Tạp chí Khoa học Trường Đại học Cần Thơ
website: sj.ctu.edu.vn

ĐIỀU KHIỂN TẮC NGHẼN TRONG GIAO THỨC TRUYỀN ĐA ĐƯỜNG CHO CÁC ỨNG DỤNG MULTIMEDIA

Lê Phong Dũ[1] và Lê Tuấn Anh[2]

[1] *Ban Phát triển Hệ thống Công nghệ Thông tin, Trường Đại học Trà Vinh*
[2] *Khoa Công nghệ Thông tin, Trường Đại học Thủ Dầu Một*

Thông tin chung:
Ngày nhận: 03/09/2013
Ngày chấp nhận: 21/10/2013

Title:
Congestion control in multipath protocol for multimedia applications

Từ khóa:
Thuật toán điều khiển tắc nghẽn, TFRC, Multipath TCP

Keywords:
Congestion control algorithm, TFRC, Multipath TCP

ABSTRACT

Recently, multipath transport control protocol (multipath TCP) allows spreading its data packets on several paths simultaneously. Such the multipath transfer can improve TCP throughput, can balance congestion among paths, and can provide native handover in a network. Current coupled multipath congestion control (MPTCP) was designed for back-compatibility with single-path TCP Reno and for general applications. However, data rate of MPTCP has high variance that not suitable for multimedia applications which require smooth data rate.

In this paper, we propose an extension algorithm of single path TFRC, named MPTFRC, designed from both the analytical model of TCP Reno at flow level and the technique of flappiness prevention between paths at packet level. The simulation results demonstrate that the proposed MPTFRC algorithm not only satisfies the three design goals of multipath congestion control algorithm but also provides data rate smoother than that of MPTCP while preservingfair sharing to the existing TCP Reno and MPTCP flows.

TÓM TẮT

Gần đây, giao thức truyền đa đường TCP cho phép truyền tải các gói dữ liệu của nó trên nhiều đường (path) đồng thời. Việc truyền dữ liệu trên đa đường như thế sẽ cải thiện thông lượng truyền, có thể cân bằng tắc nghẽn trong các đường và cung cấp khả năng chuyển vùng tự nhiên trong mạng. Thuật toán điều khiển tắc nghẽn đa đường phối hợp (MPTCP) hiện nay được thiết kế để tương thích với thuật toán điều khiển tắc nghẽn đơn đường TCP Reno và các ứng dụng thông thường. Tuy nhiên, tốc độ truyền dữ liệu của MPTCP biến thiên lớn, không phù hợp cho các ứng dụng multimedia mà chúng yêu cầu tốc độ dữ liệu mượt.

Trong báo cáo này, chúng tôi đề xuất một thuật toán mở rộng của TFRC đơn đường, được đặt tên là MPTFRC, được thiết kế xuất phát từ mô hình phân tích của TCP Reno gốc tại mức luồng và kỹ thuật tránh đánh võng giữa các đường ở mức gói. Các kết quả mô phỏng đã chứng tỏ rằng thuật toán đề xuất MPTFRC không những đảm bảo ba tiêu chí thiết kế thuật toán đa đường mà còn cung cấp tốc độ dữ liệu mượt hơn MPTCP trong khi vẫn duy trì được sự chia sẻ công bằng với các luồng TCP Reno và MPTCP hiện có.

1 GIỚI THIỆU

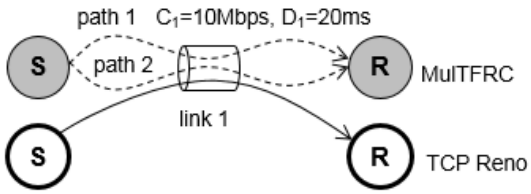
Kiến trúc multipath TCP [8] là giao thức mở rộng các đặc điểm từ giao thức TCP, cho phép một kết nối phân chia thành nhiều đường (path) và dữ liệu được truyền trên các đường đồng thời. Multipath TCP hoạt động giống như TCP và mở rộng thêm các API nhằm cung cấp thêm chức năng điều khiển cho các ứng dụng multipath TCP.

Một kết nối multipath TCP là một tập hợp của nhiều subflow mà mỗi subflow điều khiển một đường, sử dụng cửa sổ điều khiển tắc nghẽn để điều chỉnh tốc độ trên mỗi đường. Gần đây, thuật toán điều khiển tắc nghẽn MPTCP [5] đã được IETF phê duyệt hoạt động trên kiến trúc multipath TCP [8]. MPTCP thực hiện điều khiển tốc độ gửi dữ liệu phối hợp giữa các đường đảm bảo ba tiêu chí: *(i) nâng cao thông lượng truyền, (ii) đảm bảo tính công bằng đối với các luồng TCP hiện có và (iii) có khả năng cân bằng tắc nghẽn*: chuyển dữ liệu từ đường có tắc nghẽn nhiều sang đường có tắc nghẽn ít hơn. Vì MPTCP được thiết kế cho đa ứng dụng, nên tốc độ truyền dữ liệu của MPTCP biến thiên lớn, không phù hợp cho các ứng dụng multimedia mà ở đó chúng yêu cầu *tốc độ dữ liệu truyền mượt (chúng tôi gọi yêu cầu này tiêu chí (iv)* cho các ứng dụng multimedia).

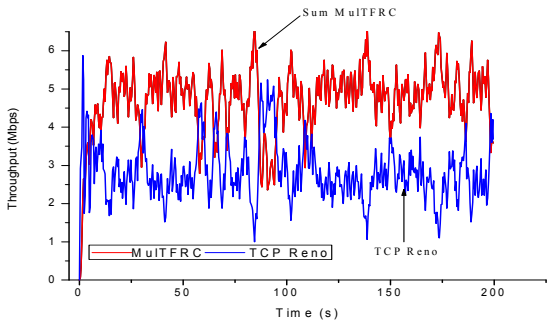
Đã có nhiều nghiên cứu điều khiển tắc nghẽn cho các ứng dụng multimedia trong giao thức đa đường, cụ thể: MultiTCP [12] là một điều khiển định hướng bên nhận (receiver-driven control), nó tăng tốc độ như TCP đơn đường và giảm tốc độ tỉ lệ nghịch với bình phương của số lượng đường được sử dụng. Ngược lại, DMP scheme [10] là một điều khiển định hướng bên gửi (sender-driven control), nó không quan tâm các luồng con có chia sẻ đường truyền cổ chai hay không, cả hai giải pháp trên không quan tâm đến tốc độ mượt và cân bằng tắc nghẽn. Ngoài ra, TCP-ROME [11] giảm sự biến động của tốc độ tức thời bằng cách sử dụng điều khiển dựa trên tốc độ như trong TFRC [3], đầu nhận trong TCP-ROME điều chỉnh tốc độ của luồng con tùy theo tốc độ video yêu cầu và tỉ lệ thông lượng của mỗi luồng, không quan tâm có đường truyền cổ chai.

