text{MB} \times 13}{2400\ \text{KB/sec} + 200\ \text{KB/sec} \times 10 + 400\ \text{KB/sec} \times 3}$$

= 215.89


Sec
Client
傳統C/S架構
Peer List = 10
Peer List = 3
Peer List = 13 (3 Power Leecher)
3 Power Leecher
單一 Client 最快完成時間
整體最快完成時間