TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

——————— * ———————

ĐỒ ÁN

TỐT NGHIỆP ĐẠI HỌC

NGÀNH CÔNG NGHỆ THÔNG TIN

HIỆU NĂNG TCP TRÊN KÊNH VỆ TINH

Sinh viên thực hiện : **Chu Quang Tú**

Lớp TTM – K51

Giáo viên hướng dẫn: GS.TS **Nguyễn Thúc Hải**

HÀ NỘI 5-2011

PHIẾU GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

1. Thông tin về sinh viên

Họ và tên sinh viên: Chu Quang Tú

Điện thoại liên lạc: 0906679888 Email: quangtu.chu@gmail.com

Lớp: Truyền thông mạng – K51 Hệ đào tạo: Chính quy

Đồ án tốt nghiệp được thực hiện tại: trường Đại học Bách Khoa Hà Nội

Thời gian làm ĐATN: Từ ngày /0 /2011 đến 27/05/2011

2. Mục đích nội dung của ĐATN:

Tìm hiểu và đưa ra phương pháp cải tiến hiệu năng hoạt động của TCP trên kênh vệ tinh. Đánh giá hiệu quả của các phương pháp cải tiến hiệu năng dựa vào mô phỏng.

3. Các nhiệm vụ cụ thể của ĐATN

- Tìm hiểu cơ sở lý thuyết về thông tin vệ tinh và phương pháp cái tiến hiệu năng TCP.
- Lập trình mô phỏng mô hình thông tin vệ tinh. Đề xuất kịch bản thử nghiệm và đánh giá hiệu quả của các phương pháp cái tiến hiệu năng.

4. Lời cam đoan của sinh viên:

Tôi – *Chu Quang Tú* – cam kết đồ án tốt nghiệp này là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của *GS.TS Nguyễn Thúc Hải*

Các kết quả nêu trong ĐATN là trung thực, không phải là sao chép toàn văn của bất kỳ công trình nào khác.

Hà Nội, ngày tháng năm

Tác giả ĐATN

Chu Quang Tú

5. Xác nhận của giáo viên hướng dẫn về mức độ hoàn thành của ĐATN và cho phép bảo vệ:

Hà Nội, ngày tháng năm

Giáo viên hướng dẫn

GS.TS Nguyễn Thúc Hải

LỜI CẢM ƠN

Để có thể thực hiện bài đồ án này, trước tiên em xin gửi lời cảm ơn đến các thầy, cô của Viện Công Nghệ Thông Tin và Truyền Thông cũng như tất cả các thầy cô của trường đại học Bách Khoa Hà Nội đã tận tình dạy dỗ, dìu dắt em trong suốt thời gian em học tập tại trường..

Em xin cảm ơn sâu sắc đến thầy giáo hướng dẫn Giáo sư – Tiến sĩ Nguyễn Thúc Hải đã trực tiếp hướng dẫn, tận tình giúp đỡ và chỉ bảo em trong suốt quá trình thực hiện đồ án này.

Em cũng xin cảm ơn anh Nguyễn Tuệ Linh, phó ban Kỹ thuật Công Nghệ, Tổng công ty Bưu Chính Việt Nam đã cho em nhiều kinh nghiệm quí báu và những ý kiến đóng góp trong quá trình em xây dựng chương trình mô phỏng sử dụng trong bài đồ án này.

Con cũng xin cảm ơn Bố, Mẹ vì đã luôn yêu thương động viên con trong suốt quá trình thực hiện bài đồ án này.

Hà Nội, Tháng 5 – 2011
Sinh viên thực hiện

Chu Quang Tú

TÓM TẮT NỘI DUNG ĐỒ ÁN

Bài đồ án này tập trung nghiên cứu vào hoạt động của giao thức TCP trên kết nối vệ tinh. Nhiệm vụ đặt ra là phải tối ưu hoạt động TCP sao cho có thể hoạt động với hiệu năng cao nhất trong điều kiện kết nối vệ tinh.

Nội dung bài đồ án này được trình bày gồm hai phần chính :

Phần một : Đặt vấn đề và định hướng giải pháp

Chương 1 : Đặt vấn đề và định hướng giải pháp

Nội dung của chương này là xác định mục tiêu cần đạt được của bài đồ án và phương pháp thực hiện mục tiêu đó.

Chương 2 : Cơ sở lý thuyết và công cụ

Chương này trình bày về sơ sở lý thuyết mạng vệ tinh, giao thức TCP và các biến thể TCP có áp dụng các thuật toán điều khiển tắc nghẽn. Đồng thời giới thiệu công cụ mô phỏng NS-2.

Phần hai : Các kết quả đạt được

Chương 3 : Đánh giá hiệu năng TCP trên kết nối End-to-end vệ tinh

Chương ba là phần thực hiện mô phỏng hoạt động của các biến thể TCP trên kết nối End-to-end trong các kịch bản khác nhau. Kết quả mô phỏng được sử dụng để đánh giá về hiệu quả hoạt động của các biến thể TCP.

Chương 4 : Một số phương pháp cái tiến hiệu năng TCP

Trong chương này, đồ án đề xuất mốt số phương pháp cải tiến hiệu năng TCP và đặc biệt đi sâu vào phương pháp cải tiến TCP bằng Snoop.

Chương 5 : Đánh giá hiệu năng giữa kết nối End-to-end và Snoop

Nội dung của chương này là đánh giá hiệu quả hoạt động của kết nối End-to-end vệ tinh với cải tiến Snoop TCP.

Kết luận : Tổng hợp kết quả đạt được và những hạn chế của bài đồ án, động thời định hướng mở rộng bài đồ án.

ABSTRACT OF THE THESIS

The content of this graduation thesis is about the studying performance of TCP protocol over satellite links. After studying standard mechanisms and TCP variants' algorithms, the thesis introduces an enhancement method named Snoop and by using simulation-base considering, it assess the effect of improvement performance TCP variant in standard End-to-end and Snoop.

MỤC LỤC

DANH MỤC CÁC TỪ VIẾT TẮT

STT	Ký hiệu	Tiếng Anh	Tiếng Việt
1	ACK	Acknowledgement	Số tuần tự thu
2	SACK	Selective ACK	ACK có lựa chọn
3	dupACK	Duplicate ACK	ACK lặp
4	E2E	End-to-end	Đầu cuối – Đầu cuối
5	GEO	Geosynchronous Satellite	Vệ tinh địa tĩnh
6	RTT	Round-trip time	Thời gian trễ toàn phần
7	cwnd	Congestion Window	Cửa sổ tắc nghẽn
8	seq	Sequence Number	Số tuần tự phát

DANH MỤC HÌNH VẼ

DANH MỤC ĐỒ THỊ

PHẦN 1 : CƠ SỞ LÝ THUYẾT

1. Đặt vấn đề và định hướng giải pháp

Ngày nay, hệ thống thông tin vệ tinh được áp dụng rộng rãi để có thể cung cấp dịch vụ truyền thông Internet tới hầu khắp mọi nơi bất kể địa hình và điều kiện thời tiết phức tạp. Việc tối ưu giao thức truyền vận TCP để nâng cao hiệu năng truyền dẫn, giúp giảm thiểu những hạn chế trên kênh vệ tinh là một vấn đề đã và đang được nhiều trường đại học cùng các tổ chức trên thế giới nghiên cứu. Mục tiêu của bài đồ án này là tìm hiểu về hoạt động của TCP trên kênh vệ tinh và thử nghiệm các cái tiến khác nhau trên mô hình thông tin vệ tinh để qua đó, có những đánh giá về mức độ hiệu quả của từng phương pháp cải tiến hiệu năng được áp dụng.

Phương pháp luận để thực hiện mục tiêu bài đồ án này là "phương pháp tiếp cận thực nghiệm dựa trên mô phỏng". Sau khi bốn thuật toán cơ sở về quản lý điều khiển tắc nghẽn dữ liệu TCP được nghiên cứu và tìm hiểu, chúng được kết hợp với nhau để tạo ra các biến thể TCP Tahoe, Reno, NewReno và SACK. Các TCP này được mô phỏng hoạt động trên mô hình vệ tinh lập trình bằng phần mềm mô phỏng mạng NS-2, kết quả mô phỏng được sử dụng để so sánh và đánh giá hiệu quả của các phương pháp cải tiến khác nhau. Dựa vào kết quả mô phỏng, ta có cơ sở để lựa chọn và áp dụng vào điều kiện thực tế.

2. Cơ sở lý thuyết và công cụ

2.1. Mạng vệ tinh

2.1.1. Khái niệm

Hệ thống thông tin vệ tinh là sự kết hợp của hai thành phần chính : Vệ tinh và trạm mặt đất.

- *Vệ tinh :* Được phóng ra ngoài Trái đất. Vệ tinh có hệ thống chuyển tiếp thông tin nên có thể thu nhận tín hiệu từ trạm mặt đất. Mặt khác, chúng còn được sử dụng để phát xuống trạm nhận tín hiệu thông qua hệ thống khuếch đại tín hiệu mạnh.
- *Trạm mặt đất :* Được đặt trên mặt đất, bao gồm hai chức năng chính : Truyền thông lên vệ tinh và nhận thông tin từ vệ tinh.

Truyền thông vệ tinh được sử dụng rộng rãi trong nhiều lĩnh vực như mạng điện thoại truyền thống, truyền tín hiệu truyền hình, truyền thông trong hàng hải ...

2.1.2. Những đặc tính cơ bản của hệ thống vệ tinh

Ưu điểm của mạng thông tin vệ tinh là dung lượng thông tin lớn, do sử dụng băng tần công tác rộng và kỹ thuật đa truy nhập cho phép đạt thông lượng lớn trong thời gian ngắn mà ít loại hình thông tin khác có thể đạt được. Mặt khác, do liên lạc giữa vệ tinh và trạm mặt đất là trực tiếp nên độ tin cậy và chất lượng thông tin của mạng vệ tinh cao hơn các loại hình thông tin khác. Ngoài ra, mạng vệ tinh còn cung cấp vùng bao phủ rộng, cho phép người sử dụng có khả năng ứng dụng thông tin di động và thông tin liên lạc toàn cầu.

Tuy nhiên, ngoài những ưu điểm kể trên, hệ thống thông tin vệ tinh vẫn còn một thách thức lớn với hiệu suất của các ứng dụng Internet bởi nhiều nhược điểm bài đồ án chỉ trình bày hai nhược điểm nổi bật là Độ trễ và Nhiễu như sau :

- *Độ trễ :* Trong hệ thông thông tin vệ tinh địa tĩnh GEO, độ trễ tối thiểu là 250ms; và đôi khi việc đóng gói dữ liệu, hàng chờ số liệu, hệ thống chuyển mạch bên trong có thể tạo thêm độ trễ kết nối giữa các trạm đầu cuối làm cho độ trễ lên đến 400ms. Ước lượng độ trễ này cao hơn khoảng 25 lần độ trễ kết nối mạng trục Internet Bắc-Nam Việt Nam (khoảng 16ms - Số liệu từ trung tâm VDC1). Độ trễ này làm ảnh hưởng đến các ứng dụng có tính tương tác cần sự bắt tay làm việc giữa hai phía - ví dụ như giao thức TCP.
- *Nhiễu :* Cường độ tín hiệu sóng radio tương ứng với khoảng cách truyền đi. Khoảng cách không gian giữa trạm mặt đất và vệ tinh là rất lớn, vì thế nhiễu sinh ra, làm giảm cường độ tín hiệu thông tin.

Ngoài những nhược điểm trên, hệ thống thông tin còn đòi hỏi đầu tư ban đầu cao trong khi thời gian làm việc vệ tinh là tương đối ngắn (từ 7 - 15 năm). Thêm vào đó, chi phí bảo dưỡng hệ thống rất tốn kém và vấp phải nhiều khó khăn về mặt kỹ thuật.

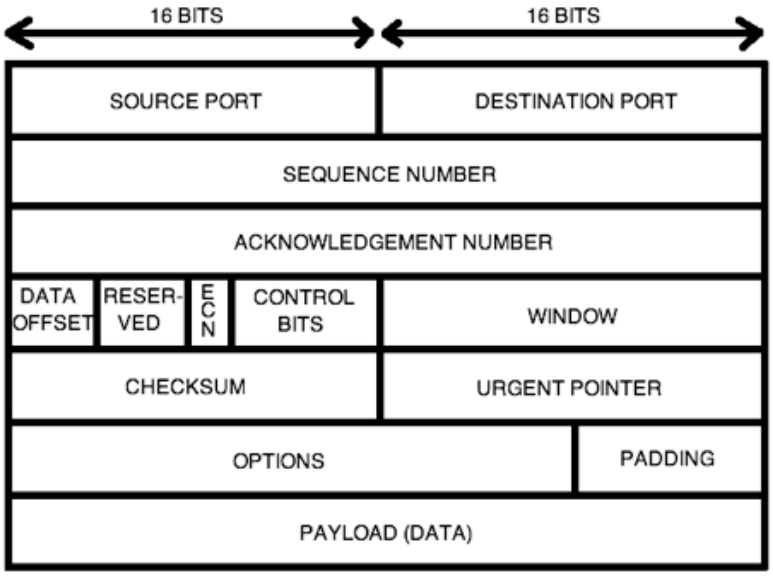
2.2. Giao thức TCP

Giao thức truyền vận TCP hoạt động ở tầng giao vận – mô hình OSI, hay tầng TCP trong mô hình TCP/IP. TCP đặc trưng là giao thức hướng kết nối. Giao thức TCP đảm bảo chuyển giao dữ liệu tới nơi nhận một cách đáng tin cậy và đúng thứ tự thông qua cơ chế điều khiển lỗi (error control) và điều khiển luồng (flow control). TCP kiểm tra để đảm bảo không có gói tin nào bị thất lạc bằng cách gán cho mỗi gói tin một số thứ tự (*sequence number*). Số thứ tự này còn được sử dụng để đảm bảo dữ liệu được trao cho ứng dụng đích theo đúng thứ tự. Đầu nhận gửi lại ACK (*acknowledgement*) cho các gói tin đã nhận được thành công; một "đồng hồ" (*timer*) tại nơi gửi sẽ báo hết hạn (*time-out)* nếu không nhận được tin báo nhận trong khoảng thời gian bằng một thời gian trễ toàn phần (Round Trip Time-RTT), và dữ liệu (được coi là bị thất lạc) sẽ được gửi lại. TCP sử dụng checksum (*giá trị kiểm tra*) để xem có byte nào bị hỏng trong quá trình truyền hay không; giá trị này được tính toán cho mỗi khối dữ liệu tại nơi gửi trước khi nó được gửi, và được kiểm tra tại nơi nhận.

Vấn đề tồn tại lớn nhất của giao thức TCP là tắc nghẽn. Trong một khoảng thời gian dài, TCP được phát triển phục vụ triệt để cho yêu cầu chống tắc nghẽn trên đường truyền.

2.2.1. Tổng quan về TCP

Cấu trúc gói tin TCP bao gồm hai phần là phần tiêu đề (header) và phần dữ liệu. Minh họa như hình dưới đây :



Hình 2.1 Cấu trúc gói tin TCP

Trong đó ý nghĩa các trường số liệu là :

- *Source Port và Destination Port :* Số hiệu của cổng TCP tại đầu-cuối
- *Sequence Number :* Số thứ tự hay Số tuần tự phát, có chức năng định danh byte đầu tiên của phần số liệu TCP trong dòng số liệu từ thực thể TCP phát đến thực thể TCP nhận. Trường này có 2 nhiệm vụ. Nếu cờ SYN bật thì nó là số thứ tự gói ban đầu và byte đầu tiên được gửi có số thứ tự này cộng thêm 1. Nếu không có cờ SYN thì đây là số thứ tự của byte đầu tiên.
- *Acknowledgement Number :* Giá trị trường này còn được gọi là số tuần tự thu. Vị trí tương đối của byte cuối cùng đã nhận đúng bởi thực thể gửi gói ACK cộng thêm 1. Giá trị này đúng khi bit cờ ACK bằng 1.
- *Data Offset :* Khoảng cách tương đối của trường số liệu với phần tiêu đề của TCP (TCP Header) tính theo từ 32 bit.
- *Reserved :* Luôn đặt là 0, sử dụng trong tương lai.
- *Control Bits* : Bao gồm 6 cờ với mục đích điều khiển : URG, ACK, PSH, RST, SYN và FIN.
- *Window :* Quy định tổng số byte dữ liệu mà thực thể nhận có thể nhận được, đồng nghĩa với độ lớn của bộ đệm nhận. Tính khởi đầu từ giá trị trường số tuần tự thu (Acknowledgement Number).
- *Checksum :* 16 bit kiểm tra cho cả phần header và dữ liệu.
- *Urgent Pointer :* Vị trí tương đối của byte trong trường số liệu TCP cần được xử lý đầu tiên. Nếu cờ URG bật thì giá trị trường này chính là số từ 16 bít mà số tuần tự phát (*sequence number*) cần dịch trái.
- *Option :* Trường tùy chọn. Nếu có, thì độ dài là bội số của 32 bit
- *Padding :* Phần vá thêm để phần tiêu đề của gói TCP có độ lớn là bội của 4 byte

2.2.2. Cơ chế kiểm soát tắc nghẽn TCP

Cơ chế kiểm soát tránh tắc nghẽn TCP gắn liền với việc điều khiển luồng. TCP xác lập một cửa sổ cho việc điều khiển. Bằng cách so sánh giá trị cần quan tâm

hiện tại với phạm vị cửa sổ (trong phạm vi, vượt trên hay thấp hơn) để đưa ra các quyết định điều khiển khác nhau. Như vậy, hiệu năng hoạt động của TCP trên bất kì kênh nào cũng chịu ảnh hưởng trực tiếp từ các cơ chế này.

2.2.2.1. Thuật toán Bắt đầu chậm

"Bắt đầu chậm" là một trong những thuật toán mà giao thức TCP dùng để điều khiển kiểm soát tắc nghẽn trong mạng. Nó được sử dụng kết hợp với các thuật toán khác để tránh việc gửi dữ liệu nhiều hơn khả năng truyền của mạng, có nghĩa là, để tránh gây tắc nghẽn mạng.

Thuật toán "Bắt đầu chậm" định nghĩa khái niệm "cửa sổ tắc nghẽn" (Congestion Window - cwnd) cho thực thể phát. Mục đích của cwnd là ngăn đầu phát phát quá mức dữ liệu so với khả năng đáp ứng của đường truyền. Việc điều khiển tắc nghẽn được thực hiện bằng cách thay đổi cwnd. Trong thực tế, việc thay đổi này được thực hiện khi có gói bị mất. Việc phát hiện mất gói có thể phát hiện thông qua time-out hay nhận được duplicate ACKs (nhập các gói ACK giống nhau) – dup ACKs.

Time-outs: liên kết với mỗi gói là một timer (bộ định thời). Nếu hết thời hiệu (time-out) mà không nhận được ACKs từ đầu thu, đầu phát sẽ tiến hành gửi lại gói này. Giá trị của bộ định thời còn được gọi là RTO (Retransmit time-out), hay RTT (round time-trip). Tuy nhiên giá trị của RTT không thể biết được trong thực tế, nó được đo thông qua giải thuật Jacobson/Karels.

DupACKs hay Duplicate ACKs: khi một gói bị mất, đồng nghĩa với việc đầu thu không nhận được gói đó. Nó phá vỡ sự tuần tự các gói nhận được ở đầu thu. Lúc này, đầu thu liên tục gửi ACK với cùng một giá trị. Đây là căn cứ cho đầu phát biết đã có mất gói.

Mô tả thuật toán :

- Sau khi một kết nối mới được thiết lập, Thực thể TCP bắt đầu phát với khởi tạo cwnd = 1 đơn vị gói dữ liệu.
- Mỗi khi nhận được thông báo trả lời ACK, thực thể TCP lại tăng cwnd thêm 1 đơn vị gói dữ liệu : cwnd = cwnd + 1
- Hoạt động này vẫn tiếp tục cho đến khi kích thước cửa sổ tắc nghẽn (cwnd) đạt tới kích thước cửa sổ quảng bá của thực thể nhận hoặc khi xảy ra hiện tượng mất mát thông tin thể hiện qua giá trị RTT tăng. Đến đây, thực thể phát biết độ lớn cwnd đã quá lớn và cần điều chỉnh phù hợp [1].

2.2.2.2. Thuật toán tránh tắc nghẽn

Đặc trưng của hiện tượng tắc nghẽn dữ liệu là thời gian trễ toàn phần (Round Time Trip-RTT) tăng. Những nguyên nhân chính dẫn đến sự gia tăng giá trị RTT là :

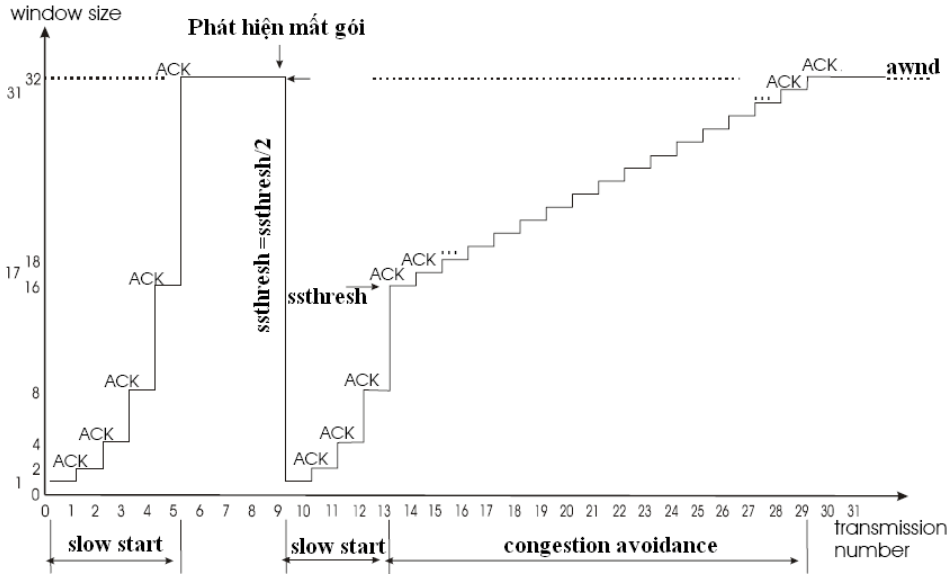
- Hiện tượng mất gói dữ liệu tại một hệ định tuyến nào đó trong mạng (nguyên nhân có thể do thiếu bộ nhớ), dẫn đến Time-out và thực thể phát phải phát lại gói dữ liệu bị mất. Hoặc;

- Thực thể phát nhiều lần nhận được thông báo trả lời ACK cho biết các gói dữ liệu đến không đúng theo thứ tự phát (out-of-order) tại thực thể nhận.

Biện pháp tránh tắc nghẽn chính là giảm lưu lượng dữ liệu phát vào kênh TCP được thiết lập, từ đó hạn chế tình trạng mất gói dữ liệu.

Thuật toán *"Bắt đầu chậm"* và "*Tránh tắc nghẽn*" là những kỹ thuật điều khiển lưu lượng và tránh tắc nghẽn hoạt động độc lập, song lại có quan hệ mật thiết với nhau, và thường được cài đặt cùng nhau để tăng hiệu quả hoạt động. Các thuật toán kết hợp hoạt động như sau:

- Hai thuật toán này yêu cầu duy trì hai biến thể trong mỗi kết nối đó là cửa sổ tắc nghẽn, cwnd, và kích thước ngưỡng bắt đầu chậm, ssthresh.
- Khởi tạo cwnd = 1 đơn vị gói dữ liệu và ssthresh = 65535 byte.
- Thực tế TCP không bao giờ phát nhiều hơn giá trị giá trị nhỏ nhất giữa độ lớn cửa sổ tắc nghẽn cwnd và độ lớn cửa sổ thu rcnd được thực thể thu thông báo trước đó. Tức là W = min {cwnd, rcnd}
- Khi xảy ra tắc nghẽn dữ liệu, đặt ssthresh = W(tắc nghẽn)/2 và đặt cwnd = 1 đơn vị gói dữ liệu.
- Mỗi khi nhận được thông báo ACK, giá trị cwnd được tăng lên phụ thuộc vào thuật toán *"Bắt đầu chậm"* hay thuật toán *"Tránh tắc nghẽn"* được thực hiện, cụ thể hơn :
    - Nếu cwnd < ssthresh, thuật toán *"Bắt đầu chậm"* được thực hiện : Giá trị cwnd được tăng 1 đơn vị gói dữ liệu với mỗi thông báo ACK nhận được : cwnd = cwnd + 1
    - Ngược lại, nếu cwnd = ssthresh, thuật toán *"Tránh tắc nghẽn"* được thực hiện. Giá trị *cwnd* được tăng thêm 1/cwnd với mỗi thông báo ACK nhận được : cwnd = cwnd + 1/cwnd.



