TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN ĐIỆN TỬ VIỄN THÔNG

====o0o====

BÁO CÁO KẾT QUẢ THỰC TẬP KỸ THUẬT

Đề tài:

CÀI ĐẶT VÀ TRIỂN KHAI HỆ THỐNG HTTP/ 2.0

Sinh viên thực hiện: **Nguyễn Minh**

Mã số sinh viên: **20132586**

Lớp: **KSTN – ĐTVT – K58**

ESRC Lab, phòng nghiên cứu 420, nhà C9

GVHD: **PGS.TS Phạm Ngọc Nam**

Th.S Nguyễn Kim Thoa

Hà Nội, Ngày 7 tháng 9 năm 2016

MỤC LỤC

1. LỜI NÓI ĐẦU

Thực tập kỹ thuật là học phần vô cùng bổ ích đối với những sinh viên năm ba . Học phần giúp chúng em có thêm nhiều kiến thức thực tế, ứng dụng những kiến thức học trên lớp vào 1 đề tài cụ thể. Em cảm thấy vô cùng may mắn khi xin được thực tập trong ESRC Lab – một Lab chuyên về lập trình nhúng với thành viên là những anh chị rất nhiệt huyết và có nền tảng kiến thức thật tuyệt vời. Trong quá trình thực tập, em đã nhận được sự chỉ bảo tận tình của thầy cô và các anh chị trong Lab để giúp em hoàn thành được đề tài. Tuy nhiên, do hệ thống HTTP/2.0 khá phức tạp và chưa có nhiều tài liệu hướng dẫn cùng với thời gian thực hiện bị hạn chế nên kết quả không được như em mong muốn.

Qua đây, em cũng xin được gửi lời cảm ơn chân thành đến PGS.TS Phạm Ngọc Nam và Th.S Nguyễn Kim Thoa cùng các anh chị K57 trong Lab đã giúp đỡ để em có thể hoàn thành đợt thực tập này!

2. NỘI DUNG

2.1 Giới thiệu chức năng, nhiệm vụ, cơ cấu tổ chức của đơn vị tiếp nhận

2.1.1 Chức năng nhiệm vụ

ESRC Lab là một lab chuyên nghiên cứu về lập trình nhúng nhưng hiện nay Lab đã mở rộng thêm một số các hướng nghiên cứu mới đó là:

- Open Flow
- HTTP Streaming
- Internet of Things
- Truyền thông tin bằng ánh sáng nhìn thấy
- Network on Chip

2.1.2 Cơ cấu tổ chức

Chủ nhiệm Lab: PGS.TS Phạm Ngọc Nam – Phó viện trưởng Viện Điện tử viễn thông

Trưởng Lab: Lê Đình Tuyên - sinh viên K57

Thành viên: Sinh viên các khóa K57, K58 cùng các thầy cô nghiên cứu sinh

2.2 Nội dung thực tập

Công việc chính em được giao trong đợt thực tập này là tìm hiểu và cài đặt hệ thống HTTP/2.0 trên PC

2.2.1 Giới thiệu chung

Video streaming qua giao thức HTTP đã trở thành một kỹ thuật nổi trội trong việc vận chuyển dữ liệu đa phương tiện qua mạng. Trái với dịch vụ truyền tải đa phương tiện truyền thống, máy khách (thay vì máy chủ) giờ đây quyết định khi nào thì dữ liệu đa phương tiện được truyền. Điều này cho phép máy khách có thể lựa chọn tốc độ bit phù hợp để thích nghi với sự biến động của băng thông. Để đạt được điều này, nhà cung cấp dịch vụ cần tạo nhiều phiên bản cho cùng một nội dung đa phương tiện (với chất lượng khác nhau), mỗi phiên bản được chia ra làm các đoạn nhỏ với

thời gian bằng nhau. Máy khách sử dụng bản tin HTTP GET để nạp đoạn video phù hợp nhất dựa vào tình trạng hiện tại của mạng và thông tin trong MPD. Vấn đề đặt ra là thuật toán để lựa chọn các phiên bản sao cho trong một giới hạn về đường truyền, cảm nhận người xem thu được là tốt nhất có thể. Tuy nhiên, một trong những nhược đểm chính của HTTP/1.1 là việc sử dụng các phân đoạn video có thời gian không đổi. Thông thường thời gian này nằm trong khoảng từ 2 tới 10 giây. Hiển nhiên, thời gian phân đoạn càng ngắn thì máy khách càng thích ứng nhanh hơn với biến động mạng, ngoài ra, trễ đầu cuối cũng giảm. Tuy nhiên, do HTTP/1.1 chỉ cho phép một yêu cầu trên một kết nối TCP, việc sử dụng phân đoạn ngắn làm cho chi phí truy vấn và độ phức tạp tính toán của nút mạng tăng.

Tháng 5/2015, phiên bản mới của giao thức HTTP, gọi là HTTP/2, được đề xuất. Phiên bản mới này giới thiệu nhiều cải tiến và chỉnh sửa, trong đó có chức năng đẩy. Chức năng này cho phép máy chủ gửi nhiều hơn một đoạn video tới máy khách cho một yêu cầu. Để làm được điều này, truy vấn HTTP/2 có thể chỉ rõ số lượng phân đoạn sẽ được tải. Trong bối cảnh này, các phương pháp thích nghi không chỉ xem xét đến tốc độ bit (như truyền thống) mà cần tính đến việc đẩy bao nhiêu đoạn video cho phù hợp.

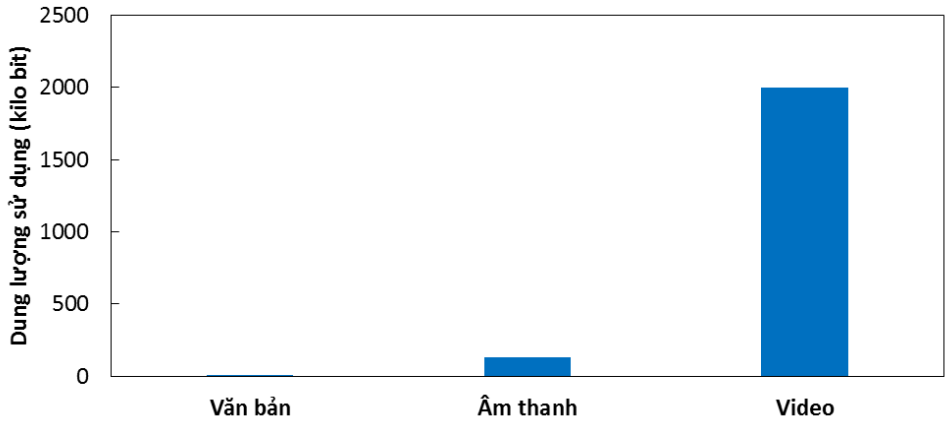
Trong những năm gần đây, nhiều phương thức thích nghi tốc độ bit đã được đề xuất. Hầu hết các phương pháp này hỗ trợ giao thức HTTP/1.1. Thông thường các phương pháp có thể chia ra làm hai nhóm là: nhóm dựa vào thông lượng và nhóm dựa vào mức bộ đệm. Theo như tên gọi của mình, các phương pháp này dựa vào thông lượng hoặc mức bộ đệm để quyết định tốc độ bit. Trong khi TCP được biết đến với sự biến động mạnh của thông lượng, phương pháp thích nghi tốc độ bit tốt là phương pháp giảm được những tác động xấu của biến động đường truyền (tốc độ bit thay đổi mượt mà, mức bộ đệm ổn định, chất lượng cảm nhận tốt).

