

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

1. Cấu trúc hệ thống P2P Streaming	5
2. WebRTC	7
3. Hoạt động chung của project	9
4. P2P protocol.....	11
5. Cấu trúc project và chức năng của từng phần	14
5.1. Media Server	14
5.2. WebServer	15
5.3. Signaling server.....	19
5.4. Reporter.....	19
6. Kịch bản mô phỏng	20
7. Triển khai.....	20
8. Kết quả và nhận xét	25
Kết Luận	30
Tài liệu tham khảo	31

Danh Mục hình ảnh

<i>Hình 1. Cấu trúc hệ thống P2P Livestreaming.....</i>	<i>5</i>
<i>Hình 2. Hoạt động chung của project.....</i>	<i>9</i>
<i>Hình 3. P2P protocol</i>	<i>11</i>
<i>Hình 4. Cấu trúc project P2P LiveStreaming</i>	<i>14</i>
<i>Hình 5. Cấu trúc file WebServer DemoP2P Clappr</i>	<i>15</i>
<i>Hình 6. Cấu trúc P2Pplugin.....</i>	<i>16</i>
<i>Hình 7. p2p_manager.....</i>	<i>17</i>
<i>Hình 8. Peer và các thành phần của Peer</i>	<i>17</i>
<i>Hình 9. Kết quả chạy Peer1</i>	<i>25</i>
<i>Hình 10. Kết quả chạy Peer2</i>	<i>26</i>
<i>Hình 11. Kết quả chạy Peer3</i>	<i>27</i>
<i>Hình 12. Peer gửi các bản tin handshaking.....</i>	<i>28</i>

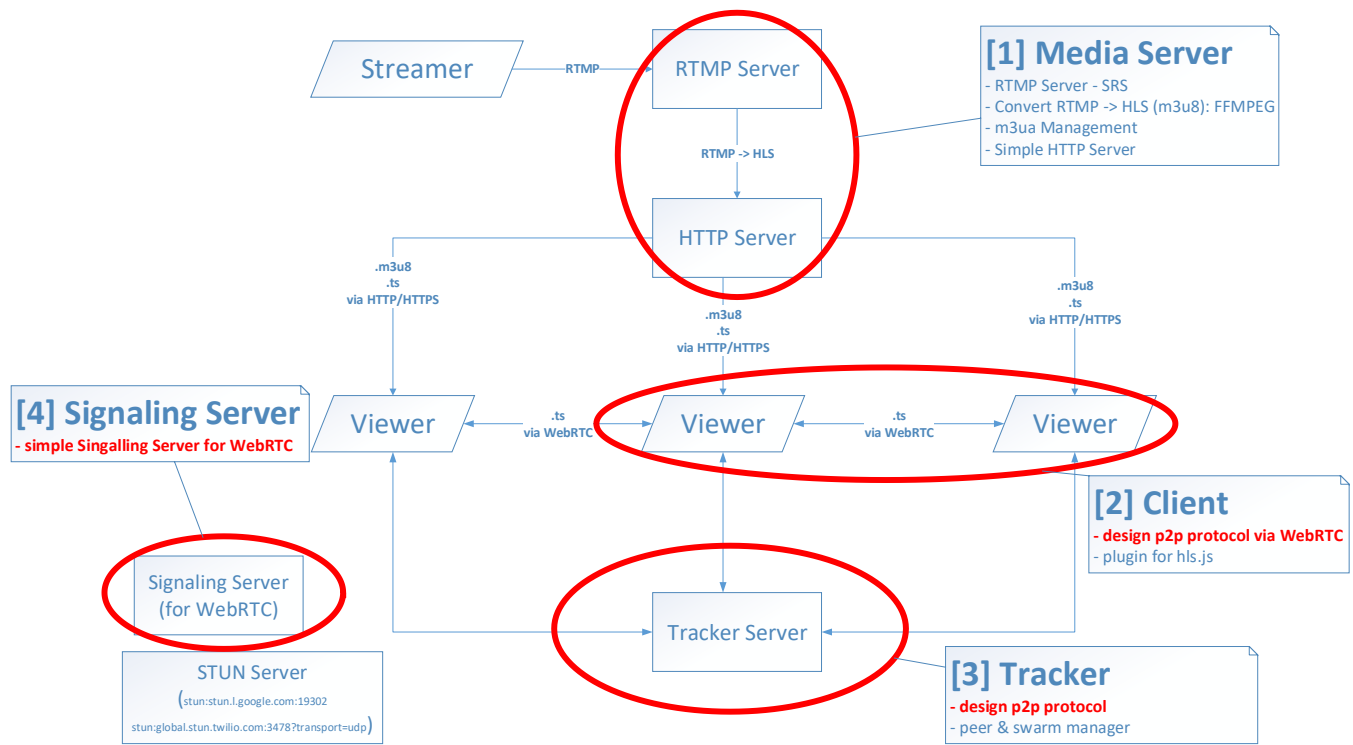
Mở đầu

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, tiết 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 RTCDataChannel 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 *RTCDDataChannel* 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. WebRTC

Trước tiên, ta tìm hiểu nền tảng để tạo phiên P2P là WebRTC.

WebRTC (Web Real-Time Communication) là một web API được phát triển bởi World Wide Web Consortium (W3C), khả năng hỗ trợ trình duyệt (browser) giao tiếp với nhau thông qua VideoCall, VoiceCall hay transfer data "Peer-to-Peer" (P2P) mà không cần browser phải cài thêm plugins hay phần mềm hỗ trợ nào từ bên ngoài.

Các phần chính của WebRTC bao gồm:

- **getUserMedia:** cho phép trình duyệt web truy cập vào camera và/hoặc microphone để lấy dữ liệu hình ảnh âm thanh cho việc truyền tải.
- **RTCPeerConnection:** dùng để cài đặt videocall/voicecall dùng cho việc truyền tải.
- **RTCDataChannel:** cho phép trình duyệt chia sẻ dữ liệu peer-to-peer.

