THUẬT TOÁN ƯỚC LƯỢNG BĂNG THÔNG KẾT HỢP DỰA TRÊN BĂNG THÔNG PHÂN ĐOẠN CUỐI CÙNG VÀ LÀM MỊN BĂNG THÔNG

THROUGHPUT ESTIMATION BASED ON LAST SEGMENT AND SMOOTH THROUGHPUT

Giáp Văn Dương [1,*]

TÓM TẮT

Truyền video trên giao thức HTTP là công nghệ đang và sẽ được ứng dụng rộng rãi trong các ứng dụng xem video trực tuyến trên các thiết bị điện tử. Tuy nhiên, một trong những yêu cầu truyền video là đảm bảo chất lượng dịch vụ QoS tốt nhất có thể cho người dùng bằng việc điều chỉnh các thông số của video kịp thời theo sự biến đổi của băng thông đường truyền. Vì vậy, ước lượng băng thông đường truyền nhằm nâng cao chất lượng dịch vụ là bài toán quan trọng đang được nghiên cứu trên khắp thế giới. Đã có nhiều nghiên cứu về ước lượng băng thông, đó là băng thông phân đoạn cuối cùng và làm mịn băng thông, nhưng mỗi thuật toán đều có ưu và nhược điểm riêng. Bài báo này sẽ đưa ra ý tưởng xây dựng một thuật toán mới có sự kết hợp ưu điểm của hai thuật toán trên để cho ra một bộ đệm ổn định và tốc độ bit video ít thay đổi. Kết quả mô phỏng cho minh chứng về hiệu quả của thuật toán mới nhằm nâng cao chất lượng dịch vụ.

Từ khóa: *Băng thông phân đoạn cuối cùng, làm mịn băng thông, băng thông kết hợp.*

ABSTRACT

Video transmission over HTTP protocol is being and will be widely used in applications of online video streaming in electronic devices. However, one of the video streaming requirements is guaranteeing the best QoS for users by adjusting the parameters of video timely according to the variation of throughput. Therefore, the problem of throughput estimation for improving the quality of service is an important problem which has been studied around the world. There have been many studies on the throughput estimation, which are Last segment throughput and Smooth throughput, but each algorithm has its own advantages and disadvantages. This paper will give the idea of building a new algorithm that combines the advantages of the two over algorithms to achieve a buffer stable and less volatile video bitrate. Simulation results demonstrate the effectiveness of the new algorithm to improve the quality of services.

Keywords: *Last Segment Throughput, Smooth Throughput, Combined Throughput.*

[1]Trường Đại học Kinh tế Kỹ thuật Công nghiệp

[*]Email: gvduong@uneti.edu.vn

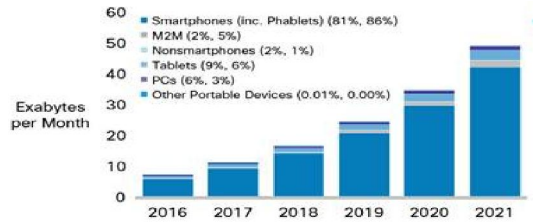
Ngày nhận bài: 26/03/2017

Ngày nhận bài sửa sau phản biện: 15/05/2017

Ngày chấp nhận đăng: 26/02/2018

1. ĐẶT VẤN ĐỀ

Những nghiên cứu gần đây [1] đã chỉ ra rằng số lượng thiết bị sử dụng các dịch vụ xem truyền hình trực tuyến đang ngày một đa dạng. Trong đó, dữ liệu cho dịch vụ xem truyền hình trực tuyến, truyền hình theo yêu cầu trở thành dịch vụ được sử dụng nhiều lưu lượng mạng nhất. Hình 1 mô tả dự đoán lưu lượng sử dụng mạng thông qua các thiết bị trong một vài năm tới. Dự đoán cho thấy lưu lượng kết nối mạng trên các thiết bị điện thoại di động tăng một cách đáng kể, từ 81% trong năm 2015 tới khoảng 86% tổng lưu lượng sử dụng mạng trong năm 2021. Dữ liệu cho dịch vụ xem truyền hình trực tuyến, truyền hình theo yêu cầu sẽ sử dụng nhiều lưu lượng mạng nhất trong các ứng dụng trên thiết bị di động. Hình 2 chỉ ra lưu lượng video ước tính tăng nhanh trong vài năm tới, với khoảng 60% trong năm 2015 lên 78% trong năm 2021 và chỉ 22% cho tất cả các dịch vụ còn lại.