Như vậy, ở HTTP/2, các phương pháp thích nghi không những cần giảm tác động xấu của đường truyền đến cảm nhận người xem, mà còn giảm chi phí truy vấn một cách tối đa.

2.2.2 Video streaming qua HTTP

2.2.2.1 Truyền tải video qua mạng

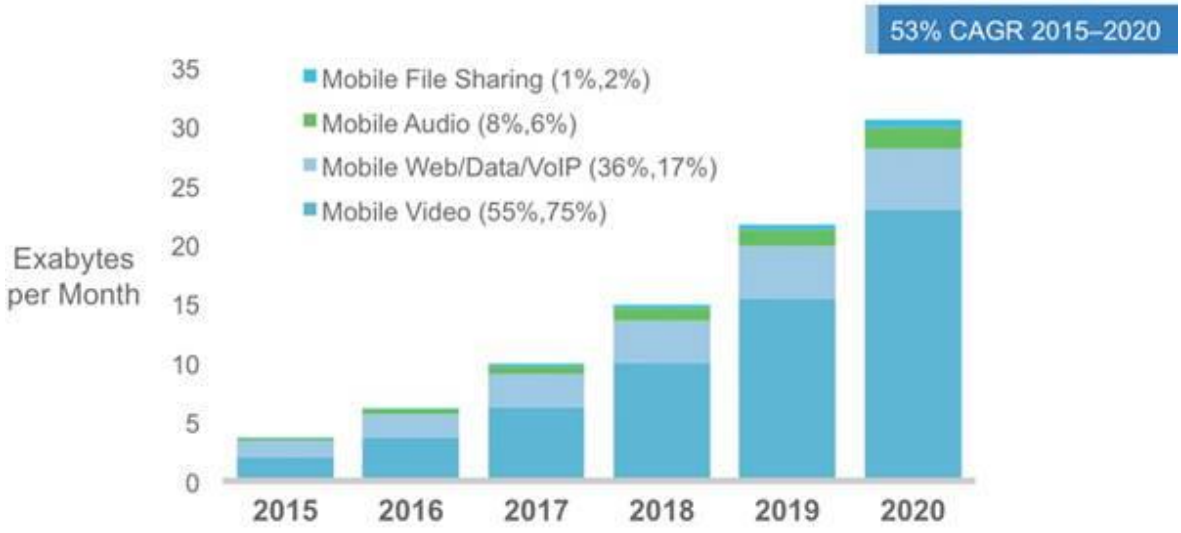
Ngày nay, việc sử dụng Internet để xem video qua mạng trở nên phổ biến do giá thành rẻ của chuyển mạch gói thông qua bộ giao thức TCP/IP. Thuật ngữ streaming ra đời ám chỉ việc xem video trong quá trình tải thay vì phải chờ cho tới khi nhận được toàn bộ nội dung video như trước đây. Nhắc đến streaming, ta có thể kể tới các ông lớn như Apple với HLS, Microsoft với Smooth Streaming hoặc Adobe với HDS. Streaming gồm phần hình và phần tiếng, tuy nhiên, như đánh giá sau đây, phần tiếng chiếm lượng dữ liệu rất nhỏ trong quá trình truyền. Do vậy, để cho đơn giản, đề tài này chỉ xem xét đến video streaming.



Hình 2.1 Dung lượng sử dụng của ba người dùng khác nhau trong một phút

Dữ liệu video có một đặc điểm nổi bật là dung lượng lớn. Lấy ví dụ trên ba người dùng sử dụng ba loại dữ liệu khác nhau là văn bản, âm thanh và video. Người dùng thứ nhất sử dụng dịch vụ văn bản để nhắn tin với bạn bè với tốc độ 500 ký tự trên 1 phút, mỗi ký tự 8 bit. Như vậy, tốc độ bit tương ứng với dịch vụ này là 4kbps.

Người dùng thứ hai nghe một bài hát với tốc độ 128kbps. Người dùng thứ ba xem một đoạn video chất lượng 2Mbps. Hình 2.1 so sánh dung lượng dữ liệu ba người dùng sử dụng trong một phút. Rõ ràng dung lượng yêu cầu để phục vụ cho việc xem video lớn hơn rất nhiều so với văn bản và âm thanh.



Hình 2.2 Dung lượng dữ liệu di động của các nguồn tài nguyên khác nhau

Theo như dự đoán của Cisco [1], đến năm 2020, dung lượng dữ liệu video chiếm đến khoảng 75% tổng dung lượng dữ liệu toàn cầu (Hình 2.2). Điều này cho thấy tầm quan trọng của việc sử dụng cũng như truyền tải video qua mạng.

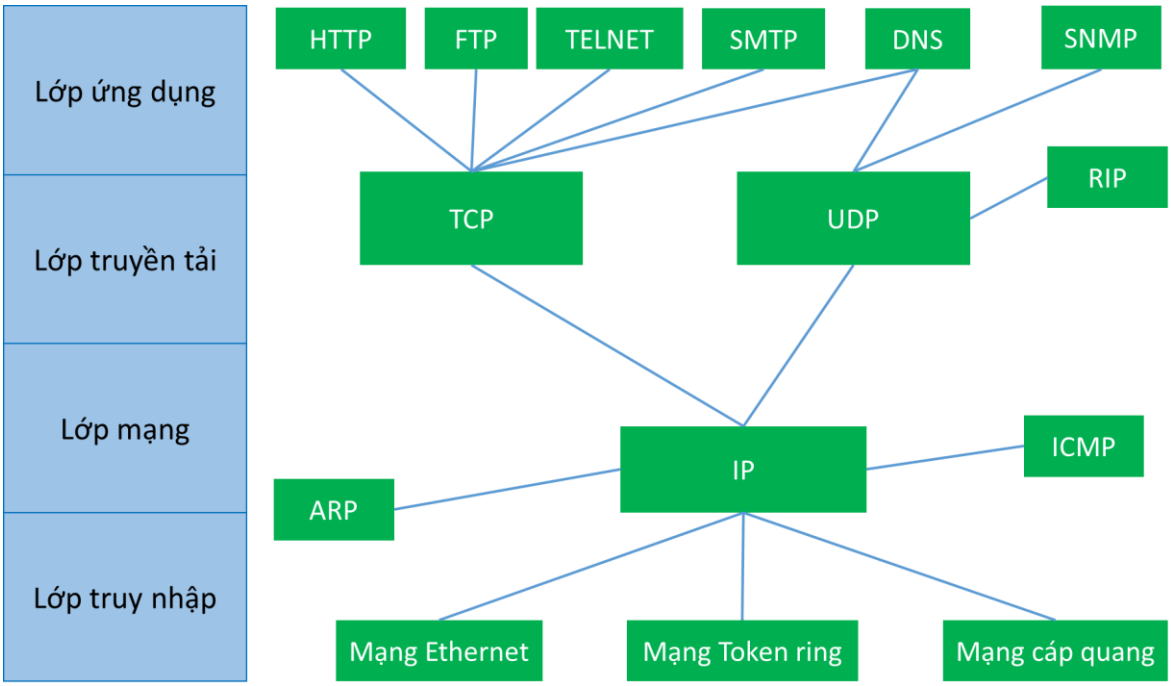
2.2.2.2 Bộ giao thức TCP/IP và giao thức HTTP