Signaling

Signaling là quá trình phối hợp truyền thông. Để ứng dụng WebRTC thiết lập 'cuộc gọi', clients cần trao đổi thông tin:

- Thông điệp điều khiển phiên được sử dụng để mở hoặc đóng thông tin liên lạc.
- Thông báo lỗi
- Media metadata như codec và cài đặt codec, băng thông và loại media.
- Key data, được sử dụng để thiết lập kết nối an toàn.
- Dữ liệu mạng, chẳng hạn như địa chỉ IP và cổng public của máy chủ.

Bản tin SDP (Session description protocol) và ICE (Interactive Connectivity Establishment) là thành phần trong quá trình Signaling.

Bản tin SDP: cho biết các thông tin về dữ liệu truyền thông, cách thức mã hóa dữ liệu đó. Bản tin SDP được trao đổi giữa các Peers thông qua Signaling Server.

Bản tin ICE: cho biết các thông tin để tìm cách vượt qua NAT và FireWall để các Peers kết nối với nhau. Mỗi Peer sẽ kết nối với máy chủ STUN, STUN trả về một loạt các dự đoán về phương thức kết nối. Client sẽ sử dụng lần lượt các dự đoán này cho đến khi có một dự đoán hoạt động. Sau 1 thời gian STUN không thành công, tức là không thể P2P, ICE quyết định sử dụng máy chủ TURN để chuyển tiếp dữ liệu.

Hoạt động của WebRTC

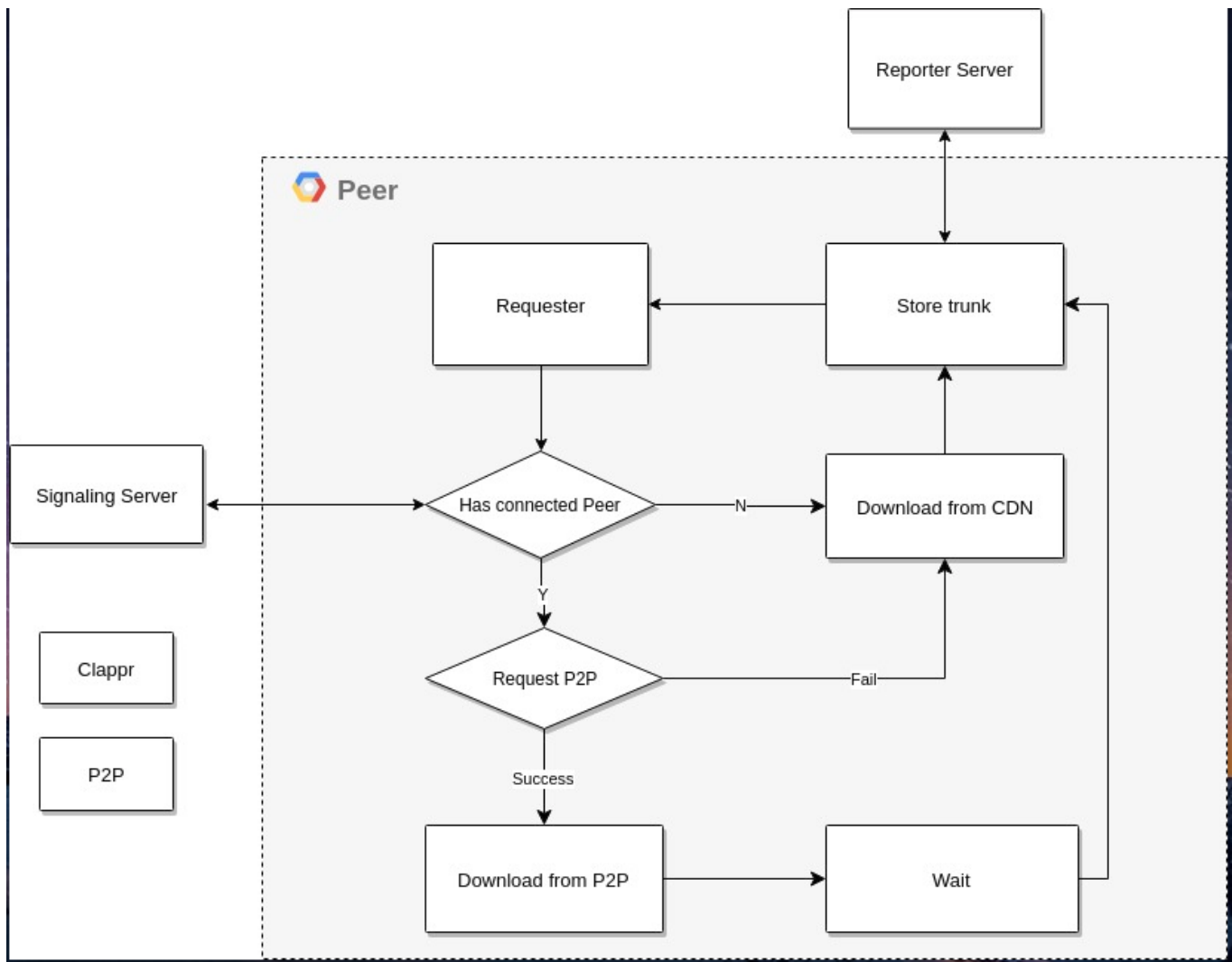
Caller: Người thực hiện cuộc gọi

Callee: Người nhận cuộc gọi

Các bước chính (Trong trường hợp STUN thành công – không sử dụng TURN):

