

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN ĐIỆN TỬ - VIỄN THÔNG



BÁO CÁO BÀI TẬP LỚN

Môn học: Đa Phương Tiện Nâng Cao

**Đề tài: Truyền thông đa phương tiện theo mô hình P2P
Live Streaming**

Giảng viên hướng dẫn: TS. Phạm Văn Tiến

Sinh viên thực hiện: Bùi Văn Bao 20140293

Đinh Tôn Thép 20144242

Ngô Gia Tiến 20144468

Lớp: KSTN – ĐTVT K59

Hà Nội, ngày 23 tháng 11 năm 2018

Mục lục

Mở đầu	4
1. Cấu trúc hệ thống P2P Streaming	5
2. Hoạt động chung của project.....	7
3. P2P protocol	9
4. Cấu trúc project và chức năng của từng phần	12
4.1. Media Server.....	13
4.2 WebServer	13
4.3 Signaling server	17
4.4 Reporter	17
5. Kịch bản mô phỏng	18
6. Triển khai	18
7. Kết quả và nhận xét	23
Kết Luận.....	28
Tài liệu tham khảo.....	29

Danh Mục hình ảnh

Hình 1. Cấu trúc hệ thống P2P Livestreaming.....	5
Hình 2. Hoạt động chung của project.....	7
Hình 3. P2P protocol	9
Hình 4. Cấu trúc project P2P LiveStreaming	12
Hình 5. Cấu trúc file WebServer DemoP2PClappr	13
Hình 6. Cấu trúc P2Pplugin.....	14
Hình 7. p2p_manager.....	15
Hình 8. Peer và các thành phần của Peer	15
Hình 9. Kết quả chạy Peer1	23
Hình 10. Kết quả chạy Peer2	24
Hình 11. Kết quả chạy Peer3	25
Hình 12. Peer gửi các bản tin handshaking.....	26

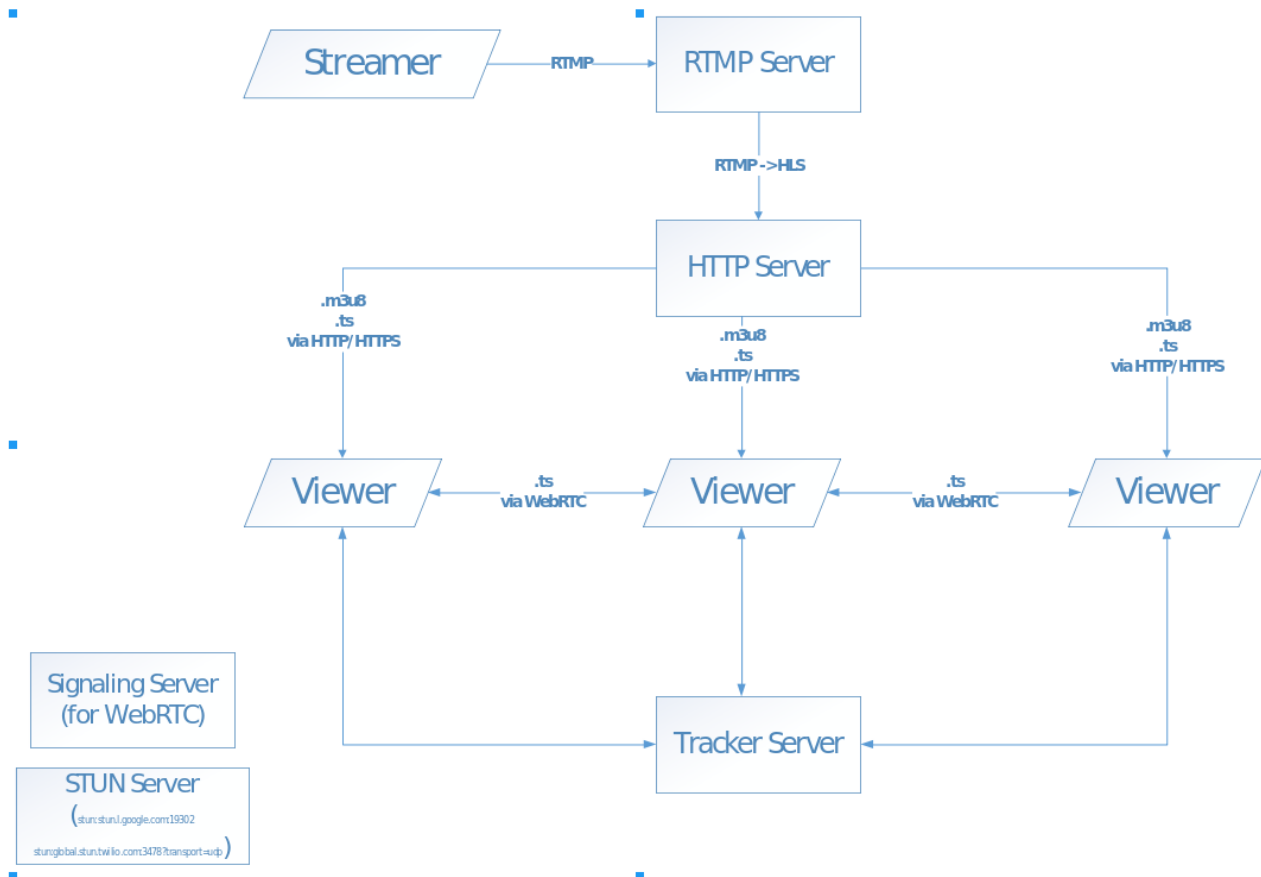
Mở đầu

Trong những ngày đầu phát triển của ứng dụng đa phương tiện, khoảng nửa cuối thập niên 90, việc xem một video trên mạng gần như là điều không thể. Ngày nay, cùng với sự bùng nổ của Internet, các ứng dụng đa phương tiện trong đó có live streaming đã trở thành nhu cầu không thể thiếu của nhiều cư dân mạng. Theo thống kê, riêng tại Mỹ đã có khoảng 33 tỉ video được xem trong tháng 12-2012 (nguồn comScore). Con số trên đủ cho ta thấy được sự lớn mạnh không ngừng của các ứng dụng live streaming.

Hiện nay, trên thế giới đã và đang phát triển rất nhiều phương pháp truyền tin multicast trên tầng ứng dụng khác nhau. Trong đó truyền tin multicast dựa trên giao thức Peer to Peer hứa hẹn có nhiều ưu điểm. Đặc thù của truyền tin multicast là phải tạo được một cây multicast tối ưu, có sự liên kết chặt chẽ giữa các node với nhau, giảm tải được lượng lưu cho phía server, tiếp kiệm chi phí đáng kể.

Mặc dù đã cố gắng nhưng do kiến thức còn hạn chế, thời gian làm bài tập không nhiều, nên bài tập dài của nhóm chúng em không thể tránh khỏi những thiếu sót về nội dung và hình thức. Do đó chúng em rất mong nhận được sự góp ý của thầy để bài tập dài của em được hoàn thiện hơn.

1. Cấu trúc hệ thống P2P Streaming



Hình 1. Cấu trúc hệ thống P2P Livestreaming.