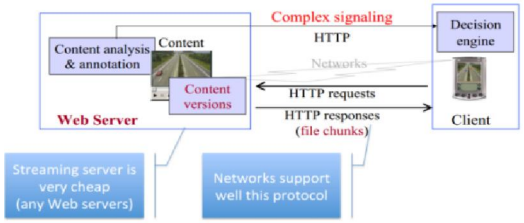


Hình 1. Lưu lượng sử dụng mạng dự đoán trên các thiết bị khác nhau

Hình 2. Lưu lượng các dịch vụ khác nhau được sử dụng trên điện thoại thông minh

Hệ thống HTTP Streaming là hệ thống mới được sử dụng trong kỹ thuật truyền video. Hình 3 chỉ ra máy chủ HTTP Streaming [3, 5] khối công cụ đưa ra quyết định được chuyển về máy khách nên sẽ giảm được khối lượng công việc rất lớn cho máy chủ. Trong máy chủ HTTP có khối phân

tích phiên bản, do nó có chứa các tập tin cùng một nội dung nhưng với nhiều phiên bản chất lượng khác nhau. Đối với đường truyền chậm nó sẵn sàng hỗ trợ cho truyền đi những phiên bản chất lượng thấp để máy khách vẫn có thể xem được. Nếu đường truyền tốt thì chất lượng phiên bản sẽ cao hơn. Tín hiệu truyền trên giao thức HTTP là tín hiệu phức tạp và mạng hỗ trợ tốt giao thức này.



Hình 3. Hệ thống HTTP Streaming

Muốn có được chất lượng hình ảnh tốt nhất cho người dùng thì ước lượng băng thông là bài toán quan trọng đang được nghiên cứu trên khắp thế giới.

Nội dung bài báo phân tích hai thuật toán ước lượng băng thông đường truyền mới hiện nay đó là Last segment throughput [3] và Smooth throughput [4]. Từ đó, đề xuất một thuật toán mới dựa trên ưu điểm của hai thuật toán trên nhằm cho ra một bộ đệm ổn định và tốc độ bit video ít thay đổi đối với những biến động của băng thông đường truyền.

2. KIẾN THỨC CƠ BẢN

2.1. Thuật toán ước lượng băng thông dựa trên phân đoạn cuối cùng

Các cơ chế ước lượng băng thông để thích ứng với điều kiện mạng được đề xuất trong [3] thực hiện theo ba yêu cầu:

- Phát lại thì không được dừng lại (tức là bộ đệm tràn dưới nên tránh).

- Sử dụng tối ưu tài nguyên mạng, trong khi có thể chọn mức tốc độ bit cao nhất đáp ứng một yêu cầu.

- Chuyển sang mức chất lượng phù hợp cần được thực hiện càng nhanh càng tốt.

Đoạn mã sử dụng cho thuật toán ước lượng băng thông dựa trên phân đoạn cuối cùng

```
privatevoid aggressiveAlgorithm()
 { if (currentSegment == 0)
     estimatedSegmentThrps[0] = 0;
   else estimatedSegmentThrps[currentSegment] =
segmentThrps[currentSegment-1];
  }
```

Các đặc tính của cơ chế điều khiển này như sau:

- Phân đoạn đầu tiên được tải về luôn có mức tốc độ bit thấp nhất.

- Luôn chọn phân đoạn có mức tốc độ bít thấp nhất khi bộ đệm bị trống.

- Thường xuyên thay đổi mức tốc độ bit của phân đoạn được tải về.

Do cơ chế này chỉ dựa vào lưu lượng băng thông mạng đo được cuối cùng nên tốc độ bit thường xuyên thay đổi nếu đường truyền mạng không ổn định. Khi lưu lượng băng thông mạng đo được cuối cùng cao hơn mức tốc độ bit của luồng hiện tại thì phân đoạn tiếp theo được tải về sẽ có tốc độ bit tăng lên. Ngược lại, nếu lưu lượng băng thông mạng đo được lần cuối nhỏ hơn tốc độ bit của luồng hiện tại thì phân đoạn tiếp theo được tải về sẽ có tốc độ bit giảm xuống.