- Initialization phase: Caller khởi tạo kết nối tạo offer và gửi SDP cho Callee.
- ICE Negotiation phase: truyền thông các bản tin ICE
- Callee receive SDP and ICE: Callee nhận bản tin SDP offer và ICE từ caller, xử lý và tạo SDP answer và ICE để gửi lại Caller.
- Caller receive SDP and ICE: Caller nhận bản tin SDP answer và ICE từ callee.
- Kết nối thành công Caller và Callee truyền dữ liệu

3. 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

P2Pplugin/src/p2p-loader.js: *loadInternal()*: Sử dụng http request để yêu cầu chunk từ CDN

P2Pplugin/src/p2p-loader.js: *readystatechange()*: Nhận chunk từ CDN bằng handle, sau đó callback success cho HLS

(2) Download từ P2P

P2Pplugin/src/p2p-loader.js: *p2pManager.requestResource(..)*: 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:

$$waitTime = 0.95 * chunkTime - downloadP2PTime$$

Nếu download thất bại Callback fail, peer download trunk từ CDN

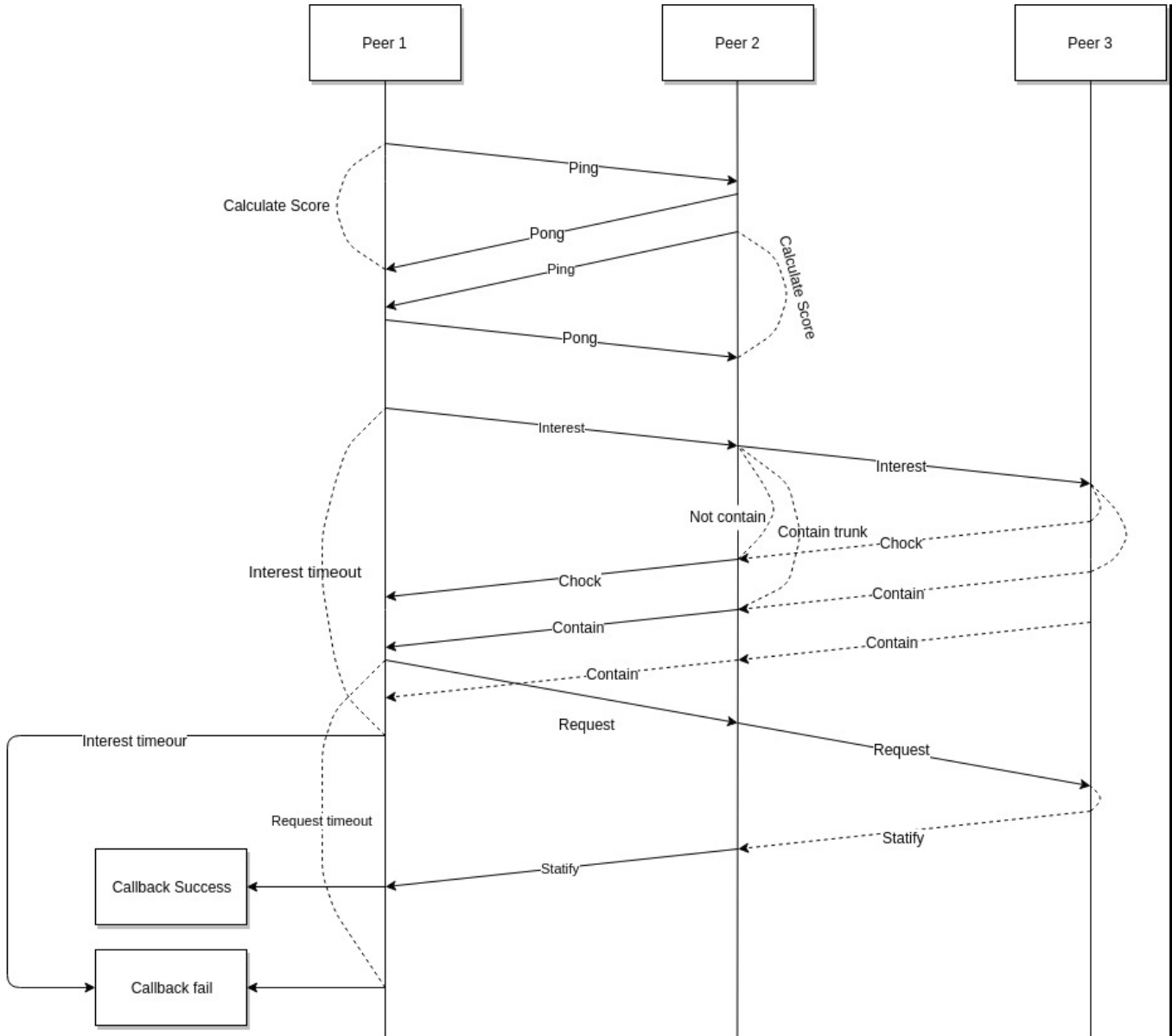
P2Pplugin/src/p2p-loader.js: *receiveCDN(resource, method)*: Trả về URL và nguyên nhân bị fail (interest timeout/request timeout), yêu cầu download trunk từ CDN

Download chunk đó từ CDN (1).

Sau khi đã download thành công (từ peer hoặc CDN), chunk đó được lưu lại tại storage 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.