Gần đây, MulTFRC [1] đã được đề xuất. Nó là phiên bản mở rộng của TFRC [3] vốn là giao thức đơn đường và được thiết kế cho các ứng dụng multimedia. Vì MulTFRC được thiết kế với giả định rằng các đường có cùng round-trip-time (RTT). Trên thực tế, giả định này hiếm xảy ra, cho nên MulTFRC không đảm bảo thỏa mãn tiêu chí trong giao thức đa đường, do nó không có khả năng bù khác biệt RTT, mặc dù nó có khả năng cung cấp tốc độ truyền dữ liệu mượt.

Để minh họa sự chia sẻ không công bằng giữa hai luồng của MulTFRC với một luồng đơn TCP Reno tại một đường truyền nút cổ chai. Mô hình mạng mô phỏng như Hình 1.



Hình 1: Mô hình mạng đánh giá chia sẻ công bằng của MulTFRC



Hình 2: Thông lượng chia sẻ công bằng giữa MulTFRC và TCP Reno

Hình 2 cho thấy rằng, tổng thông lượng của MulTFRC gấp đôi so với thông lượng của TCP Reno. Khi chia sẻ công bằng với các luồng TCP đơn đường. Một kết nối trong giao thức đa đường chia sẻ công bằng với kết nối trong giao thức đơn đường khi nhiều luồng con của giao thức đa đường cạnh tranh với nhau. Do đó, thuật toán MulTFRC không thỏa mãn tiêu chí thứ hai về chia sẻ công bằng tại đường truyền cổ chai.

Trong bài báo này, chúng tôi đề xuất một thuật toán mở rộng của TFRC [3] đơn đường, được đặt tên là MPTFRC, xuất phát từ mô hình phân tích của TFRC gốc [3] và kết hợp với sự cải tiến để tránh đánh võng giữa các đường ở mức gói. Các kết quả mô phỏng đã chứng tỏ rằng thuật toán đề xuất MPTFRC không những đảm bảo ba tiêu chí trên mà còn cung cấp tốc độ dữ liệu mượt hơn MPTCP trong khi vẫn duy trì được sự chia sẻ công bằng với TCP chuẩn, TFRC và MPTCP hiện có.

Phần còn lại của bài báo chúng tôi tổ chức như sau: Chúng tôi mô tả chi tiết thiết kế thuật toán MPTFRC trong Phần II. Chúng tôi đánh giá hiệu

năng của thuật toán trong Phần III. Và kết luận sẽ trình bày trong Phần IV.

2 THIẾT KẾ GIAO THỨC MPTFRC

Trong phần này chúng tôi trình bày giao thức đề xuất MPTFRC mở rộng từ giao thức TFRC [3]. Trước tiên, chúng tôi tiến hành mở rộng thuật toán điều khiển tắc nghẽn TFRC đơn đường thành đa đường có sự phối hợp giữa các đường. Sau đó, kết hợp với điều chỉnh tránh đánh võng để đảm bảo thuật toán hoạt động hiệu quả trong các điều kiện mạng khác nhau. Hiện tượng đánh võng sẽ không thể nhìn thấy trong lúc thiết kế đa đường (nghĩa là quá trình thiết kế chỉ thực hiện ở mức luồng) mà chỉ có thể thấy được ở mức gói. Giả sử có hai đường có cùng điều kiện mạng (cùng RTT, băng thông và cùng mức độ nghẽn mạng), sự đánh võng được mô tả là: dữ liệu gần như chỉ truyền trên một đường trong khoảng thời gian ngẫu nhiên và sau đó chỉ truyền trên đường còn lại và lặp lại như thế [2].

Để cải tiến công thức truyền tốc độ từ thuật toán TFRC gốc [3], việc chứng minh công thức dựa vào giao thức TCP đa đường. Xuất phát từ công thức điều khiển tốc độ trong TFRC gốc được đề xuất trong [3].

$$B(p) \approx \frac{1}{RTT\sqrt{\frac{2pb}{3}} + T_0 \min\left(1, 3\sqrt{\frac{3pb}{8}}\right) p\left(1+32p^2\right)},$$

trong đó, công thức tính thông lượng $B(p)$ được tính bằng gói tin/giây, RTT là round-trip-time, tỉ lệ mất gói tin là p, b là số gói tin được xác nhận bởi một ACK ($b = 1$), T_0 thời gian truyền lại hết thời gian chờ (RTO). Giá trị $T_0 = 4 \times RTT$. Chúng tôi giả sử rằng R là tập hợp tất cả các đường, $r \in R$ là một đường cụ thể. Trọng số đề xuất điều chỉnh phối hợp tốc độ truyền dữ liệu đưa vào giao thức MPTFRC ở trạng thái ổn định là

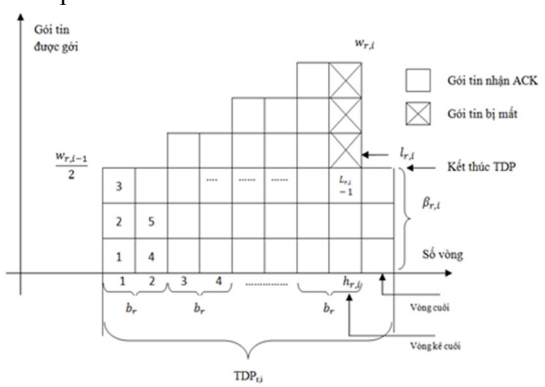
$$\alpha_r = \left(\frac{\hat{x}_r}{\sum \hat{x}_r}\right)^2,$$

Trong đó $\hat{x}_r$ là tốc độ truyền dữ liệu ở trạng thái ổn định, gọi b_r là số gói tin được xác nhận bởi một ACK trên đường r.