Lợi thế của thuật toán này là việc sử dụng tốt nhất băng thông có sẵn. Nhưng tốc độ bit của mỗi phân đoạn thay đổi quá nhiều, làm cho chất lượng hình ảnh trên máy khách liên tục bị thay đổi nếu điều kiện mạng thay đổi.

2.2. Thuật toán ước lượng dựa trên làm mịn băng thông

Đặc điểm của thuật toán này [4] là lặp đi lặp lại và cố gắng để suy ra tốc độ bit có thể đạt được từ các tốc độ bit đo được trong các lần lặp trước đó.

Việc ước lượng được thực hiện như sau: Tại lần lặp *i*, để tính toán tốc độ bit hiện thời B_i, khách hàng nhân số byte nhận với tám và chia số này bởi thời gian trôi qua.

$$B_i = Bytes*8 / elapsed \quad (1)$$

Tốc độ bit hiện thời này được lọc thông thấp để xác định một ước lượng đáng tin cậy.

$$avg_{i+1} = (1-\alpha).avg_i + \alpha.B_i \quad (2)$$

Trong đó, B_i là tốc độ bit đo được, avg_i là tốc độ bit trung bình cho lần lặp hiện tại và α là trọng số tốc độ đo được cho các bit hiện tại.

Ngoài tốc độ bit trung bình, thuật toán còn ước lượng tốc độ bit phương sai.

$$\Delta_i = |B_i - avg_i| \quad (3)$$

$$var_{i+1} = (1-\beta).var_i + \beta.\Delta_i \quad (4)$$

Trường hợp Δ_i là sự khác biệt giữa tốc độ bit đo được và tốc độ bit trung bình cho lần lặp hiện thời, var_i là phương sai tính toán cho các lần lặp hiện tại và β là trọng số cho các phương sai đo lường hiện thời.

Trên mỗi lần lặp, ước lượng hiện tại (hoặc tốc độ bit đạt được) tính như sau:

$$\hat{\beta}_i = avg_i - c.var_i \quad (5)$$

Tham số *c* là một hằng số kiểm soát các hành vi của khách hàng. Nếu phương sai lớn thì khách hàng được sử dụng băng thông ít hơn nhiều so với băng thông trung bình.

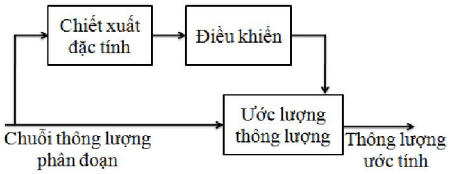
Đoạn mã sử dụng cho thuật toán ước lượng dựa trên làm mịn băng thông

```
privatevoid simpleDynamicAlgorithm()
 { if (currentSegment == 0)
     estimatedSegmentThrps[0] = 0;
   elseif(currentSegment == 1)
    {estimatedSegmentThrps[0] = segmentThrps[0];
     estimatedSegmentThrps[1] = segmentThrps[0]; }
   Else
     { doubledelta = 0.2;
estimatedSegmentThrps[currentSegment] = (1.0-delta) *
estimatedSegmentThrps[currentSegment-1] + delta *
segmentThrps[currentSegment-1];
   if (PRINT)
System.out.printf("estimatedSegmentThrp=%f\n",estima
tedSegmentThrps[currentSegment]);} }
```

Thuật toán này giải quyết được vấn đề tốc độ bit thay đổi liên tục nhưng bộ đệm lại giảm mạnh khi băng thông đột ngột xuống thấp, thậm chí có thể làm cho bộ đệm bị trống rỗng khiến màn hình bị đóng băng.

3. THUẬT TOÁN ƯỚC LƯỢNG KẾT HỢP DỰA TRÊN PHÂN ĐOẠN CUỐI CÙNG VÀ LÀM MỊN BĂNG THÔNG

Mục tiêu của cơ chế này là xác định một giá trị lưu lượng được sử dụng để làm chuẩn đánh giá sau này, có giá trị ổn định đối với những khoảng biến thiên nhỏ và có khả năng đáp ứng nhanh đối với các biến thiên lớn của lưu lượng qua mạng [5].



Hình 4. Mô hình xác định lưu lượng dùng đánh giá cho mạng

Trong hình 4, khối chiết xuất đặc tính chứa thông tin của một vài phân đoạn đã được tải về từ trước. Dựa trên những thông tin này, khối điều khiển sẽ thực hiện các hàm tính toán, kết hợp với thông tin lưu lượng mạng đo được trong lần gần nhất rồi đưa ra mức lưu lượng dùng làm tham chiếu để chọn tải phân đoạn tiếp theo phù hợp.