4. P2P protocol



Hình 3. P2P protocol

Trước tiên, để khởi tạo kết nối giữa 2 peers, bản tin Ping/pong được gửi đi 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 bản tin **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 bản tin **contain** từ partner peer, gửi bản tin **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 tối đa 2 tín hiệu **request**, peer 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 nhận dữ liệu thành công

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 bản tin **contain**

➔ Nếu không có: gửi tín hiệu **chack**

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 bản tin *request*

Timeout cho quá trình request được tính:

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

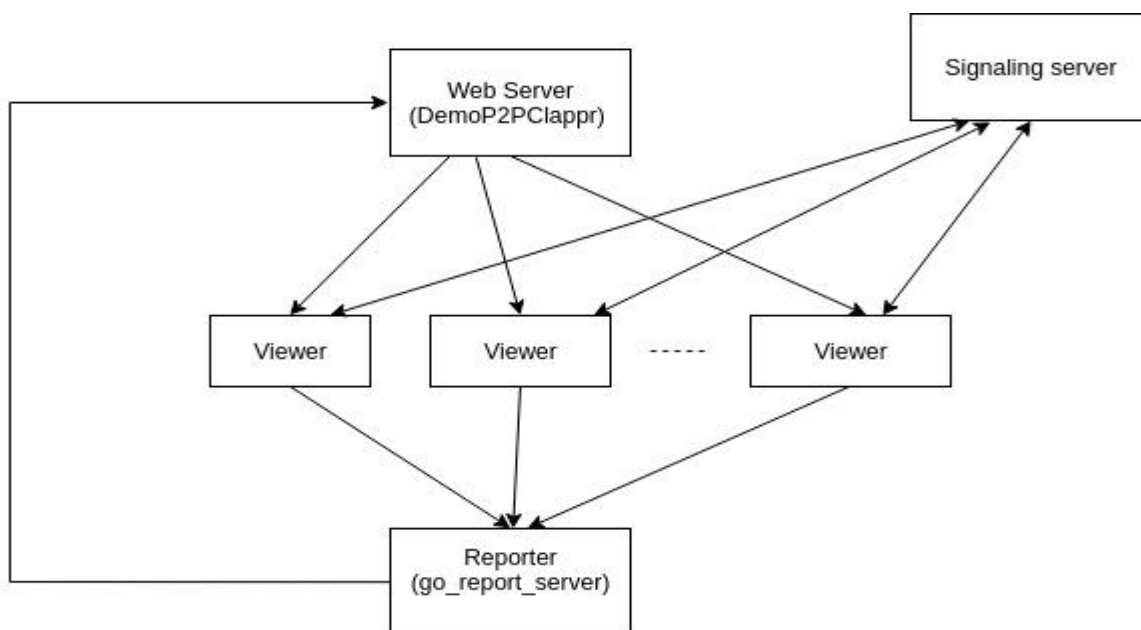
Bản tin *request* gửi cho peer xác nhận đầu tiên có chứa chunk cần download. Peer sẽ gửi lại dữ liệu chunk ở dạng Base64, nếu tại thời điểm nhận *request* peer không còn chứa dữ liệu chunk cần download thì sẽ gửi tín hiệu *chock*.

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)

5. 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: sau 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.

5.1. Media Server

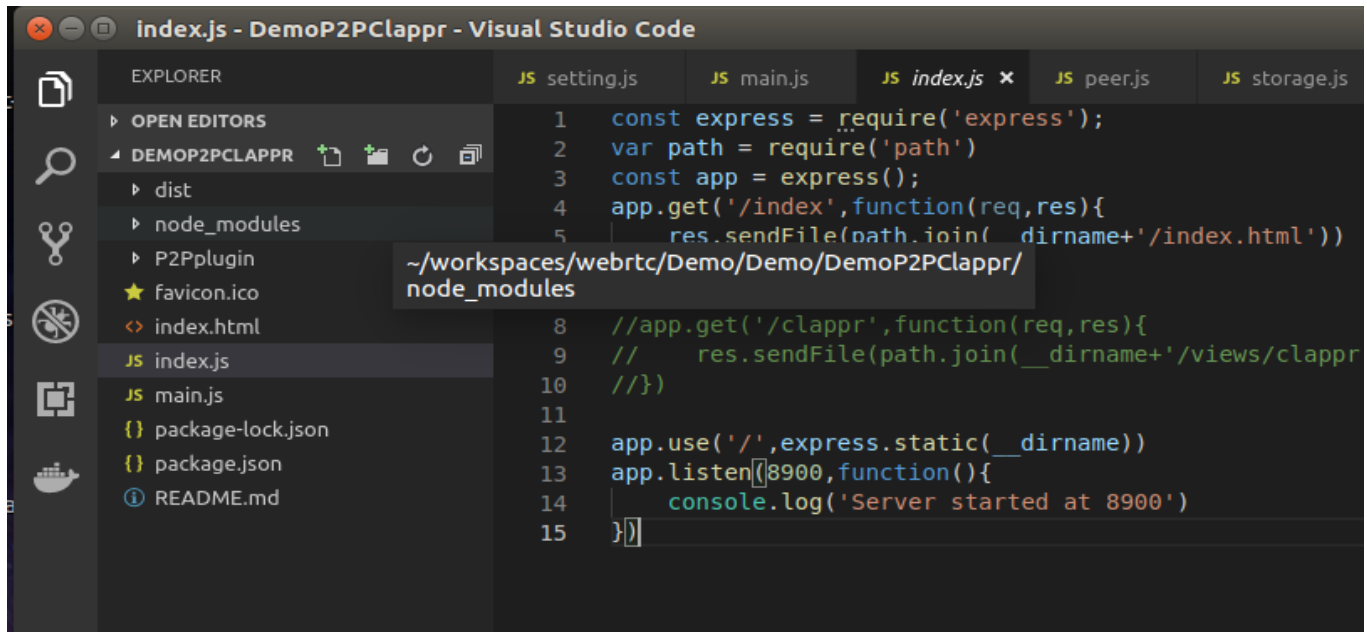
Chúng em sử dụng 2 server CDN public của streamroot và peer5 có địa chỉ tại :

- <https://demo-live.streamroot.io/index.m3u8> ^[2]
- https://wowza.peer5.com/live/smil:bbb_abr.smil/playlist.m3u8 ^[3]