a) Bộ giao thức TCP/IP

Bộ giao thức TCP/TP là một bộ giao thức truyền thông cài đặt chồng giao thức mà hầu hết các máy tính thương mại đang chạy trên đó. Bộ giao thức này được đặt tên theo hai giao thức chính của nó, đó là giao thức điều khiển giao vận TCP và giao thức liên mạng IP.

Như nhiều bộ giao thức khác, bộ giao thức TCP/IP có thể coi là tập hợp của các lớp, mỗi lớp giải quyết một vấn về có liên quan đến việc truyền dữ liệu và cung cấp cho các giao thức lớp cấp trên một dịch vụ được định nghĩ rõ ràng dựa trên việc sử dụng các dịch vụ của lớp thấp hơn. Về mặt lô-gic, các lớp trên gần với người dùng

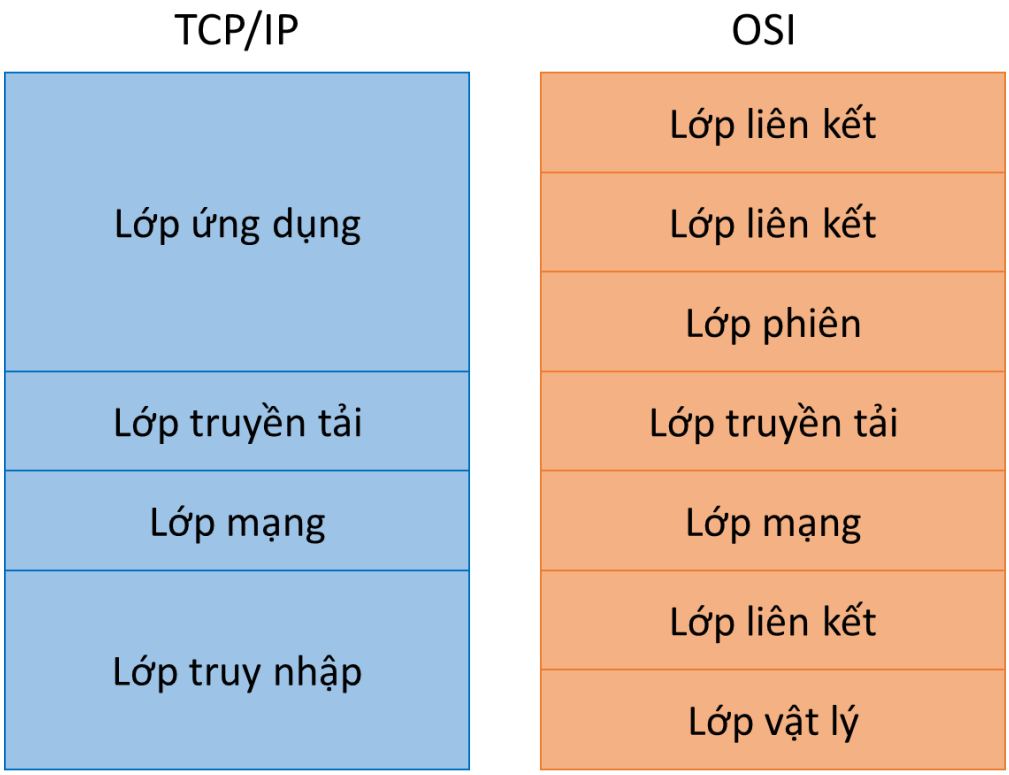
hơn và làm việc với dữ liệu trừu tượng hơn, chúng dựa vào các giao thức lớp cấp dưới để biến đổi dữ liệu thành các dạng mà cuối cùng có thể truyền đi một cách vật lý.



Hình 2.3 Mô hình TCP/IP

Hình 2.3 và Hình 2.4 mô tả các lớp của mô hình TCP/IP và so sánh với mô hình tham chiếu OSI.

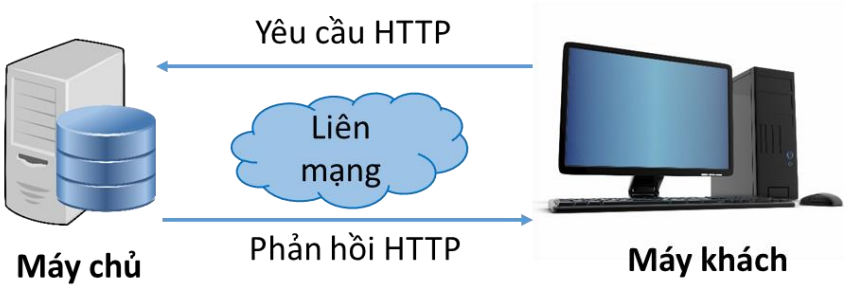
- Lớp ứng dụng: cung cấp dịch vụ người dùng, trao đổi dữ liệu ứng dụng. Các giao thức ứng dụng gồm: HTTP, TELNET, FTP, SMTP...
- Lớp truyền tải: thực hiện kết nối giữa hai thiết bị đầu cuối bằng giao thức TCP hoặc UDP.
- Lớp mạng: Xác định địa chỉ, tìm đường đi tốt nhất cho gói tin.
- Lớp liên kết: cung cấp các phương tiện kết nối vật lý như cáp, bộ chuyển đổi, giao thức kết nối... Cung cấp các dịch vụ cho tầng mạng.