Hình 2.2 Quá trình "Bắt đầu chậm" và "Tránh tắc nghẽn" [2]

2.2.2.3. Thuật toán "Phát lại nhanh"

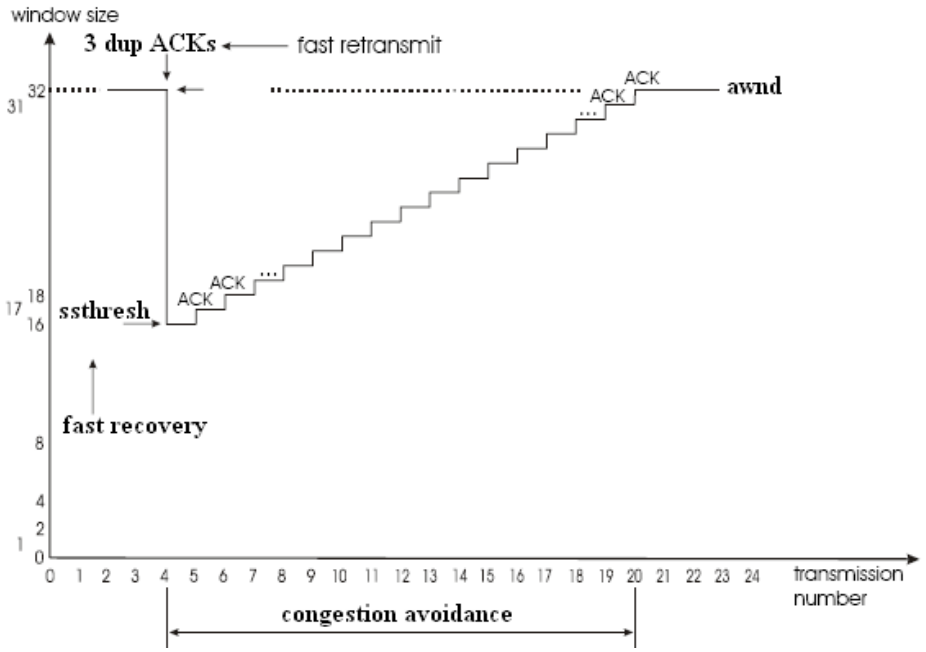
Thông thường, thực thể TCP phát sử dụng một đồng hồ để xác định mất mát dữ liệu. Nếu không nhận được ACK tương ứng với dữ liệu được phát trong một khoảng thời gian cụ thể (được tính theo thời gian trễ toàn phần), thì thực thể TCP phát sẽ coi dữ liệu đó đã bị mất mát trong quá trình truyền và sẽ tiến hành phát lại gói dữ liệu. Thuật toán "Phát lại nhanh" là một cải tiến cho phép thực thể TCP phát không cần chờ Time-out trong trường hợp nhận được nhiều hơn ba dup ACKs.

2.2.2.4. Thuật toán "Khôi phục nhanh"

Thuật toán "Khôi phục nhanh" là một biến thể của thuật toán "Khởi động chậm". Thuật toán này quy định việc thực hiện thuật toán "Tránh tắc nghẽn" ngay sau khi thực hiện thuật toán "Phát lại nhanh". Đồng thời, "Khôi phục nhanh" là một cải tiến cho phép thông lượng cao ở mức tắc nghẽn trung bình, nhất là với các cửa sổ kích thước lớn.

Thuật toán "Khôi phục nhanh" và "Phát lại nhanh" thường được cài đặt cùng nhau như sau :

- Sau khi nhận được ba dupACKs liên tiếp :
  - Thực thể phát thiết lập ssthresh = cwnd/2 (nhưng không nhỏ hơn hai đơn vị gói dữ liệu) và thực hiện phát lại gói dữ liệu thất lạc.
  - Tiếp đó, thiết lập cwnd = ssthresh + 3*smss (với smss là kích thước gói dữ liệu). Điều này cho phép mở rộng kích thước cửa sổ tắc nghẽn cwnd tương ứng với số gói dữ liệu đã rời mạng và đã được lưu đệm ở đầu bên kia.
- Nếu nhận được dupACK khác, tăng cwnd = cwnd + 1*smss. Điều này mở rộng kích thước cửa sổ tắc nghẽn cho gói dữ liệu rời mạng. Sau đó gửi dữ liệu nếu giá trị cwnd mới cho phép.
- Nếu nhận được thông báo ACK mới, phục hồi cwnd = ssthresh ở bước đầu tiên. ACK mới này là thông báo của quá trình truyền lại từ bước 1, một thời gian trễ toàn phần kể từ lúc truyền lại. Thêm nữa, ACK này còn là thông báo về tất cả các gói dữ liệu trung gian gửi giữa lúc mất gói tin và lúc nhận được dupACK đầu tiên [1]



Hình 2.3 Quá trình "Phát lại nhanh" và "Phục hồi nhanh" [2]

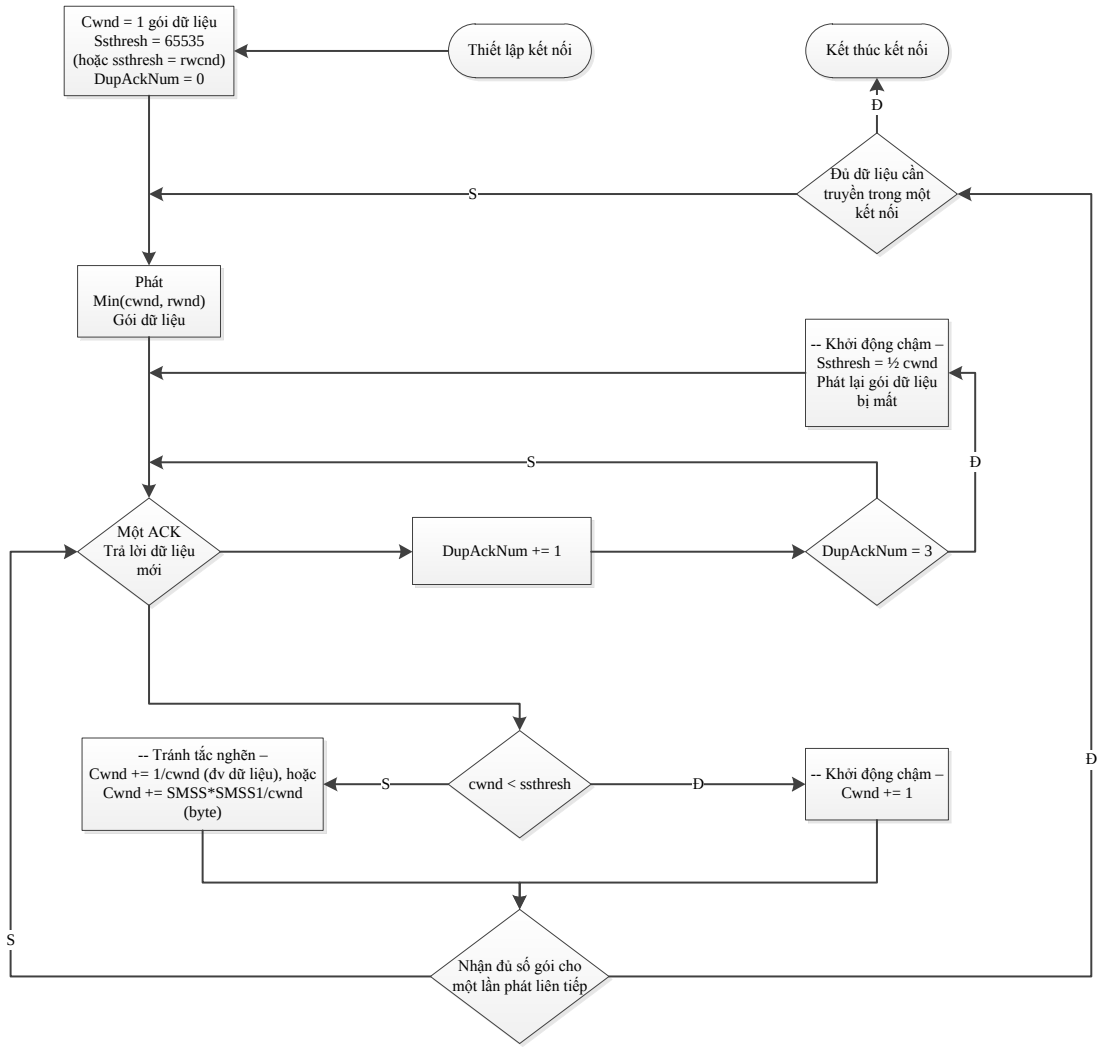
2.3. Một số phiên bản TCP mở rộng

Sau khi bốn thuật toán cơ sở về quản lý điều khiển tắc nghẽn dữ liệu TCP được nghiên cứu và tìm hiểu, chúng được kết hợp với nhau để tạo ra các biến thể TCP. Các phiên bản TCP này hoạt động tập trung vào kiểm soát tắc nghẽn mạng những vẫn đảm bảo duy tri thông lượng cho người sử dụng.

2.3.1. Tahoe TCP

TCP Tahoe được đề xuất bởi Jacobson, 1988. Tahoe là giải thuật điều khiển tránh tắc nghẽn đầu tiên. Các thuật toán mới bao gồm "Khởi động chậm", "Tránh tắc nghẽn" và "Phát lại nhanh" mà đã được mô tả trong phần trên của bài đồ án này.

Thuật toán : Thuật toán điều khiển tắc nghẽn Tahoe TCP là kết hợp của của ba thuật toán cơ sở "Khởi động chậm", "Tránh tắc nghẽn" và "Phát lại nhanh" đã được mô tả ở phần trên. Đặc trưng của Tahoe TCP là khi phát hiện mất dữ liệu thông qua ba dupACKs thì thực thể phát sẽ phát lại gói dữ liệu bị mất, đặt cwnd = 1 gói dữ liệu và thực hiện "Khởi động chậm". Chiến lược phục hồi của Tahoe ở đây là không hạn chế việc truyền lại nhiều nhất một gói dữ liệu trong mỗi thời gian trễ toàn phần RTT và như thế, Tahoe có thể phát lại cả các gói dữ liệu đã được truyền thành công trước đó [4].



Hình 2.4 Sơ đồ thuật toán của Tahoe – TCP

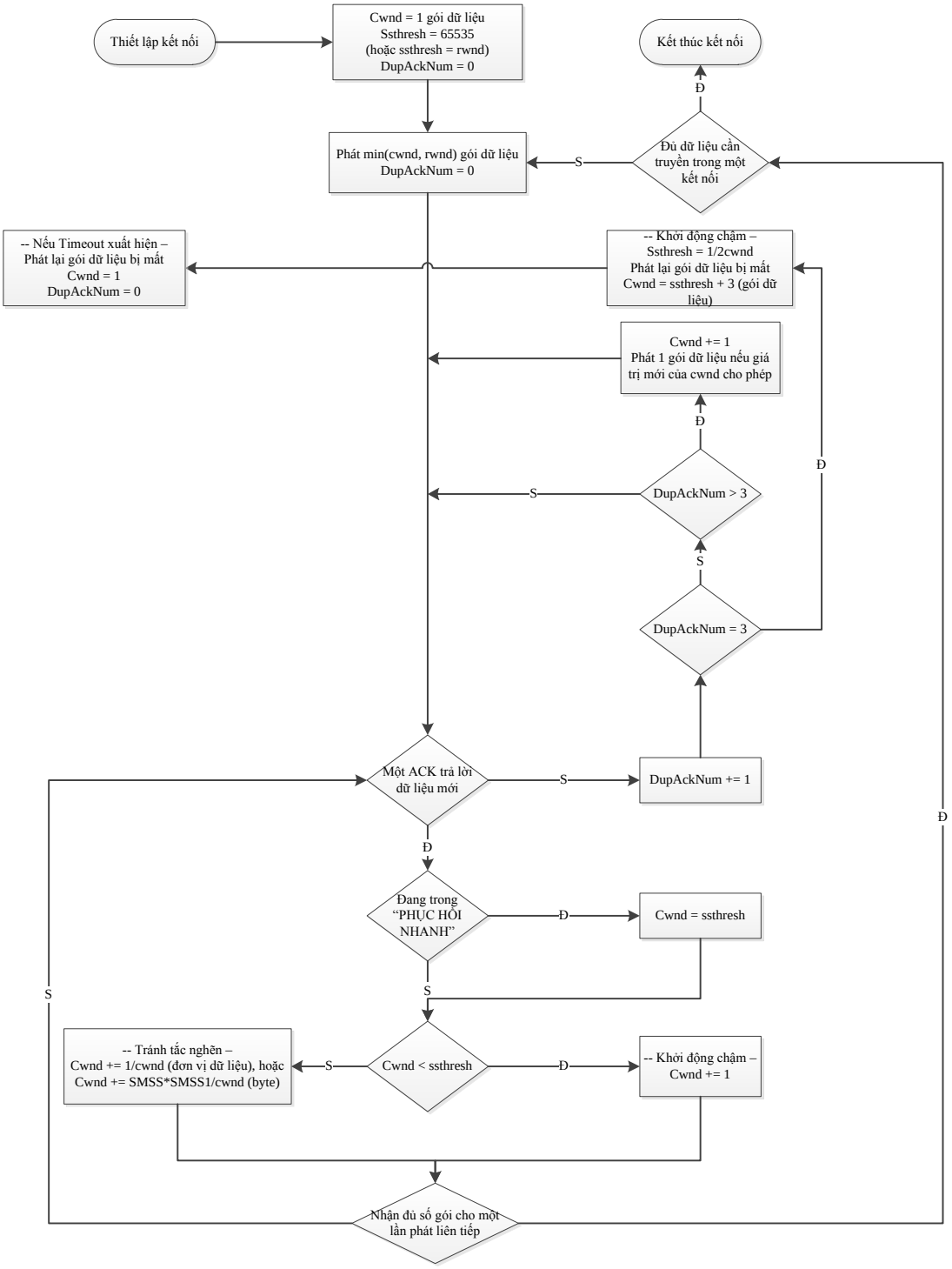
2.3.2. Reno TCP

TCP Reno được cải tiến từ TCP Tahoe, nhưng đã sửa đổi quá trình hoạt động của thuật toán "Phát lại nhanh" để bao hàm luôn cả thuật toán "Phục hồi nhanh". "Phục hồi nhanh" hoạt động với giả thiết mỗi một dupACK nhận được đại diện cho một gói dữ liệu đơn đã rời khỏi mạng. Như vậy trong thời gian "Phục hồi nhanh" thực thể phát TCP có khả năng ước lượng về tổng số dữ liệu còn tồn đọng

Thuật toán "Phục hồi nhanh" của Reno được tối ưu hóa cho trường hợp khi một gói dữ liệu đơn bị mất từ một cửa sổ dữ liệu. Thực thể phát phát lại nhiều nhất một gói dữ liệu bị mất trong mỗi RTT. Reno cái tiến một cách đáng kể việc xử lý của Tahoe khi một gói dữ liệu đơn bị mất, nhưng không hiệu quả khi nhiều gói dữ liệu bị mất từ một cửa sổ dữ liệu.

Thuật toán : Thuật toán điều khiển tắc nghẽn Reno – TCP là kết hợp của bốn thuật toán cơ sở "Khởi động chậm", "Tránh tắc nghẽn", "Phát lại nhanh" và "Phục hồi nhanh" được mô tả ở trên.

Đặc trưng của Reno – TCP là khi phát hiện thấy mất dữ liệu thông qua 3 dupACK, thực thể phát TCP phát lại gói dữ liệu bị mất, giảm cửa sổ tắc nghẽn một nửa và thay "Khởi động chậm" của Tahoe bằng "Phục hồi nhanh". Chiến lược phục hồi của Reno là có thể phát lại nhiều nhất một gói dữ liệu bị mất cho mỗi thời gian trễ toàn phần RTT. Khi có nhiều gói tin bị mất trong một cửa sổ dữ liệu (số gói mất lớn hơn hai) thì Reno phải nhờ vào đồng hồ phát lại để phục hồi các gói dữ liệu bị mất đó.



Hình 2.5 Sơ đồ thuật toán Reno – TCP

2.3.3. NewReno TCP

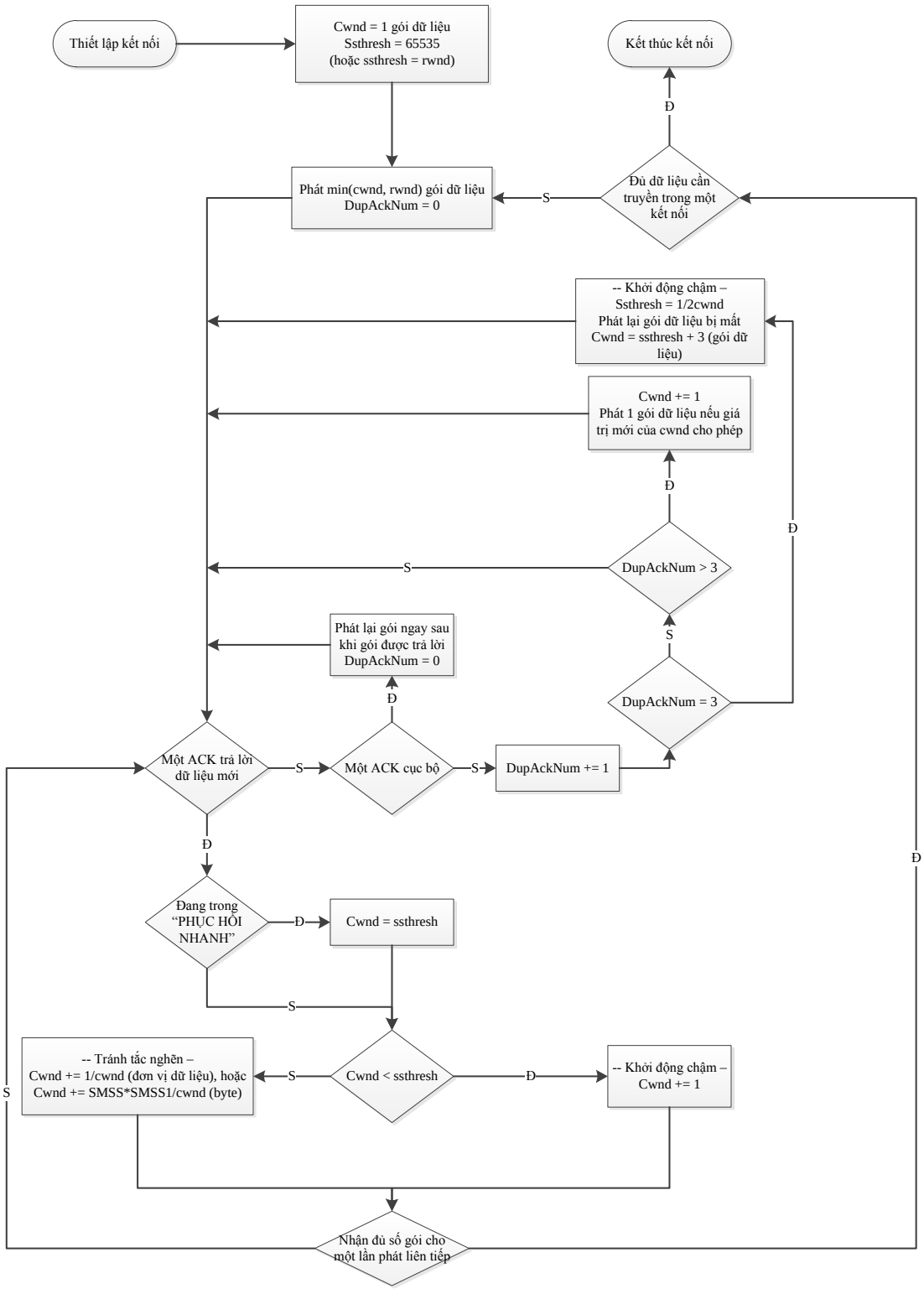
TCP New-Reno là một cải tiến của Reno, có thể phát hiện ra mất nhiều gói trong cùng một cửa sổ. Giống như Reno, New-Reno cũng bắt đầu giai đoạn "truyền lại nhanh" khi nhận được các dup ACK. Tuy nhiên, khác với RENO, New-Reno không thoát khỏi giai đoạn "Phục hồi nhanh" cho đến khi tất cả các gói tin outstanding được xác nhận. Khái niệm outstanding để chỉ các gói dữ liệu đã được gửi sau nhưng chưa được xác nhận tại thời điểm phát hiện mất gói. Do đó, New-Reno giải quyết được vấn đề giảm cwnd nhiều lần khi mất nhiều gói.

Trong giai đoạn "Phục hồi nhanh" của NewReno, nếu thực thể phát nhận được một ACK mới thì :

- Nếu ACK thông báo tất cả các gói tin outstanding được xác nhận, thì New-Reno hoạt động giống như Reno.
- Ngược lại (gọi là ACK cục bộ - Partial ACK), suy ra rằng, có nhiều hơn một gói bị mất. Trong trường hợp này, New-Reno tiến hành gửi lại gói mất cho đến khi tất cả các gói tin outstanding được xác nhận. Chi tiết được mô tả trong [5].

Hạn chế của New-Reno là mặc dù hổ trợ gửi lại nhiều gói, nhưng phải mất một RTT để New-Reno phát hiện ra mỗi gói mất.

Thuật toán : Thuật toán điều khiển tắc nghẽn NewReno – TCP là kết hợp giữa bốn thuật toán cơ sở "Khởi động chậm", "Tránh tắc nghẽn", "Phát lại nhanh" và "Phục hồi nhanh" đã được mô tả ở trên.

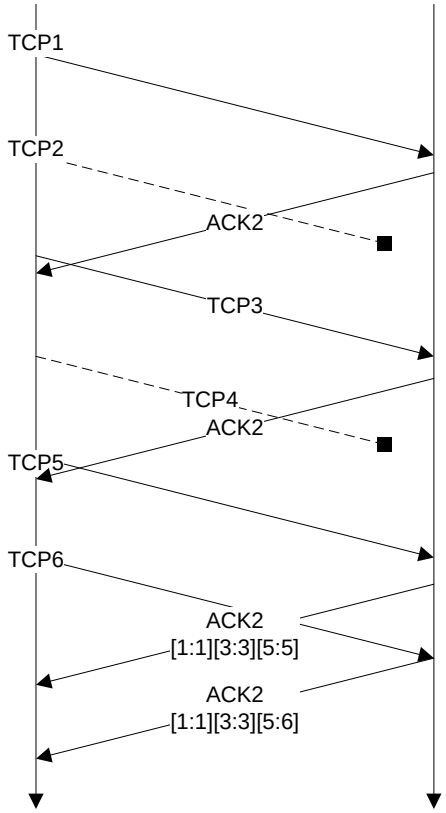


Hình 2.6 Sơ đồ thuật toán NewReno – TCP

2.3.4. SACK TCP

SACK được công bố bởi M.Mathis, J.Mahdavi, S.Floyd và A.Romanow và được mô tả trong [6]. SACK là thuật ngữ viết tắt từ Selective Acknowedgment Options – ACK có lựa chọn. Khác với cơ chế gửi ACK như Tahoe, Reno, hay NewReno (còn gọi là cơ chế cumulative acknowledgment – ACK tích lũy), SACK không gửi ACK cho từng gói tin đến mà gửi ACK cho một nhóm các gói

tin. Trong đó, có các thông tin về các khối gói tin liên tiếp. Từ đó, có thể suy ra được các gói tin bị mất.