Hệ thống P2P Live Streaming có cấu trúc gồm 4 phần như hình 1: Nguồn nội dung đa phương tiện từ Streamer được truyền tải đến server. Sau đó, server sẽ gửi nội dung tới viewer theo cách mà người dùng có thể sử dụng. Các viewer có thể hợp tác và chia sẻ các phần nội dung trực tiếp giữa họ, giảm tải từ máy chủ.

P2P Live Streaming hoạt động tốt nhất tại đỉnh Viewers, nơi nhiều người xem cùng một nội dung cùng một lúc thông qua RTCDatChannel của WebRTC. Sau đây, ta đi vào chi tiết chức năng của từng phần

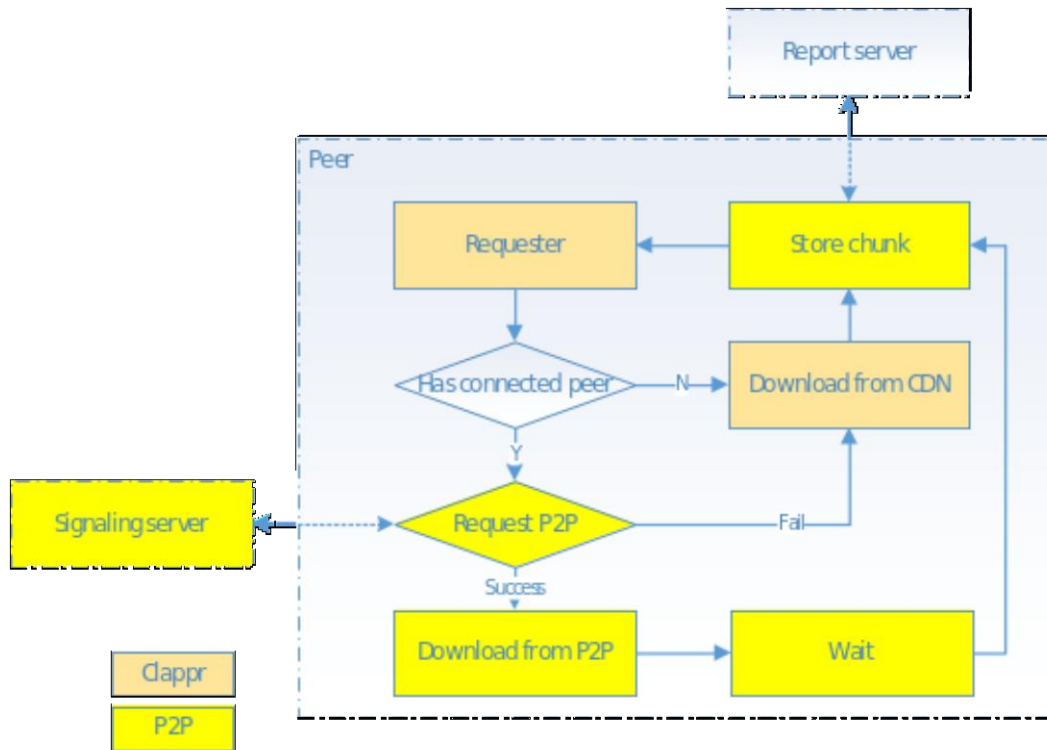
- **Media Server:** đóng vai trò của CDN server, chứa những bản sao về nội dung được từ nguồn streamer truyền qua. Tại đây nội dung ban đầu của stream được khởi tạo bằng chuyển đổi RTMP

stream thành HLS. HLS phá vỡ luồng stream thành chuỗi file HTTP-base file *.ts (transport stream)*. Sau đó transport-stream (.ts) files được đánh index và lưu danh sách trong file *.m3u8*. File .m3u8 được client yêu cầu download đầu tiên để có thể phát live stream.

- **Client:** bao gồm views hay peers. Các views có 2 cách để có được nội dung (file .ts hay còn gọi là *trunk*) của stream: download trực tiếp trunk từ media server thông qua giao thức HTTP/HTTPS, hoặc download từ views nào báo hiệu có trunk đó đầu tiên qua giao thức P2P. Peers trao đổi trunk thông qua giao thức *RTCDataChannel* của WebRTC. Trước khi 2 peer có thể trao đổi dữ liệu dữ liệu cần phải thực hiện thủ tục "signaling" thông qua Signaling server.
- **Tracker:** Mỗi peer được quản lý bởi 1 tracker bao gồm 2 thành phần: *Swarm* và *P2PManager*. Tracker của mỗi peer quản lý số peer partner, kiểm tra kết nối P2P để quyết định sẽ download trunk từ media server hay từ P2P.
- **Signaling Server:** cầu nối trao đổi ban đầu trước khi trao đổi dữ liệu thông qua p2p được thành lập.

Nơi chuyển tiếp bản tin SDP và ICE: cho biết thông tin về dữ liệu, cách thức mã hóa (bản tin SDP), thông tin để tìm cách vượt qua NAT và FireWall (ICE) để các peer có thể kết nối với nhau.

2. Hoạt động chung của project



Hình 2. Hoạt động chung của project

Requester: Xác định URL của chunk cần download (HLSJS)

Khi xác định được URL của chunk, sẽ khởi tạo P2P Manager, P2P loader (custom loader config cho HLSJS) và gọi phương thức *load*.

function load (context, config, p2pManager, storage, controlTimeout, callback)

Mô tả:

Nếu không có kết nối P2P thì sử dụng HTTPRequester để download chunk từ CDN (1). Nếu có kết nối P2P thì sử dụng P2P protocol để download chunk từ peer khác (2).

Input:

Context: Thông tin URL của chunk

Config: Thông tin timeout cho download CDN

P2P manager: p2p protocol cho trường hợp download P2P, chức năng khởi tạo swarm cho peer, thêm hoặc xóa peer partner

Storage: Lưu trữ chunk sau khi Peer download thành công. Mỗi peer có 1 storage riêng và được quản lý bản swarm của peer đó.

Swarm: mỗi peer sẽ có 1 swarm, chức năng quản lý các peer partner, gửi bản tin interest and request tới các peer partner, nhận và xử lý các bản tin từ peer partner

Control timeout: Update thông tin thời gian download để cập nhật timeout

Callbacks: callback cho HLS khi download thành công

(1) Download từ CDN

Sử dụng http request để yêu cầu chunk từ CDN

P2Pplugin/src/p2p-loader.js: *loadInternal()*

Nhận chunk từ CDN bằng handle, sau đó callback success cho HLS

P2Pplugin/src/p2p-loader.js: *readystatechange()*

(2) Download từ P2P

Sử dụng p2p manager request resource để yêu cầu từ P2P

Callback success nếu download thành công từ P2P:

P2Pplugin/src/p2p-loader.js: *receiveP2P(chunk, method, resource, sender)*

Trả về, nội dung chunk, URL và id của peer đã gửi chunk đó

Sau khi download thành công từ *sender*, hệ thống đợi một khoảng thời gian để tổng thời gian download bằng thời gian video trong 1 chunk:

$$waitTime = 0.95 * chunkTime - downloadP2PTime$$

Callback fail nếu download thất bại

P2Pplugin/src/p2p-loader.js: *receiveCDN(resource, method)*

Trả về URL và nguyên nhân bị fail (interest timeout/request timeout)