Hình 2.4 So sánh với mô hình TCP/IP với mô hình OSI

b) Giao thức HTTP

Giao thức HTTP nằm trong tầng ứng dụng của mô hình TCP/IP, được sử dụng để truyền nội dung đa phương tiện từ máy chủ đến trình duyệt ở máy khách. Cơ chế hoạt động chính của HTTP là yêu cầu – phản hồi (Hình 2.5). Phiên bản hoàn chỉnh đầu tiên của HTTP là HTTP 0.9, công bố năm 1991. Tiếp theo là HTTP 1.0 (1996), HTTP 1.1 (1997) và mới đây nhất là HTTP 2.0 (2015). Các phiên bản sau ra đời nhằm thay thế cho phiên bản trước, kế thừa những chức năng cốt lõi, cải tiến và bổ sung.



Hình 2.5 Cơ chế yêu cầu – phản hồi của HTTP

Để làm việc với tài nguyên, HTTP 1.0 giới thiệu 3 phương thức:

- GET: gửi yêu cầu để lấy tài nguyên thông qua URL. URL chứa tất cả các thông tin cần thiết mà các máy chủ cần xác định vị trí và trả lại tài nguyên.
- POST: tạo ra một nguồn tài nguyên mới bằng cách gửi tài nguyên tới máy chủ.
- HEAD: lấy tiêu đề để kiểm tra thông tin cơ bản về tài nguyên.

Phiên bản HTTP 1.1 bổ sung thêm 5 phương thức[2]:

- OPTIONS: mô tả những lựa chọn giao tiếp cho tài nguyên. Máy khách sẽ được trả về thông tin về những cấu hình cơ bản của máy chủ.
- PUT: cập nhật một nguồn tài nguyên hiện có
- DELETE: xóa một nguồn tài nguyên hiện có
- TRACE: gửi bản tin phản hồi cho yêu cầu của máy khách, thường sử dụng để gỡ rối trong quá trình phát triển.
- CONNECT: thiết lập kết nối tới máy chủ được định nghĩa bởi URI.

Tuy nhiên, các phiên bản truyền thống của HTTP yêu cầu mỗi một bản tin yêu cầu cho một tài nguyên xác định. Điều này dẫn tới số lượng yêu cầu cho một phiên làm việc lớn. Do đó làm tăng độ phức tạp xử lý cho các nút mạng. Tháng 5 năm 2015,

HTTP 2.0 (HTTP/2) đã chính thức được công bố đưa ra những thay đổi nhằm nâng cao hiệu năng trong đó có phương thức đẩy. Phương thức đẩy cho phép máy chủ đẩy nhiều tài nguyên đến máy khách mà chỉ cần một bản tin yêu cầu. Do đó, chi phí cho yêu cầu cũng như độ phức tạp xử lý của các nút mạng giảm đi rõ rệt.

2.2.2.3 Những điểm mới trong HTTP/2

Tuy đã có những bước tiến quan trọng so với HTTP 1.0 (cải tiến bộ nhớ đệm, mã trạng thái mới, hỗ trợ nén nâng cao...), HTTP 1.1 không cho phép người tồn tại nhiều hơn một truy vấn trên mỗi kết nối TCP. Trong khi, các trang web ngày nay không chỉ có những đoạn mã HTML đơn giản, ngoài ra nó còn được kết hợp với các công cụ cho việc trang trí, thiết kế, các đoạn mã thực thi, hình ảnh, video... Để truyền tải lượng dữ liệu đó, trình duyệt phải tạo ra nhiều kết nối tới máy chủ. Nếu có quá nhiều truy vấn được thực hiện, nó sẽ làm ảnh hưởng đến hiệu suất mạng cũng như máy chủ. Ví dụ, người dùng A sử dụng nhấn vào một đường dẫn có năm ảnh. Mỗi ảnh được lưu tại một URL. Như vậy phải cần phải tốn năm truy vấn để hiện được năm ảnh này lên trình duyệt. Khi đó, thời gian tải sẽ bị ảnh hưởng, ngoài ra, các nút mạng cũng phải tiếp nhận và xử lý năm yêu cầu này. Chính vì vậy, HTTP/2 được tạo ra nhằm sử dụng hiệu quả hơn các nguồn lực mạng và giảm độ trễ bằng cách nén tệp tiêu đề và cho phép các lưu lượng đồng thời được lưu thông trên một kết nối. HTTP/2 là phiên bản chính thứ hai của giao thức mạng HTTP dựa trên SPDY/2 để cải tiến tốc độ truy cập. HTTP/2 mang lại một số lợi ích như giúp trình duyệt tải nội dung nhanh hơn, các kết nối có thời gian sống dài hơn, nội dung xuất hiện nhanh hơn, hỗ trợ nhiều kết nối song song. Ngoài ra, các yêu cầu HTTP được gửi tới máy chủ cũng nhẹ hơn nên rất nhiều yêu cầu có thể được thực hiện cùng lúc, hạn chế tình trạng nghẽn hoặc từ chối truy cập.

Nhìn chung, ở cấp độ cao, HTTP/2 có các đặc điểm:

- Nhị phân
- Đa kết nối

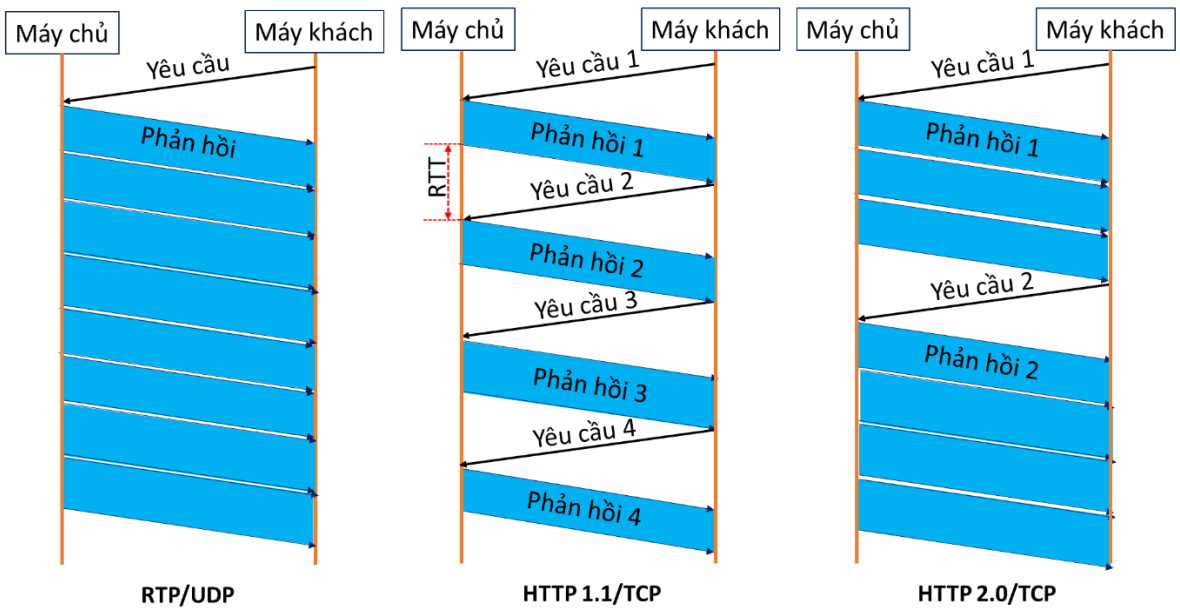
- Sử dụng một kết nối cho trạng thái song song
- Sử dụng nén tiêu đề để giảm tải cho máy chủ