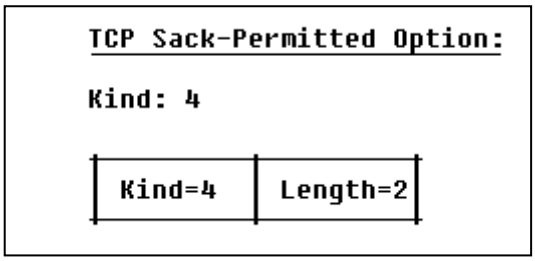
Hình 2.7 Ví dụ về phương thức hoạt động của SACK

Trong ví dụ trên, thực thể nhận báo SACK 3 khối gói tin liên tục [1:1], [3:3], [5:6]. Điều này cho biết các gói 2, 4 đã bị mất.

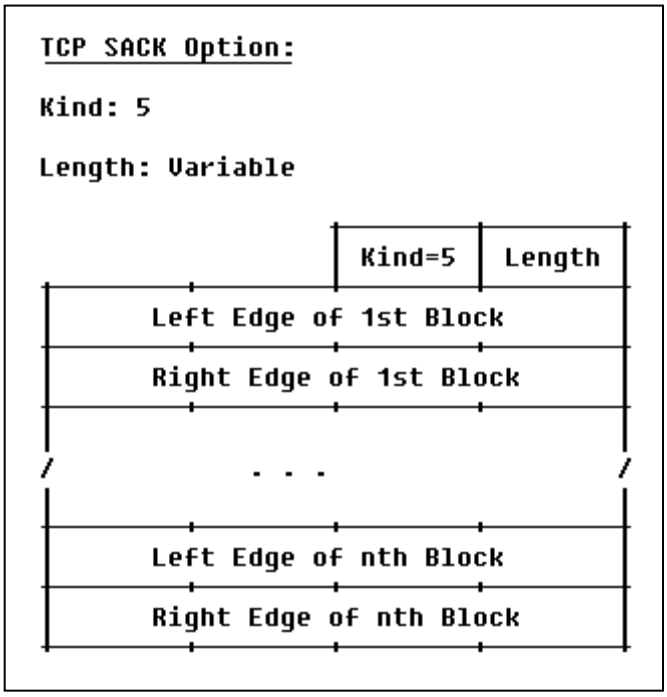
Ưu điểm của SACK so với ACK tích lũy là SACK cho phép thực thể phát phát hiện ra đồng thời nhiều gói tin bị mất mà ACK tích lũy không làm được.

Thuật toán : Thuật toán điều khiển tắc nghẽn SACK – TCP là kết hợp bốn thuật toán cơ sở "Khởi động chậm" , "Tránh tắc nghẽn", "Phát lại nhanh" và "Phục hồi nhanh" đã được mô tả ở trên.

SACK – TCP có một sự cải tiến so với Reno và NewReno là có sử dụng ACK có lựa chọn (Selective ACK - SACK) để thông tin kịp thời những gói dữ liệu bị mất thông qua tùy chọn ACK (ACK Option) được lưu trong nó. Và qua đó, thực thể phát có thể sử dụng chiến lược phát lại có lựa chọn để phục hồi khi nhiều gói dữ liệu bị mất. Có 2 loại tùy chọn dành cho SACK, Tùy chọn type 4 dùng để báo hiệu cho phép SACK và tùy chọn type 5 dùng để chứa thông tin SACK (các khối gói tin liên tiếp) và trong đó các khối gói tin được lưu thành từng cặp (left, right) liên tục. Cấu trúc các option như sau :

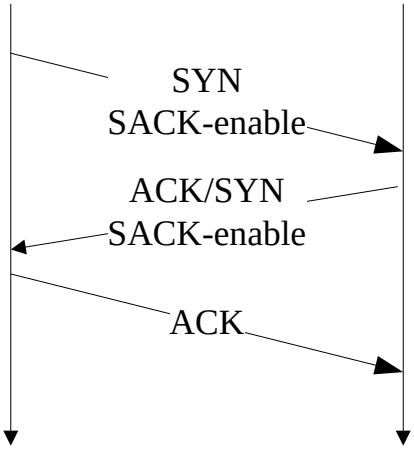


Hình 2.8 Option thông báo cơ chế điều khiển SACK



Hình 2.9 Option thông tin SACK

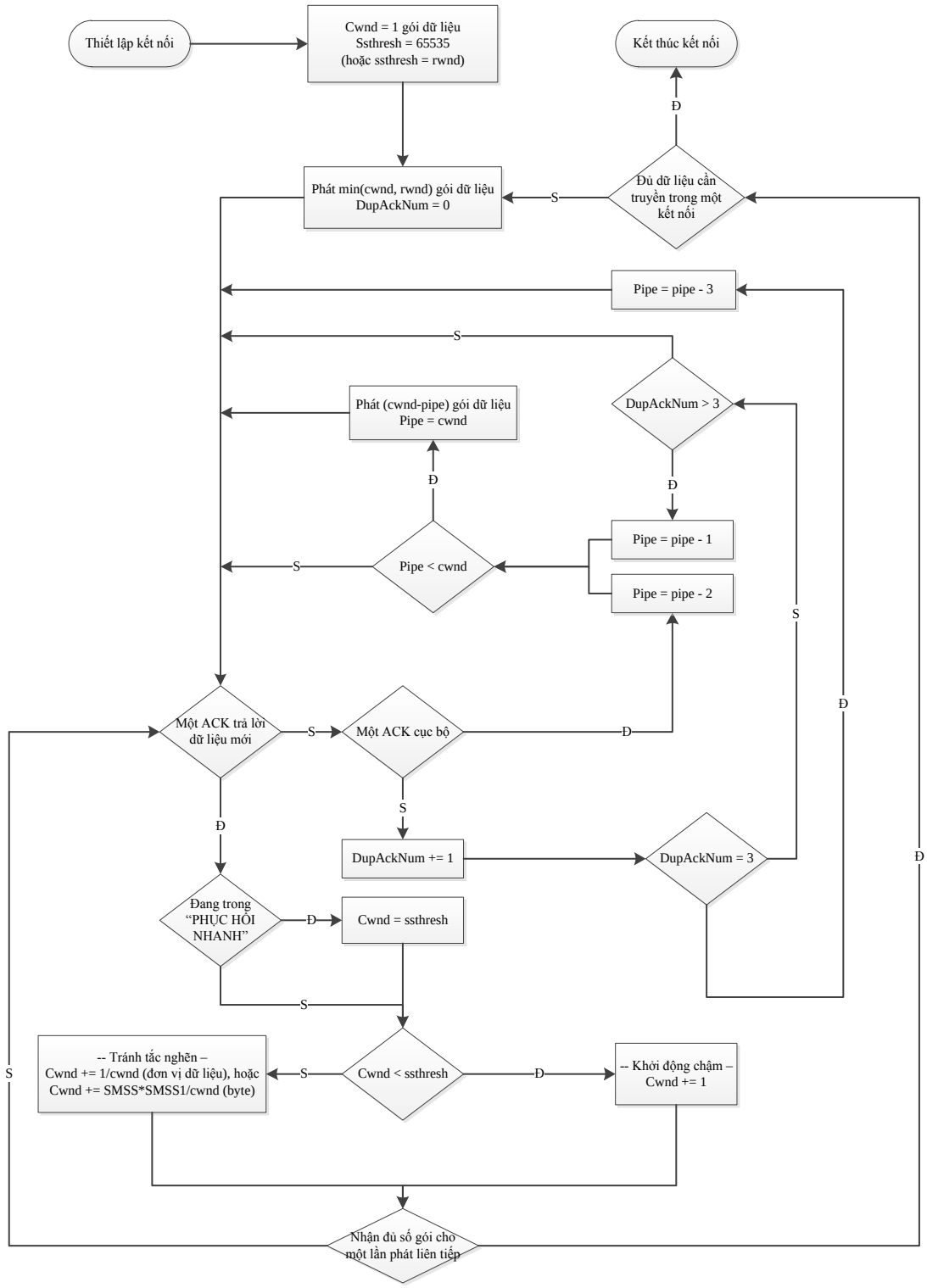
Giai đoạn thiết lập kết nối cũng là giai đoạn để thực thể phát và thực thể thu thỏa thuận về việc có sử dụng SACK hay không.



Hình 2.10 Quá trình khai báo sử dụng SACK

TCP SACK kế thừa "Bắt đầu chậm", "Truyền lại nhanh" của Reno và cơ chế Time-out của Tahoe trong trường hợp gói bị mất không phát hiện được bởi SACK. Các khối gói tin trong tùy chọn của SACK cung cấp thông tin cho thực thể phát biết được các gói bị mất, các gói outstanding để có thể gửi lại. Khi bắt đầu thời kỳ "Phục hồi nhanh", thực thể thu ghi nhận lại số lượng dữ liệu outstanding (lưu vào biến pipe). Đồng thời giảm đi cwnd còn một nửa. Mỗi khi nhận được ACK, pipe giảm đi 1 và khi truyền lại một gói tin thì pipe được tăng thêm 1. Bất cứ khi nào pipe trở nên nhỏ hơn cwd, thực thể phát kiểm tra những gói tin nào chưa được gửi và gửi chúng đi. Khi không còn các gói tin outstanding, dữ liệu mới sẽ được gửi đi. Như vậy, trong vòng một RTT, có nhiều hơn một gói tin được truyền lại.

Trong quá trình truyền lại, nếu tiếp tục mất mát, gói tin bị mất sẽ được phát hiện bởi đồng hồ phát lại. Sau khi đồng hồ phát lại hết hạn, SACK thực hiện "Bắt đầu chậm". Thực thể phát ra khỏi quá trình "Phục hồi nhanh" sau khi nhận được một ACK phục hồi (Recovery ACK) và được bên nhận thông báo đã nhận tất cả dữ liệu tồn đọng khi ở trong "Phục hồi nhanh".



Hình 2.11 Sơ đồ thuật toán SACK – TCP

2.4. Giới thiệu công cụ sử dụng trong đồ án

2.4.1. Phần mềm mô phỏng NS-2

NS-2 là một bộ mô phỏng có cấu trúc hướng đối tượng. Nó được xây dựng dựa trên hai ngôn ngữ hướng đối tượng là C++ và phiên bản hướng đối tượng của Tcl (hay còn gọi là OTcl). NS-2 có thể được mở rộng bởi người sử dụng (tức là người dùng có thể lập trình tùy ý trên nền của bộ mô phỏng NS). Thông qua cách tiếp cận này, coi mỗi một mô hình mô phỏng như một chương trình hơn là

các mô hình tĩnh - không thể thay đổi. Mục đích của NS-2 là tạo ra một môi trường giả lập cho việc nghiên cứu, kiểm tra, thiết kế các giao thức, các kiến trúc mới, so sánh các giao thức và tạo ra các mô hình mạng phức tạp.

Phiên bản thứ nhất của NS được phát triển vào năm 1995 và phiên bản thứ hai ra đời năm 1996. NS-2 là phần mền mã nguồn mở có thể chạy được trong môi trường Linux và Window. Hiện tại, phiên bản NS-3 đang được phát triển mới nhiều bổ sung và cải tiến mới so với phiên bản thứ 2.

NS-2 được nhiều cá nhân và tổ chức ở nhiều nơi trên thế giới sử dụng để phục vụ cho việc học tập và nghiên cứu mạng máy tính. NS-2 hỗ trợ gần như đầy đủ mọi khía cạnh khác nhau trong mạng máy tính. Từ các lớp mô phỏng cơ sở như các Node, các liên kết tạo thành mô hình mạng (Topology), các gói tin – hay các gói dữ liệu, việc chuyển tiếp các gói tin, độ trễ truyền dẫn, rồi đến lớp dùng để phân kênh các gói tin nhằm mục đích hướng các gói tin đến các trạm xác định, các Agent xác định. Cho đến việc quản lý các hàng đợi tại các trạm trung chuyển, lập lịch trình cho các gói tin, các thuật toán tránh tắc nghẽn, kiểm soát dữ liệu, nâng cao hiệu năng truyền tin. Thông qua NS-2, người dùng có thể mô phỏng các mạng LAN, WLAN, hệ thống mạng vệ tinh … rồi các ứng dụng FTP, HTTP, Webcache, Telnet… dựa trên các Agent của tầng vận chuyển, các giao thức tầng vận chuyển …

2.4.2. Công cụ thực hiện đồ án

Bài đồ án này sử dụng bộ mô phỏng NS-2 đẻ thực hiện việc mô phỏng các mô hình vệ tinh với những thay đổi thiết lập phù hợp.

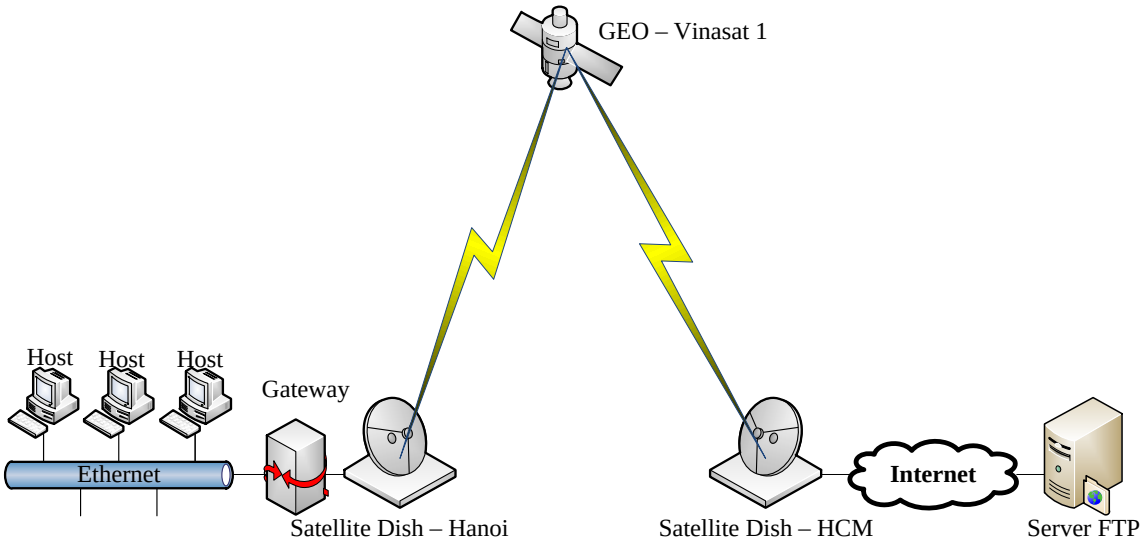
Chương trình mô phỏng được cài đặt trên hệ điều hành Ubuntu 10.10 (phiên bản Desktop - 32bit). Trong đó phiên bản NS-2 (2.34) được dịch trên bộ dịch GCC-4.3.

Máy tính cài đặt cấu hình và chạy chương trình mô phỏng có cấu hình như sau :

- CPU: Intel P8400 2.26Ghz, 2 nhân
- RAM: 4GB DDR3
- Ổ cứng 7200rpm

3. Đánh giá hiệu năng TCP trên kết nối TCP End-to-End vệ tinh

3.1. Kết nối TCP end-to-end



Hình 3.1 Mô hình mô phỏng kết nối End-to-End

Kết nối TCP E2E (End-to-End) là hình thức kết nối TCP truyền thống mà trong đó, tại các trạm đầu-cuối của giao thức TCP bắt tay và làm việc với nhau trực tiếp. Tức là thông tin từ trạm phát thông qua môi trường mạng được truyền trực tiếp tới trạm thu và thông tin trả lời trực tiếp từ trạm thu qua được truyền qua môi trường mạng đến trạm phát.

Với kết nối TCP end-to-end, mỗi gói dữ liệu hay thông tin phản hồi đều chịu ảnh hưởng trực tiếp của tổng hành trình trên mạng.

3.2. Hiệu năng các biến thể TCP trên kết nối end-to-end

Thực nghiệm

Mục đích thực hiện mô phỏng hoạt động các biến thể TCP trên kết nối End-to-end vệ tinh là để xác định hiệu quả hoạt động của từng biến thể TCP riêng biệt và so sánh với các biến thể TCP khác. Qua đó thấy được hiệu quả của các thuật toán điều khiển tắc nghẽn cơ sở.

Mô hình mô phỏng sử dụng ở đây được minh họa như Hình 3.1 , trong đó mỏ phỏng một mạng LAN kết nối với mát chủ Internet thông qua hệ thống vệ tinh :

- Mạng LAN trong mô phỏng này là mạng Ethernet có tốc độ 10Mbps và độ trễ 2ms. Trong mô hình này, giả thiết rằng mạng LAN làm việc không có lỗi, tức là trong một thời điểm chỉ có một trạm truyền dữ liệu trên mạng và truyền song công. Do đó loại bỏ các khả năng tác động đến hiệu năng truyền trên kênh vệ tinh
- Từ mạng LAN kết nối với một Gateway để liên kết với trạm mặt đất. Nó được coi là trạm lặp (Repeater) kết nối với trạm mặt đất. Trạm lặp này có

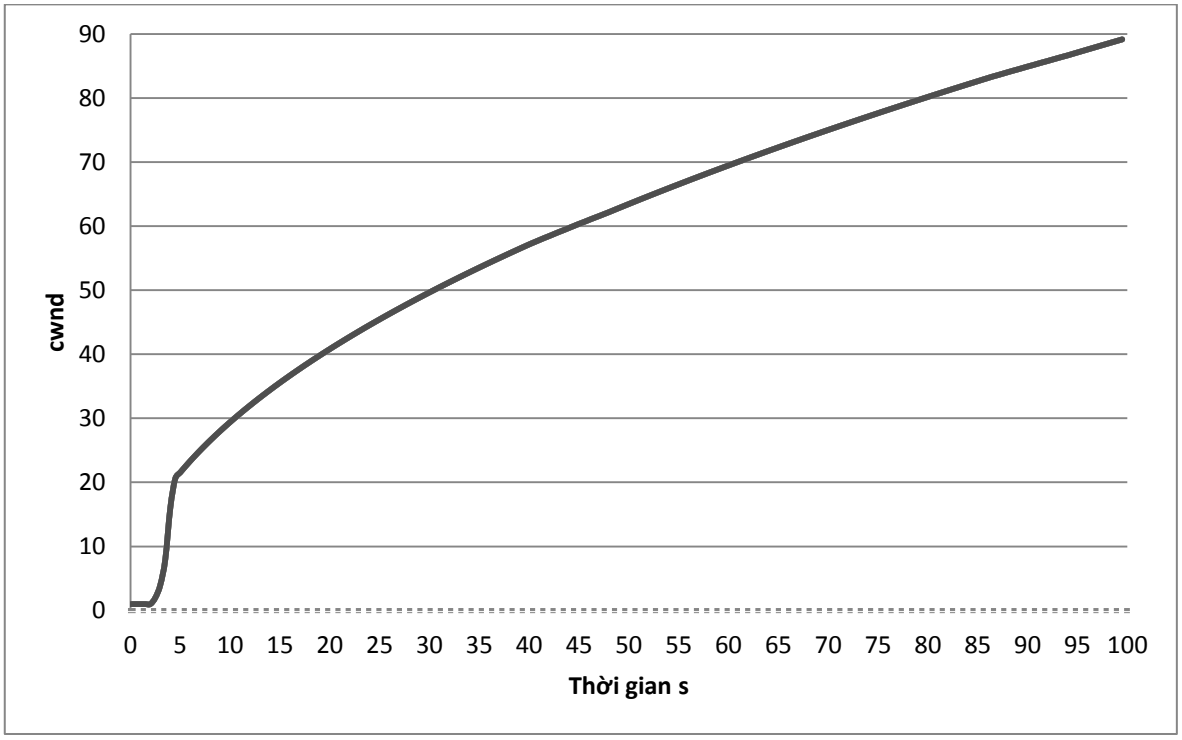
kết nối trực tiếp với trạm mật đất với tốc độ 100bps và có độ trễ 0ms. Với thiết lập này, giả thiết xây dựng cổng kết nối với vệ tinh và giả thuyết không có lỗi khi truyền trên kết nối giữ trạm lặp và trạm mặt đất.

- Hệ thống vệ tinh được mô phỏng ở đây là hệ thống GEO liên kết đối xứng, song công. Trong đó, vệ tinh GEO được lấy tọa độ theo vệ tinh VINASAT-1, vị trí quĩ đạo 132 độ Đông. Hai trạm mặt đất được đặt Hà Nội - HN (tọa độ *21.02 105.50*) và Hồ Chí Minh - HCM (tọa độ *10.46 106.40*). Trên kênh truyền vệ tinh có thể thay đổi tham số về khả năng hư hỏng và mất mát gói dữ liệu để sổ thể quan sát hiệu quả các phương pháp cải thiện hiệu năng TCP.
- Từ trạm mặt đất bên thu kết nối tới trạm thu (Server) có tốc độ 10Mbps, độ trễ 2ms, không có lỗi khi kết nối. Kết nối này có thể là kết nối quang hoặc đồng tốc độ cao. Tương tự như trong LAN, giả định rằng không phát sinh lỗi truyền tin trên kết nối này.
- Trong mạng LAN, thực hiển triển khai trạm phát TCP. Trạm phát được cài đặt làm việc với giao thức TCP và có thể triển khai các biến thể của TCP khác như : Tahoe, Reno, NewReno, SACK…
- Trạm thu cũng được triển khai giao thức TCP và có khả năng đáp ứng với các TCP trên.
- Để tạo ra lưu lượng lưu thông trên mạng, sử dụng giao thức truyền dữ liệu FTP. Giao thức FTP được cài đặt trên tầng TCP của trạm phát và có thể truyền số liệu liên tục trong suốt thời gian thực nghiệm.
- Trong bài mổ phỏng, kích thước mỗi gói tin là 1000 byte – không đổi trong tất cả các kịch bản mô phỏng để dễ dàng quan sát cách xử lý của cửa sổ tắc nghẽn cwnd và phương pháp làm việc của các thuật toán TCP và các biện pháp nâng cao hiệu suất.
- Tại tầng TCP của trạm phát, thiết lập hệ thống quan sát quá trình làm việc của TCP, trong đó thực hiện ghi lại các giá trị : Số tuần tự phát (t_seq_no); cửa sổ tắc nghẽn (cwnd_) và các thông số truyền dẫn khác để có thể tính toán các định đặc tính đường truyền. Chu kì ghi ở đây là 0.5s. Các tham số này được sử dụng để khảo sát đánh giá hiệu quả của các thuật toán điều khiển tắc nghẽn cơ sở và các phương pháp cải tiến hiệu năng TCP khác nhau.