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

5.2. WebServer

WebServer được customize từ open source *BemTV Plugin for Clappr Media Player* ^[4]. 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 DemoP2P Clappr

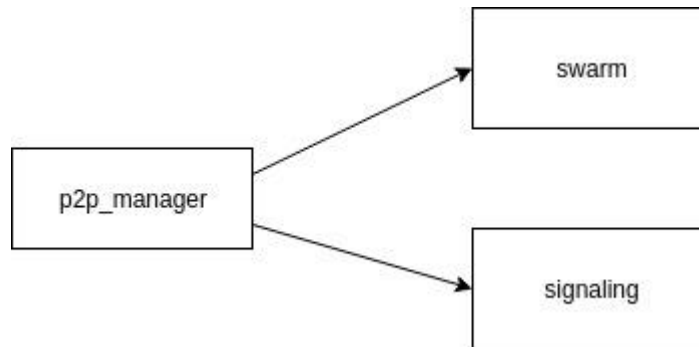
- **index.js**: khởi tạo web server.
- **index.html**: chứa thiết kế giao diện web.
- **main.js**: file script xử lý các event của web, được import trong file index.html
- Thư mục P2Pplugin: thành phần quan trọng nhất, chứa bản thiết kế plugin, đại diện cho mỗi Peer, được import trong file index.html
- **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) ^[5], được import trong file index.html
 - + level-selector.js: Clappr Level Selector Plugin ^[6]
- **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		
└─ JS adaptiveTimeout.js	258		
└─ JS base64-binary.js	259		
└─ JS md5.js	260		
└─ JS p2p_manager.js	261		
└─ JS peer.js	262		
└─ JS storage.js	263		
└─ JS swarm_utils.js	264		
└─ JS swarm.js	265		
└─ rtc-bufferedchannel	266		
└─ JS p2p-loader.js	267		
└─ JS setting.js	268		
└─ JS index.js	269		
└─ {} package-lock.json	270		
└─ {} package.json	271		
└─ {} webpack.config.js	272		

Hình 6. Cấu trúc P2Pplugin

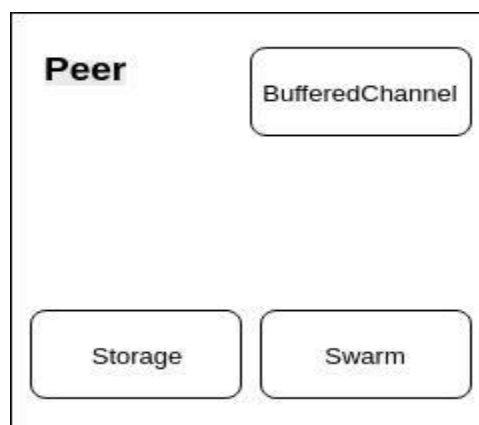
- **p2p_manager.js:**



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.interestedReceived(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 8).
- 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** ^[7]: 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

5.3. Signaling server

- Folder: signaling
- Sử dụng module rtc-swiffborad ^[8] để tạo signaling server. Nó sử dụng websocket để giao tiếp với các signaling client.

5.4. 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.

6. Kịch bản mô phỏng

Environment

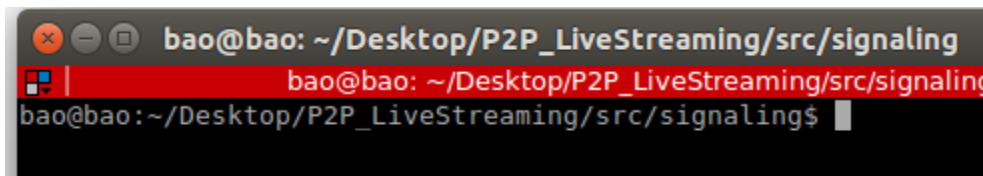
- OS: Ubuntu 16.04 LTS
- Nodejs: version 10.4.1
- Npm: 6.3.0
- Golang
- Browser: google chrome. Player: Clappr

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.