Trên mỗi đường r, giả sử w_r là kích thước cửa sổ tránh tắc nghẽn. Cơ chế điều khiển tránh tắc nghẽn của các đường r được thực hiện theo các vòng truyền tin, một vòng bắt đầu bằng việc gửi w_r gói tin trong cửa sổ tắc nghẽn và trong quá trình các gói tin được gửi đi sẽ không có bất kỳ gói tin nào được gửi, trước khi ACK đầu tiên được gửi về. Khi ACK đầu tiên được nhận, đánh dấu vòng đầu tiên kết thúc và chuyển sang vòng thứ 2. Kích thước cửa sổ ở vòng thứ 2 $w_r^{'} = w_r + \frac{a_r}{b_r}$, kích thước cửa sổ sẽ tăng tuyến tính với chu kỳ $\frac{a_r}{b_r}$ trên một vòng hay RTT_r trong giai đoạn tránh tắc nghẽn và không mất gói tin. Khoảng thời gian của một vòng bằng với RTT_r và gói tin bị mất trong một vòng thì độc lập với các vòng khác.

2.1 Phát hiện mất gói tin theo cơ chế dupACK

Xét trên mỗi đường r, chúng tôi định nghĩa thông lượng $B_r(p)$ được xác định số gói tin được truyền đi trong khoảng thời gian nhất định và TDP_r là khoảng giữa hai lần xảy ra mất gói tin trong cơ chế dupACK.



Hình 3: Cơ chế phát hiện mất gói tin dạng dupACK

Trong một TDP_r thứ i, thì $Y_{r,i}$ là số gói tin được gửi đi, $A_{r,i}$ là khoảng thời gian và $w_{r,i}$ là kích thước cửa sổ. Mối quan hệ giữa thông lượng và xác suất mất gói tin được tính như sau:

$$B_r(p) = \frac{E[Y_r]}{E[A_r]}$$

Giả sử hệ số $l_{r,i}$ là gói tin bị mất đầu tiên trong TDP_r thứ i tại vòng $h_{r,i}$. Tổng $Y_{r,i} = l_{r,i} + w_{r,i} - 1$ gói tin sẽ được thêm vào vòng $h_{r,i} + 1$ theo công thức:

$$E\left[Y_r\right] = E\left[L_r\right] + E\left[W_r\right] - 1$$

Do $l_{r,i}$ trong vòng độc lập với các gói tin trong các vòng khác, vì thế $\{L_{r,i}\}_i$ là một chuỗi các biến ngẫu nhiên độc lập và phân phối giống nhau. Xác suất $l_{r,i} = k_r$ tức là $k_r - 1$ gói tin đã gửi thành công trước khi gói tin bị mất xảy ra.

$P\left[l_{r,i} = k_r\right] = (1-p_r)^{k_r-1} p_r, k_r = 1,2,3,\ldots$ tức là $l_{r,i}$ được xác định là:

$$E\left[L_{r,i}\right] = \sum_{k_r=1}^{\infty} (1-p_r)^{k_r-1} p_r k_r = \frac{1}{p_r}$$

Đặt $t_{r,ij}$ là khoảng thời gian của vòng thứ j trong TDP$_r$ thứ i. Khi đó, thời gian của một TDP$_r$ thứ i là $A_{r,i} = \sum_{j=1}^{k_{r,i}+1} t_{r,ij}$. $t_{r,ij}$ là biến ngẫu nhiên và độc lập với kích thước cửa sổ, vì thế nó độc lập với vòng thứ j.

$$E\left[A_r\right] = \left(E\left[H_r\right] + 1\right) E\left[t_r\right],$$

trong đó biểu diễn $RTT_r = E[t_r]$.

Cuối cùng xác định được các công thức sau:

$$E\left[H_r\right] = \frac{2\alpha_r + b_r}{6\alpha_r} + \sqrt{\left(\frac{2\alpha_r + b_r}{6b_r}\right)^2 + \frac{2b_r(1-p_r)}{3\alpha_r p_r}}$$

$$E\left[A_r\right] = RTT_r\left(\frac{2\alpha_r + b_r}{6\alpha_r} + \sqrt{\left(\frac{2\alpha_r + b_r}{6b_r}\right)^2 + \frac{2b_r(1-p_r)}{3\alpha_r p_r}} + 1\right)$$

$$B_r(p) = \frac{1}{RTT_r}\sqrt{\frac{3\alpha_r}{2b_r p_r}} + O\left(\frac{1}{\sqrt{p_r}}\right)$$

2.2 Phát hiện mất gói tin theo cơ chế dupACK và Time-out

Đặt Z_r^{TO} là khoảng thời gian trong trường hợp mất gói tin dưới dạng time-out và Z_r^{TD} khoảng thời gian trong trường hợp mất gói tin dưới dạng dupACK. S_r là tổng thời gian được định nghĩa như sau:

$$S_r = Z_r^{TO} + Z_r^{TD}.$$

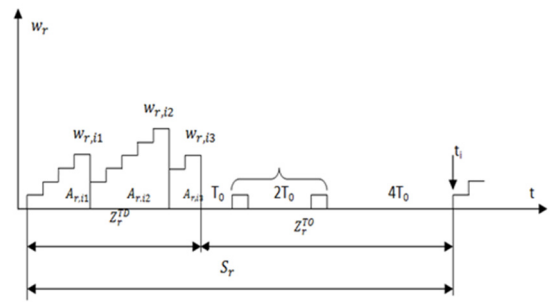
Đặt M_r là số gói tin đã truyền trong khoảng thời gian S_r, khi đó $\{S_r, M_r\}$ là một chuỗi các biến ngẫu nhiên. Thông lượng được tính như sau:

$$B_r(p) = \frac{E[M_r]}{E[S_r]}$$

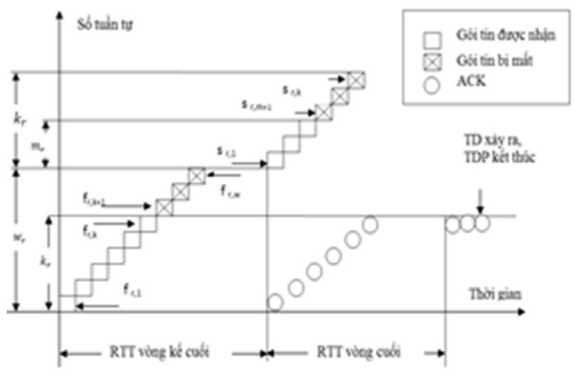
Đặt n_r là số TDP$_r$ trong khoảng thời gian Z_r^{TD}. Trong TDP$_r$ thứ i, xác định các giá trị $Y_{r,i}$ là số gói tin được gửi, $A_{r,i}$ là khoảng thời gian và $W_{r,i}$ là kích thước cửa sổ. Tương tự, R_r là số gói tin gửi trong khoảng thời gian Z_r^{TO}.