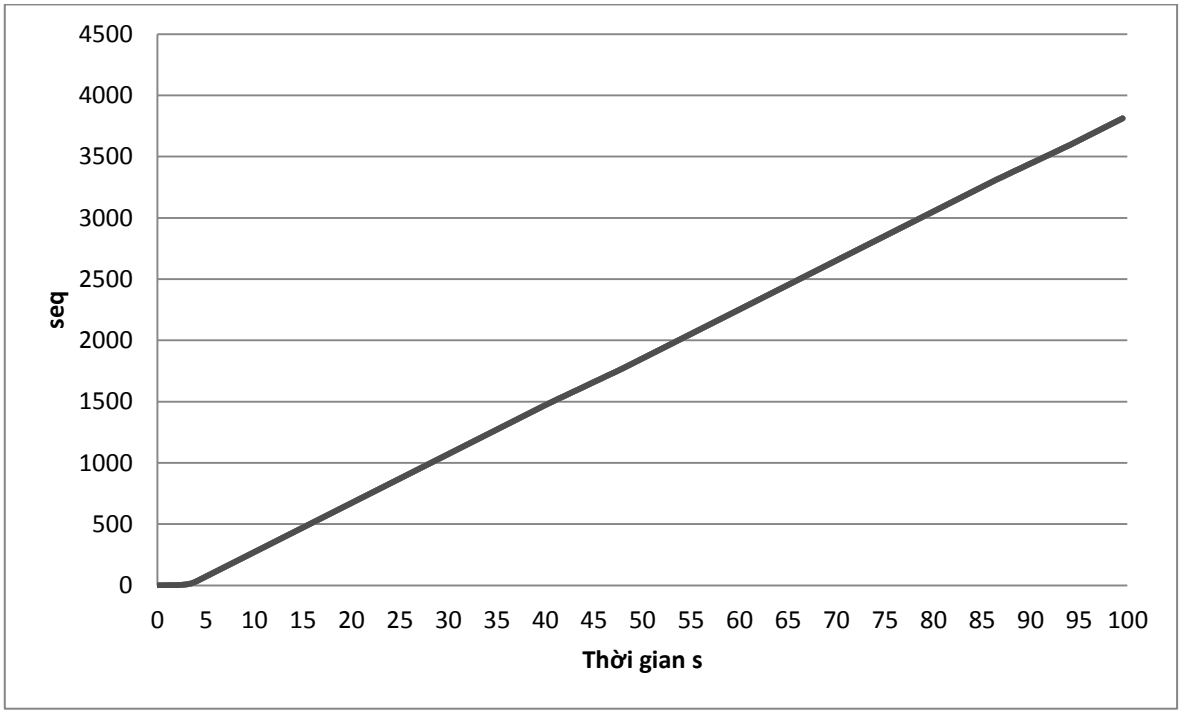
Như vậy, mô hình mô phỏng này được thiết lập là một hệ thống truyền thông vệ tinh có những thông số sát với thực tế. Đồng thời có thể thay đổi khả năng mất mát thông tin, tỷ lệ lỗi để thấy được hiệu quả làm việc của các phương pháp cải tiến hiệu năng. Giúp kết quả thu được có tính thực tế cao hơn.

Kịch bản thử nghiệm 1 :

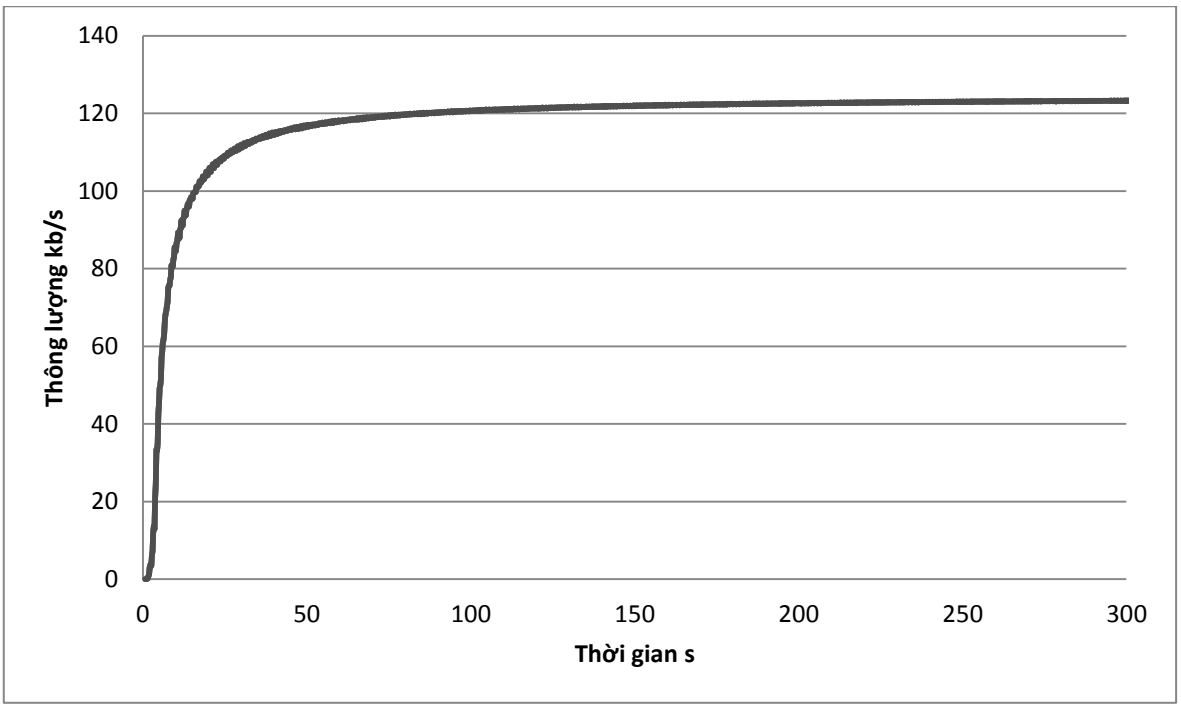
Không có lỗi và mất mát thông tin trên liên kết vệ tinh



Đồ thị 3.1 Đồ thị cwnd của các TCP khi không có mất mát trên E2E vệ tinh



Đồ thị 3.2 Đồ thị seq của các TCP khi không có sự mất mát trên E2E vệ tinh



Đồ thị 3.3 Đồ thị thông lượng các TCP khi không có mất mát trên E2E vệ tinh

Đánh giá : Thông qua các Đồ thị có thể thấy, khi *trên kênh vệ tinh không có sự mất mát thông tin (lý tưởng) thì các thuật toán TCP Tahoe, Reno, NewReno và SACK thực thi hoàn toàn giống nhau.*

Đồ thị cửa sổ tắc nghẽn 3.1 là một đường đồng biến. Chứng tỏ không có sự xung đột và va chạm trong quá trình truyền dữ liệu. Tức là kết quả mô phỏng phản ánh đúng điều kiện đặt ra ban đầu.

Thêm nữa, từ Đồ thị 3.1 và 3.2, trong vòng 4 giây đầu, cwnd và seq tăng chậm. Điều này là do ảnh hưởng của thuật toán *"Bắt đầu chậm"*. Đến khi cwnd = ssthresh (từ giây thứ 4), thuật toán "Tránh tắc nghẽn" hoạt động và với mỗi thông báo ACK nhận được cwnd = cwnd + 1/cwnd, nên cwnd tăng nhanh theo đồ thị hàm Hypebol.

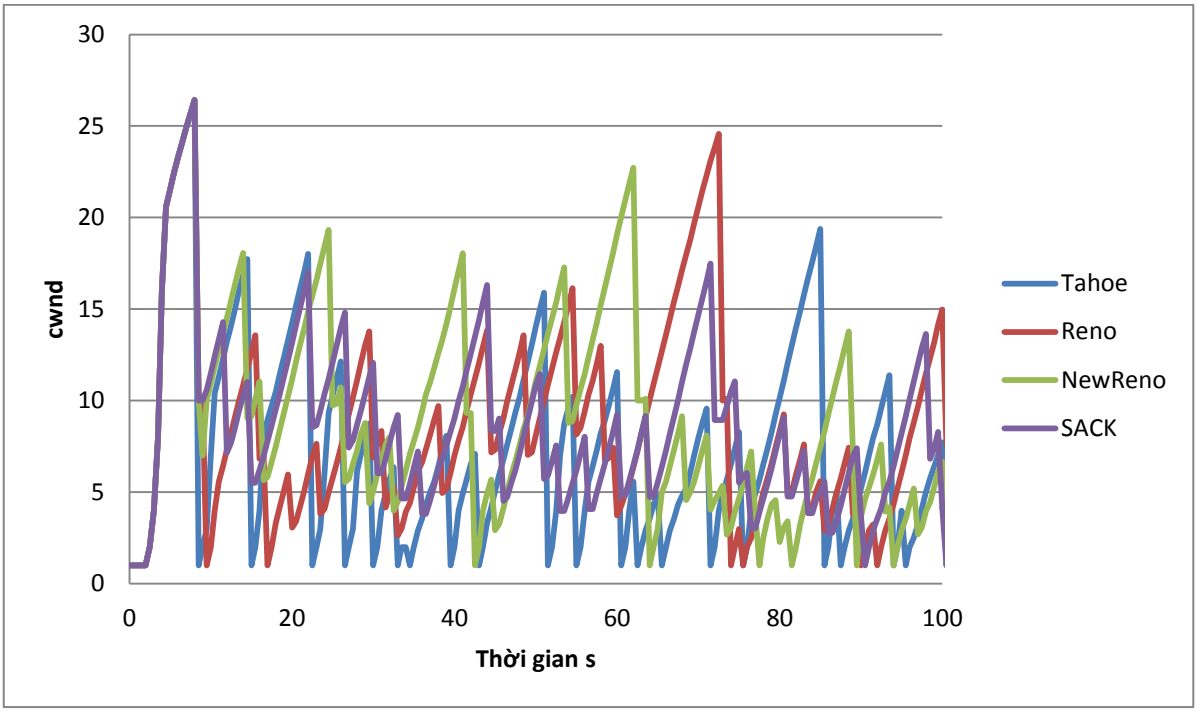
Dựa vào Đồ thị 3.3 , xác định được thông lượng tối đa có thể đạt được khi sử dụng các giao thức Tahoe – TCP, Reno – TCP, NewReno – TCP và SACK – TCP trên kênh vệ tinh địa tĩnh GEO là như nhau, khoảng 125 kbps.

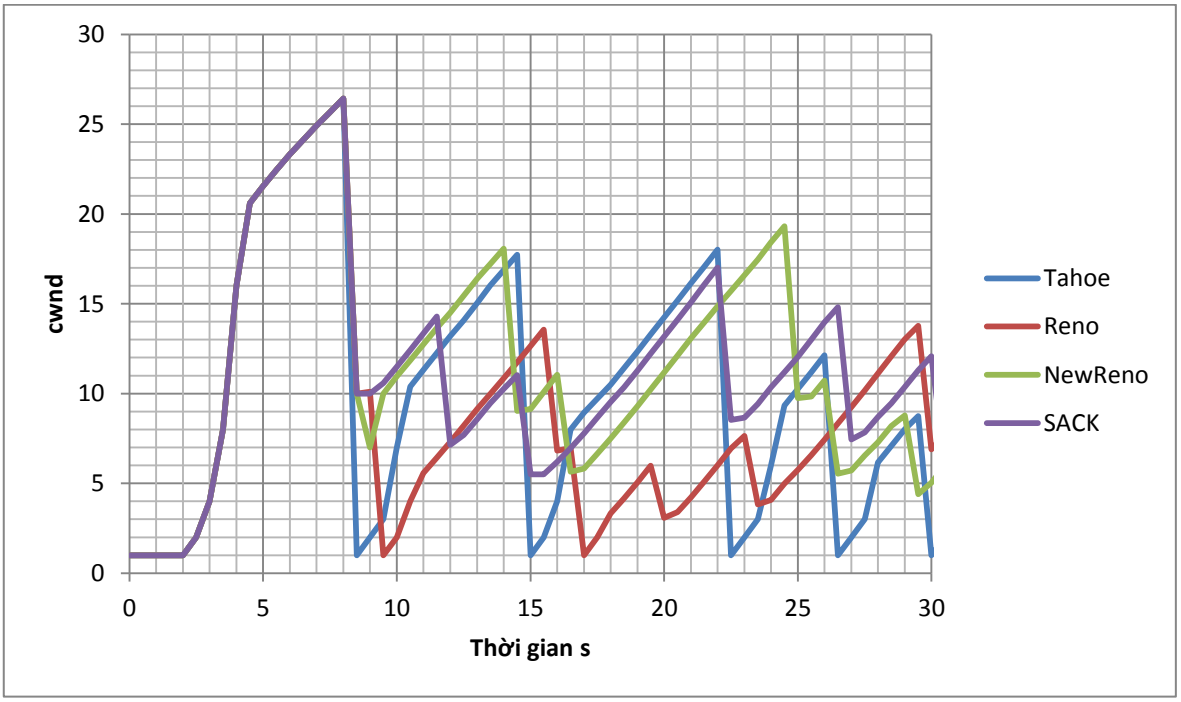
Nguyên nhân của sự chênh lệch rất lớn giữa thông lượng tối đa có thể đạt được (125 kbps) với băng thông kênh truyền vệ tinh (2048 kbps) là do độ trễ lớn của kênh vệ tinh (do đã loại bỏ mọi khả năng lỗi do mất mát). Mô hình giả lập này có độ trễ toàn phần RTT = 510 ms (chưa kể đến độ trễ tại các node). Đề đạt được tốc độ băng thông kênh truyền là 2048 kbps thì theo lý thuyết, kích thước cửa sổ tắc nghẽn cwnd = throughput * RTT = 130.56 KByte. Kích thước này lớn hơn nhiều so với kích thước cửa sổ tối đa là 64 Kbyte [3].

Như vậy, dựa vào kịch bản thử nghiệm này đã xác định được sự tồn tại của thuật toán "bắt đầu chậm" và "tránh tắc nghẽn", ngoài ra, còn xác định được thông lượng tối đa có thể đạt trên kênh vệ tinh của các TCP và ảnh hưởng của độ trễ tới hiệu năng mạng vệ tinh.

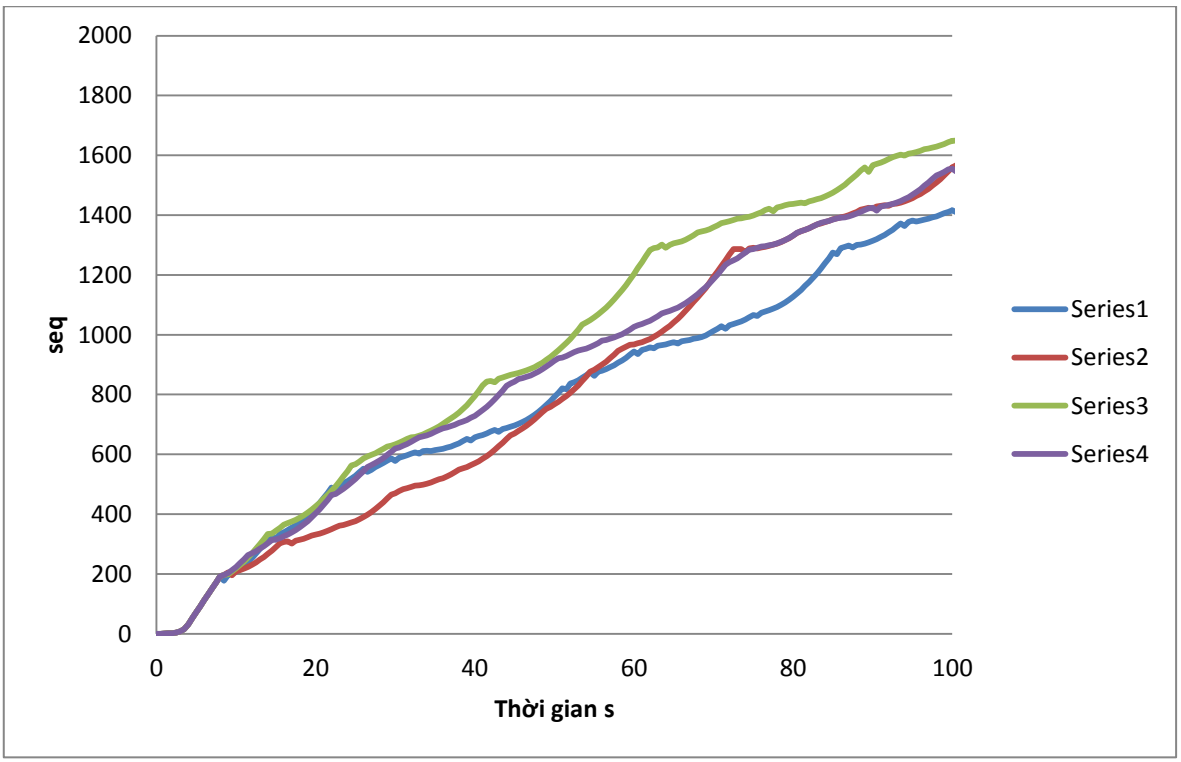
Kịch bản thử nghiệm 2 :

Có lỗi và mất mát thông tin trên liên kết vệ tinh. Sự mất mát trễn mỗi chặng liên kết vệ tinh là 2% số lượng gói tin truyền qua.

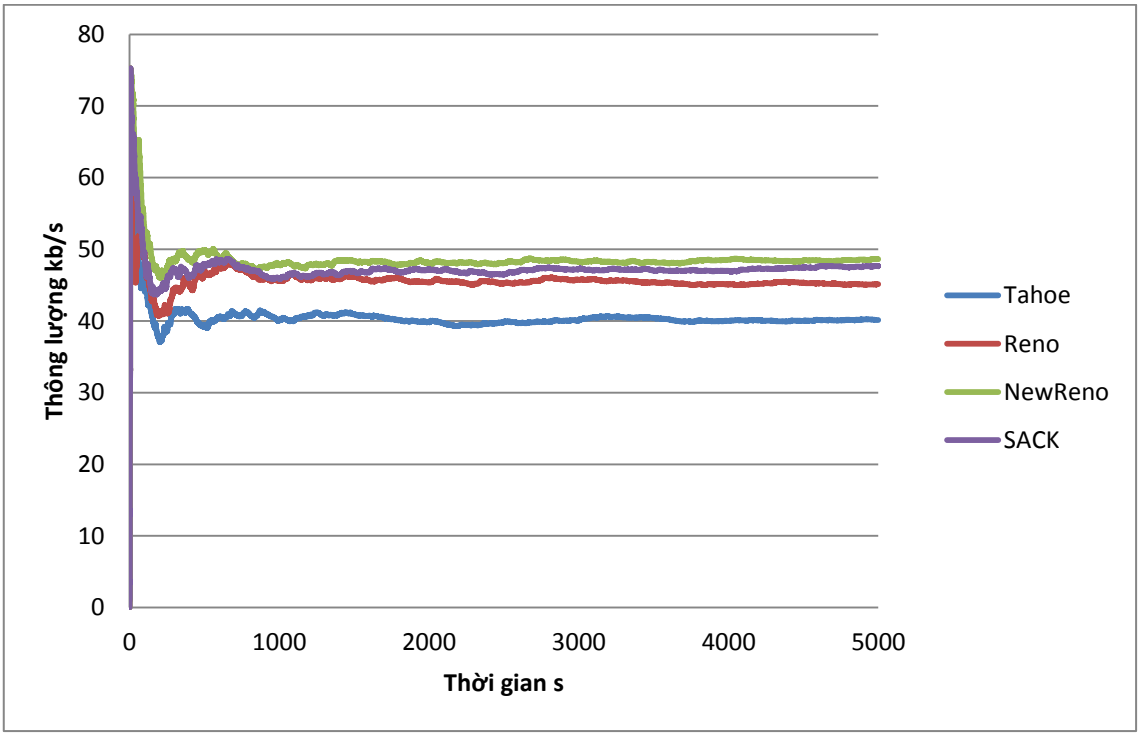


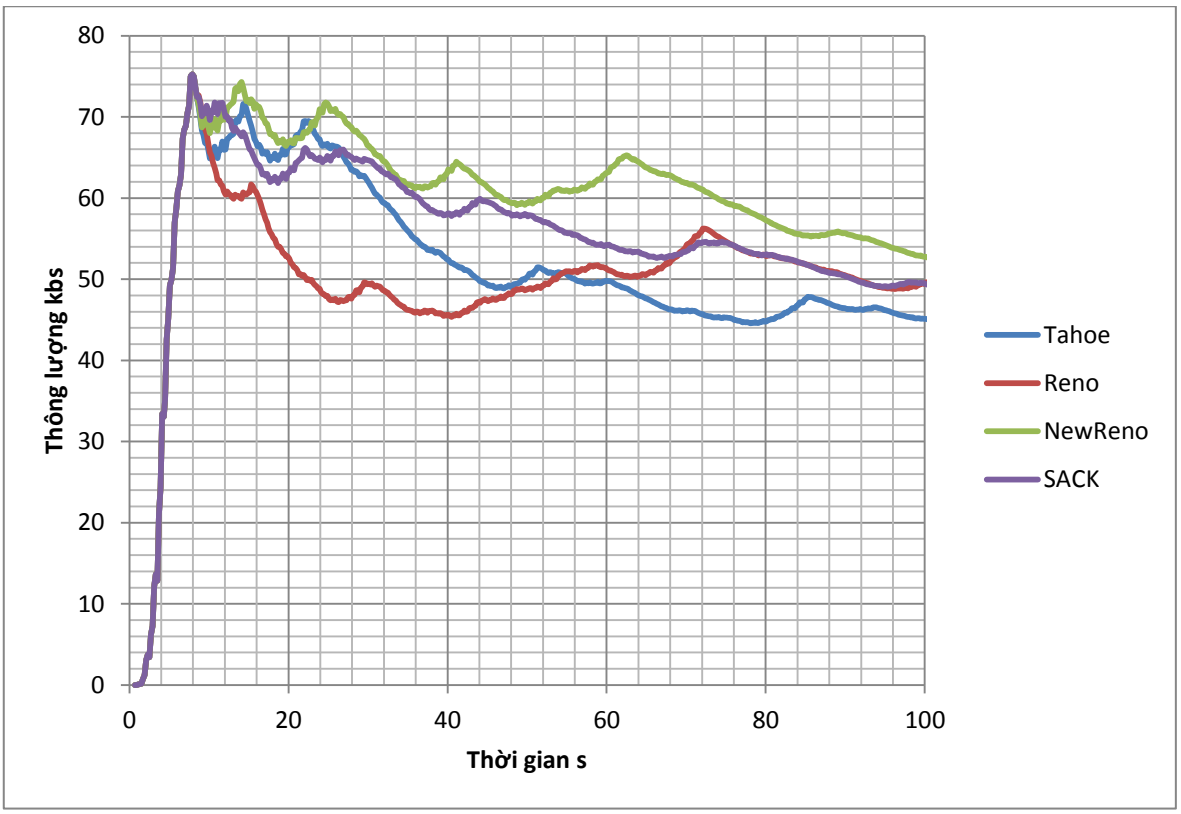


Đồ thị 3.4 Đồ thị cwnd của các TCP khi có mất mát trên E2E vệ tinh



Đồ thị 3.5 Đồ thị seq của các TCP khi có mất mát trên E2E vệ tinh





Đồ thị 3.6 Đồ thị thông lượng của các TCP khi có mất mát trên E2E vệ tinh

Đánh giá

Trong kịch bản này, sử dụng điều kiện mất mát là 2% trên mỗi kết nối vệ tinh. Đây là một tỷ lệ lỗi lớn so với các kết nối thông thường, sử dụng tỷ lệ lỗi này một phần để phản ánh những ảnh hưởng của điều kiện thời tiết đến kết nối vệ tinh.

Dựa vào Đồ thị cwnd và seq, có thể thấy hiệu quả khác nhau của các TCP trên kênh vệ tinh trong điều kiện có mất mát thông tin. Trong khoảng 8 giây đầu, các biến thể TCP đều thực hiện "bắt đầu chậm" do đó có sự tương đồng về mặt đồ thị. Sau đó, bắt đầu có lỗi xảy ra và cách xử lý của mỗi thuật toán là khác nhau nên thấy được ngay sự thay đổi :

Nhờ khả năng phát hiện nhiều dữ liệu mất mát trên một cửa sổ, SACK – TCP và NewReno – TCP có thể duy trì cwnd ở mức cao do đó tần số phát vẫn đảm bảo khi xảy ra lỗi trên đường truyền. Ngoài ra, nhờ kết hợp "Truyền lại nhanh" của Reno và cơ chế Time-out của Tahoe nên SACK rất ít khi phải thực hiện lại "Bắt đầu chậm" (thiết lập cwnd = 1). Đồng thời nhờ cơ chế ACK có lựa chọn, SACK có thể đồng thời nhận biết nhiều gói tin bị mất về thực hiện truyền lại ngay, do đó cwnd của SACK luôn được duy trì ổn định và không bị giảm đột ngột khi gặp lỗi.