2.2.3 Truyền tải video thích ứng qua giao thức HTTP

2.2.3.1 Truyền tải video sử dụng giao thức HTTP/TCP

Khi streaming qua giao thức HTTP chưa được phát triển một cách mạnh mẽ như hiện tại, việc truyền tải video qua mạng chủ yếu sử dụng bộ giao thức RTP/UDP. Điều này dẫn tới một số những nhược điểm sau:

- Không tin cậy: Khi UDP gửi bản tin, ta không thể biết được gói tin có đến được đích hay không, gói tin có thể bị mất trong quá trình truyền, không có sự xác nhận từ máy khách cũng như truyền lại từ máy chủ.
- UDP cho phép máy chủ đẩy liên tục tài nguyên từ máy chủ tới máy khách, điều này dẫn tới việc lãng phí tài nguyên nến như người dùng tải cả video về nhưng không xem hết.
- Do không có cơ chế chống tắc nghẽn, RTP/UDP không đảm bảo cho người dùng có thể xem video một cách liên tục khi băng thông biến động.
- Máy chủ RTP hoàn toàn điều khiển quá trình truyền, trong khi mỗi máy chủ tiếp nhận một lượng lớn yêu cầu đồng thời từ các máy khách. Điều này làm tăng chi phí và độ phức tạp của việc triển khai hệ thống truyền tải video với quy mô lớn.
- Dữ liệu truyền qua giao thức UDP có thể bị tường lửa chặn.



Hình 2.6 Truyền gói tin qua các bộ giao thức

HTTP trở thành giao thức phổ biến nhất trong việc truyền tải video qua mạng là do giao thức này khắc phục hoàn toàn những nhược điểm trên. Trong HTTP streaming, máy khách nắm toàn bộ quá trình truyền, điều này làm giảm độ phức tạp xử lý cho máy chủ. Nhờ việc sử dụng lại cơ sở hạ tầng sẵn có của liên mạng, chi phí xây dựng cũng như triển khai của HTTP streaming rất thấp. Ngoài ra, việc truyền dữ liệu qua HTTP thân thiện với tường lửa và được chống tắc nghẽn từ giao thức TCP.

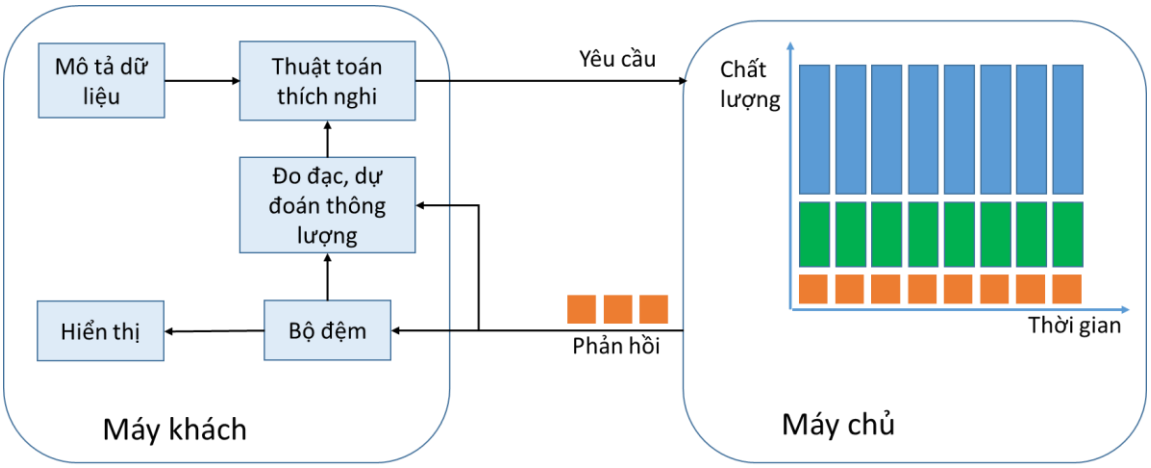
Trong HTTP streaming, video được lưu ở máy chủ như một tệp thông thường với một URL xác định. Khi người dùng muốn xem video, máy khách thiết lập một kết nối TCP tới máy chủ và gửi bản tin GET cho URL ứng với video được yêu cầu. Máy chủ gửi video đã được yêu cầu kèm theo một bản tin phản hồi đến người dùng nhanh nhất có thể thông qua hoạt động điều khiển luồng và chống tắc nghẽn ở giao thức TCP. Ở phía máy khách, dữ liệu được tiếp nhận và được lưu trữ tạm thời ở bộ đệm. Khi bộ đệm đạt được một ngưỡng xác định, ứng dụng ở máy khách bắt đầu chạy video thông qua việc nạp khung hình từ bộ đệm, giải mã và hiển thị video trên màn hình.

Hình 2.6 mô tả cơ bản quá trình truyền dữ liệu của các bộ giao thức. Dễ thấy ở bộ giao thức RTP/UDP, do không có chế độ xác nhận giữa máy chủ và máy khách

nên quá trình truyền không tốn chi phí truy vấn. Ngoài ra, thông lượng trong trường hợp này cũng lớn hơn do không tốn những khoảng thời gian chờ giữa những lần xác nhận. Khoảng thời gian này được gọi là RTT, được tính từ lúc máy khách gửi yêu cầu cho tới khi nhận được bit đầu tiền. Ở bộ giao thức HTTP 1.1/TCP, do có cơ chế xác nhận và chỉ truy vấn được một URL cho mỗi kết nối TCP, nên chi phí truy vấn nhiều hơn, dẫn tới sự suy giảm thông lượng cũng như tăng độ phức tạp xử lý cho nút mạng. Ở bộ giao thức HTTP 2.0/TCP, nhược điểm về chi phí truy vấn đã được giảm do mỗi lần truy vấn, máy chủ có thể gửi cùng lúc nhiều đoạn video cho máy khách.