7. Triển khai

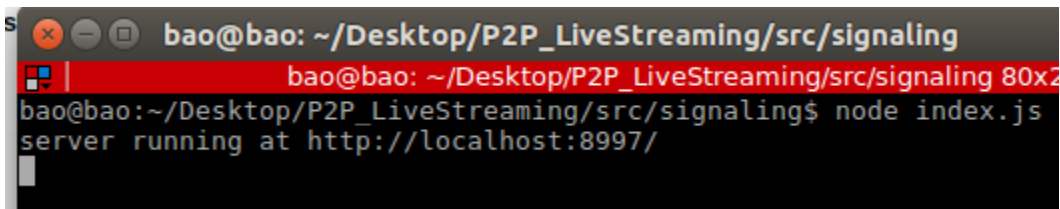
- **B1:** Khởi động signaling server

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

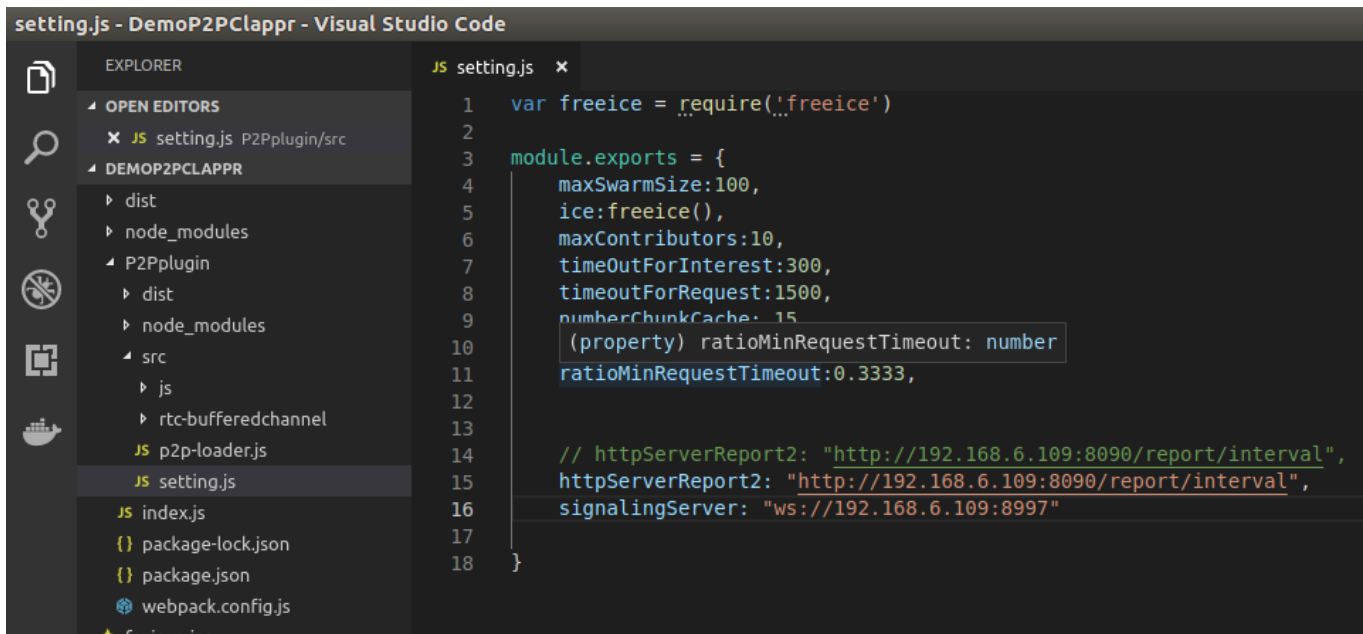
Lấy địa chỉ IP Private để config cho WebServer: ifconfig

```
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)
```

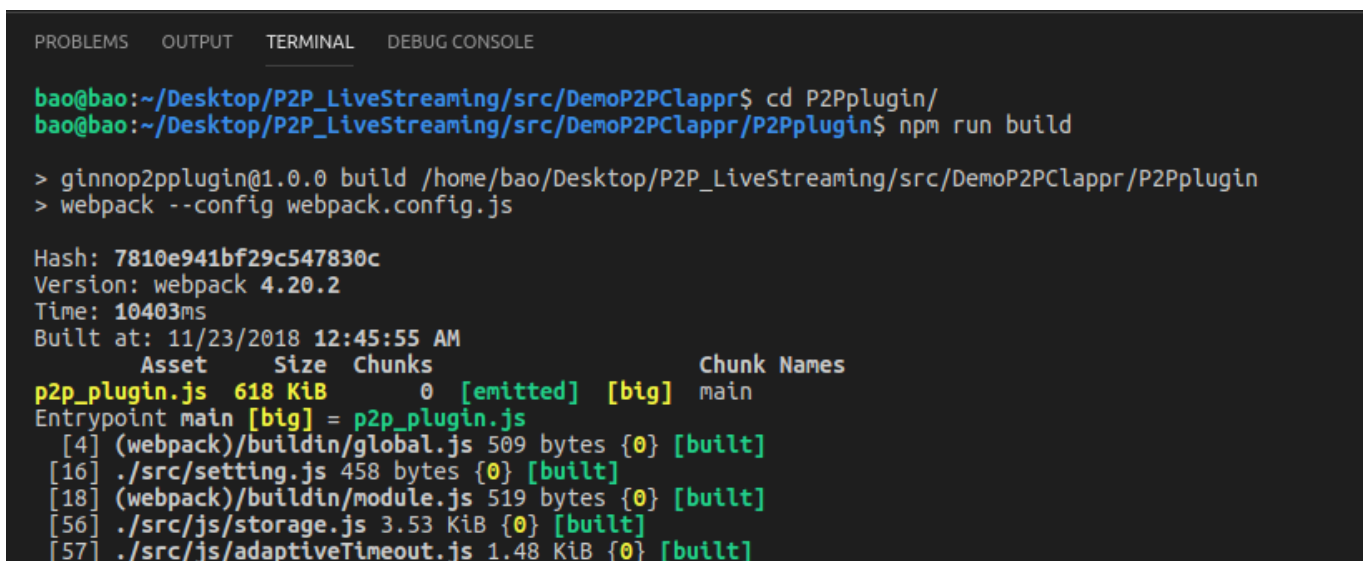
➔ IP Private là: 192.168.6.109

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



```
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)

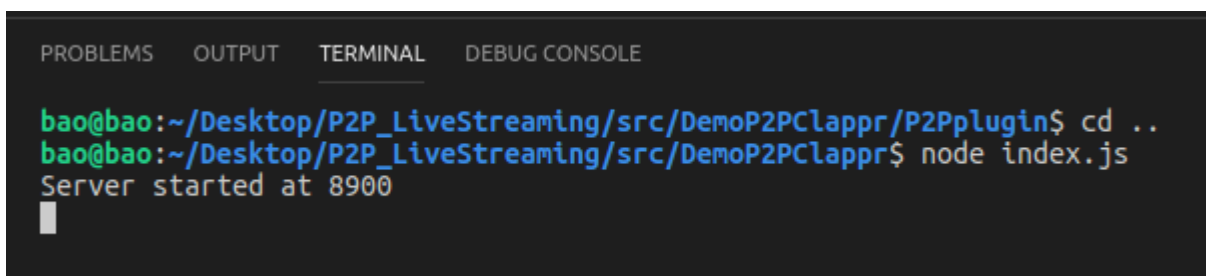