Trong khi đó, NewReno nhờ cải tiến loại bỏ thời gian chờ khi nhiều gói tin bị mất so với Reno nên NewReno không thoát khỏi "Phục hồi nhanh" cho đến khi các gói dữ liệu outstanding được xác định. Vì thể NewReno không cần giảm cwnd nhiều lần khi mất nhiều gói như Reno, do đó cwnd của NewReno luôn được duy trì ở mức cao và có phần nhỉnh hơn so với SACK – TCP.

Dựa vào Đồ thị 3.6 có thể thấy sự chênh lệch giữa Tahoe và Reno thể hiện rõ ràng hơn so với hai TCP kia. Ở khoảng 80s đầu của kịch bản, hiện tượng mất gói xảy ra liên tục và sau khi nhận được 3 dupACK, Tahoe liên tục thực hiện "Bắt đầu chậm" (cwnd =1) và truyền lại gói bị mất. Vì thì thông lượng của Tahoe liên tục giảm trong khoảng 100s đầu và dần dần về mức cân bằng.

Trong khi đó, Khi nhận được 3 dupACK, Reno không quay về "Bắt đầu chậm" mà chuyển sang "Phục hồi nhanh" tức là thiết lập ssthresh = cwnd/2 rồi cwnd = ssthresh + 3*smss, nên khi gặp lỗi liên tục Reno liên tục phải giảm cwnd, do đó ban đầu Reno duy trì cwnd ở mức thấp hơn so với Tahoe. Nhưng khi không gặp lỗi, nhờ "Phục hồi nhanh" Reno duy trì mức cửa sổ cao hơn so với Tahoe. Do đó mang lại hiệu quả lâu dài hơn so với Tahoe.

Đồ thị 3.7 thể hiện hiệu quả của các TCP lần lượt giảm dần theo thứ tự NewReno > SACK > Reno > Tahoe. Như vậy, với điều kiện kênh vệ tinh có lỗi, Tahoe TCP là kém hiệu quả nhất. Trong khi đó, NewReno và SACK là hai TCP có hiệu quả tốt nhất.

Ở kịch bản này, thông lượng tối đa có thể đạt được là 75 kbps (tại thời điểm bắt đầu gặp lỗi – giây thứ 8) và sau đó duy trì ổn định ở khoảng 50 – 40 kbps, tương đương với 40 – 32 % so với thông lượng cực đại trong điều kiện không có lỗi. Từ đó thấy được mức độ ảnh hưởng của môi trường truyền dẫn đến hiệu năng TCP trên kênh vệ tinh.

Như vậy, từ kịch bản thử nghiệm này ta nhận thấy sự có mặt của thuật toán "Phục hồi nhanh" và hiệu quả của cải tiến NewReno so với Reno, hiệu quả của cải tiến SACK so với Reno và Tahoe. Ta còn xác định được thông lượng tối đa mô hình có thể đạt được trong điều kiện có mất mát thông tin mà qua đó thấy được mức độ ảnh hưởng của môi trường đến hiệu năng hoạt động của các TCP.

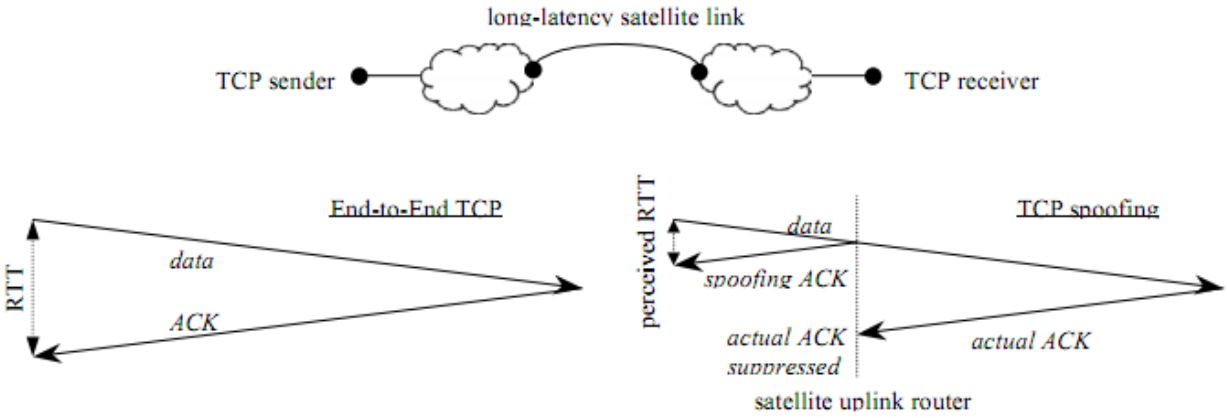
4. Một số phương pháp cái tiến hiệu năng TCP

4.1. Cải tiến hiệu năng bằng phương pháp chia cắt

4.1.1. TCP Spoofing

TCP Spoofing là kỹ thuật sử dụng một cổng kết nối trung gian để sớm trả lời trạm phát về gói tin TCP mà không cần chờ thông tin trả lời thật từ trạm thu. Điều này làm trạm phát lầm tưởng rằng mạng có độ trễ nhỏ và vì thế giai đoạn "Bắt đầu chậm" của TCP tiến hành một cách nhanh chóng hơn. Cổng kết nối trung gian có tác dụng làm vùng đệm cho các gói tin TCP qua nó, khi thông tin phản hồi thực sự từ trạm thu trở lại kết nối này, nó sẽ loại bỏ để ngăn ngừa sự trùng lặp thông tin trả lời đến trạm phát. Nếu thông tin phản hồi của trạm thu không trở lại hoặc thời gian chờ vượt quá ngưỡng cho phép của cổng kết nối, nó sẽ phát lại phần thông tin bị mất từ bộ đệm cục bộ của mình [7].

TCP Spoofing phá vỡ khái niệm kết nối end-to-end về mặt bản chất khi trạm phát nhầm tưởng phân đoạn dữ liệu đã đến đích, trong khi thực chất gói tin vẫn đang được chuyển tiếp quá giang thông qua cổng kết nối trung gian.



Hình 4.1 Lược đồ TCP Proofing [8]

4.2. Cải tiến hiệu năng TCP bằng phương pháp Link Layer

Ý tưởng chính là nâng cao hiệu năng hoạt động TCP bằng cách hỗ trợ ở tầng Link Layer. Thay vì sử dụng phương thức hoạt động truyền lại End-to-End, phương pháp này áp dụng việc truyền lại cục bộ ở tầng link. Khi hoạt động trong môi trường lỗi nhiều như môi trường vệ tinh, việc truyền lại từng chặn giảm khả năng lỗi trên đường truyền khi phải truyền dữ liệu trên một quãng đường dài.

4.2.1. Giao thức Snoop TCP

PEPs (Performace Enhancement Proxies) là phương pháp được dùng đến để giảm hiện tượng hiệu năng hạ thấp do ảnh hưởng từ đặc điểm của các đường kết nối riêng biệt. Giao thức Snoop là một trong các phương pháp đó. Mục tiêu lớn nhất của Snoop là cải thiện hiệu năng của truyền thông thông qua liên kết không dây và liên kết vệ tinh mà không cần tác động truyền lại và biến đổi cửa sổ tại tầng giao vận [9].

Thực chất Snoop TCP là một biến thể của TCP Spoofing. Nếu TCP Spoofing tự thực hiện việc trả lời ACK thì Snoop TCP chỉ chuyển tiếp ACK khi nhận được trả lời từ trạm thu. Tương tự TCP Spoofing, Snoop TCP cũng là trạm trung gian sử dụng bộ đệm và đồng hồ để kiểm soát việc phát lại. Tuy nhiên, giữa chúng cũng có sự khác biệt cơ bản : TCP Spoofing được triển khai trên tầng 4 của mô hình OSI, vì thể hình thức chia cắt TCP khi triển khai trên thực tế là phải tác động đến đầu cuối truyền số liệu, thực thể phát và trạm Spoofing phải triển khai một bộ giao thức làm việc để có thể gián tiếp truy nhập đến thực thể thu. Trong khi đó, Snoop TCP được triển khai chủ yếu trên tầng 2 của mô hình OSI, do đó trong suốt với các thực thể đầu cuối, các thực thể đầu cuối có thể làm việc và giao tiếp trực tiếp với nhau thông qua các Snoop mà không cần phải triển khai đồng bộ một giao thức đặc biệt nào khác.

Giao thức Snoop được thiết lập chạy trên trạm Snoop và được triển khai trên trạm mặt đất của hệ thống thông tin vệ tinh hoặc mạng không dây. Trạm làm việc giám sát các gói tin được truyền qua và lưu lại các gói tin tại vùng nhớ đệm. Sau khi lưu đệm, gói tin được chuyển tiếp đến đích đồng thời giám sát các thông tin trao đổi phản hồi.

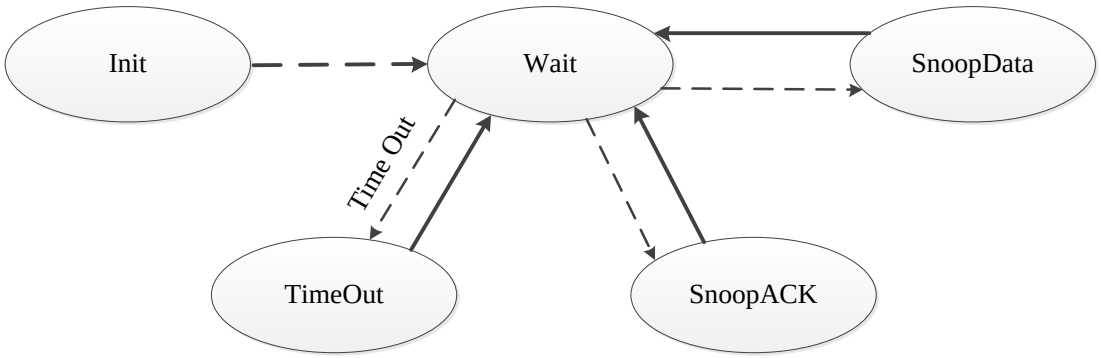
Quy tắc của trạm Snoop là lưu trữ tạm thời các gói tin cho kết nối TCP. Khi dữ liệu bị mất (thể hiện bằng việc nhận được các dupACKs), trạm Snoop sẽ gửi lại các gói tin bị mất từ vùng đệm. Vì thế, tầng TCP sẽ không nhận thấy việc mất gói tin và do đó thuật toán điều khiển tắc nghẽn không bị kích hoạt. Thêm vào đó, trạm Snoop khởi động đồng hồ kiểm soát việc phát lại cho từng kết nối TCP. Khi đồng hồ kiểm soát phát lại quá thời gian, trạm Snoop sẽ phát lại các gói tin không nhận được thông báo trả lời. Đồng hồ này được gọi là đồng hồ ổn đinh (persist) bởi vì không giống các đồng hồ phát lại TCP, nó có một giá trị cố đinh.

Giao thức Snoop chặn lại các gói TCP, phân tích chúng và phát lại chúng nếu cần thiết. Kết quả là không có định dạng gói tin bổ sung nào được đưa vào trong giao thức, tức là tất cả các gói tin gửi hay nhận đều không bị thay đổi và vẫn theo dạng của giao thức TCP.

Nhược điểm của Snoop TCP là đòi hỏi phải có một thiết bị trung gian tốn kém, đủ mạnh để có thể lưu đệm gói tin, chuyển tiếp và phát lại nếu có sự cố. Thiết bị thuộc loại này thường rất tốn kém. Ngoài ra Snoop TCP không hoạt động với cơ chế bảo mật, do không can thiệp vào các tầng trên tầng liên kết.

4.2.2. Hoạt động của trạm Snoop

Mô hình trạng thái làm việc của trạm Snoop



Hình 4.2 Mô hình trạng thái làm việc của trạm Snoop

Bộ đệm Snoop (Snoop Cache) : Được sử dụng để lưu trữ tạm thời các gói tin nhận được ở tầng trên. Nội dung của bản lưu trữ tạm thời chứa bản sao của các gói tin và thông tin điều khiển giao diện (ICI – Interface Control Information).

Bảng lưu kết nối Snoop (Snoop Connection Table) : Trạm Snoop duy trì bảng thông tin theo dõi các kết nối TCP. Bảng này là cần thiết do hai trạm kết nối có thể tồn tại rất nhiều các liên kết thiết lập và mỗi bộ kết nối đó cần được duy trì một cách riêng biệt (do số tuần tự không phải là duy nhất cho các kết nối riêng biệt) . Mỗi thông tin trong bảng lưu kết nối Snoop xác định một kết nối TCP bằng địa chỉ IP nguồn – đích và kết hợp với cổng TCP. Chúng cũng giữ lại số tuần tự sau cùng và thông báo phản hồi ACK sau cùng của mỗi liên kết. Ngoài ra, mỗi liên kết được bổ sung thêm tham số *timeout_evt* để xác định thời gian truyền lại.

Trạng thái khởi tạo(Init State) : Quá trình xử lý của trạm Snoop được bắt đầu ở trạng thái khởi tạo khi mà bảng lưu kết nối Snoop và bộ đệm Snoop được thiết lập.

Trạng thái chờ (Wait State) : Sau khi phiên khởi tạo được thiết lập, quá trình xử lý chuyển từ "*trạng thái khởi tạo"* sang "*trạng thái chờ"*. Khi đó, tiến trình tiếp tục duy trị *ở "trạng thái chờ"* đến khi :

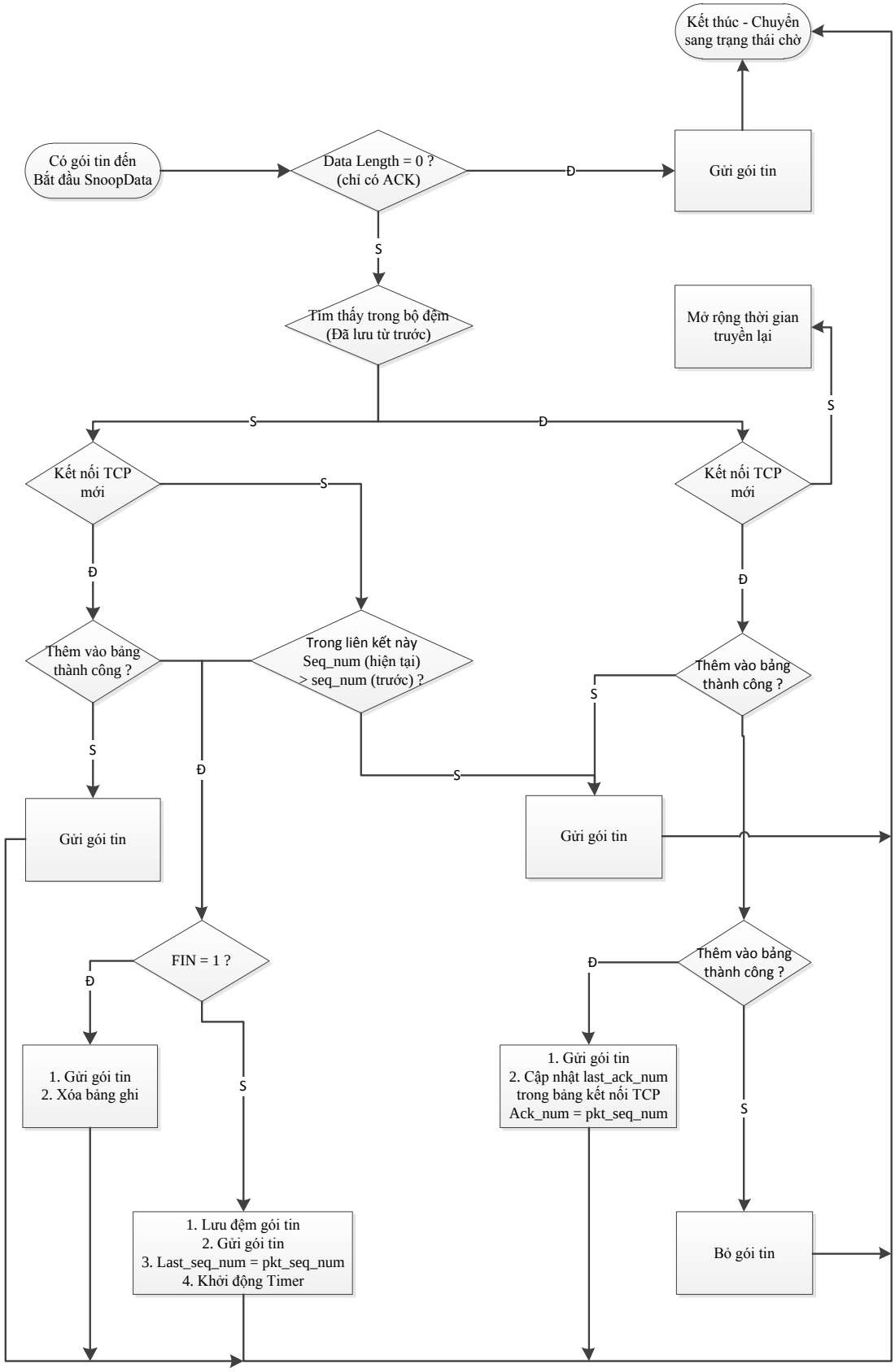
- Xuất hiện gói tin đến từ lớp MAC
- Xuất hiện gói tin đến từ tầng trên tầng TCP/IP
- Đồng hồ xác định thời gian truyền lại hết hạn.

Tất các các sự kiện đều như lệnh ngắt. Việc gói tin đến được hiểu là một luồng ngắt. Sự kiện đồng hồ xác định thời gian truyền lại hết hạn được thực hiện như một lệnh ngắt đã được lập lịch từ trước.

Trạng thái Snoop dữ liệu (SnoopData Sate): Khi một gói tin từ tầng trên tầng TCP/IP tới, quá trình chuyển trạng thái từ *"trạng thái chờ"* sang *"trạng thái Snoop dữ liệu"*. Trạm Snoop lưu lại số tuần tự cuối cùng lấy được từ tầng cao hơn và gói tin được xử lý tùy vào số tuần tự của nó :

- Nếu gói tin là mới với số tuần tự TCP bình thường : Gói tin mới có số tuần tự cao hơn là trường hợp bình thường, khi đó gói tin được thêm vào bộ đệm Snoop và chuyển tiếp đến tầng thấp hơn.
- Nếu gói tin có số tuần tự không đúng, và đã được lưu đệm trước đây : Điều này xảy ra khi các gói tin mất gây ra timeout tại thực thể phát.
    - Nếu số ACK cao hơn số ACK cuối cùng trạm Snoop nhận được, nó sẽ cho rằng gói tin đã bị mất. Khi đó gói tin sẽ được chuyển tiếp đến đích.
    - Còn nếu số tuần tự nhỏ hơn số tuần tự cuối cùng trạm Snoop nhận được, gói tin sẽ bị loại bỏ.

  Sau khi xử lý xong gói tin, quá trình xử lý Snoop sẽ chuyển lại về "trạng thái chờ" và chờ sự kiện tiếp sau.

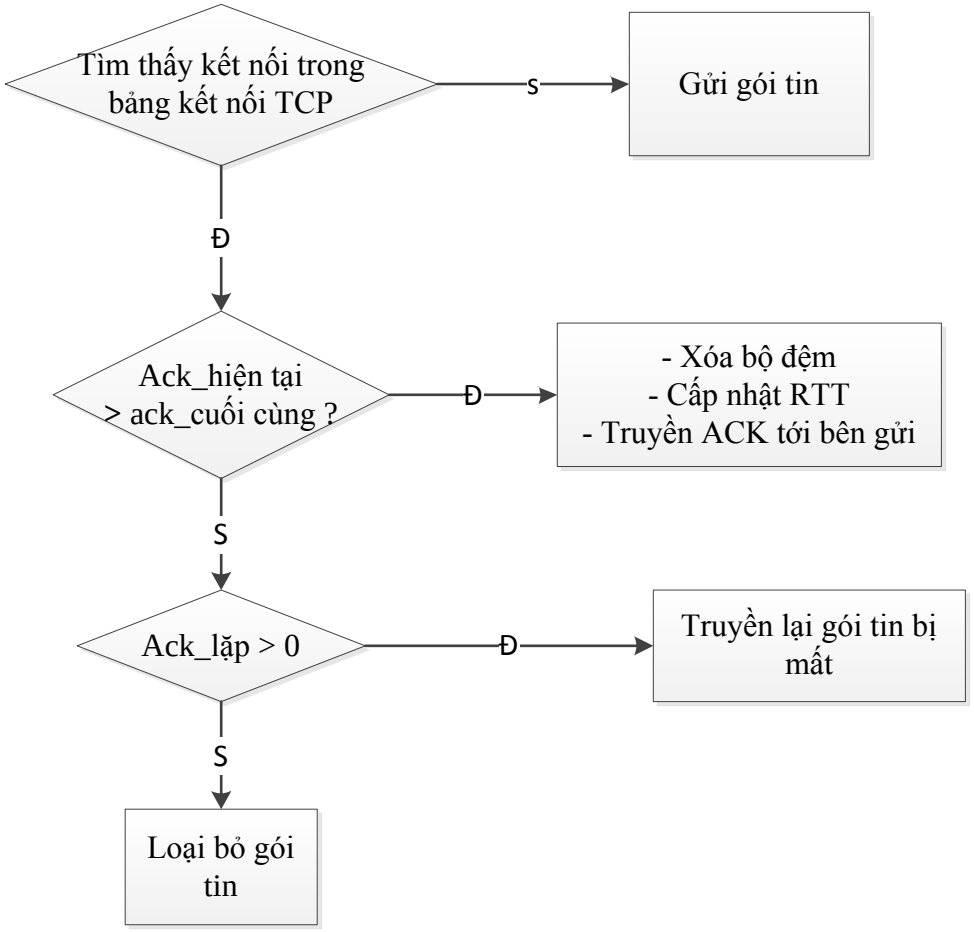


Hình 4.3 Sơ đồ thuật toán SnoopData

Trạng thái phản hồi thông tin (SnoopACK Sate) : Khi trạm Snoop nhận được gói tin từ lớp MAC, quá trình xử lý Snoop sẽ chuyển từ "trạng thái chờ" sang

"trạng thái phản hồi thông tin". Gói tin sẽ được xử lý dựa trên số tuần tự phản hồi ACK :

- Nếu thông báo ACK mới : Tức là gói tin ACK với số tuần tự phản hồi ACK cao hơn số cuối cùng trạm Snoop nhận được. Gói tin này khởi động việc xóa bản ghi tương ứng ở vùng đệm. Thông tin phản hồi được chuyển tiếp lên tầng cao hơn TCP/IP
- Nếu thông báo ACK sai : Tức là gói tin ACK với số tuần tự phản hồi ACK thấp hơn số cuối cùng trạm Snoop nhận được. Điều này rất hiếm và thông báo ACK bị loại bỏ.
- Nếu là DupACKs : Tức là gói tin ACK giống với ACK cuối cùng trạm Snoop nhận được. Điều này làm cho trạm Snoop nhận biết rằng gói tin đã gửi với số tuần tự phát cao hơn đã bị mất. Khi đó, trạm Snoop sẽ gửi lại toàn bộ các gói tin bắt đầu từ gói tin bị mất đầu tiên.