2.2.3.2 Truyền tải video thích nghi và chuẩn DASH

Để đưa HTTP streaming trở nên công nghiệp hóa, ISO/IEC MPEG đã đưa ra chuẩn DASH, định nghĩa tiêu đề cũng như định dạng nội dung cho dữ liệu đa phương tiện. Trong DASH, một video gốc được mã hóa tại nhiều phiên bản khác nhau. Mỗi phiên bản có thể có các cấu hình khác nhau (độ phân giải, tốc độ khung hình, chất lượng cảm nhận) và được đại diện bởi một tốc độ bit. Thông thường, với cùng một nội dung, tốc độ bit càng lớn thì chất lượng video càng cao. Mỗi phiên bản lại được tiếp tục chia ra thành các đoạn với độ dài (theo thời gian) không đổi. Thông thường độ dài đoạn nằm trong khoảng từ hai giây đến mười giây. Khi băng thông cao, máy khách có thể yêu cầu chơi video ở chất lượng cao và ngược lại, khi băng thông hạn chế, máy khách yêu cầu giảm chất lượng video để tránh việc xem bị ngắt quãng. Do vậy, với băng thông biến động, máy khách có thể thích nghi bằng cách lựa chọn các đoạn với những chất lượng hợp lý sao cho chất lượng video chơi là tối ưu và tránh được sự gián đoạn.



Hình 2.7 Các khối cơ bản trong hệ thống video streaming

Một kiến trúc cơ bản của hệ thống HTTP streaming theo chuẩn DASH bao gồm một máy chủ lưu trữ nội dung và một máy khách quyết định chất lượng video sẽ được chơi (Hình 2.7). Thông thường, video được truyền tải thông qua một dãy các yêu cầu-phản hồi. Các phân đoạn video tải về được lưu trữ tại khối *Bộ đệm*, trước khi được chơi tại khối *Hiển thị*. Thành phần quan trọng ở phía máy khách là khối *Thuật toán thích nghi*, nơi mà các thuật toán được áp dụng và lựa chọn phiên bản nào tiếp theo sẽ được tải. Các quyết định này sẽ dựa trên mức bộ đệm thực tế (quan sát được từ khối *Bộ đệm*), thông lượng ước lượng được (cung cấp bởi khối *Đo đạc và ước lượng thông lượng*) và thông tin *Mô tả dữ liệu*. Mỗi phiên bản của video gốc được lưu bởi một URL. Liên mạng được sử dụng để giao tiếp giữa máy chủ và máy khách. Khi bắt đầu phiên làm việc, máy khách sẽ gửi yêu cầu, máy chủ phản hồi bằng bản tin tiêu đề để máy khách nắm được những thông tin cơ bản về video sắp được tải. Sau đó, dựa vào thông tin này, cũng như tình trạng mạng, máy khách sẽ gửi bản tin GET kèm theo URL tương ứng. Thông thường, nếu hiện tại mức bộ đệm và băng thông đo được ở mức cao, phiên bản có tốc độ bit cao sẽ được chọn và ngược lại, nếu mức bộ đệm thấp cũng như băng thông biến động, phiên bản có tốc độ bit thấp hơn được chọn để tránh cho việc xem video bị ngắt quãng.

Tuy nhiên, khi chất lượng thay đổi, chất lượng cảm nhận của người dùng cũng bị ảnh hưởng. Ví dụ, người dùng thường cảm thấy không dễ chịu khi một giây trước

chất lượng video ở mức rất cao nhưng đột nhiên một giây sau chất lượng lại bị giảm đi rất thấp. Thay vì giảm chất lượng một cách đột ngột, ta chơi video ở mức chất lượng tương đối cao và giảm chất lượng một cách từ từ để giảm những tác động tiêu cực đến người dùng. Như vậy, những thông số đầu ra (ví dụ: số lần thay đổi chất lượng, sự (mức) suy giảm chất lượng cao nhất, chất lượng trung bình, độ ổn định mức bộ đệm...) là một trong nhưng tiêu chí quan trọng để đánh giá hiệu năng của thuật toán quyết định tốc độ bit.

HTTP streaming có thể được sử dụng cho dịch vụ streaming trực tuyến và dịch vụ streaming theo yêu cầu. Ở streaming theo yêu cầu, video đã sẵn có ở máy chủ, máy khách có thể tải về một lượng lớn thông tin và lưu trữ ở bộ đệm. Trong khi đó, ở streaming trực tuyến, một trong những khó khăn lớn nhất là mức bộ đệm không thể vượt quá một giá trị ngưỡng. Lý do là vì nội dung cho những đoạn video tiếp theo chưa sẵn có theo thời gian thực. Hơn nữa, streaming trực tuyến cũng rất nhạy cảm với trễ. Ví dụ khi xem một trận bóng trực tuyến, sẽ là rất phiền nếu hàng xóm đã ăn mừng bàn thắng trong khi ở màn hình của mình bóng vẫn đang ở giữa sân. Khái niệm mức bộ đệm khởi tạo chỉ ra số lượng đoạn video cần được đệm trước khi máy khách bắt đầu chơi. Khái niệm mức bộ đệm mục tiêu chỉ ra mức bộ đệm mà máy khách cố gắng duy trì trong phiên làm việc. Trong streaming trực tuyến hai khái niệm trên là như nhau và cũng chính là mức bộ đệm tối đa mà bộ đệm có thể đạt được.

2.2.3.3 Đánh giá chất lượng của video streaming qua HTTP/2

Một trong những mục đích cơ bản của HAS là tối ưu chất lượng video. Chất lượng video phụ thuộc chủ yếu vào tốc độ bit cũng như sự gián đoạn khi xem. Sự gián đoạn (gây ra bởi mức bộ đệm rỗng) được xem xét là có ảnh hưởng xấu nhất đến chất lượng video. Khi thông lượng giảm và thấp hơn tốc độ bit, mức bộ đệm sẽ giảm mạnh. Khi lượng dữ liệu trong bộ đệm không đủ để chơi, máy khách phải dừng và chờ tới khi bộ đệm nạp đủ số đoạn yêu cầu. Trong video streaming, sự gián đoạn có thể xảy

ra, đặc biệt là trong trường hợp streaming trực tuyến, do mức bộ đệm tối đa bị giới hạn. Rất nhiều nghiên cứu chỉ ra rằng sự gián đoạn này cần tránh để cải thiện chất lượng cảm nhận.

Chất lượng video thay đổi rất quan trọng khi đánh giá QoE của người dùng, mặc dù nó có vai trò chính trong HAS là khắc phục tình trạng gián đoạn khi xem. Một nghiên cứa chủ quan của chất lượng video chỉ ra rằng việc chất lượng video thay đổi dần dần tới các mức chất lượng thấp hơn cho kết quả QoE tốt hơn so với sự giảm đột ngột và mạnh của chất lượng. Tuy nhiên, sự tăng chất lượng với mức độ cao không làm giảm QoE trong tất cả các trường hợp. Một nghiên cứu gần đây chỉ ra rằng video có số lần thay giảm (giảm) chất lượng ít sẽ ảnh hưởng ít hơn tới chất lượng cảm nhận người dùng. Trong khi đó, sự tăng chất lượng đột ngột có thể cải thiện QoE khi mà người dùng có thể dễ dàng cảm nhận được sự cải thiện chất lượng.