```
bao@bao:~/Desktop/P2P_LiveStreaming/src/DemoP2Pclappr$ cd P2Pplugin/
bao@bao:~/Desktop/P2P_LiveStreaming/src/DemoP2Pclappr/P2Pplugin$ npm run build

> 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:

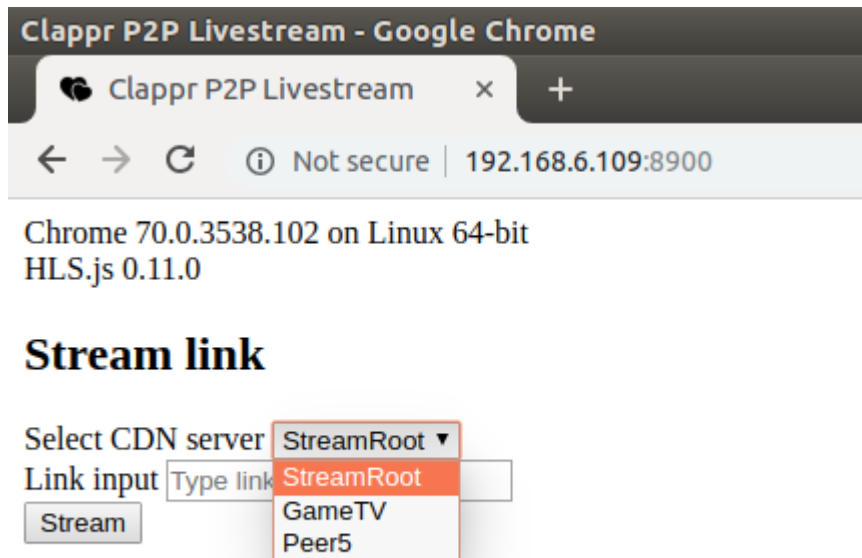


```
bao@bao:~/Desktop/P2P_LiveStreaming/src/DemoP2Pclappr/P2Pplugin$ cd ..
bao@bao:~/Desktop/P2P_LiveStreaming/src/DemoP2Pclappr$ node index.js
Server started at 8900
```

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 x +

← → ↻ ⓘ 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

8. Kết quả và nhận xét

The screenshot shows a Google Chrome browser window titled "Clappr P2P Livestream". The address bar displays "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. Overlaid text lists credits:

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

The bottom of the screenshot shows the Chrome DevTools console with 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

Messenger x GitHub - rtc-io/rtc-buff x GitHub - rtc-io/rtc-swit x The payload received fr x Clappr P2P Livestre

← → ↻ ⓘ Không bảo mật | 192.168.6.109:8900

Ứng dụng Google Dịch Spring Boot + RESTful Cours pour apprend GitHub - CellularPriv DE THI DTVT - Googl ESP8266 v

P2P ID: cjoswnecj00003b6r6rpul82o.
 Swarm ID: test
 CDN chunk number:3
 P2P chunk number:217
 Level: 0. TimeP2P: 237. Time CDN: 0
 Chunk:4633493.ts. Size: 274kB
 Sender: cjoswimf40000316rs3xndn00
 Speed: 9234 kbps
 -----3 peers-----
 CDN chunk number:339 : 51%
 P2P chunk number:320 : 49%