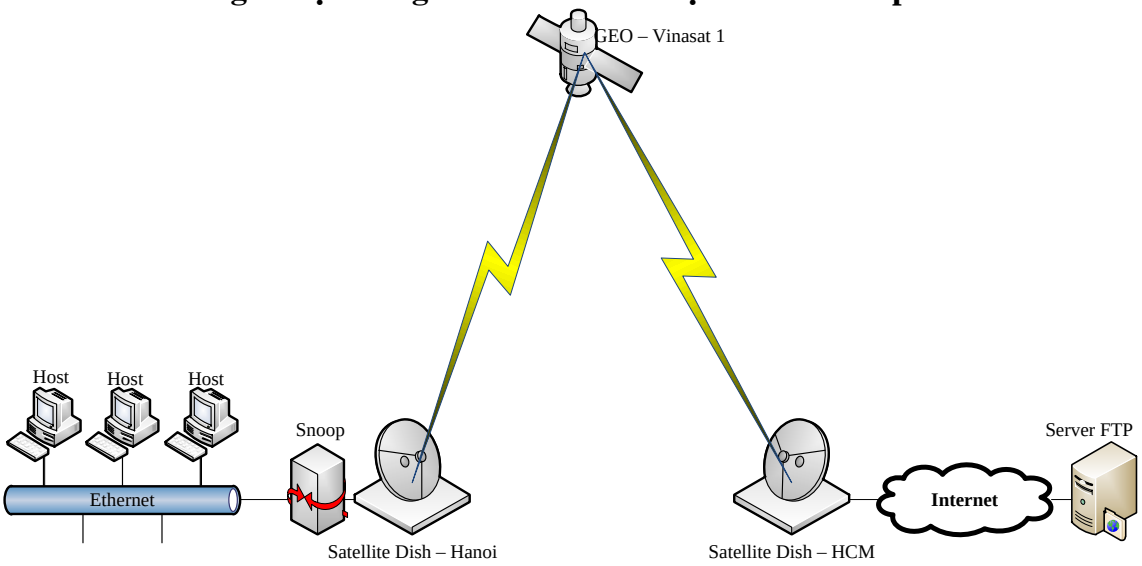


Hình 4.4 Sơ đồ thuật toán SnoopACK

Trạng thái quá ngưỡng thời gian (TimeOut State) : Khi đồng hồ kiểm soát thời gian phát lại của một kết nối hết hạn, quá trình xử lý chuyển từ *"trạng thái chờ"* sang *"trạng thái quá ngưỡng thời gian"*. Quá trình xử lý trong suốt "*trạng thái quá ngưỡng thời gian*" rất giống với quá trình xử lý khi nhận gói tin dupACK trong "*trạng thái phản hồi thông tin"* tức là : Tất các các gói tin không nhận được phản hồi ACK đều được truyền lại.

Đồng hồ kiểm soát thời gian phát lại của một kết nối được gia hạn khi trạm Snoop nhận được gói tin mới từ tầng cao hơn, một thông báo ACK từ tầng thấp hơn hoặc khi một đồng hồ phát lại hết hạn.

4.2.3. Đánh giá hiệu năng TCP trên kênh vệ tinh có Snoop TCP

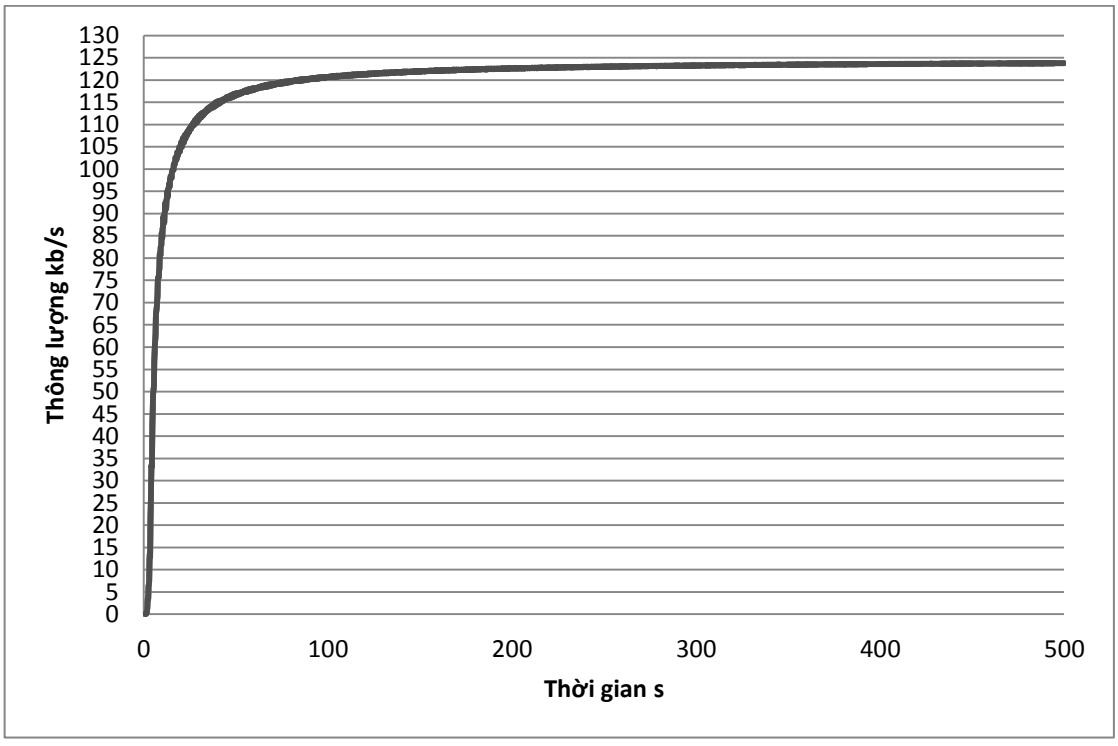


Hình 4.5 Mô hình vệ tinh sử dụng Snoop TCP

Kịch bản thử nghiệm 3 :Không có mất mát trên liên kết vệ tinh

Đồ thị 4.1 Đồ thị cwnd của các TCP khi không có mất mát trên Snoop vệ tinh

Đồ thị 4.2 Đồ thị seq của các TCP khi không có mất mát trên Snoop vệ tinh

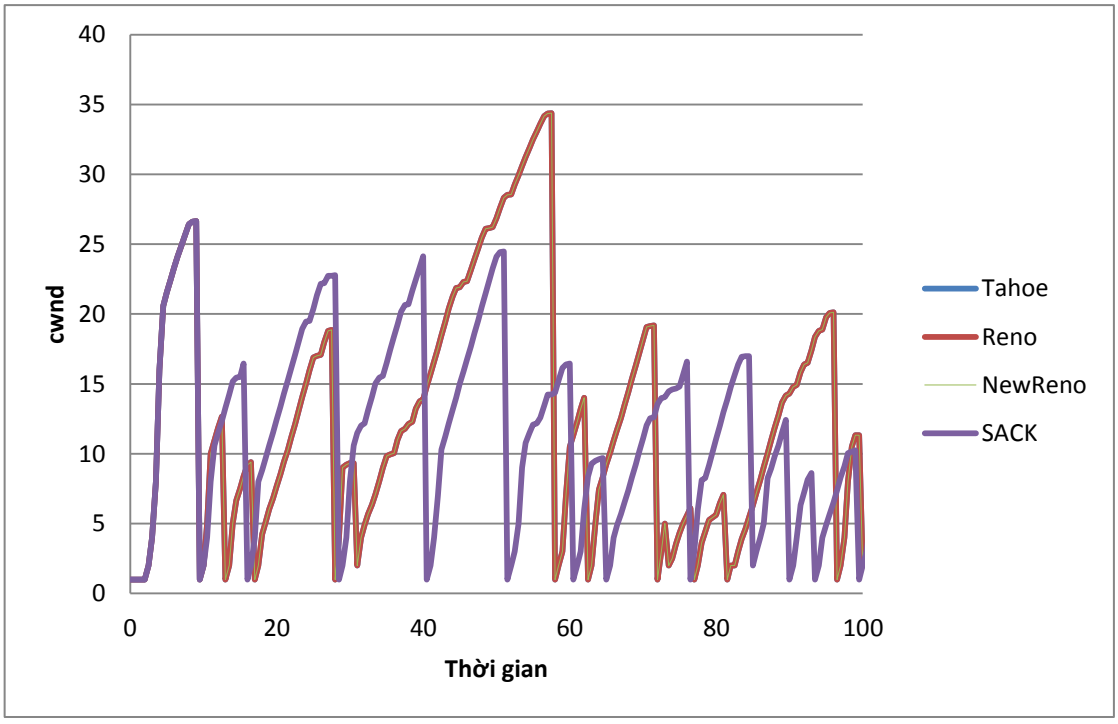


Đồ thị 4.3 Đồ thị thông lượng các TCP khi không có mất mát khi có Snoop

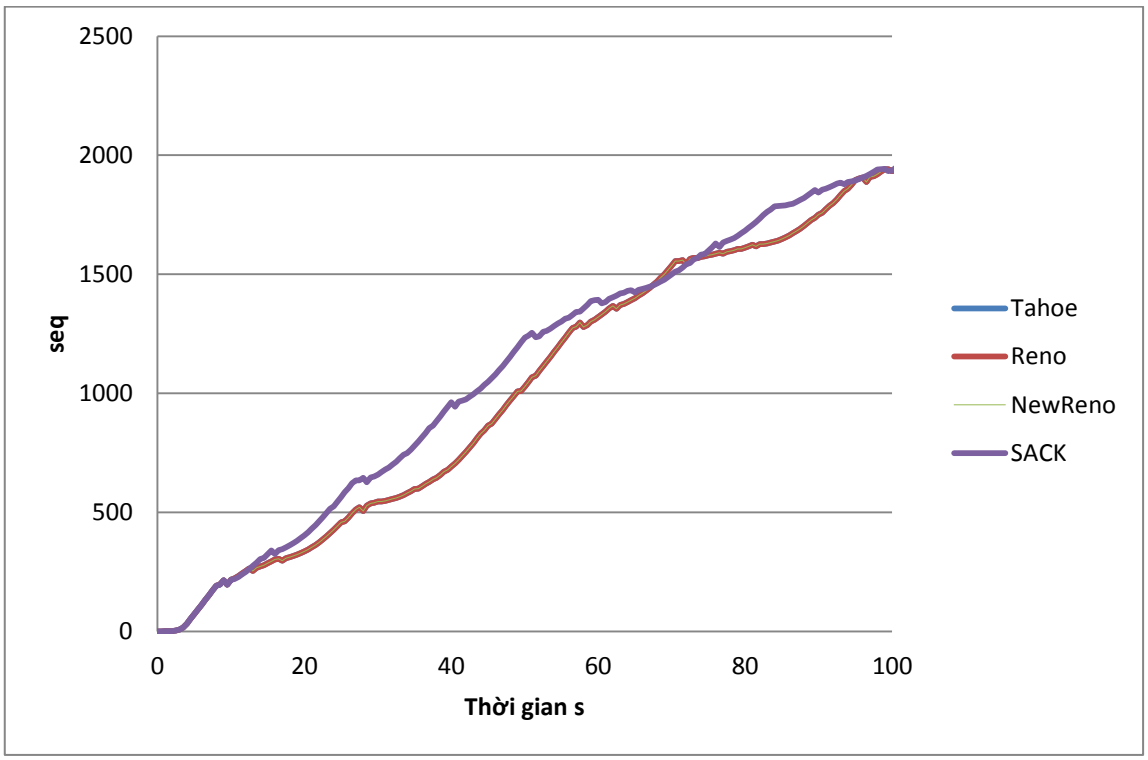
Đánh giá : Từ thử nghiệm này có thể kết luận : *Nếu kênh thông tin không bị mất mát thông tin, hoạt động của các Snoop TCP là như nhau*. Điều này có thể giải thích giống với kịch bản 1, là do các thuật toán hoạt động hoàn toàn như nhau trong điều kiện không có lỗi. Và vì không có lỗi xảy ra, nên hoạt động gửi

gói tin không gặp vấn đề, nên việc sử dụng Snoop thay vì Repeater cũng không mang lại điều gì khác biệt.

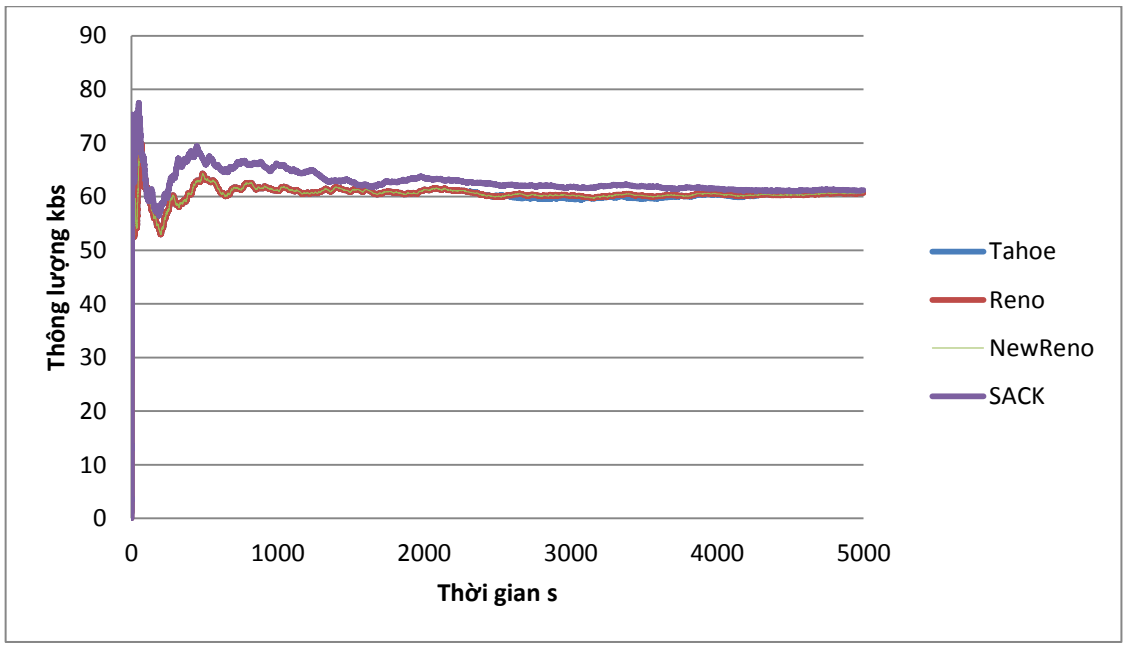
Kịch bản thử nghiệm 4 : Có lỗi và mất mát thông tin trên liên kết vệ tinh. Sự mất mát trễn mỗi chặng liên kết vệ tinh là 2% số lượng gói tin truyền qua.

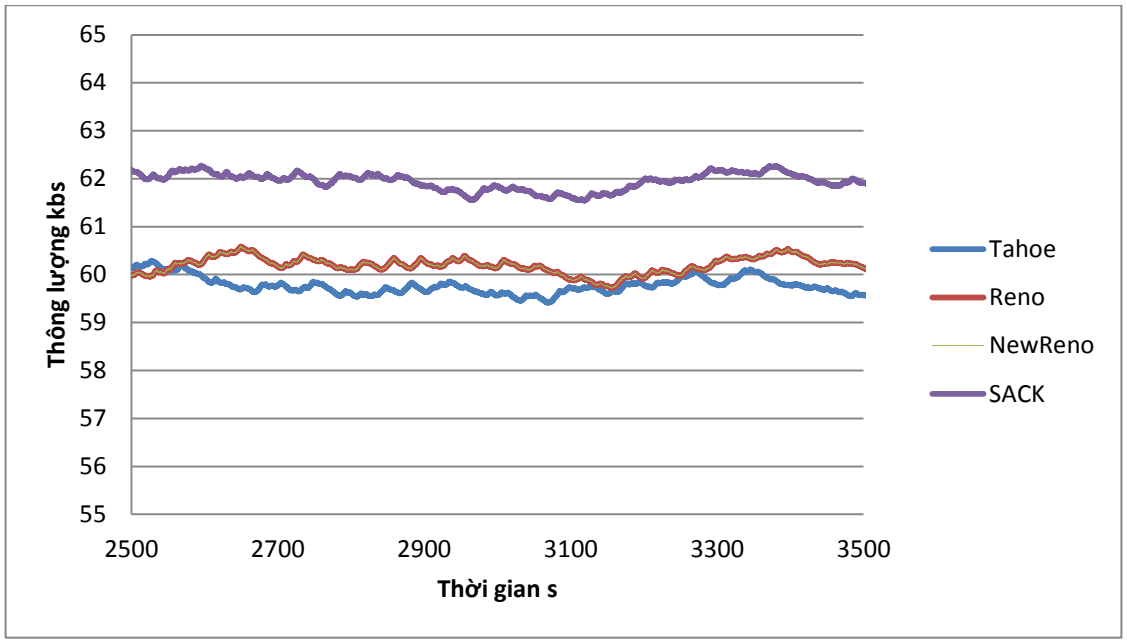


Đồ thị 4.4 Đồ thị cwnd của các TCP khi có mất mát trên Snoop vệ tinh



Đồ thị 4.5 Đồ thị cwnd của các TCP khi có mất mát trên Snoop vệ tinh





Đồ thị 4.6 Đồ thị thông lượng của các TCP khi có mất mát trên Snoop vệ tinh

Đánh giá : Khác với kịch bản thử nghiệm 3, trong trường hợp này bổ sung lỗi mất mát thông tin ngẫu nhiên ở cả hai chiều lên-xuống của kênh vệ tinh là 2%. Từ các Đồ thị, có thể xác định được :

Trong kịch bản thử nghiệm này, Hoạt động của Snoop Tahoe, Reno và NewReno là gần như giống nhau. Sự khác biệt là rất nhỏ và gần như không đáng kể. Điều này chứng tỏ tác động rất rõ ràng của Snoop TCP đối với các biến thể TCP sử dụng ACK tích lũy. Đây là điểm rất khác biệt so với kết nối E2E.

Snoop SACK có hiệu quả hơn so với các biến thể TCP còn lại. Thời gian đầu, khả năng mở của số các tất các các biến thể TCP là như nhau, tuy nhiên do có hỗ trợ Snoop, SACK tận dụng được đường truyền tốt hơn so với các TCP còn lại

(SACK hỗ trợ gửi lại đồng thời nhiều gói tin bị lỗi trong mỗi RTT) nên SACK phục hồi nhanh hơn và duy trì hiệu quả mở cửa sổ tốt hơn.

Thông lượng trung bình của các TCP trong kịch bản này là 60 – 62 kbps, tức là tương đương với khoảng gần 3% so với băng thông vệ tinh (2048kbps)

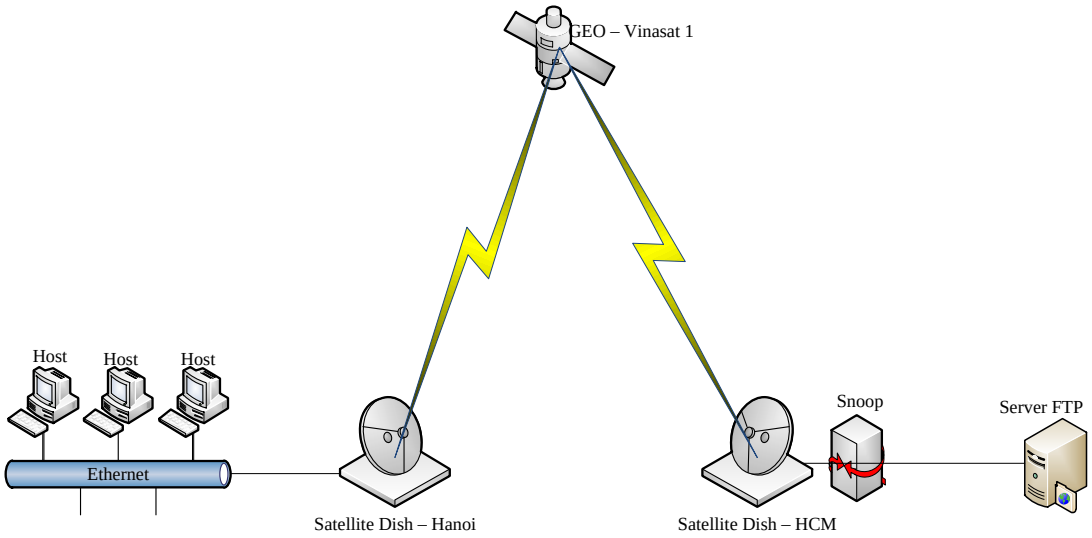
4.2.4. Mở rộng kịch bản Snoop TCP

Trong mô hình mô phỏng của bài đồ án, cũng như trên một số các công trình nghiên cứu khác. Hầu như chỉ sử dụng trạm Snoop đặt cạnh trạm phát (mặt đất). Do đó, tôi đề xuất đưa ra kịch bản thử nghiệm với trường hợp trạm Snoop đặt gần trạm thu (mặt đất) với cùng một mô hình thử nghiệm ban đầu, để việc so sánh được nhất quán.

Do trong điều kiện không có mất mát thông tin, trạm Snoop hoạt động như một Repeater bình thường mà không làm ảnh hưởng gì tới hiệu năng nên tôi chỉ thử nghiệm với điều kiện tồn tại mất mát thông tin trên kênh vệ tinh để có thể thấy được hiệu quả của các thử nghiệm một cách đầy đủ nhất.

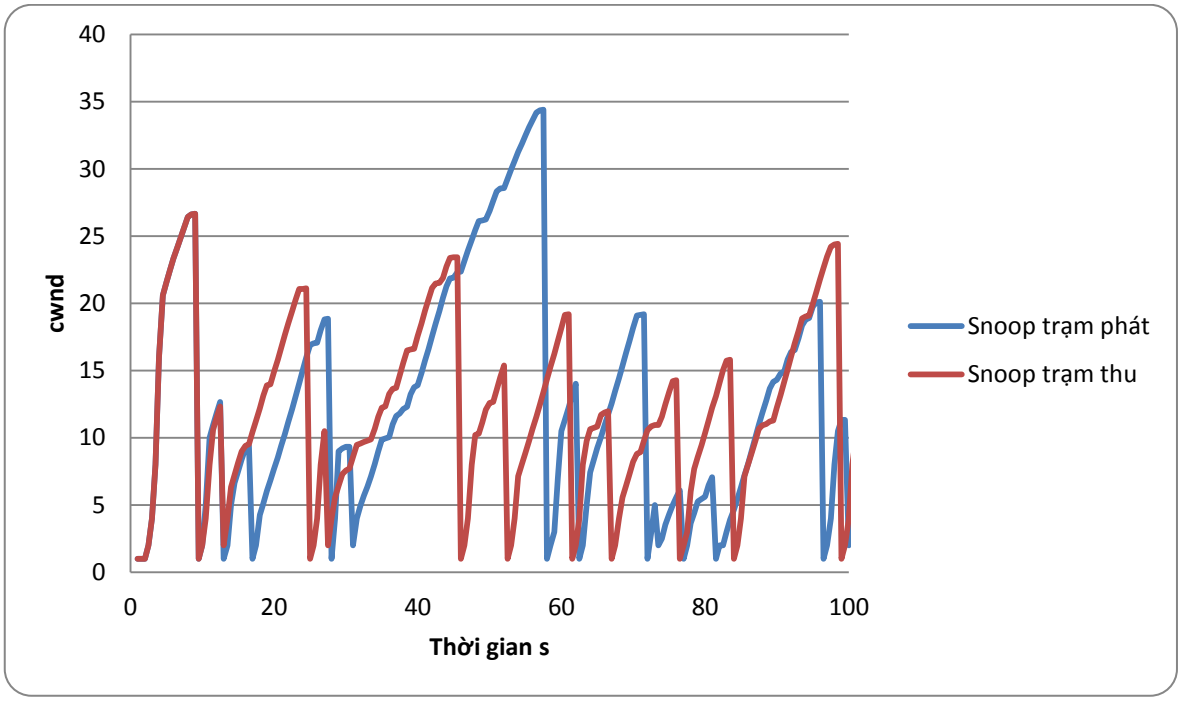
Các điều kiện ở kịch bản này được đặt như trong kịch bản thử nghiệm 4.