Xét trong khoảng thời gian Z_r^{TD} có n_r TDP$_r$ thì có $n_r - 1$ TDP$_r$ kết thúc dưới dạng dupACK và TDP$_r$ cuối cùng kết thúc dưới dạng time-out [9]. Khi đó $\mathbf{Y}_{r,i}$ và $\mathbf{A}_{r,i}$ không phụ thuộc vào time-out, có nghĩa là nó sử dụng lại công thức tính dạng dupACK. Thông lượng cụ thể được tính như sau:

$$B_r(p) = \frac{E\left[Y_r\right] + Q_r \times E\left[R_r\right]}{E\left[A_r\right] + Q_r \times E\left[Z_r^{TO}\right]}$$



Hình 4: Mất gói tin dạng dupACK và time-out



Hình 5: Gói tin bị mất trong vòng kế cuối

Xét các gói tin từ $\mathbf{f}_{r,1} \ldots \mathbf{f}_{r,w}$ được gửi tại vòng kế cuối, các gói $\mathbf{f}_{r,1} \ldots \mathbf{f}_{r,k}$ nhận được ACK và giả

sử gói tin $f_{r,k+1}$ là gói tin đầu tiên bị mất. Do đó, tất cả các gói tin từ $f_{r,k+1}$ trong vòng kế cuối đều bị mất. Tuy nhiên, khi các gói tin $f_{r,1} \dots f_{r,k}$ nhận được ACK thì có k_r gói tin khác $s_{r,1} \dots s_{r,k}$ được gửi trong vòng kế tiếp mà có thể được xem là vòng cuối cùng có một số gói tin bị mất, giả sử gói tin bị mất là $s_{r,m+1}$, tức là gói tin từ $s_{r,m+2} \dots s_{r,k}$ bị mất trong vòng cuối cùng [9].

Đặt m_r là số gói tin đã gửi thành công trong vòng cuối cùng được xác nhận bởi ACK cho các gói tin $f_{r,k}$. Giả sử $A(w_r, k_r)$ là xác suất mà k_r gói tin đầu tiên nhận được ACK trong một vòng của w_r gói tin thì:

$$A\left(w_r, k_r\right) = \frac{\left(1 - p_r\right)^{k_r} + p_r}{1 - \left(1 - p_r\right)^{w_r}}$$

Giả sử $C(n_r, m_r)$ là xác suất mà m_r gói tin nhận được ACK trong vòng cuối cùng (n_r là gói tin được gửi) và phần còn lại của những gói tin trong vòng. Nếu có bị mất gói tin thì

$$C\left(n_r, m_r\right) = \begin{cases} \left(1 - p_r\right)^{m_r} + p, m_r \le n_r - 1 \\ \left(1 - p_r\right)^{m_r}, m_r = n_r \end{cases}$$

Đặt $\widehat{Q_r}(w_r)$ là xác suất mất gói tin trên cửa sổ với kích thước w_r dưới dạng time-out.

Xác suất có gói tin $f_{r,k+1}$ bị mất trong vòng kế cuối và gói tin bị mất $s_{r,m+1}$ trong vòng cuối cùng và có giá trị $\min\left(1, \frac{3}{w_r}\right)$.

Tiếp theo, xem xét khi một gói tin được truyền giữa hai chuỗi time-out liên tiếp. Một chuỗi k_r dạng time-out xảy thì có $k_r - 1$ liên tiếp mất gói tin. Do đó:

$$P\left[R_r = k_r\right] = p_r^{k_r - 1}\left(1 - p_r\right)$$

R_r được tính như sau:

$$E\left[R_r\right] = \sum_{k_r=1}^{\infty} k_r p_r \left[R_r = k_r\right] = \frac{1}{1 - p_r}$$

Khoảng thời gian trung bình của chuỗi time-out bao gồm cả việc truyền lại. Thấy rằng 6 time-out đầu tiên kết thúc trong 1 chuỗi có chiều dài là $2^{i-1}T_0, i = 1 \dots 6$ với time-out trung bình theo sau có chiều dài $64T_0$ [9]. Khoảng thời gian của một chuỗi với k_r time-out kết thúc được tính như sau:

$$G_{k_r} = \begin{cases} \left(2^{k_r} - 1\right)T_0, k \le 6 \\ \left(63 + 64\left(k_r - 6\right)\right)T_0, k \ge 7 \end{cases}$$

$$E\left[Z_r^{TO}\right] = \sum_{k_r=1}^{\infty} G_{k_r} p_k \left[R_r = k_r\right]$$

$$E\left[Z_r^{TO}\right] = T_0 \frac{1 + p_r + 2p_r^2 + 4p_r^3 + 8p_r^4 + 16p_r^5 + 32p_r^6}{1 - p_r}$$

$$B_r(p) = \frac{1}{RTT_r\sqrt{\frac{2b_r p_r}{3\alpha_r}} + T_0 \min\left(1, 3\sqrt{\frac{3b_r p_r}{8\alpha_r}}\right) p_r\left(1 + 32p_r^2\right)}$$

2.3 Thuật toán MPTFRC

Trong quá trình thiết kế MPTFRC, chúng tôi nhận thấy rằng, tốc độ truyền dữ liệu của MPTFRC nếu chỉ cập nhật tăng, giảm theo công thức (1) thì xảy ra hiện tượng đánh võng. Hiện tượng này được mô tả như sau:

Giả sử rằng MPTFRC truyền dữ liệu trên hai đường độc lập có cùng băng thông, độ trễ và cùng mức độ nghẽn mạng thì sự đánh võng được mô tả là: dữ liệu gần như chỉ truyền trên một đường trong khoảng thời gian ngẫu nhiên và sau đó chỉ truyền trên đường còn lại và lặp lại như thế. Hiện tượng này cũng xuất hiện trong thiết kế MPTCP [8]. Để tránh hiện tượng này, chúng tôi sử dụng kỹ thuật được đề xuất trong [7] là thêm tham số ε_r trên đường r. Thuật toán tăng tốc độ truyền trên đường r được trình bày:

Thuật toán điều khiển tăng tốc độ trên đường r trong giao thức MPTFRC:

```
if (curr_rate_r<target_rate_r)
{
 mult=(now-last_change)/rtt_r
 if (mult > 2) mult=2;

curr_rate_r=curr_rate_r+(size_r/rtt_r
) × mult × α_r + ε_r × size_r/rtt_r ;
 }
```

```
last_change_ = now;
```