Lưu lượng dùng tham chiếu cho một phân đoạn thứ *i* được xác định dựa trên biểu thức sau:

$$T_e(i) = \begin{cases} (1-\delta)T_e(i-2) + \delta T_s(i-1) & i > 0 \\ T_s(i-1) & i = 1,2 \end{cases} \quad (6)$$

Trong đó: $T_e(i)$ là lưu lượng giới hạn cho phân đoạn thứ *i* được truyền tải. Được dùng làm ngưỡng xác định mức tốc độ bit cho phân đoạn sẽ được truyền tải tiếp theo về máy khách.

$T_e(0)$ ở đây không xác định, do đó đối với phân đoạn đầu tiên, hệ thống thường mặc định truyền tải về phân đoạn có mức tốc độ bit thấp nhất.

T_s là lưu lượng phân đoạn thứ *i*. Lưu lượng phân đoạn được tính dựa trên khoảng thời gian từ lúc gửi yêu cầu tới khi nhận xong thông điệp phản hồi từ máy chủ.

Tham số δ được điều chỉnh để điều khiển sự phụ thuộc của lưu lượng giới hạn và lưu lượng phân đoạn với băng thông lưu lượng hiện tại để từ đó chọn tải mức tốc độ bit tiếp theo. Từ công thức ta có thể thấy khi δ nhỏ, $T_e(i)$ phụ thuộc phần lớn vào lưu lượng được ước lượng trước đó. Do đó, chất lượng của các phân đoạn trong luồng này có tính đồng đều cao. Ngược lại, khi δ càng lớn thì lưu lượng đánh giá càng phụ thuộc vào lưu lượng mạng lần gần nhất đo được.

Trong khối chiết xuất đặc tính, một tham số xác định độ lệch lưu lượng chuẩn được xác định theo công thức như sau:

$$p = \frac{|T_s(i) - T_e(i)|}{T_e(i)} \quad (7)$$

Khi *p* có giá trị lớn, tức lưu lượng kết nối trong mạng có một sự thay đổi lớn, máy khách cần phải nhanh chóng đưa ra một phản hồi để điều khiển tăng, giảm mức lưu lượng tham chiếu, điều này tương đương với δ lớn dần tới 1. Khi mức lưu lượng kết nối không thay đổi nhiều hay giá trị nhỏ, thì giá trị lưu lượng tham chiếu cũng không thay đổi nhiều. Lúc này, giá trị *p* cũng tiến dần tới giá trị δ_{min} (khoảng 0,05 ~ 0,2). Mối liên hệ giữa *p* và δ được thể hiện theo như sau:

$$\delta = \frac{1}{1 + e^{-k(p-p_0)}} \quad (8)$$

Trong đó: giá trị của *k* và P_0 là các tham số, phụ thuộc vào các điều kiện mạng cụ thể.

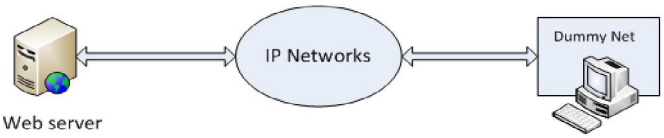
Từ biểu thức (7) và (8), kết hợp với các đoạn mã được sử dụng trong hai thuật toán trên để cho ra một đoạn mã mới được sử dụng cho thuật toán ước lượng kết hợp.

Đoạn mã sử dụng cho thuật toán ước lượng kết hợp dựa trên phân đoạn cuối cùng và làm mịn băng thông

```
privatevoid dynamicAlgorithm()
 { if(currentSegment == 0)
    estimatedSegmentThrps[0] = 0;
   elseif(currentSegment == 1)
    { estimatedSegmentThrps[0] = segmentThrps[0];
     estimatedSegmentThrps[1] = segmentThrps[0]; }
   Else
    { doubledeviation = Math.abs((segmentThrps
[currentSegment-1] - estimatedSegmentThrps[current
Segment-1]))/estimatedSegmentThrps[currentSegment-1];
     doubledelta = 1 / (1 + Math.exp(-k * (deviation -
P0)));
  estimatedSegmentThrps[currentSegment] =
(1.0-delta) * estimatedSegmentThrps[currentSegment-1]
+ delta * segmentThrps[currentSegment-1];
     if (PRINT)
System.out.printf("deviation=%f;delta=%f;estimatedSeg
mentThrp=%f\n",deviation,delta,estimatedSegmentThrp
s[currentSegment]); }
}
```