Đặt trạm Snoop gần trạm thu vệ tinh

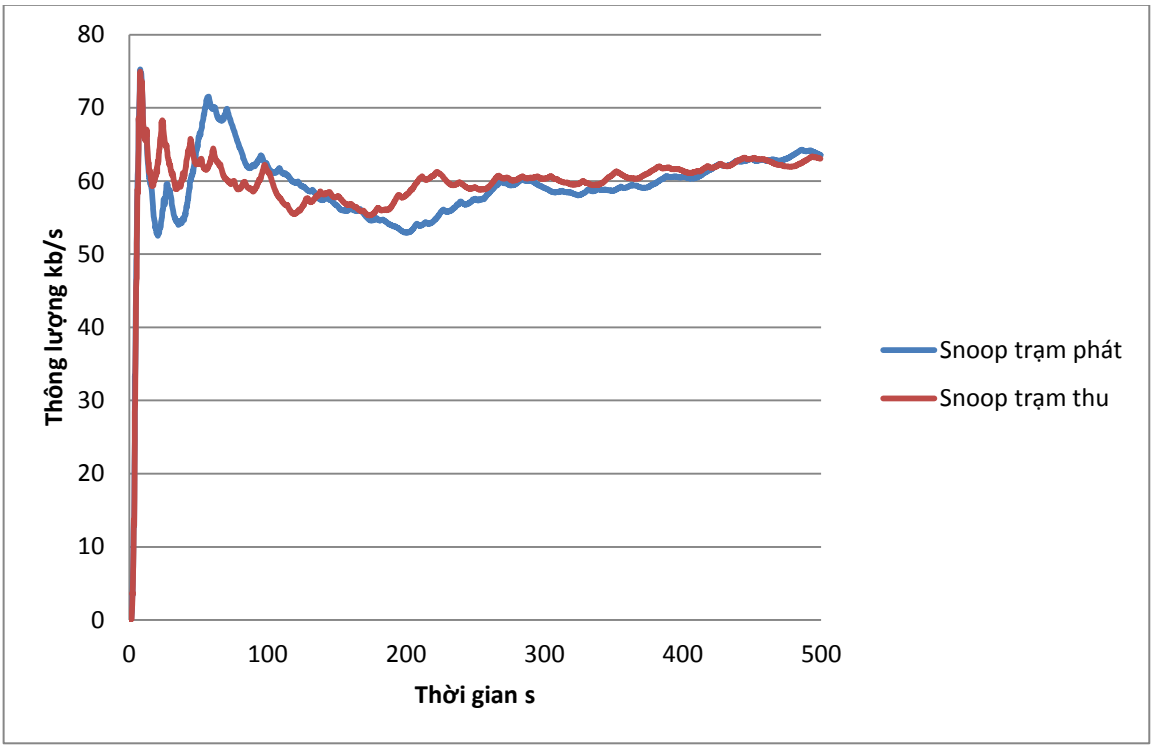


Hình 4.6 Mô hình vệ tinh sử dụng Snoop TCP tại trạm thu vệ tinh

Tahoe – TCP

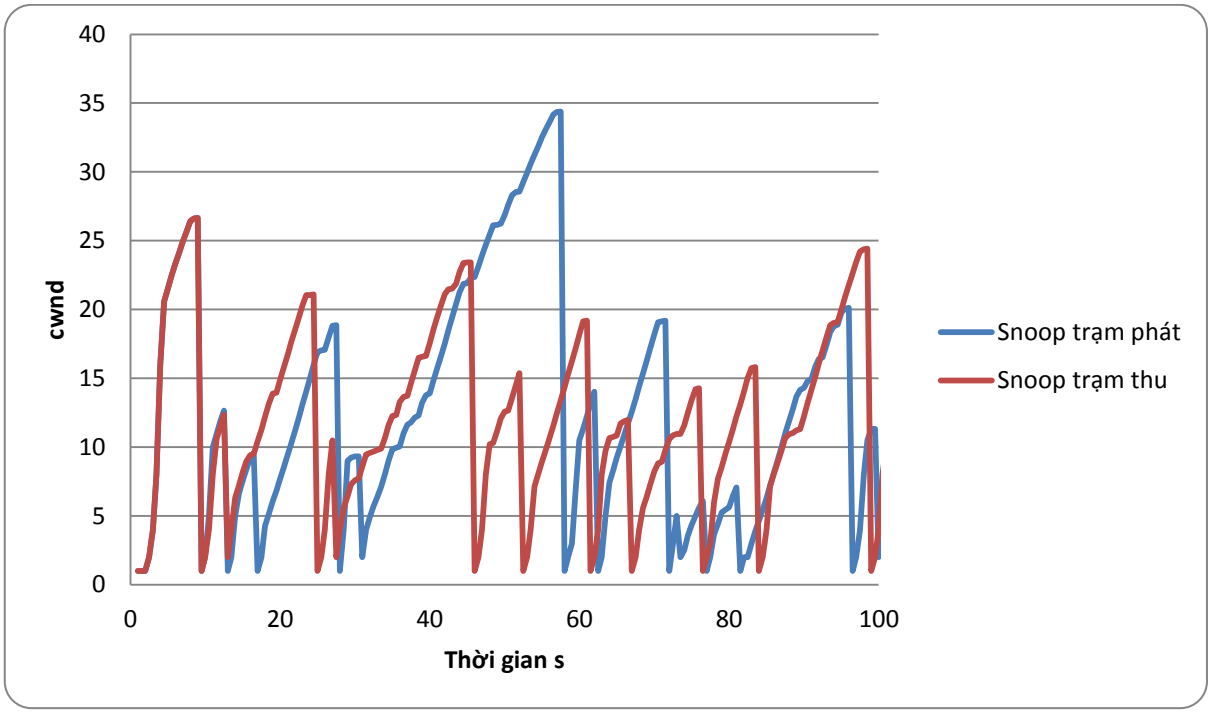


Đồ thị 4.7 Đồ thị cwnd của Tahoe ở các vị trí trạm Snoop

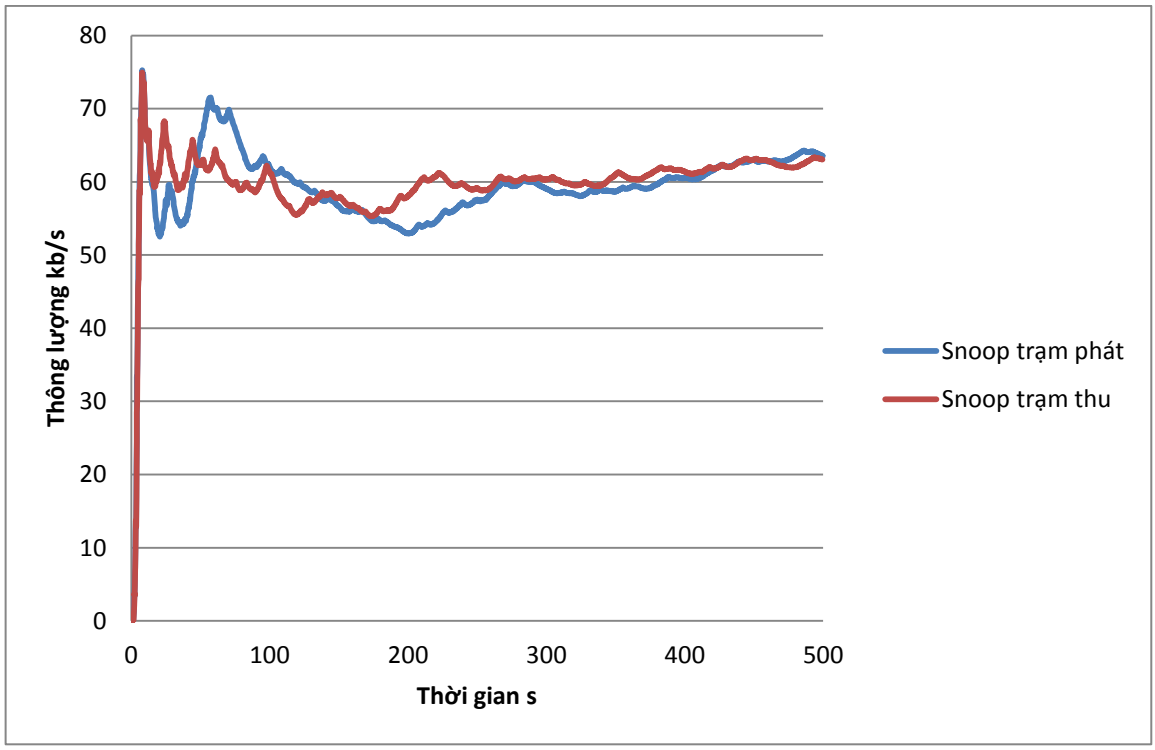


Đồ thị 4.8 Đồ thị thông lượng của Tahoe ở các vị trí trạm Snoop

Reno – TCP

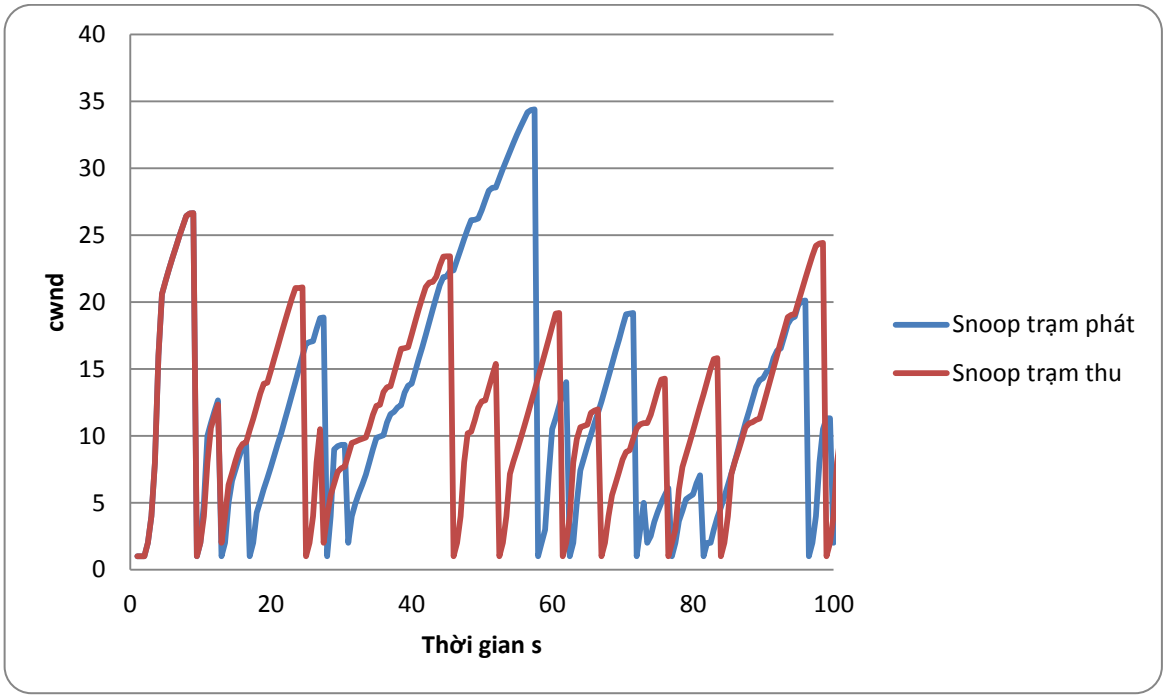


Đồ thị 4.9 Đồ thị cwnd của Reno ở các vị trí trạm Snoop

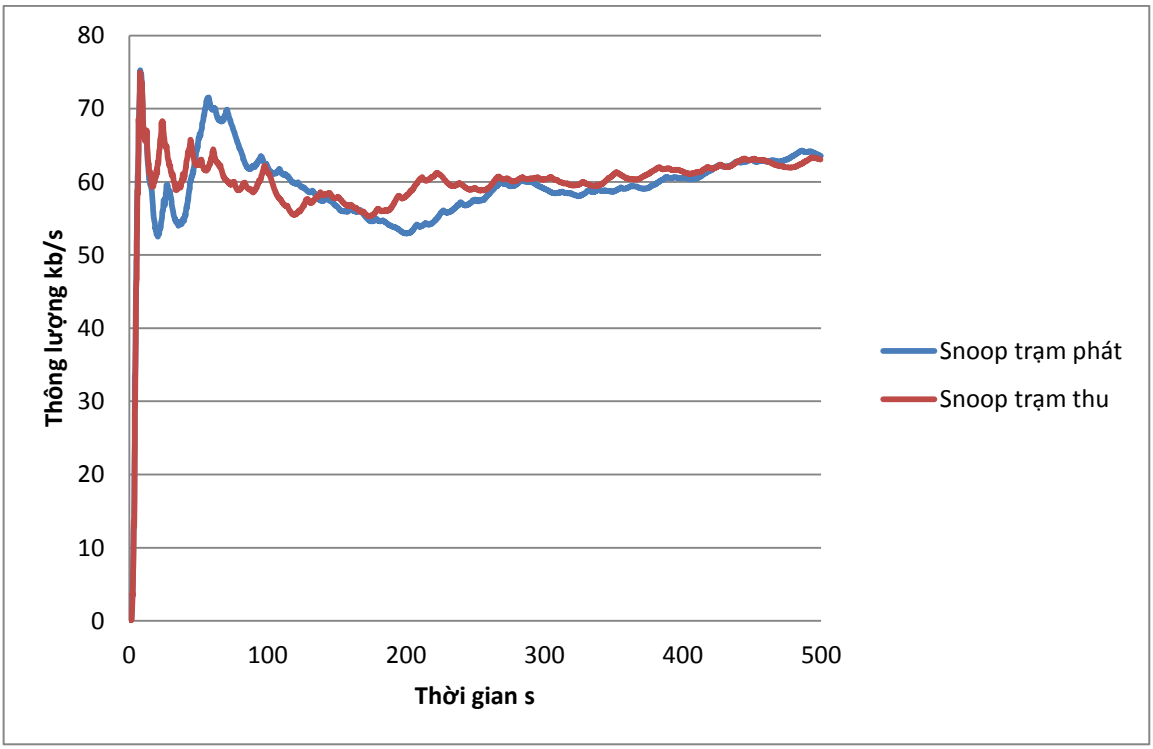


Đồ thị 4.10 Đồ thị thông lượng của Reno ở các vị trí trạm Snoop

NewReno – TCP

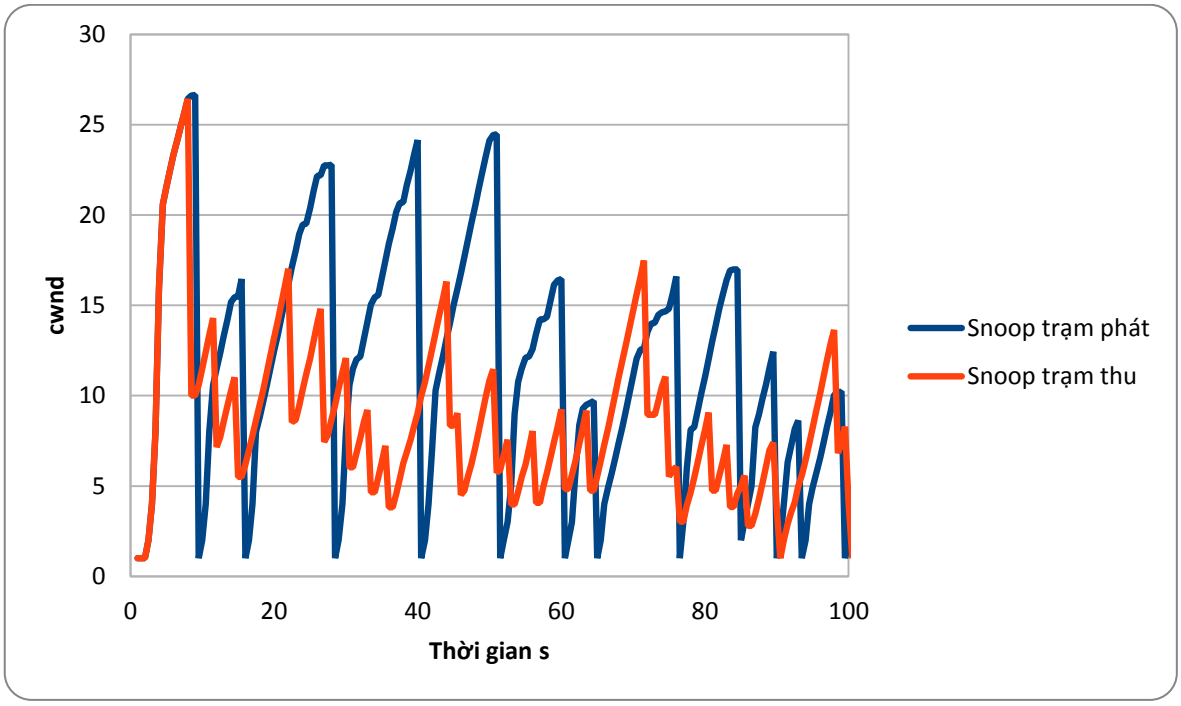


Đồ thị 4.11 Đồ thị cwnd của NewReno ở các vị trí trạm Snoop

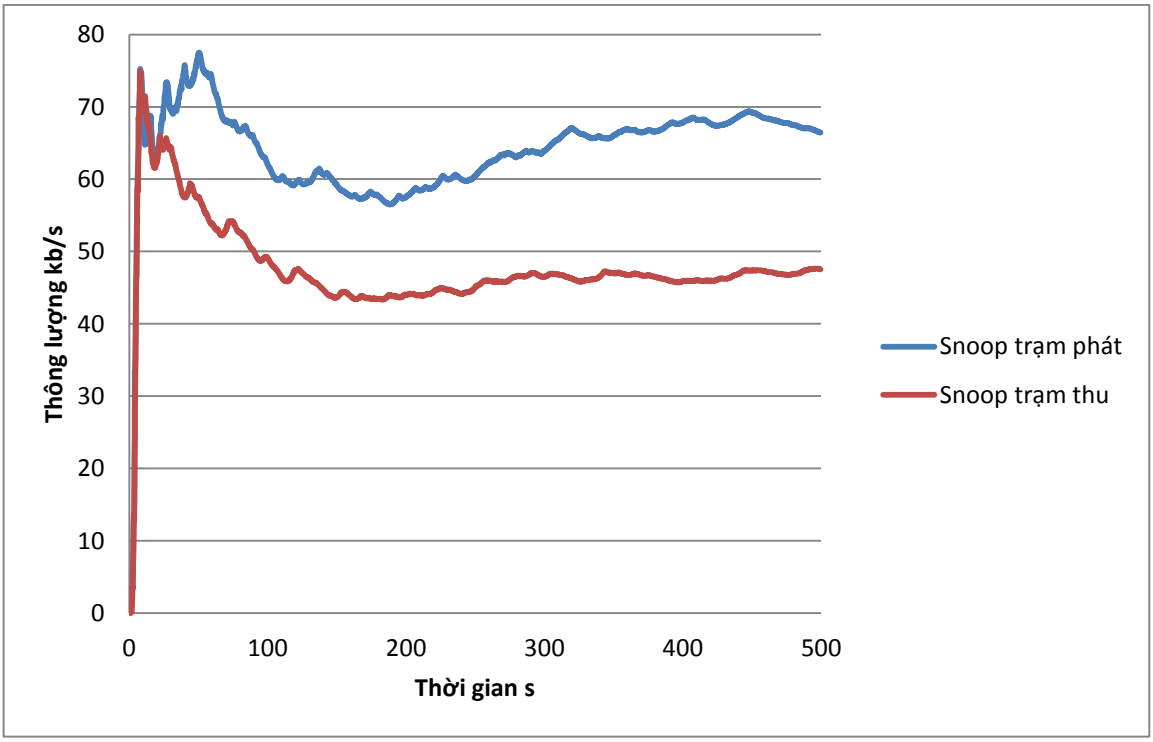


Đồ thị 4.12 Đồ thị thông lượng của NewReno ở các vị trí trạm Snoop

SACK – TCP



Đồ thị 4.13 Đồ thị cwnd của SACK ở các vị trí trạm Snoop



Đồ thị 4.14 Đồ thị thông lượng của SACK ở các vị trí trạm Snoop

Đánh giá :

Với việc thử nghiệm các vị trí khác nhau của trạm Snoop và kết quả thu được ở kịch bản thử nghiệm 4 có thể kết luận rằng :

Trong trường hợp trạm Snoop đặt cạnh trạm thu, các TCP như Tahoe, Reno và NewReno đều hoạt động có phần hiệu quả hơn so với khi trạm Snoop được đặt cạnh trạm thu. Điều này được thể hiện bằng các đô thị cwnd và thông lượng trung bình , khi các TCP Snoop trạm phát luôn duy trì cwnd ở mức ổn định dù trong điều kiện lỗi xảy ra liên tục thì các TCP Snoop trạm thu lại thay đổi cwnd liên tục và đột ngột. Nguyên nhân gây ra sự bất thường này có thể là do vị trí của thực thể TCP phát và thu, một đầu ở Hà Nội và một đầu ở Hồ Chí Minh.

Ở mô hình vệ tinh đang sử dụng, Quá trình gửi yêu cầu được thực hiện ở trạm HN và trạm HCM thực hiện gửi dữ liệu trả về, nên gói tin hồi đáp có kích thước lớn hơn gói tin yêu cầu (do có kèm dữ liệu trả về). Mặt khác, do thiết lập lỗi mất mát ngẫu nhiên ở cả hai kênh vệ tinh nên trong cả quá trình gửi yêu cầu và gửi trả lời đều gặp phải vấn đề về mất mát. Như vậy, trường hợp trạm HCM gửi dữ liệu trả về có khả năng gặp mất mát thông tin nhiều hơn so với trường hợp trạm HN gửi yêu câu vì gói dữ liệu trả về có kích thước lớn hơn, do đó khả năng "vượt lỗi" là kém hơn. Vì thế khi đặt trạm Snoop ở cạnh trạm thu mặt đất, việc truyền dữ liệu lên vệ tinh được thực hiện hiệu quả hơn do có hỗ trợ "vượt lỗi" trong quá trình gửi trả lời từ khả năng phát lại nhanh gói tin bị lỗi đã được lưu đệm của trạm Snoop. Do đó, các TCP Tahoe, Reno, NewReno hoạt động có phần hiệu quả hơn so với lúc đặt trạm Snoop ở cạnh trạm thu mặt đất.

Trong khi đó, qua đồ thị 5.17 có thể thấy việc đặt trạm Snoop cạnh trạm thu làm giảm hiệu năng của SACK so với khi Snoop đặt cạnh trạm phát. Nguyên nhân có thể do SACK là TCP được cài đặt ở thực thể thu. Trong quá trình truyền nhận, thực thể thu gửi ACK có lựa chọn cho thực thể phát để xác định các gói tin bị lỗi. Việc đặt trạm Snoop bên cạnh trạm thu làm giảm khả năng truyền lại dữ liệu khi mất mát liên tục xảy ra ở phía thực thể phát. Hơn nữa, vì trạm HN thực hiện phát lại yêu cầu trên kết nối có độ trễ lớn hơn lại không có hỗ trợ khuếch đại từ trạm Snoop nên việc phát lại các gói tin bị mất mát kém hiệu quả, làm cho SACK liên tục phải giảm cwnd mà không phục hồi được. Do đó, hiệu năng của SACK trong trường hợp này kém hơn so với khi trạm Snoop đặt gần trạm phát.