Tham số ε_r được thêm vào công thức (1) để tránh đánh võng được thực hiện như sau:

Đặt $\mathbf{RTT}_r$ và w_r là round-trip-time và kích thước cửa sổ trên đường r. Tiếp theo cần xác định đường có kích thước cửa sổ lớn nhất và đường tốt nhất. Công thức xác định được định nghĩa như sau:

$$M = \left\{ i(t) \middle| i(t) = \arg\max_{r \in R} w_r \right\}$$

$$B = \left\{ j(t) \middle| j(t) = \arg\max_{r \in R} \frac{1}{RTT_r^2 (p_r)} \right\}$$

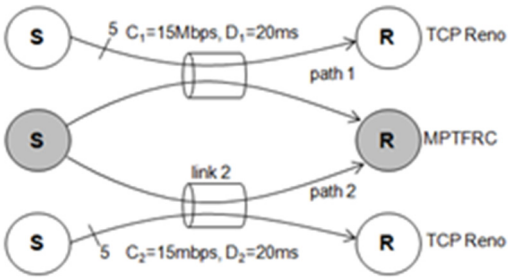
M là tập hợp những đường với kích thước cửa sổ lớn nhất, B là tập hợp những đường mà được xem xét là đường tốt nhất. Để đảm bảo không đánh võng trong quá trình truyền dữ liệu, sử dụng tham số ε_r tránh đánh võng trên từng đường như sau:

$$\varepsilon_r = \begin{cases} \dfrac{\frac{1}{|R|}}{|B \backslash M|} \text{nếu } r \in B \backslash M \neq \emptyset \\ -\dfrac{\frac{1}{|R|}}{|M|} \text{ nếu } r \in M \text{ và } B \backslash M \neq \emptyset \\ 0 \qquad \text{ngược lại} \end{cases}$$

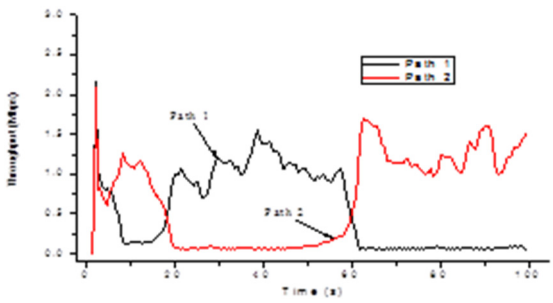
Trong đó, $B \backslash M$ là tập hợp những phần tử có trong B nhưng không có trong M, $\emptyset$ là phần tử rỗng và $|R|$ là tổng số đường truyền dữ liệu. Khi đó $\sum_{r \in R} \varepsilon_r = 0$.

Nếu đường tốt nhất có kích thước cửa sổ lớn nhất, thì $\varepsilon_r = 0$ bất kỳ với $r \in R$. Tuy nhiên, nếu đường tốt nhất có kích thước cửa sổ nhỏ, tức là nếu $B \backslash M \neq \emptyset$ thì ε_r là một số dương nếu $r \in B \backslash M$, ε_r là số âm nếu $r \in M$.

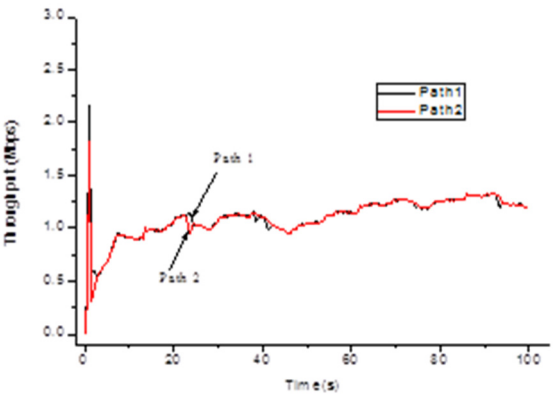
Chúng tôi sẽ mô phỏng sự đánh võng khi không có tham số ε_r và tránh đánh võng bằng tham số ε_r. Mô phỏng được thực hiện như Hình 6. Ở đó có 5 luồng TCP Reno trên các link 1 và 2 để tạo lưu lượng nền (background traffic).



Hình 6: Mô hình mạng đánh giá đánh võng



Hình 7: không sử dụng tham số ε_r



Hình 8: sử dụng tham số ε_r

Trong Hình 7 khi truyền dữ liệu chúng tôi không sử dụng tham số tránh đánh võng ε_r. Thông lượng truyền trên hai đường thay đổi. Đầu tiên thông lượng tăng nhanh trên đường truyền thứ 2 (path 2) sau đó bắt đầu giảm tốc độ, thông lượng truyền thay đổi chỉ truyền trên đường 1 (path 1) và gần như chỉ truyền trên một đường trong khoảng thời gian ngẫu nhiên, tốc độ truyền trên hai đường luân phiên thay đổi, hiện tượng đánh võng xảy ra

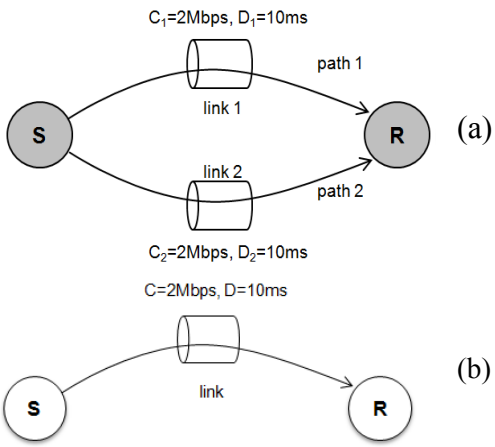
trên hai đường. Khi sử dụng tham số tránh đánh võng ε_r thì không xuất hiện sự đánh võng như trong Hình 8, ở đó tốc độ trên hai đường truyền (path 1 và path 2) gần như bằng nhau trong suốt 100 giây mô phỏng.