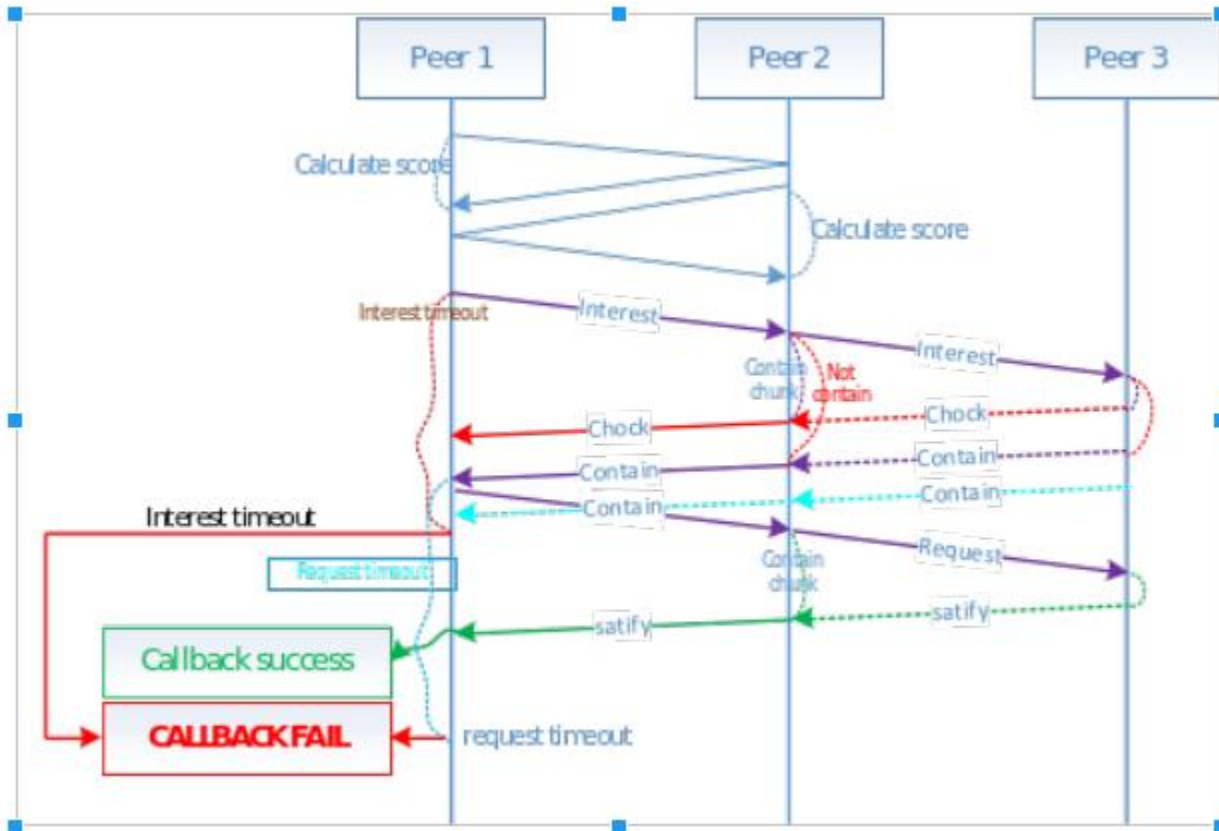
Download chunk đó từ CDN (1).

Sau khi đã download thành công (từ peer hoặc CDN), chunk đó được lưu lại để cung cấp cho các peer khác nếu được yêu cầu. Storage lưu tối đa 5 chunks.

P2Pplugin/src/js/storage.js: *setItem(url, value)*

Tiếp tục xác định URL của chunk tiếp theo.

3. P2P protocol



Hình 3. P2P protocol

Ping/pong sau khi kết nối giữa 2 peers được khởi tạo để tính score:

Gửi bản tin ping gồm 2048 ký tự “x”, nhận pong sau t ms.

$$Score = 100 - \lfloor \frac{t}{100} \rfloor$$

Score dùng để đánh giá tốc độ kết nối giữa các peers, score càng cao thì thời gian kết nối và truyền dữ liệu ngắn.

P2Pplugin/src/js/peer.js: *sendPing()*

P2Pplugin/src/js/peer.js: *sendPong()*

P2Pplugin/src/js/peer.js: *receivePong()*

Peer gửi yêu cầu

Gửi interest: P2Pplugin/src/js/swarm.js: *sendInterest(resource,callbacks)*

- SwarmUtils sẽ chọn ra tối đa 10 peers có score cao nhất để gửi interest

P2Pplugin/src/js/swarm_utils.js: get contributors()

Timeout cho interest là 100ms

- Sau khi nhận được contain từ partner peer, gửi request tới peer đó ngay để yêu cầu dữ liệu của chunk cần down. Hệ thống chỉ nhận 2 contains

P2Pplugin/src/js/swarm.js: containReceive(peer, resource): nhận contain từ peer

P2Pplugin/src/js/swarm.js: sendRequest(peerIden): gửi interest tới peer vừa gửi contain

- Do gửi tới đa 2 tín hiệu request, sẽ nhận về nhiều nhất là 2 dữ liệu của cùng 1 chunk. Dữ liệu của chunk nào được truyền thành công trước sẽ lấy chunk đó và loại bỏ dữ liệu chunk nhận sau

Request timeout = chunkTime – previousDownloadTime

- P2Pplugin/src/js/swarm.js: satisfyReceived(peer, chunk, resource)

Peer trả về dữ liệu chunk

Khi nhận được tín hiệu interest, peer kiểm tra trong storage và trả lời: (*storage:contain()*)

- Nếu có chunk: gửi tín hiệu ***contain***
- Nếu không có: gửi tín hiệu ***chock***

P2Pplugin/src/js/peer.js: *interestReceived(resource)*

Khi nhận tín hiệu request, peer lấy dữ liệu từ storage và gửi lại cho peer

P2Pplugin/src/js/peer.js: *sendSatisfy(resource)*

Gửi tín hiệu *request*

Timeout cho quá trình request được tính:

$$timeout = \max(0.3 * chunkTime; chunkTime - downloadPrevChunkTime)$$

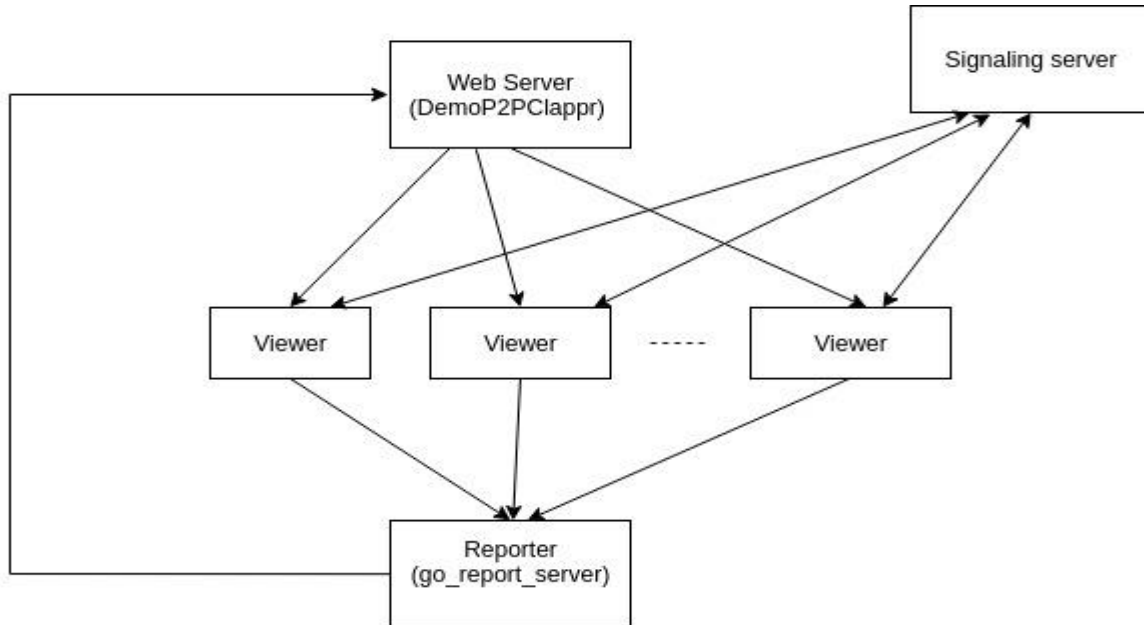
Tín hiệu *request* gửi cho peer xác nhận đầu tiên có chứa chunk cần download. Peer sẽ gửi lại dữ liệu chunk ở dạng Base64, nếu tại thời điểm nhận *request* peer không còn chứa dữ liệu chunk cần download thì sẽ gửi tín hiệu *chock*.

Nếu trong thời gian timeout, download thành công, gọi callback success, trả về nội dung chunk, URL chunk và peerID của người gửi.

Nếu hết thời gian timeout mà vẫn chưa download được thành công, gọi callback fail, trả về URL chunk và nguyên nhân fail (timeout interest hoặc timeout request)

4. Cấu trúc project và chức năng của từng phần

Trong phần này, ta xem xét cấu trúc chi tiết và làm rõ chức năng của từng phần của project P2P LiveStreaming. (Hình 4)



Hình 4. Cấu trúc project P2P LiveStreaming

Hoạt động chung: sao khi web server được khởi động tại path X, client truy nhập vào X để tải nội dung web về và trở thành các viewer (Peer). Tại giao diện web-client, ta lựa chọn server CDN để bắt đầu streaming. Viewer (peer) tùy vào tình huống sẽ download trunks từ CDN^[*] hoặc từ viewer khác^[**].

[**] Viewer sử dụng p2p để download trunks cần thực hiện thủ tục signalling thông qua signaling server. Sau mỗi chu kỳ T, các viewer báo cáo số liệu đã download về cho Reporter. Sau đó, Reporter phân tích và tổng hợp, rồi gửi để web server để hiển thị. Sau đây, ta xem xét chi tiết chức năng các phần.

4.1. Media Server

Chúng em sử dụng 2 server CDN public của streamroot và peer5 có địa chỉ tại :

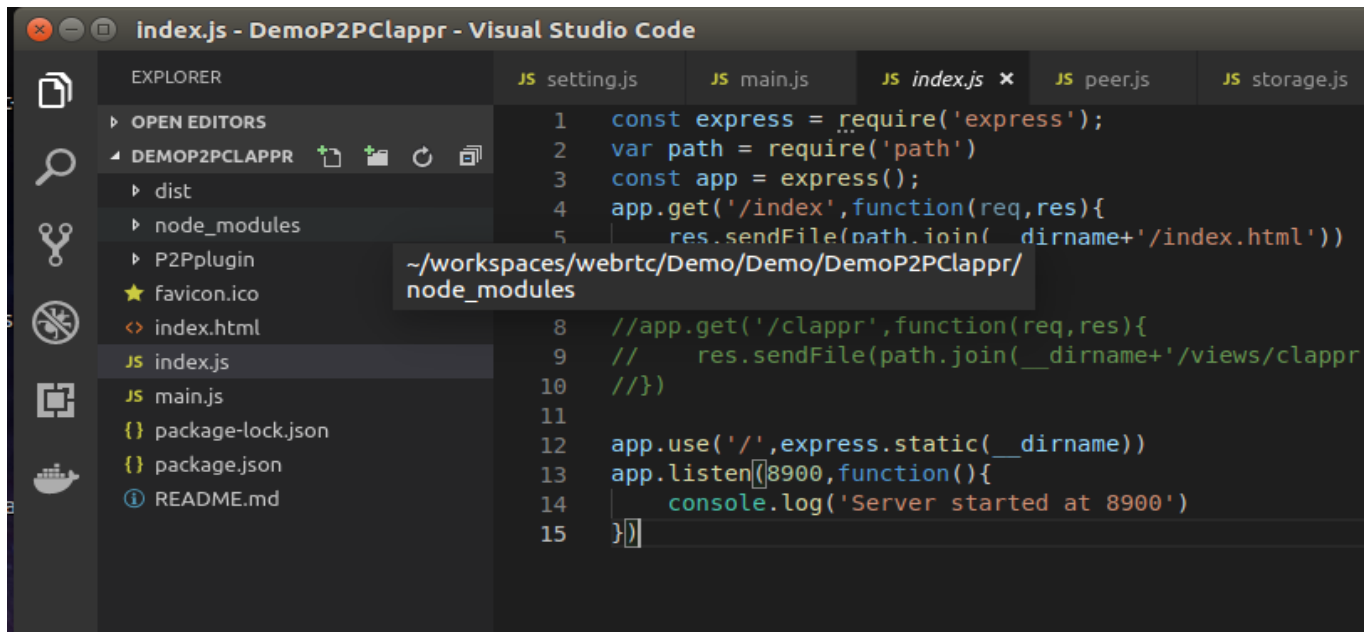
- <https://demo-live.streamroot.io/index.m3u8> ^[1]
- https://wowza.peer5.com/live/smil:bbb_abr.smil/playlist.m3u8 ^[2]

Việc lựa chọn server CDN được quy định trong file index.html

4.2 WebServer

WebServer được customize từ open source *BemTV Plugin for Clappr Media Player* ^[3].

WebServer được viết bằng ngôn ngữ nodejs và html. Cấu trúc file WebServer được thể hiện trong hình 5.