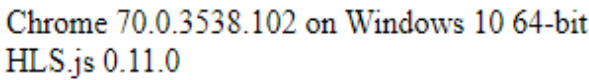
Casting

Elements Console Sources Network Performance Memory Application Security Audits

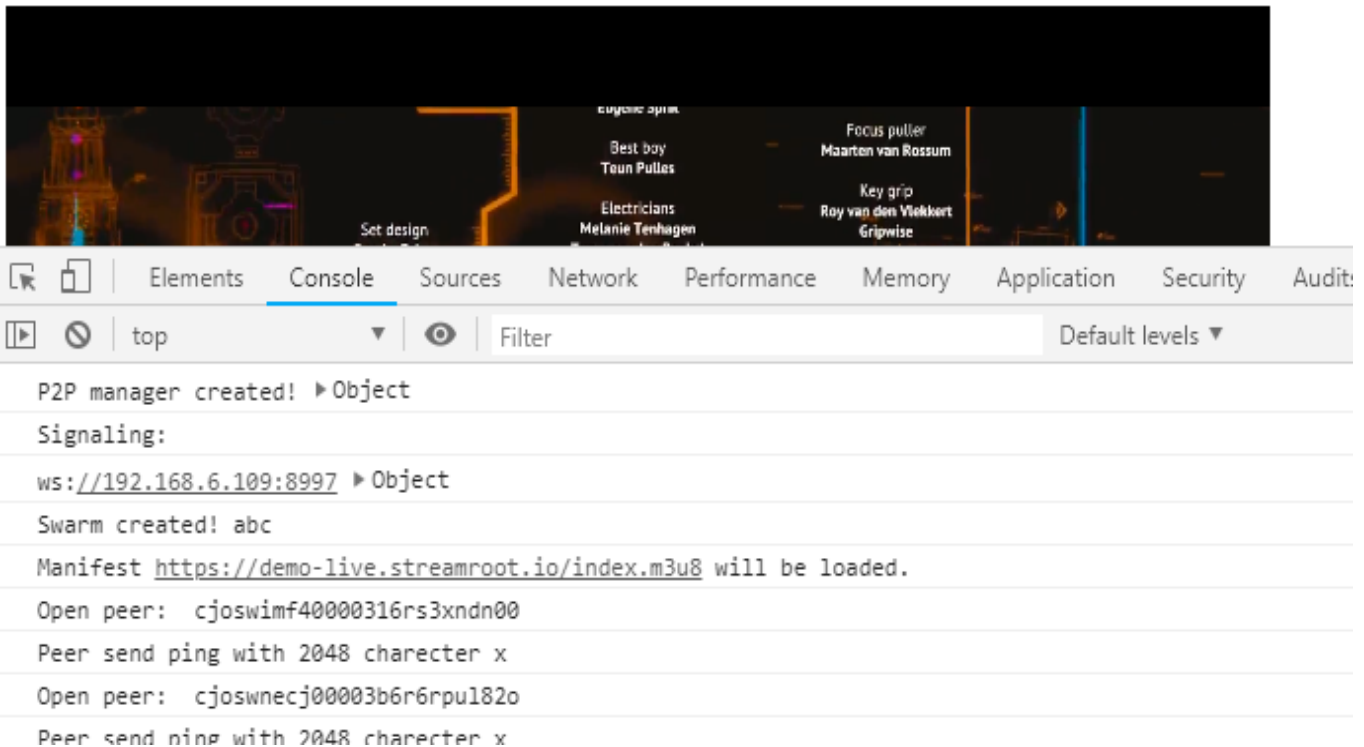
top Filter Default levels ▼

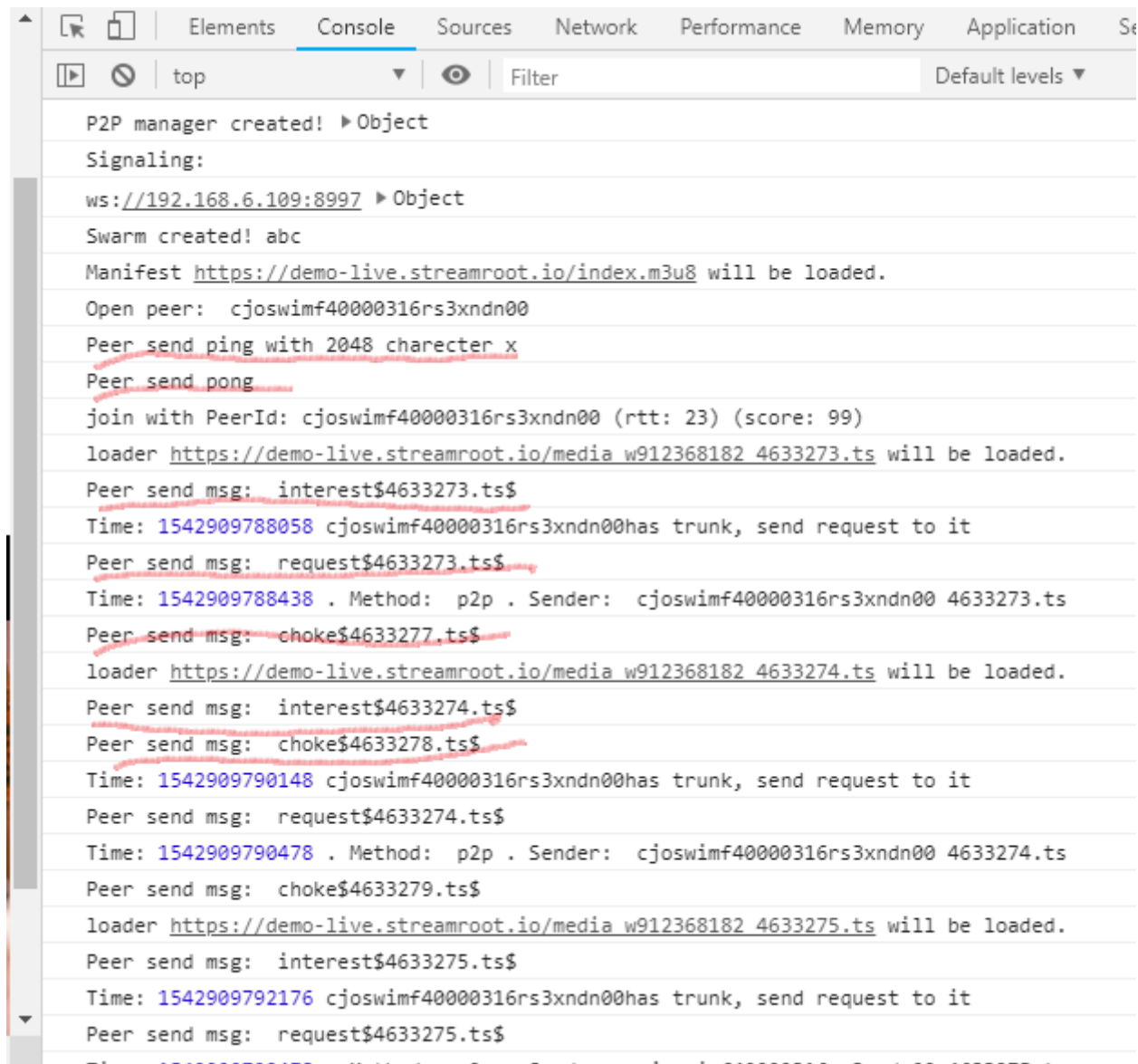
P2P manager created! ▶ Object
 Signaling:
 ws://192.168.6.109:8997 ▶ Object
 Swarm created! abc
 Manifest <https://demo-live.streamroot.io/index.m3u8> will be loaded.
 Open peer: cjoswimf40000316rs3xndn00
 Peer send ping with 2048 charecter x
 Peer send pong
 join with PeerId: cjoswimf40000316rs3xndn00 (rtt: 23) (score: 99)
 loader https://demo-live.streamroot.io/media/w912368182_4633273.ts will be loaded.
 Peer send msg: interest\$4633273.ts\$
 Time: 1542909788058 cjoswimf40000316rs3xndn00has trunk, send request to it

Hình 10. Kết quả chạy Peer2



```
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%
```





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 nền tảng để có thể hoàn thành bài tập lớn này.

Tài liệu tham khảo

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