3 ĐÁNH GIÁ HIỆU NĂNG

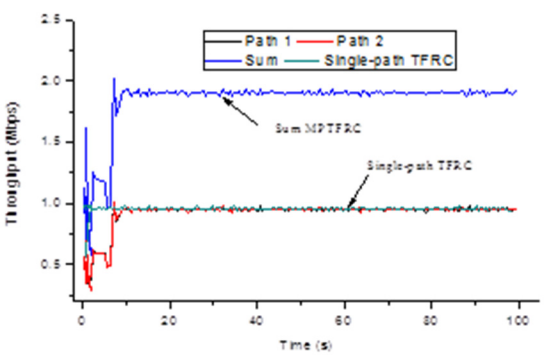
Để đánh giá thuật toán MPTFRC đã đề xuất trong phần 2, chúng tôi tiến hành mô phỏng bằng bộ công cụ mô phỏng mạng NS2-31 [6], chạy trong môi trường hệ điều hành Ubuntu 10.4. Với các tham số mô phỏng như sau: kích thước gói dữ liệu là 576 byte, sử dụng cơ chế hàng đợi active RED, tốc độ lấy mẫu sau mỗi 500 ms, kích thước bộ đệm bằng băng thông nhân với độ trễ.

a. Đánh giá theo tiêu chí tăng thông lượng

Để đánh giá tiêu chí tăng thông lượng của MPTFRC, đầu tiên chúng tôi mô phỏng hai đường MPTFRC qua hai link như Hình 9(a), sau đó chúng tôi tiếp tục mô phỏng riêng biệt một luồng TFRC qua 1 link như Hình 9(b).



Hình 9: Mô hình mạng đánh giá tiêu chí tăng thông lượng

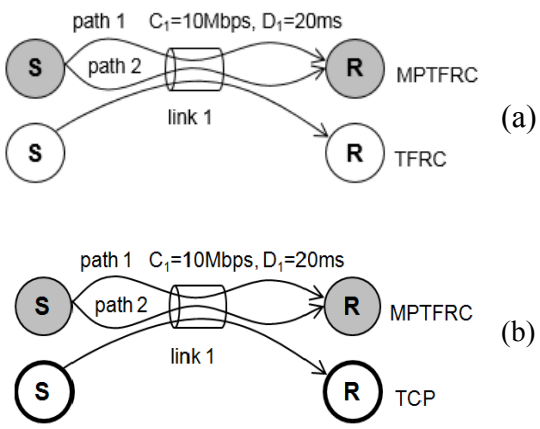


Hình 10: Tăng thông lượng của MPTFRC

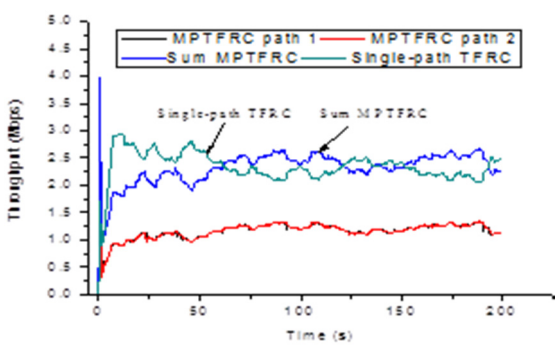
Kết quả Hình 10 thấy rằng, thông lượng tổng hai đường của MPTFRC là gần gấp đôi so với TFRC đơn đường, tăng thông lượng đạt được như trong thuật toán điều khiển tắc nghẽn giao thức truyền đa đường. Việc tăng thông lượng của MPTFRC so với TFRC đơn đường phụ thuộc vào số lượng Subflow trong một kết nối MPTFRC.

b. Đánh giá theo tiêu chí chia sẻ công bằng

Trong phần này, chúng tôi muốn đánh giá khả năng chia sẻ công bằng của MPTFRC so với các giao thức hiện có như: TCP Reno, TFRC tại một đường truyền cổ chai. Mô phỏng gồm hai mô hình mạng như Hình 11.



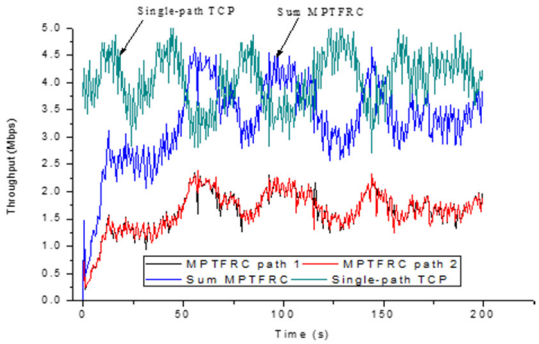
Hình 11: Mô hình mạng mô phỏng chia sẻ công bằng



Hình 12: Thông lượng chia sẻ công bằng giữa MPTFRC với TFRC

Hình 12 cho thấy, mặc dù hai đường MPTFRC cùng cạnh tranh với một luồng TFRC như Hình 11(a), nhưng hai đường MPTFRC không tích cực chiếm thông lượng của đường TFRC. Thông lượng hai đường MPTFRC chỉ bằng gần một nửa so với thông lượng trên đường TFRC. Kết quả này là do α_r đo được trung bình là: $\alpha_1 \approx \alpha_2 \approx 0.25$, nên đưa vào công thức (1) thì tốc độ trên mỗi đường sẽ

bằng 1/2 so với TFRC (vì chúng có chung RTT, băng thông và cùng mức độ tắc nghẽn).

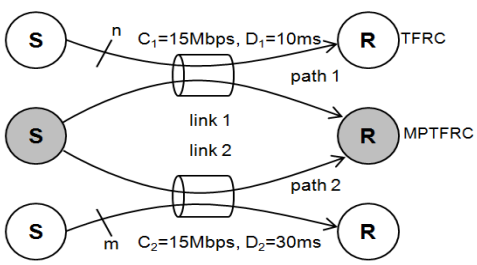


Hình 13: Thông lượng chia sẻ công bằng giữa MPTFRC với TCP Reno

Trường hợp hai đường MPTFRC cũng cạnh tranh công bằng với đường TCP như trong Hình 11(b). Tổng thông lượng của hai đường MPTFRC gần bằng với thông lượng trên đường TFRC đơn đường, cũng như TCP đơn đường (Hình 13) cùng cạnh tranh qua một link. Tức là MPTFRC chia sẻ công bằng với kết nối TFRC và TCP Reno. Bởi vì công thức TFRC được rút ra từ TCP Reno (hay thông lượng TFRC tương đương TCP Reno) nên không khó để giải thích tổng thông lượng MPTFRC hai đường truyền tương đương với thông lượng của TCP Reno trong cùng điều kiện mạng.

c. Đánh giá theo tiêu chí cân bằng tắc nghẽn

Chúng tôi đánh giá tiêu chí cân bằng tắc nghẽn bằng mô hình mạng Hình 14. Mô hình mạng này gồm một kết nối MPTFRC có 2 đường: path 1 và path 2. Trong đó path 1 cạnh tranh với đường TFRC, có n= 20 luồng qua link 1, tốc độ C_1 = 15 Mbps, độ trễ D_1 = 10 ms; path 2 cạnh tranh với một đường TFRC qua link 2 có m= 12 luồng, tốc độ C_2 = 15 Mbps, độ trễ D_2 = 30 ms. Chúng tôi tạo 20 luồng TFRC trên link 1 để mức độ tắc nghẽn cao trên đường thứ 1 (path 1).