Hình 5. Cấu trúc file WebServer DemoP2PCLappr

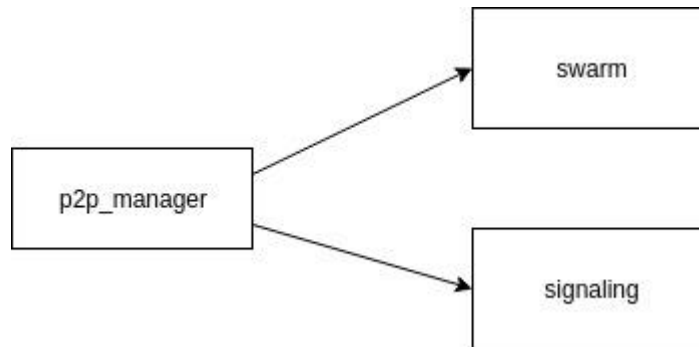
- **index.js**: khởi tạo web server.
- **index.html**: chứa thiết kế giao diện web.
- **main.js**: file script xử lý các event của web, được import trong file index.html
- Thư mục P2Pplugin: thành phần quan trọng nhất, chứa bản thiết kế plugin, đại diện cho mỗi Peer, được import trong file index.html
- **Thư mục dist**:
 - + p2p_plugin.js: được build từ thư mục P2Pplugin bằng webpack

- + clappr.min.js: plugin adds peer-to-peer powers for HTTP Live Streaming (HLS) ^[4], được import trong file index.html
- + level-selector.js: Clappr Level Selector Plugin ^[5]
- **Thư mục node_modules:** nơi chứa dependencies được liệt kê trong file package.json
- Các thành phần và chức năng P2Pplugin:

P2Pplugin	253
dist	254
node_modules	255
src	256
js	257
adaptiveTimeout.js	258
base64-binary.js	259
md5.js	260
p2p_manager.js	261
peer.js	262
storage.js	263
swarm_utils.js	264
swarm.js	265
rtc-bufferedchannel	266
p2p-loader.js	267
setting.js	268
index.js	269
package-lock.json	270
package.json	271
webpack.config.js	272

Hình 6. Cấu trúc P2Pplugin

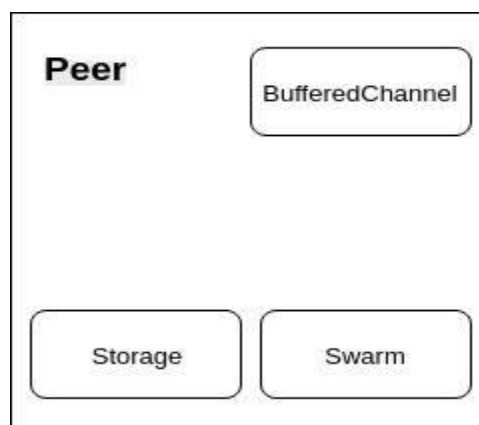
- **p2p_manager.js:**



Hình 7. p2p_manager

- + QuickConnect(tracker, this.connectConfig): kết nối tới signalling
- + connection.createDataChannel('test'): tạo channel test
- + setListener(): lắng nghe sự kiện channel test opened, channel test closed.
- + new Swarm(): Tạo swarm
- + addPeer, removePeer: Thêm hoặc loại bỏ peer khỏi swarm của peer mà nó quản lý.
- + requestResource: Gửi yêu cầu trunk cho p2p bằng cách gửi bản tin “interest” tới tất cả các peer partner trong swarm

- **peer.js**



Hình 8. Peer và các thành phần của Peer

- + Kết nối với signalling tại channel “test”

- + Sử dụng BufferedChannel để gửi và nhận các bản tin handshaking, trunk data.
- + Xử lý các bản tin handshaking (hình 8).

```

46
47     messageReceived(data){
48         var [command,resource,content] = data.split('$')
49         switch(command){
50             case 'interest':
51                 this.interestReceived(resource)
52                 break
53             case 'ping':
54                 this.sendPong()
55                 break
56             case 'pong':
57                 this.pongReceived()
58             case 'choke':
59                 //console.log('chock from ',this,":",resource)
60                 this.swarm.chokeReceived(resource)
61                 break
62             case 'contain':
63                 //console.log('contain from ',this,":",resource)
64                 this.swarm.containReceived(this, resource)
65                 break
66             case 'request':
67                 this.sendSatisfy(resource)
68                 break
69             case 'satisfy':
70                 //console.log('check md5')
71                 if(content.length > 0){
72                     //console.log('md5 success')
73                     this.swarm.satisfyReceived(this, resource, content)
74                 }
75                 break
76         }
77     }
78

```

Hình 8. Peer xử lý các bản tin handshaking.

- **storage.js:**

- Lưu trữ trunk (URL và trunks)
- Lưu trữ số CDN trunk, P2P trunk và dung lượng của chúng (numP2P, numCDN, sizeP2P, sizeCDN).
- Lưu trữ số trunk upload thành công (numUpload, sizeUpload), số lần download fail từ CDN và P2P (cdn_fail, requestfail)

- **swarm.js:**

- Mỗi peer sẽ storage riêng và được quản lý bởi 1 swarm (hình 6).
- Xử lý các tín hiệu từ các peer partner
- Gửi bản tin “interest” và “request”

- **p2p_loader.js**

- Check p2p connection, nếu có connection thỏa mãn thì quyết định download trunk từ P2P, ngược lại download trunk từ CDN.

- **rtc-bufferedchannel** ^[6]: là open source được thiết kế theo chuẩn RTCDataChannel của WebRTC, peer sử dụng bufferedchannel để truyền và gửi dữ liệu.

- **setting.js**: chứa file config của P2Pplugin

4.3 Signaling server

- Folder: signaling
- Sử dụng module rtc-swiffborad ^[7] để tạo signaling server. Nó sử dụng websocket để giao tiếp với các signaling client.

4.4 Reporter

- Folder: go_report_server
- Nhận dữ liệu từ các peer, tổng hợp vào thống kê số lượng, phần trăm trunk download từ CDN và P2P rồi gửi đến WebServer để hiển thị.
- Ngôn ngữ: golang.

5. Kịch bản mô phỏng

Environment

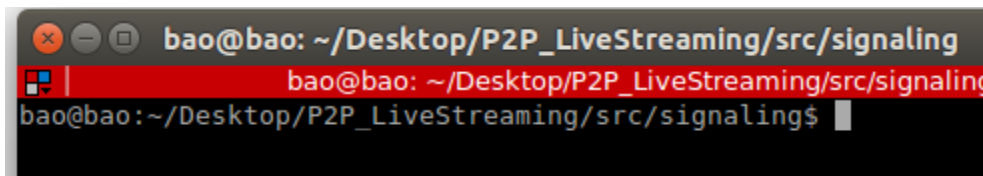
- OS: Ubuntu 16.04 LTS
- Nodejs: version 10.4.1
- Npm: 6.3.0
- Golang
- Browser: google chrome

Kịch bản: mô phỏng P2P giữa 3 peer sử dụng 3 máy vi tính gồm 2 laptop và 1 pc. Trong đó, chúng em sử dụng máy pc làm server và client. Client sẽ truy cập đến server qua IP private.