4. KẾT QUẢ MÔ PHỎNG VÀ THẢO LUẬN

4.1. Cài đặt hệ thống



Hình 5. Kiến trúc chung của hệ thống

Bảng 1. Cấu hình máy chủ và máy khách

Máy chủ HTTP	Máy chủ HTTP được cấu hình trên máy như sau: CPU: Intel® core™ i3 RAM: 2GB OS: Ubuntu 12.04 LTS HTTP server: Apache 2.2.22
Máy khách	Máy khách được chạy trên máy tính có cấu hình như sau: CPU: Intel® core™ i3-2348M với tốc độ 2,3 GHz RAM: 2GB OS: Windows 7 Ultimate, 32 bit

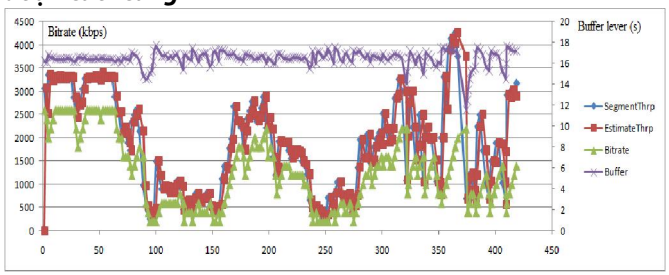
Hai phần quan trọng nhất trong hệ thống là máy chủ HTTP và máy khách có cấu hình như bảng 1. Giao tiếp giữa máy chủ và máy khách được thực hiện trên giao thức HTTP. Nội dung truyền thông được mã hóa ở nhiều mức khác nhau và phân chia thành các phân đoạn để thỏa mãn các thông số kỹ thuật của tiêu chuẩn tương ứng.

Để tạo ra các kịch bản mạng giả lập, hệ thống sử dụng một công cụ mô phỏng mạng DummyNet [2]. Để chạy chương trình trên máy khách sử dụng eclipse SDK v3.7.

Các phân đoạn video được mã hóa với 13 mức tốc độ bit từ 200 Kbps đến 2600 Kbps. Độ chênh lệch của mỗi mức là 200 Kbps. Các phân đoạn video đều có độ dài thời gian phát là 2 giây. Kịch bản giả lập băng thông của mạng ban đầu là 2,5Mbps, sau đó tăng lên 3,3Mbps, rồi tiếp tục tăng dần và biến đổi theo thời gian. Băng thông nhỏ nhất là 217Kbps, băng thông lớn nhất là 4,2Mbps, RTT = 100ms.

4.2. Kết quả mô phỏng

4.2.1. Thuật toán ước lượng băng thông dựa trên phân đoạn cuối cùng

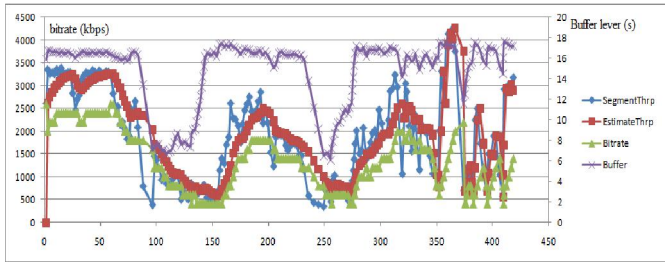


Hình 6. Kết quả đo được theo thuật toán ước lượng băng thông dựa trên phân đoạn cuối cùng

Hình 6 cho thấy đường ước lượng băng thông luôn trễ sau đường băng thông phân đoạn một chút và giống với đường băng thông phân đoạn. Lợi thế của thuật toán này là việc sử dụng tốt nhất của băng thông có sẵn nên bộ đệm luôn cao. Tuy nhiên, đường mức tốc độ bit của mỗi phân đoạn lại thay đổi quá nhiều, làm cho chất lượng hình ảnh trên máy khách liên tục bị thay đổi nếu điều kiện mạng thay đổi.