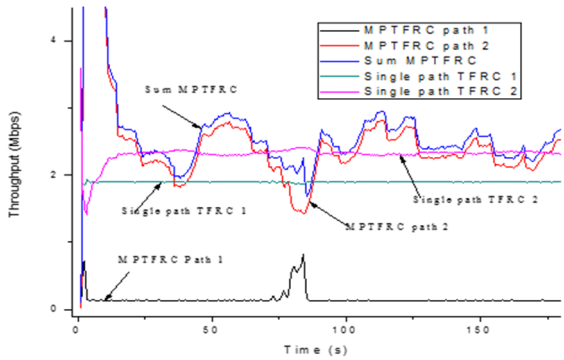
Hình 14: Mô hình mạng mô phỏng cân bằng tắc nghẽn

Tỉ lệ mất gói tin đo được trên hai đường cụ thể như sau:

Bảng 1: Tỉ lệ mất gói tin

Path 1	Path 2
$P_1 = 0.69\%$	$P_2 = 0.12\%$

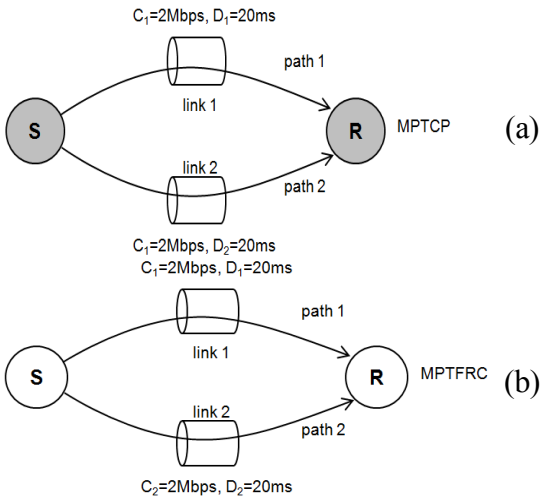
Vì tỉ lệ mất gói tin $p_1 > p_2$, tức là luồng MPTFRC chỉ tập trung truyền thông lượng vào path 2 là chính, để đạt được tiêu chí tăng thông lượng, MPTFRC gửi càng nhiều càng tốt trên path 2 trong khi tốc độ truyền dữ liệu trên path 1 gần như bằng không, nhưng tổng thông lượng của MPTFRC xấp xỉ thông lượng đường tốt nhất trên TFRC (Single-path TFRC2) trong Hình 15.



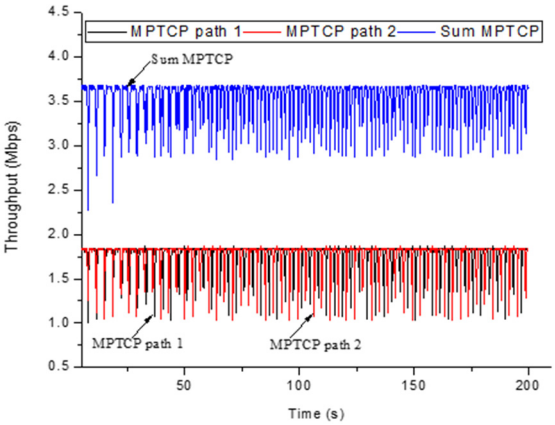
Hình 15: Thông lượng cân bằng tắc nghẽn giữa MPTFRC và TFRC

d. Đánh giá theo tiêu chí sự mượt của tốc độ truyền

Một trong những mục tiêu quan trọng trong thiết kế giao thức cho các ứng dụng multimedia là sự mượt của tốc độ truyền dữ liệu. Đánh giá tốc độ mượt của dữ liệu truyền sử dụng mô hình mạng Hình 16.



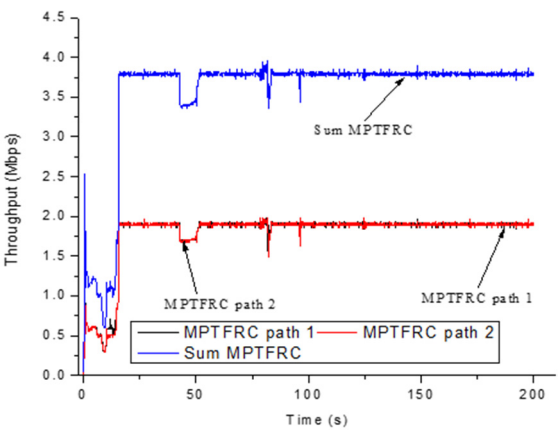
Hình 16: Mô hình mạng mô phỏng sự mượt của tốc độ truyền



Hình 17: Tốc độ dữ liệu truyền mượt của MPTCP

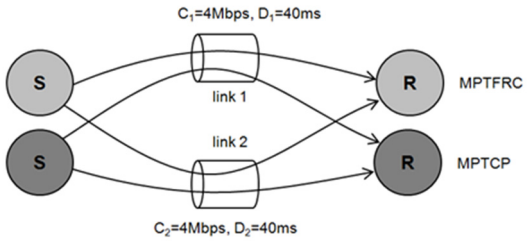
Sự mượt của tốc độ truyền dữ liệu của hai đường MPTCP có sự dao động lớn như Hình 17, tốc độ truyền dữ liệu liên tục thay đổi. Vì trong MPTCP, thuật toán điều chỉnh tốc độ trong giai đoạn giảm tương tự như TCP nên tốc độ có sự dao động lớn.

Ngược lại, sự mượt của tốc độ truyền dữ liệu của hai đường MPTFRC như Hình 18, thông lượng tăng chậm hơn so với MPTCP, tuy nhiên tốc độ truyền trong MPTFRC ít thay đổi, biên độ thay đổi nhỏ. Do MPTFRC thay đổi tốc độ dựa vào công thức (như TFRC).