2.2.4 Tiến hành cài đặt

2.2.4.1 Khái quát các công việc cần thực hiện

Hệ thống HTTP được cài đặt gồm 1 máy chủ NGINX chạy trên nền Ubuntu 14.04 LTS và 1 máy khách chạy trình duyệt đơn giản được viết bằng ngôn ngữ C sử dụng thư viện nghttp2. Thuật toán quyết định bitrates được viết bằng Java và liên kết với trình duyệt ở máy khách thông qua chương trình Expect. Máy chủ và máy khách được kết nối với nhau thông qua mạng LAN

Gói thư viện nghttp2 là sự thực thi của giao thức HTTP và thuật toán nén header của nó bằng ngôn ngữ C. Hệ thống đã được hướng dẫn sử dụng tại [3]

a) Máy chủ

Cài đặt các gói thư viện Jasson, Python

Cài đặt SPDY

Cài đặt NGHTTP/2

Cài đặt server nginx

Tạo thư mục chứa các segment của video và file index.php

Khởi chạy server

b) Máy khách

Cài đặt các gói thư viện Jasson, Python

Cài đặt SPDY

Cài đặt NGHTTP/2

Cài đặt dummynet để config băng thông theo yêu cầu

Cài đặt expect

Test thử kết nối với server

Chạy thuật toán

2.2.4.2 Chi tiết các bước cài đặt

a) Cài đặt HTPP/2.0 server

i. Cài thư viện

sudo apt-get install make binutils autoconf automake autotools-dev libtool pkg-config zlib1g-dev libcunit1-dev libssl-dev libxml2-dev libev-dev libevent-dev libjansson-dev libjemalloc-dev python3.4-dev g++ g++-mingw-w64-i686 git python3-setuptools

sudo easy_install3 pip

ii. Cài SPDY

mkdir ~/HTTP2_src

git clone https://github.com/tatsuhiro-t/spdylay.git ~/HTTP2_src/spdylay

iii. Tìm địa chỉ SPDY

sudo updatedb

locate libspdylay.so.7

Với i386:

sudo ln -s /usr/local/lib/libspdylay.so.7 /lib/i386-linux-gnu/libspdylay.so.7

sudo ln -s /usr/local/lib/libspdylay.so.7.2.0 /lib/i386-linux-gnu/libspdylay.so.7.2.0

Với x86_64

sudo ln -s /usr/local/lib/libspdylay.so.7 /lib/x86_64-linux-gnu/libspdylay.so.7

sudo ln -s /usr/local/lib/libspdylay.so.7.2.0 /lib/x86_64-linux-gnu/libspdylay.so.7.2.0

sudo ldconfig

iv. Cài đặt NGHTTP2

git clone https://github.com/tatsuhiro-t/nghttp2.git ~/HTTP2_src/nghttp2

cd ~/HTTP2_src/nghttp2

autoreconf -i

automake

autoconf

./configure PYTHON=/usr/bin/python3

make

sudo make install

sudo updatedb

locate libnghttp2.so.14

Với i386:

sudo ln -s /usr/local/lib/libnghttp2.so.14 /lib/i386-linux-gnu/libnghttp2.so.14

sudo ln -s /usr/local/lib/libnghttp2.so.14.0.2 /lib/i386-linux-gnu/libnghttp2.so.14.0.2

Với x86_64:

sudo ln -s /usr/local/lib/libnghttp2.so.14 /lib/x86_64-linux-gnu/libnghttp2.so.14

sudo ln -s /usr/local/lib/libnghttp2.so.14.0.2 /lib/x86_64-linux-gnu/libnghttp2.so.14.0.2

sudo ldconfig

v. Test nghttp server

nghttp --version

nghttpx --version

nghttpd --version

h2load --version

Sử dụng 2 file: /Server_conf/certs/self.crt và /Server_conf/certs/self.key

sudo nghttpx -f*,443 -blocalhost,444 ./self.key ./self.crt

vi. Cài dặt NGINX server

sudo apt-get install nginx

Configure NGINX server trong các thư mục /etc/nginx/sites-available/ va /etc/nginx/sites-enabled

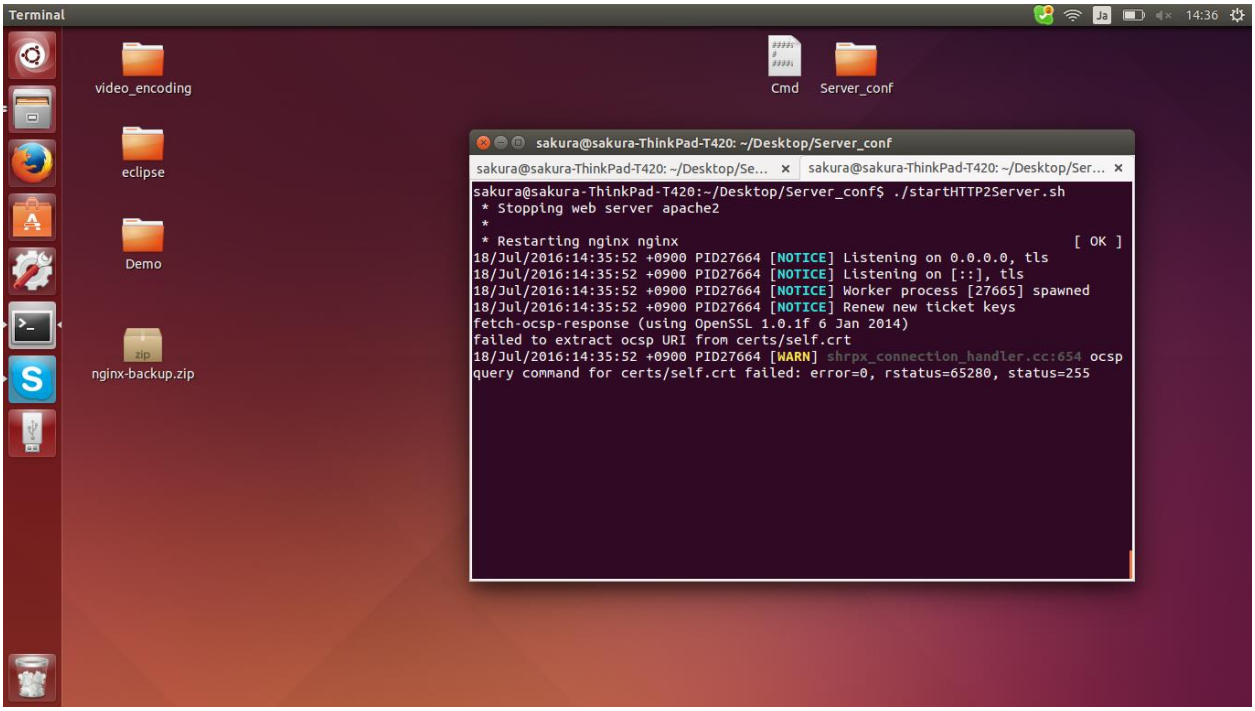
Copy các file video segment vao trong /usr/share/nginx/html/video, đổi quyền truy cập cho các file này. Cụ thể:

sudo chmod -R 755 *

vii. Cài php5-fpm

sudo apt-get install php5-fpm

viii. Chạy startHTTP2Server.sh



Hình 2.8 Kết quả cài đặt Server HTTP/2.0 thành công

Nếu xảy ra lỗi, xem log file o \var\log\nginx\error.log

b) Cài đặt HTTP/2.0 client

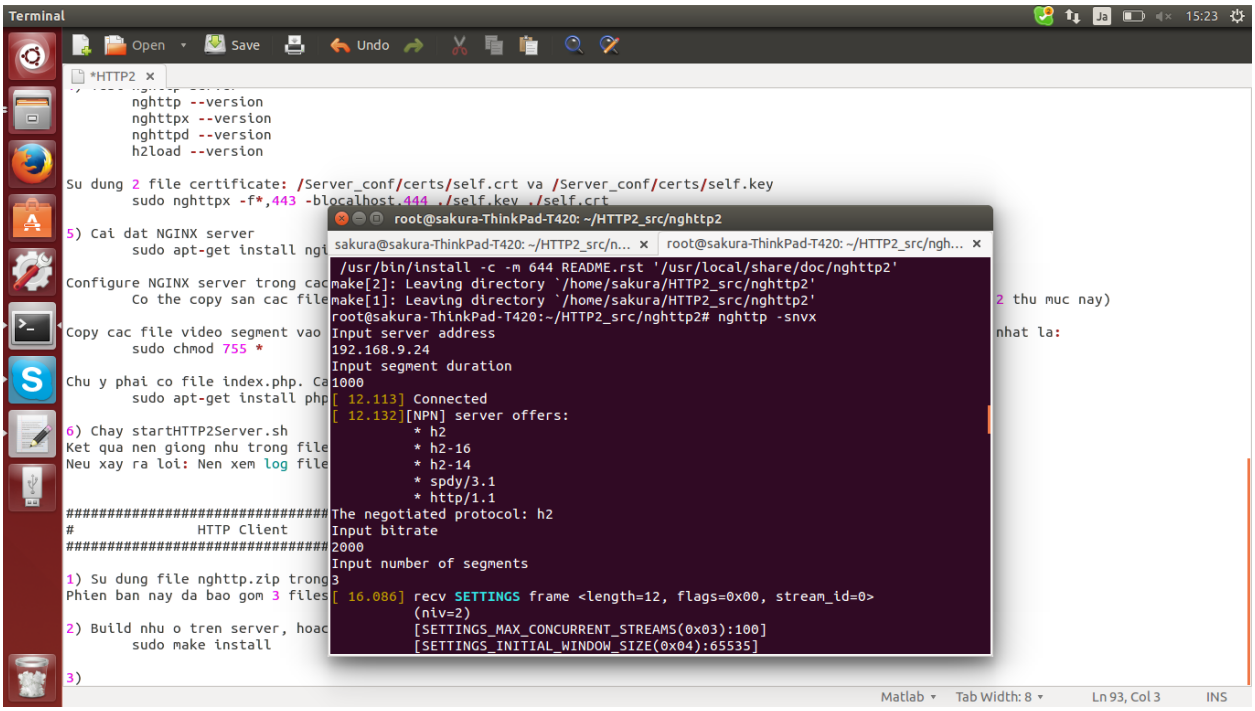
i. Thực hiện từ bước i. đến bước v. trong mục 2.2.2.1 Cài đặt HTTP/2.0 server

ii. Kiểm tra kết nối với server

nghttp -snvx

Nhập server ip address, segment duration, bitrate và số lượng pushed segments

Kết quả như trong ảnh



Hình 2.9 Kết quả cài đặt thành công NGHTTP/2.0 client

c) Cài đặt Java client

i. Cài java 1.8

Download tại [4]

ii. Cài đặt expect là chương trình giao tiếp giữa java client và nghttp client

sudo apt-get install expect

iii. Cài dummynet và kernel 3.13

Link hướng dẫn cài dummynet [5]

iv. Sửa lỗi throughput quá lớn

Vào terminal gõ 2 lệnh

sudo ipfw add pipe 3 ip from any to any

sudo ipfw pipe 3 config bw 3575Kbit/s delay 15ms

2.3 Nhận xét, đề xuất

2.3.1 Ưu điểm

- Có tinh thần tự giác cao
- Có khả năng độc lập nghiên cứu
- Có kỹ năng làm slide, thuyết trình
- Có tinh thần trách nhiệm với công việc được giao
- Cần cù, chăm chỉ

2.3.2 Nhược điểm

- Chưa có tinh thần cầu toàn

2.3.3 Đề xuất

- Thời gian thực tập ngắn nên áp lực công việc khá lớn.

3. KẾT LUẬN

Qua thời gian 2 tháng thực tập tại Lab, em đã học hỏi thêm được rất nhiều kiến thức bổ ích về hệ thống HTTP nói chung và hệ thống HTTP/2.0 nói riêng. Trong thời gian bùng nổ của Internet hiện nay thì khái niệm HTTP nghe rất quen thuộc nhưng thực sự chưa có nhiều người hiểu được cấu trúc và cách thức hoạt động của nó.

Em nghĩ những kiến thức mà mình học hỏi được qua đợt thực tập hè sẽ vô cùng hữu ích cho quá trình học tập và nghiên cứu của mình sau này.

Một lần nữa em xin được gửi lời cảm ơn chân thành và sâu sắc nhất đến PGS.TS Phạm Ngọc Nam và Th.S Nguyễn Kim Thoa đã giúp em hoàn thành tốt đợt thực tập lần này!

4. PHỤ LỤC

5. TÀI LIỆU THAM KHẢO

[1] http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-n/mobile-white-paper-c11-520862.html truy cập 13/07/2016

[2] https://en.wikipedia.org/wiki/List_of_HTTP_header_fields truy cập: 03/08/2016

[3] Hypertext Transfer Protocol Version 2 (HTTP/2) *M. Belshe, M. Thomson, May* 2015

[4] http://www.oracle.com/technetwork/java/downloads/jdk8-downloads-133151 truy cập: 05/08/2016

[5] http://hiephv.blogspot.com/2012/08/cai-at-dummynet-cho-linux.html truy cập: 05/08/2016