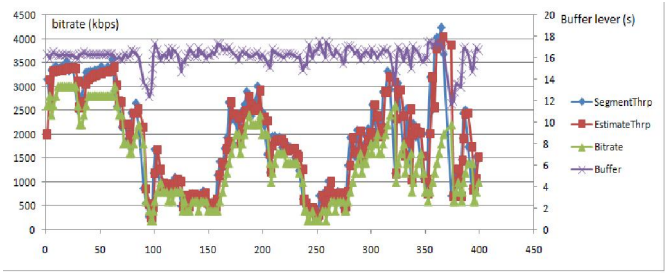
4.2.2. Thuật toán ước lượng dựa trên làm mịn băng thông

Hình 7 cho thấy thuật toán này giải quyết được vấn đề tốc độ bit thay đổi liên tục của thuật toán dựa trên phân đoạn cuối cùng nhưng bộ đệm lại giảm mạnh khi băng thông đột ngột xuống thấp. Thuật toán ước lượng này có được dựa trên việc cộng trung bình nên đường ước lượng băng thông không thể theo sát với đường băng thông phân đoạn được mà nó luôn là giá trị trung bình của những điểm trước đó.



Hình 7. Kết quả đo được theo thuật toán ước lượng dựa trên làm mịn băng thông

4.2.3. Phương pháp ước lượng kết hợp dựa trên phân đoạn cuối cùng và làm mịn băng thông



Hình 8. Kết quả đo được theo thuật toán ước lượng kết hợp dựa trên phân đoạn cuối cùng và làm mịn băng thông

Hình 8 cho thấy rằng băng thông đánh giá quá cao tại thời điểm từ 80s đến 140s và 227s đến 278s khiến cho bộ đệm cạn kiệt rất nhanh chóng, từ 17s xuống chỉ còn 6s. Trong khi đó, theo hình 8 ước lượng kết hợp sử dụng băng thông tốt nhất có sẵn nên bộ đệm luôn cao, chỉ giảm từ 17s xuống 13s do lấy ưu điểm của thuật toán ước lượng dựa trên phân đoạn cuối cùng.

Đường ước lượng băng thông luôn trễ sau đường băng thông phân đoạn một chút và giống với đường băng thông phân đoạn, do lấy ưu điểm của thuật toán ước lượng dựa trên phân đoạn cuối cùng.

Hình 6 cho thấy rằng tại thời điểm từ 125s đến 160s, 260s đến 275s, từ 365s tới 400s các mức tốc độ bit thay đổi liên tục thì với thuật toán ước lượng kết hợp đã giải quyết được vấn đề này do lấy ưu điểm của thuật toán ước lượng dựa trên làm mịn băng thông.

5. KẾT LUẬN

Bài báo này nghiên cứu việc sử dụng một thuật toán mới với tên gọi là ước lượng băng thông kết hợp dựa trên băng thông phân đoạn cuối cùng và làm mịn băng thông. Thuật toán mới do lấy ưu điểm của băng thông phân đoạn cuối cùng và làm mịn băng thông, nên nó cung cấp chất lượng tốt nhất có thể cho người dùng trong khi vẫn duy trì một bộ đệm ổn định với những thay đổi liên tục từ băng thông đường truyền. Nghiên cứu tiếp theo sẽ tiến hành thí nghiệm hệ thống đường truyền với nhiều máy chủ cùng phục vụ, kết hợp sử dụng kỹ thuật dự đoán tốc độ bit để cho ra một phương pháp chung cho các thể loại nội dung khác nhau.

TÀI LIỆU THAM KHẢO

[1]. *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update 2016 to 2021*, February 7, 2017. Available from: http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html

[2]. DummyNet project, March 2010. Available from: http://info.iet.unipi.it/~luigi/dummynet/

[3]. L. R. Romero, 2011. *"A dynamic adaptive HTTP streaming video service for Google Android"*. M.S. Thesis, Royal Institute of Technology (KTH), Stockholm, Oct 2011.

[4]. S. Gouache, G. Bichot, A. Bsila, C. Howson, 2011. *"Distributed & adaptive HTTP streaming"*. Proc. IEEE International Conference on Multimedia and Expo (ICME), Barcelona, Spain, July 2011.

[5]. T. C. Thang, Q-D Ho, J. W. Kang, A. T. Pham, 2012. *"Adaptive Streaming of Audiovisual Content using MPEG DASH"*. IEEE Trans. on Consumer Electronics, vol. 58, no. 1, pp. 78-85, Feb 2012.