6. Triển khai

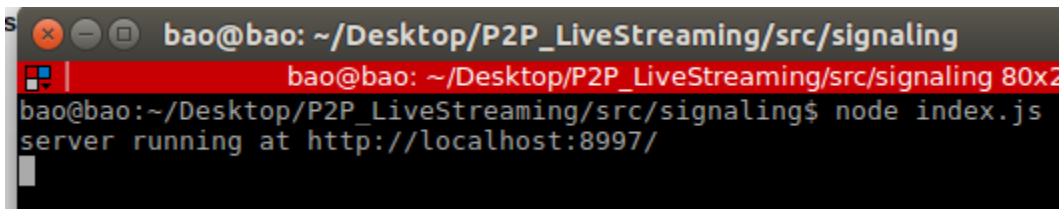
- **B1:** Khởi động signaling server

Sử dụng terminal truy cập đến thư mục: P2P_LiveStreaming/src/signaling



```
bao@bao: ~/Desktop/P2P_LiveStreaming/src/signaling
bao@bao: ~/Desktop/P2P_LiveStreaming/src/signaling$
```

Chạy signaling server: node indes.js



```
bao@bao: ~/Desktop/P2P_LiveStreaming/src/signaling
bao@bao:~/Desktop/P2P_LiveStreaming/src/signaling$ node index.js
server running at http://localhost:8997/
```

- **B2:** Khởi Reporter server

Sử dụng terminal khác truy cập đến thư mục P2P_LiveStreaming/src/go_report_server/src

Chạy reporter server: ./src

```
bao@bao: ~/Desktop/P2P_LiveStreaming/src/go_report_server/src
bao@bao: ~/Desktop/P2P_LiveStreaming/src/go_report_server/src 80x24
bao@bao:~$ cd ~/Desktop/P2P_LiveStreaming/src/go_report_server/src/
bao@bao:~/Desktop/P2P_LiveStreaming/src/go_report_server/src$ ./src
0x658e10
```

- **B3: Khởi WebServer**

Sử dụng terminal khác truy cập đến thư mục P2P_LiveStreaming/src/DemoP2PClappr

Lấy địa chỉ IP Private để config cho WebServer: ifconfig

```
bao@bao: ~/Desktop/P2P_LiveStreaming/src/DemoP2PClappr
bao@bao: ~/Desktop/P2P_LiveStreaming/src/DemoP2PClappr 80x24
bao@bao:~/Desktop/P2P_LiveStreaming/src/DemoP2PClappr$ ifconfig
eno1      Link encap:Ethernet  HWaddr e0:d5:5e:8c:03:07
          inet addr:192.168.6.109  Bcast:192.168.6.255  Mask:255.255.255.0
          inet6 addr: fe80::daf8:8262:1970:c8be/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:160365 errors:0 dropped:0 overruns:0 frame:0
          TX packets:119886 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:122182763 (122.1 MB)  TX bytes:34895960 (34.8 MB)
          Interrupt:16 Memory:51200000-51220000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:24694 errors:0 dropped:0 overruns:0 frame:0
          TX packets:24694 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1898626 (1.8 MB)  TX bytes:1898626 (1.8 MB)
```

➔ IP Private là: 192.168.6.109

➔ Cấu hình địa chỉ cho httpServerReport2 và signalingServer trong file setting.js tại
P2P_LiveStreaming/src/DemoP2Pclappr/P2Pplugin/src/setting.js

```
setting.js - DemoP2Pclappr - Visual Studio Code

EXPLORER
  OPEN EDITORS
    JS setting.js P2Pplugin/src
  DEMOP2PCLAPPR
    dist
    node_modules
    P2Pplugin
      dist
      node_modules
      src
        js
        rtc-bufferedchannel
      JS p2p-loader.js
      JS setting.js
      JS index.js
      package-lock.json
      package.json
      webpack.config.js

JS setting.js
1  var freeice = require('freeice')
2
3  module.exports = {
4    maxSwarmSize:100,
5    ice:freeice(),
6    maxContributors:10,
7    timeOutForInterest:300,
8    timeoutForRequest:1500,
9    numberChunkCache: 15
10   (property) ratioMinRequestTimeout: number
11   ratioMinRequestTimeout:0.3333,
12
13
14   // httpServerReport2: "http://192.168.6.109:8090/report/interval",
15   httpServerReport2: "http://192.168.6.109:8090/report/interval",
16   signalingServer: "ws://192.168.6.109:8997"
17
18 }
```

→ Tại thư mục P2Pplugin, mở 1 terminal, chạy lệnh: npm run build

→ Để generated file p2p_plugin.js (plugin cho web)

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

bao@bao:~/Desktop/P2P_LiveStreaming/src/DemoP2Pclappr$ cd P2Pplugin/
bao@bao:~/Desktop/P2P_LiveStreaming/src/DemoP2Pclappr/P2Pplugin$ npm run build

> ginnop2pplugin@1.0.0 build /home/bao/Desktop/P2P_LiveStreaming/src/DemoP2Pclappr/P2Pplugin
> webpack --config webpack.config.js

Hash: 7810e941bf29c547830c
Version: webpack 4.20.2
Time: 10403ms
Built at: 11/23/2018 12:45:55 AM
    Asset      Size  Chunks             Chunk Names
p2p_plugin.js  618 KiB       0 [emitted] [big]  main
Entrypoint main [big] = p2p_plugin.js
   [4] (webpack)/buildin/global.js 509 bytes {0} [built]
  [16] ./src/setting.js 458 bytes {0} [built]
  [18] (webpack)/buildin/module.js 519 bytes {0} [built]
  [56] ./src/js/storage.js 3.53 KiB {0} [built]
  [57] ./src/js/adaptiveTimeout.js 1.48 KiB {0} [built]
```

→ Chạy WebServer:

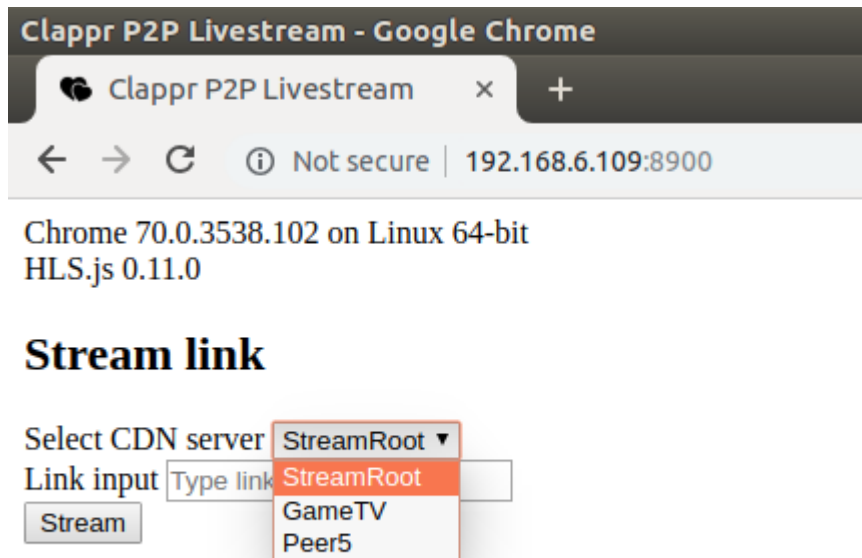
```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