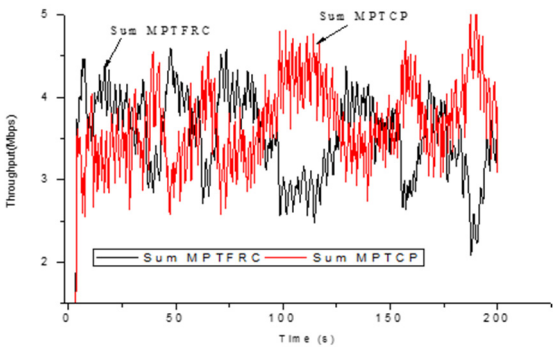


Hình 18: Tốc độ dữ liệu truyền mượt của MPTFRC

Tiếp theo, chúng tôi mô phỏng so sánh tốc độ truyền dữ liệu mượt của hai đường MPTFRC và hai đường MPTCP song song đều qua hai link: link 1 và link 2 có độ trễ D= 40 ms và tốc độ C= 4 Mbps như Hình 19.



Hình 19: Mô hình mạng mô phỏng tốc độ dữ liệu truyền mượt của hai đường MPTFRC và hai đường MPTCP truyền song song



Hình 20: Tốc độ dữ liệu truyền mượt của hai đường MPTFRC và hai đường MPTCP truyền song song

Kết quả Hình 20 cho thấy rằng, tốc độ truyền dữ liệu của hai đường MPTCP và MPTFRC qua hai link song song gần như tương đương. Tuy nhiên, thay đổi tốc độ trên MPTFRC tương đối chậm, tốc độ thay đổi giữa hai lần liên tiếp không lớn. Cụ thể xem xét thời gian từ giây 100 đến giây 150, biên độ dao động tốc độ của MPTCP lớn hơn so với MPTFRC. Đồng thời, qua kết quả mô phỏng cũng cho thấy rằng hai Subflow của hai đường MPTFRC chia sẻ công bằng và cùng tồn tại với hai Subflow của hai đường MPTCP.

4 KẾT LUẬN

Trong bài báo này, chúng tôi đã trình bày kỹ thuật mở rộng điều khiển tắc nghẽn cho ứng dụng multimedia trong giao thức truyền đơn đường vào trong giao thức truyền đa đường. Cải tiến giao thức bao gồm xác định trọng số động nhằm điều chỉnh thông lượng gửi dữ liệu giữa các đường trong nhiều điều kiện mạng khác nhau. Trọng số động của MPTFRC sẽ điều chỉnh lưu lượng tỉ lệ nghịch với mức độ tắc nghẽn của mạng, để dễ dàng đạt được 4 tiêu chí đã nêu bao gồm: *(i) nâng cao thông*

lượng truyền, (ii) đảm bảo chia sẻ công bằng đối với các luồng TCP hiện có; (iii) có khả năng cân bằng tắc nghẽn; (iv) tốc độ dữ liệu truyền mượt. Tuy nhiên, khi cài đặt ở mức gói, thiết kế ban đầu sẽ không cho kết quả mong muốn, đó là sự đánh võng ở mức gói (các đường ít khi sử dụng đồng thời ngay cả khi các đường có cùng điều kiện mạng như băng thông, độ trễ). Để sử dụng đường truyền không bị đánh võng giữa các đường được thực hiện bằng cách tính tham số tránh đánh võng giữa các đường dựa trên đường có kích thước cửa sổ lớn nhất và đường tốt nhất trong quá trình truyền dữ liệu. Mặc dù vậy, nó không làm thay đổi bài toán gốc ban đầu.

Bằng cách mô phỏng bởi nhiều mô hình mạng với các thông số mạng khác nhau, chúng tôi đã đánh giá giao thức đề xuất theo 4 tiêu chí nêu trên. Đồng thời chúng tôi so sánh sự chia sẻ công bằng và cùng tồn tại với các giao thức khác hiện có như TCP Reno, TFRC, MPTCP.

TÀI LIỆU THAM KHẢO

1. Dragana Damjanovic , Michael Welzl, "MulTFRC: Providing Weighted Fairness for Multimedia Applications (and others too!) ", 2009, ACM SIGCOMM Computer Communication Review,NY, USA,volume 39 (number 3), pages 5-12.
2. Damon Wischik, Mark Handley, Costin Raiciu, 2009, "Control of Multipath TCP and Optimization of Multipath Routing in the Internet". NET-COOP 2009, LNCS 5894, pages 204–218.
3. S. Floyd, M. Handley, 2008 "TCP Friendly Rate Control (TFRC): Protocol Specification", Network Working Group, Obsoletes: 3448, Updates: 4342, RFC 5348.
4. Damon Wischik, Costin Raiciu, Adam Greenhalgh, Mark Handley, 2011, "Design, implementation and evaluation of congestion control for multipath TCP" Proceedings of the 8th USENIX conference, CA, USA, pages 8-22.
5. C.Raiciu, M.Handly, D.Wischik, 2011, "Coupled Congestion Control for Multipath Transport Protocols ", Internet Engineering Task Force, RFC 6356, ISSN: 2070-1721.
6. NS-2 network simulator 2, http://www.isi.edu/nsnam/ns/, 26/5/2013.
7. Ramin Khalili,Nicolas Gast, Miroslav Popovic, Utkarsh Upadhyay, Jean-Yves Le Boudec "MPTCP is not Pareto-Optimal: Performance Issues and a Possible Solution", 2012, CoNEXT 12, pages 10-13
8. A. Ford, C.Raiciu, S.Barre, 2011, "Architectural Guidelines for Multipath TCP Development", Internet Engineering Task Force (IETF), RFC 6182 (Informational).
9. Jitendra Padhye, Victor Firoiu, Don Towsley, Jim Kurose, 1998, "Modeling TCP Throughput: ASimple Model and its Empirical Validation", ACM SIGCOMM, vol. 28, pp. 303-314.
10. B. Wang, W. Wei, Z. Guo, and D. Towsley, 2009, "Multipath live streaming via TCP: Scheme, performance and benefits", ACM Trans, NY, USA,vol. 5, no. 3, pages
11. J.-W. Park, R. Karrer, and J. Kim, 2011 "TCP-ROME: A transport-layer parallel streaming protocol for real-time online multimedia environments", Communications and Networks Journal, vol. 13, no. 3, pages. 277 – 285.
12. S. Tullimas, T. Nguyen, R. Edgecomb, and S.-c. Cheung, 2008, "Multimedia streaming using multiple TCP connections," ACM Trans. Multimedia Comput. Commun. Appl., vol. 4, no. 2, pages. 120-121.