bao@bao:~/Desktop/P2P_LiveStreaming/src/DemoP2Pclappr/P2Pplugin$ cd ..
bao@bao:~/Desktop/P2P_LiveStreaming/src/DemoP2Pclappr$ node index.js
Server started at 8900
```

Hệ thống Live Streaming đã chạy hoàn tất. Truy cập vào địa chỉ IP Private tại port 8900 để bắt đầu.

192.168.6.109:8900

Chọn 1 CDN server để bắt đầu:



Nhấn F12 chọn console để xem kết quả

Clappr P2P Livestream - Google Chrome


Clappr P2P Livestream

Not secure | 192.168.6.109:8900

Chrome 70.0.3538.102 on Linux 64-bit
HLS.js 0.11.0

Clappr P2P

P2P ID: cjoswimf40000316rs3xndn00.
Swarm ID: test
CDN chunk number:8
P2P chunk number:0
Level: 0. TimeP2P: 0. Time CDN: 1159
Chunk:4633172.ts. Size: 286kB
Sender: CDN
Speed: 1977 kbps
-----1 peers-----
CDN chunk number:8 : 100%
P2P chunk number:0 : 0%



Elements Console Sources Network Performance Memory Application Security Audits

top Filter Default level

P2P manager created! ▶ {room: "test", iceServers: Array(2), signaling: "ws://192.168.6.109:8997"}

Signaling:

ws://192.168.6.109:8997 ▶ {room: "test", iceServers: Array(2)}

Swarm created! abc

Manifest <https://demo-live.streamroot.io/index.m3u8> will be loaded.

loader https://demo-live.streamroot.io/media/w735619546_4633161.ts will be loaded.

time: 1542909565073 . From cdn: 4633161.ts

Update: 1118

loader https://demo-live.streamroot.io/media/w735619546_4633162.ts will be loaded.

time: 1542909565551 . From cdn: 4633162.ts

7. Kết quả và nhận xét

The screenshot shows a Google Chrome browser window titled "Clappr P2P Livestream". The address bar shows "192.168.6.109:8900". The page content includes the following text:

Chrome 70.0.3538.102 on Linux 64-bit
HLS.js 0.11.0

Clappr P2P

P2P ID: cjoswimf40000316rs3xndn00.
Swarm ID: test
CDN chunk number:333
P2P chunk number:0
Level: 0. TimeP2P: 111. Time CDN: 1337
Chunk:4633498.ts. Size: 373kB
Sender: CDN
Speed: 2231 kbps
-----3 peers-----
CDN chunk number:339 : 52%
P2P chunk number:315 : 48%

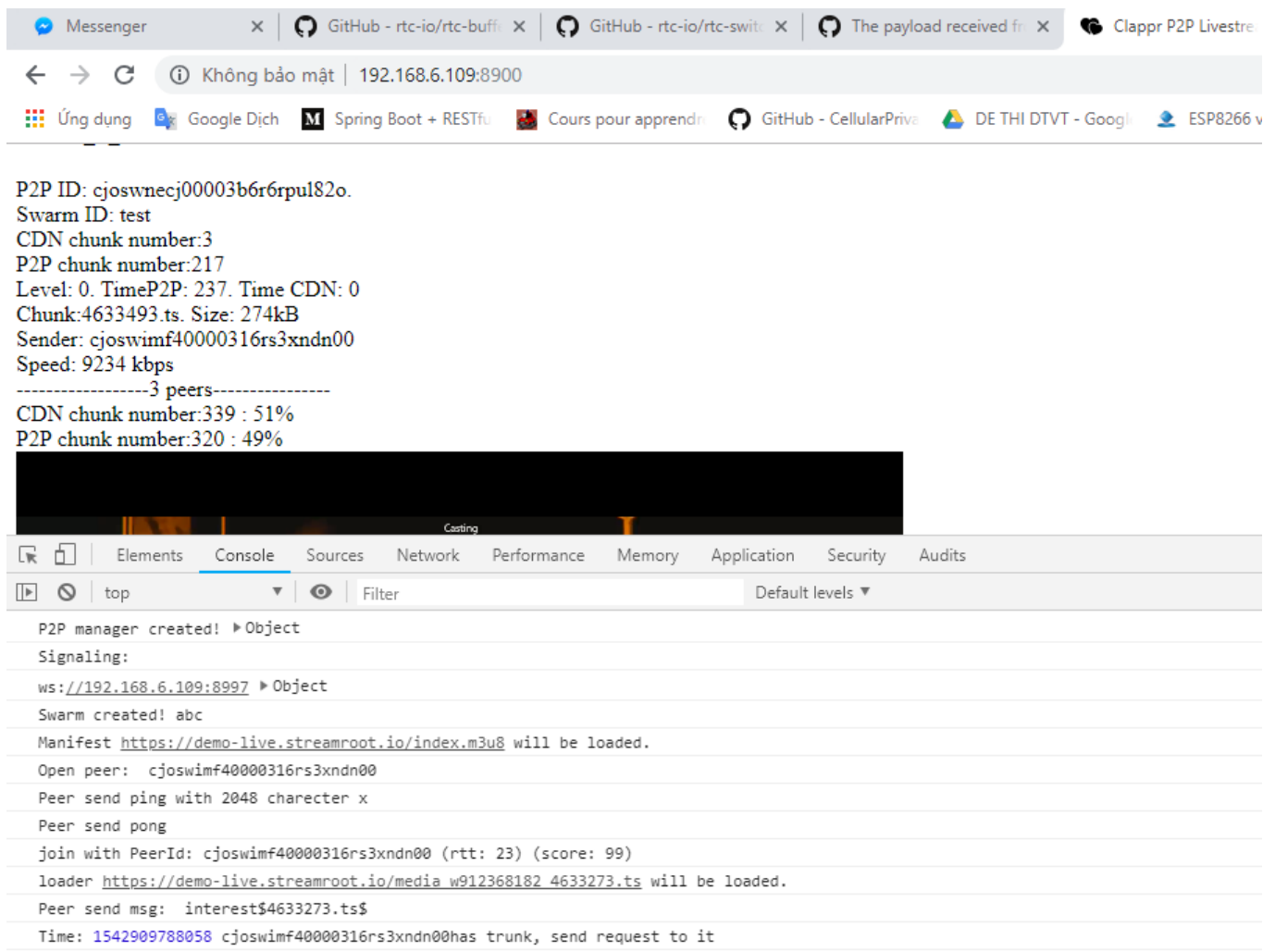
The video player interface shows a dark scene with orange and blue lines. The following roles and names are listed:

- Assistant Director: Marie Bos
- Gaffer: Eugene Spink
- Best boy: Tessa Pulles
- Electricians: Melanie Tenhagen, Tessa van den Briel
- Focus puller: Maarten van Raaij
- Key grip: Ray van den Veldert, Grietke
- Sound recorder: Victor Dekker
- Scripted extra: Daan van 't Ende, Koen Martens, Campbell Batten, Kiki Merle
- Set design: Ronke Faber
- Set construction: Seeb Dooling, Henricke Kraaij
- Props: Parake Houten
- Set dresser: Rik van Oe
- Set costumes: Lisa Lind
- Rigger: Rob Tuijthof
- Make-up and hair: Sanna Kivita
- Coloring: Delice Viba, Cafe Met

The Chrome DevTools console shows the following log messages:

```
P2P manager created! ▶ {room: "test", iceServers: Array(2), signaling: "ws://192.168.6.109:8997"}
Signaling:
ws://192.168.6.109:8997 ▶ {room: "test", iceServers: Array(2)}
Swarm created! abc
Manifest https://demo-live.streamroot.io/index.m3u8 will be loaded.
loader https://demo-live.streamroot.io/media_w735619546_4633161.ts will be loaded.
time: 1542909565073 . From cdn: 4633161.ts
Update: 1118
loader https://demo-live.streamroot.io/media_w735619546_4633162.ts will be loaded.
time: 1542909565551 . From cdn: 4633162.ts
```

Hình 9. Kết quả chạy Peer1



Hình 10. Kết quả chạy Peer2

Clappr P2P Livestream

Not secure | 192.168.6.109:8900

Chrome 70.0.3538.102 on Windows 10 64-bit
HLS.js 0.11.0

Clappr P2P

P2P ID: cjosws2xj00003b6r5c616eom.
Swarm ID: test
CDN chunk number:3
P2P chunk number:108
Level: 0. TimeP2P: 406. Time CDN: 0
Chunk:4633495.ts. Size: 334kB
Sender: cjoswimf40000316rs3xndn00
Speed: 6584 kbps
-----3 peers-----
CDN chunk number:339 : 51%
P2P chunk number:325 : 49%

Elements
Console
Sources
Network
Performance
Memory
Application
Security
Audit

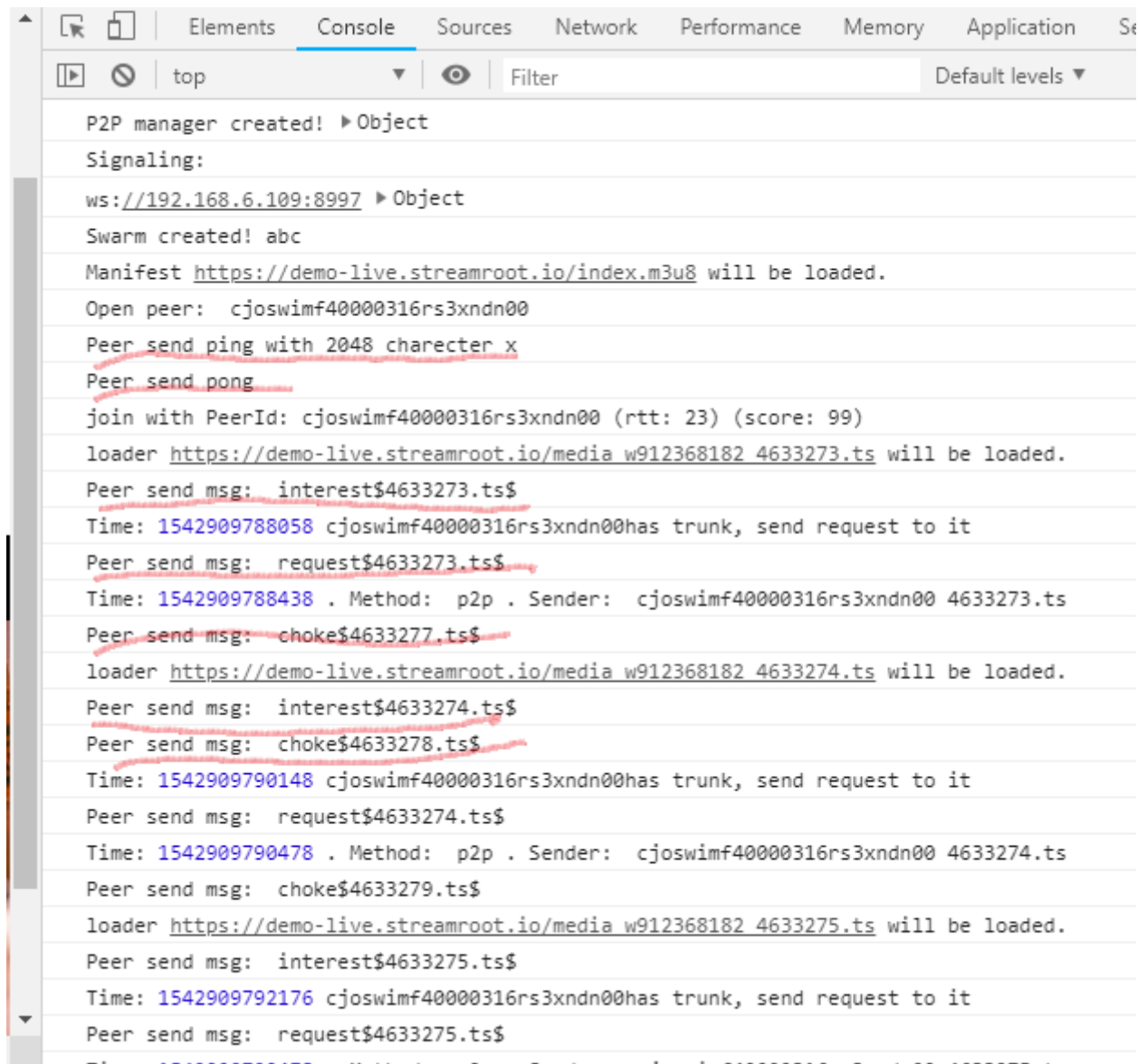
top
Filter
Default levels

```

P2P manager created! ▶ Object
Signaling:
ws://192.168.6.109:8997 ▶ Object
Swarm created! abc
Manifest https://demo-live.streamroot.io/index.m3u8 will be loaded.
Open peer: cjoswimf40000316rs3xndn00
Peer send ping with 2048 character x
Open peer: cjoswnecj00003b6r6rpul82o
Peer send ping with 2048 character x

```

Hình 11. Kết quả chạy Peer3



Hình 12. Peer gửi các bản tin handshaking.

❖ **Nhận xét:**

- Ở kết quả Peer1, Peer1 sẽ download toàn bộ các trunk từ server CDN, do nó được khởi động đầu tiên
- Ở kết quả Peer2, Peer3 cho thấy peer sẽ dùng P2P protocol để download các trunk, do các peer này được khởi động sau
- Số liệu thông kê hiện thị chính xác và đồng bộ giữa các peer.
- Sau khoảng 15ph chạy mô phỏng, ta thấy số trunk download sử dụng P2P của hệ thống là 49%, tiếp kiệm được đáng kể lưu lượng.

- Tuy nhiên, do điều kiện khách quan (không có server public), việc sử dụng p2p qua IP Private là điều kiện lý tưởng nên kết quả chưa phù hợp với thực tế. Sau 1 khoảng thời gian 30ph, các peer download sau bắt đầu bắt kịp peer download trước, lúc này cả 3 Peer sẽ request cùng 1 trunk, khiến cả 3 peer đều download trunk từ CDN.

-

Kết Luận

Qua bài tập lớn P2P Live Streaming, chúng em thu được rất nhiều vốn kiến thức cả lý thuyết lẫn thực hành, đặc biệt là kỹ năng làm việc nhóm, cách giải quyết vấn đề. Chắc chắn những kinh nghiệm này sẽ là hành trang hữu ích cho việc học tập và công việc sau này.

Một lần nữa nhóm chúng em xin chân thành cảm ơn thầy giáo Phạm Văn Tiến đã hướng dẫn em tận tình trong thời gian học tập để nhóm có đủ kiến thức để có thể hoàn thành bài tập lớn này.

Tài liệu tham khảo

- [1] <https://streamroot.io/>
- [2] <https://docs.peer5.com/>
- [3] <https://github.com/streamroot/clappr-p2phls-plugin>
- [4] <https://github.com/clappr/clappr>
- [5] <https://github.com/clappr/clappr-level-selector-plugin>
- [6] <https://github.com/rtc-io/rtc-bufferedchannel>
- [7] <https://github.com/rtc-io/rtc-switchboard>