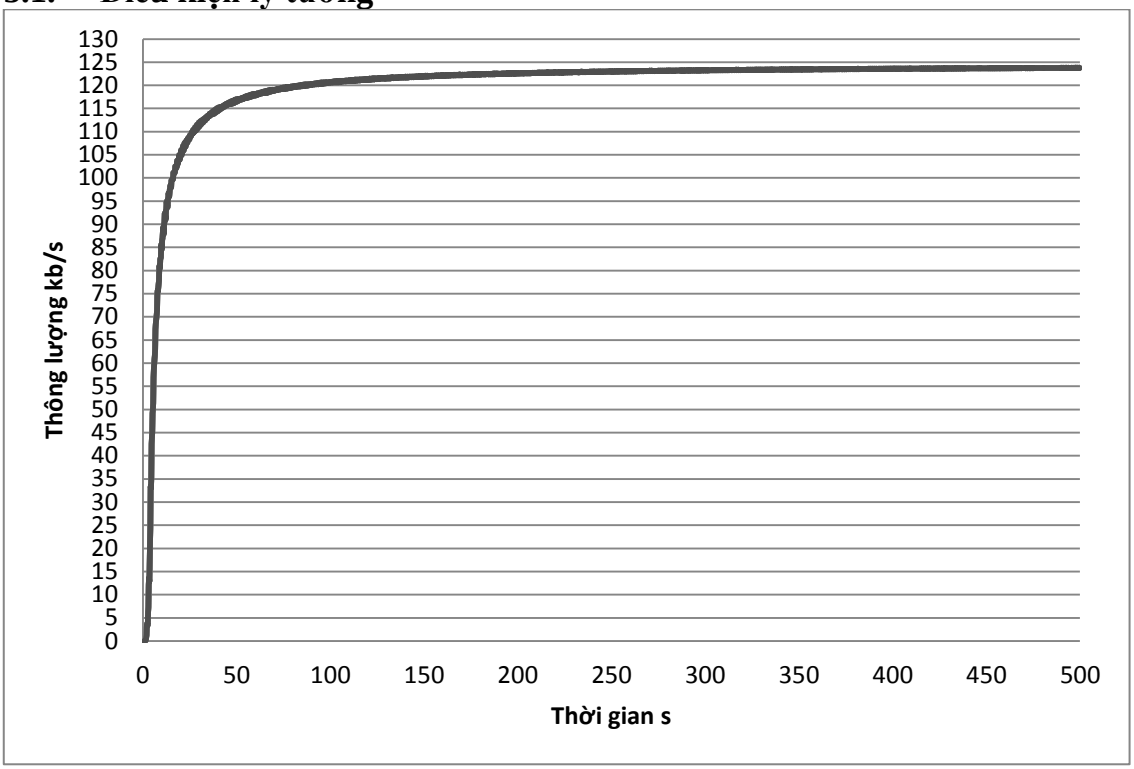
Kết nối từ trạm HN tới vệ tinh có độ trễ lớn hơn so với kết nối từ HCM đến vệ tinh là vì : Vệ tinh trong mô hình mô phỏng là loại vệ tinh địa tĩnh, hoạt động trên quỹ đạo địa tĩnh nên khoảng cách từ vệ tinh đến các điểm đầu cuối là khác nhau. Cụ thể, do vệ tinh mô phỏng là VINASAT-1, có tọa độ 132^o Đông và hai trạm đầu cuối HN – HCM nằm gần như trên một trục kinh tuyến nên khoảng cách từ vệ tinh đến trạm HN là lớn hơn so với trạm HCM, vì thế độ trễ trên kết nối từ HN đến vệ tinh là lớn hơn so với trạm HN.

Như vậy, từ việc mở rộng kịch bản Snoop, xác định được rằng: *Với các vị trí khác nhau của trạm Snoop, hiệu năng hoạt động của các TCP là khác nhau.*

5. Đánh giá hiệu năng giữa kết nối TCP end-to-end và Snoop TCP

Trong phần này tôi sẽ đánh giá mô hình vệ tinh với kết nối End-to-end truyền thống và mô hình vệ tinh có áp dụng Snoop. Mô hình mô phỏng như hình 4.5. Với kết nối E2E thì trạm Snoop được cấu hình là trạm Repeater thông thường.

5.1. Điều kiện lý tưởng



Đồ thị 5.1 Đồ thị thông lượng của các TCP trên E2E và Snoop

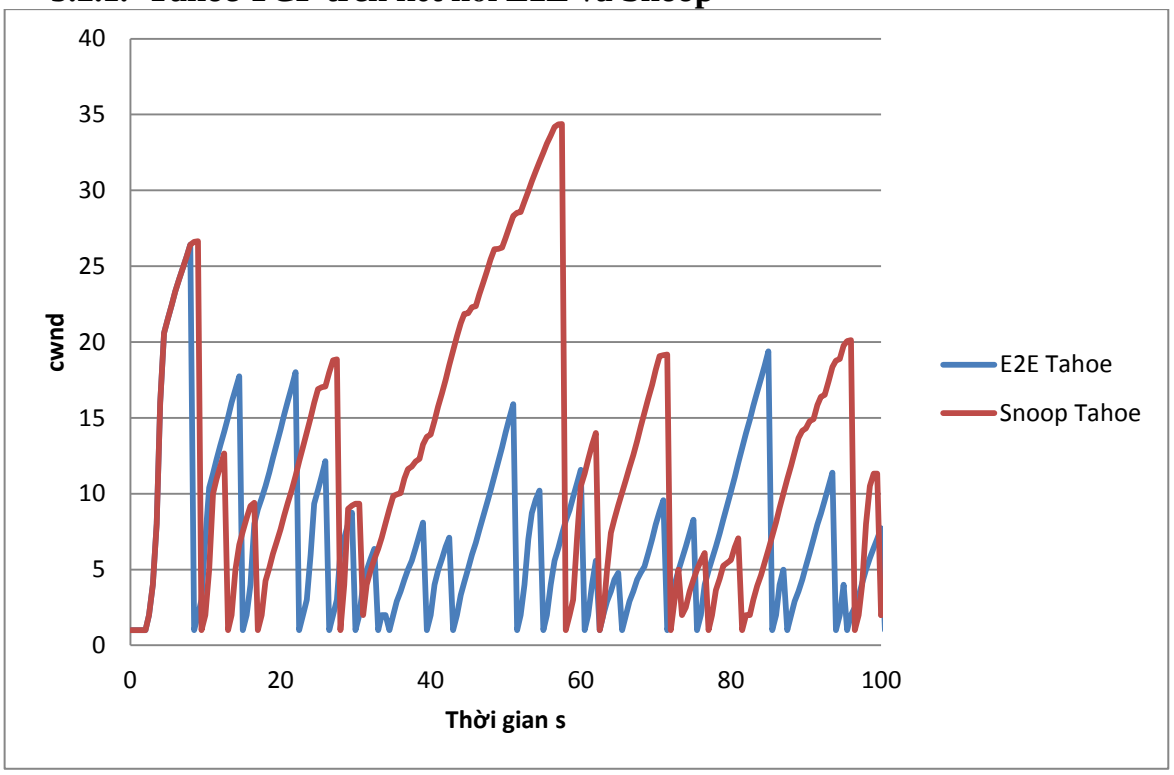
Đánh giá : Từ Đồ thị trên có thể thấy, *trong điều kiện kênh vệ tinh không có mất mát thì hiệu năng hoạt động của các TCP trên liên kết E2E và Snoop là như nhau.*

Ngoài ra, hoạt động của trạm Snoop trong hai trường hợp là như nhau. Nghĩa là trong điều kiện không lỗi, trạm Snoop hoạt động tương tự như một trạm Repeater bình thường. Tính năng "Snoop" không được kích hoạt trong trường hợp này.

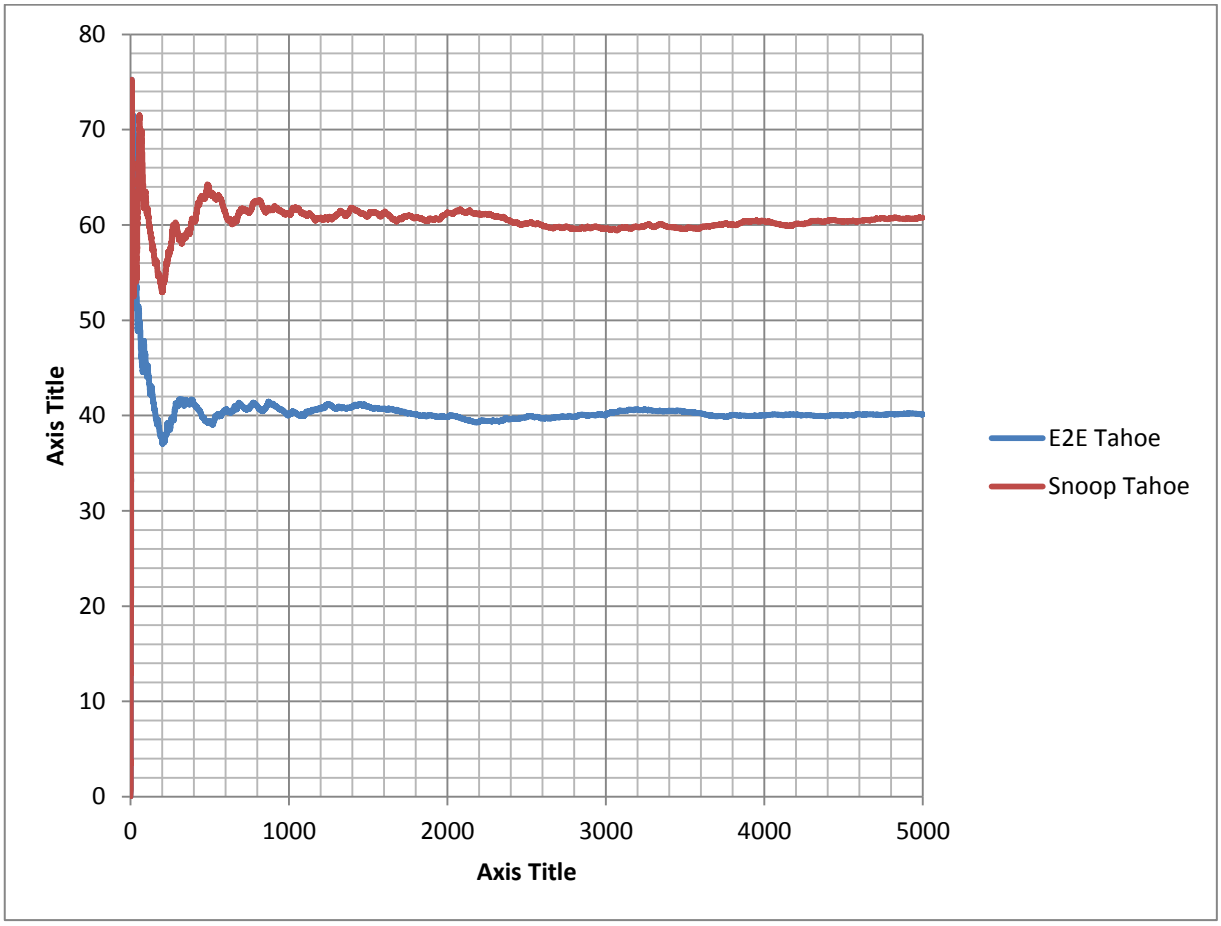
5.2. Điều kiện có lỗi với các biến thể TCP

Sử dụng kết quả của các kịch bản thử nghiệm 2 và 4 trước đây, Xem xét sự thay đổi cửa sổ tắc nghẽn và thông lượng truyền thực của các biến thể TCP trên kết nối E2E và Snoop.

5.2.1. Tahoe TCP trên kết nối E2E và Snoop

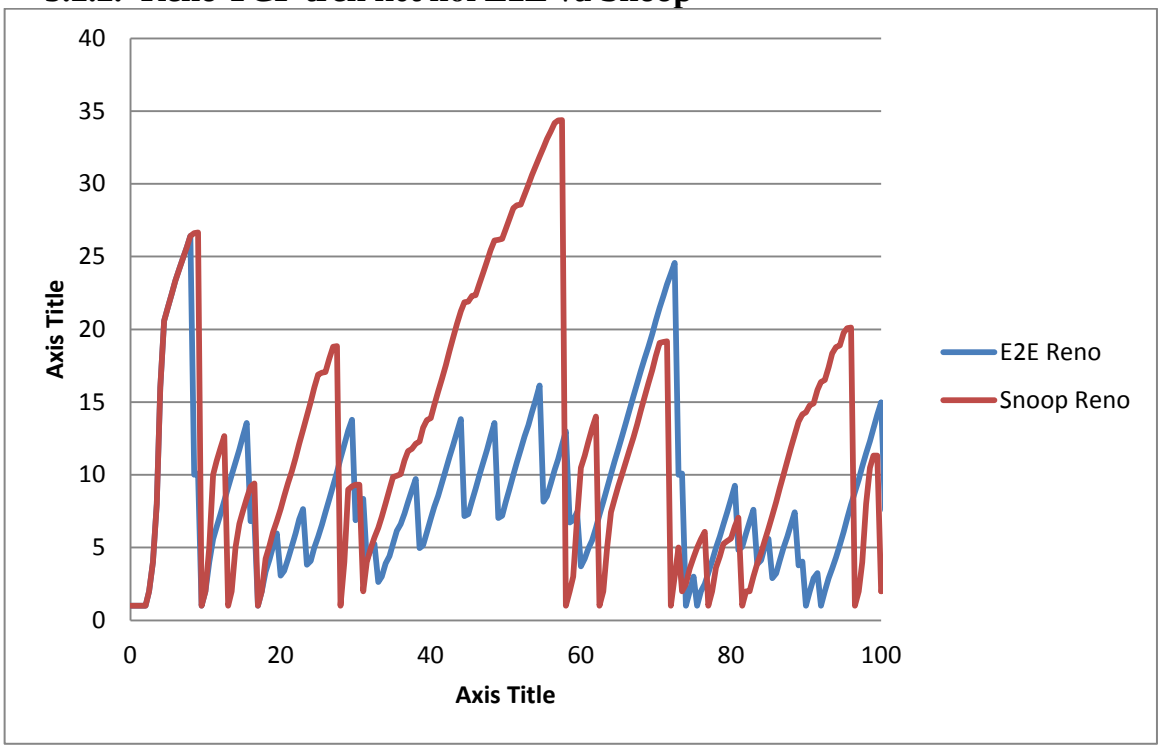


Đồ thị 5.2 Đồ thị cwnd của Tahoe trên các kết nối E2E và Snoop

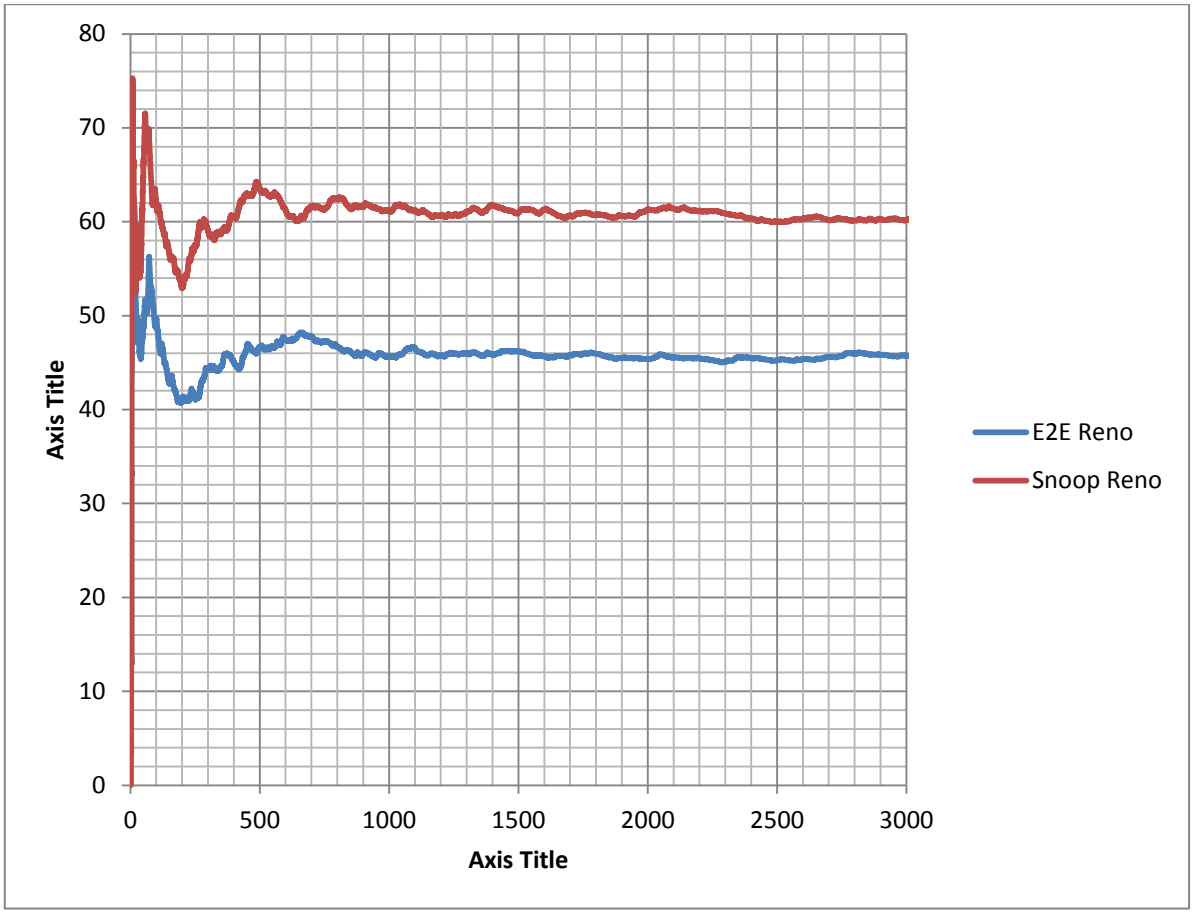


Đồ thị 5.3 Đồ thị thông lượng của Tahoe trên kết nối E2E và Snoop

5.2.2. Reno TCP trên kết nối E2E và Snoop

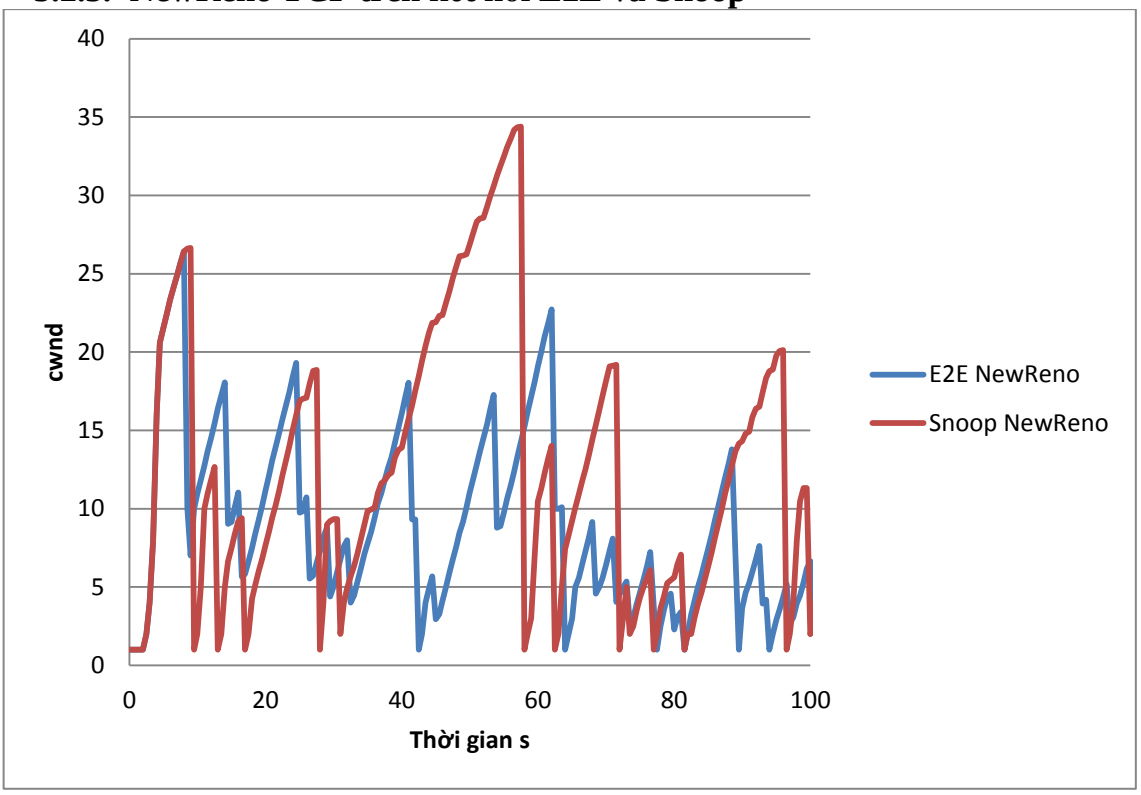


Đồ thị 5.4 Đồ thị cửa sổ tắc nghẽn của Reno trên kết nối E2E và Snoop

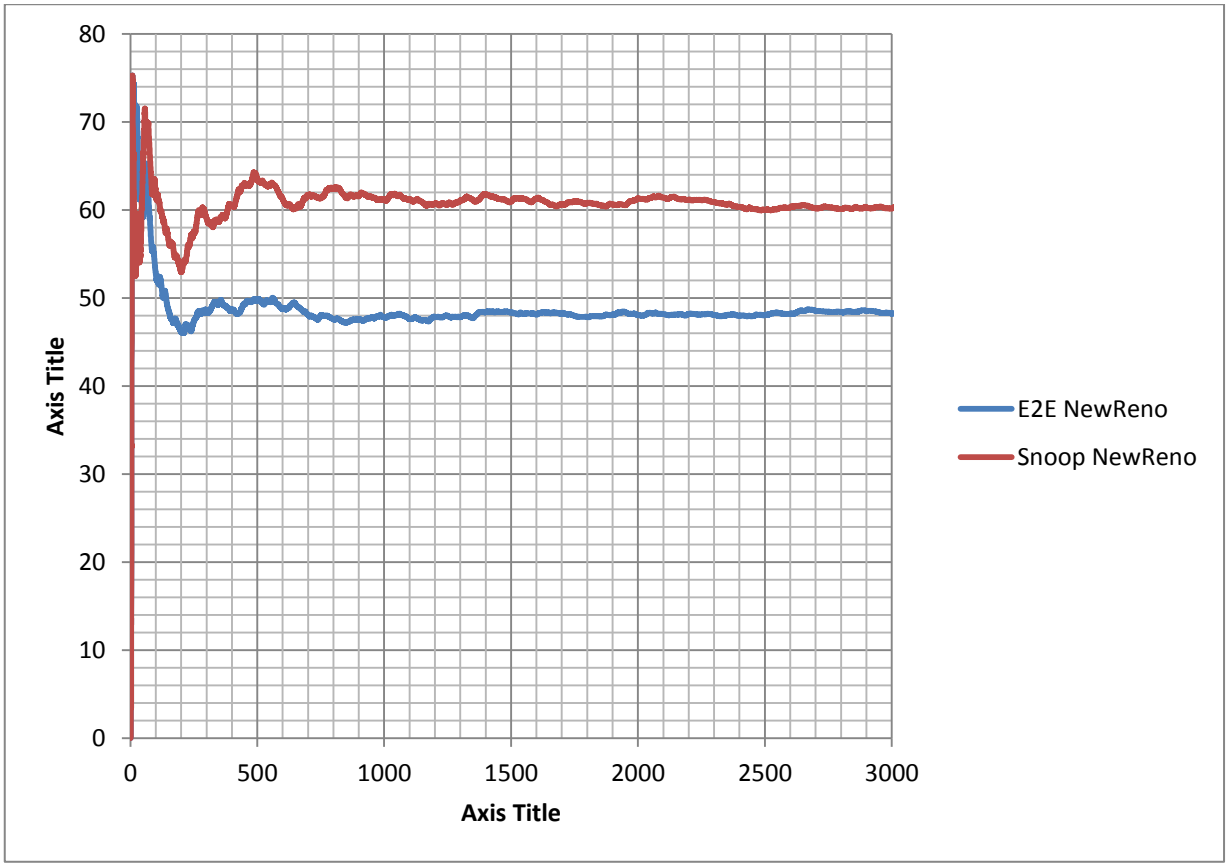


Đồ thị 5.5 Đồ thị thông lượng toàn phần của Reno trên kết nối E2E và Snoop

5.2.3. NewReno TCP trên kết nối E2E và Snoop

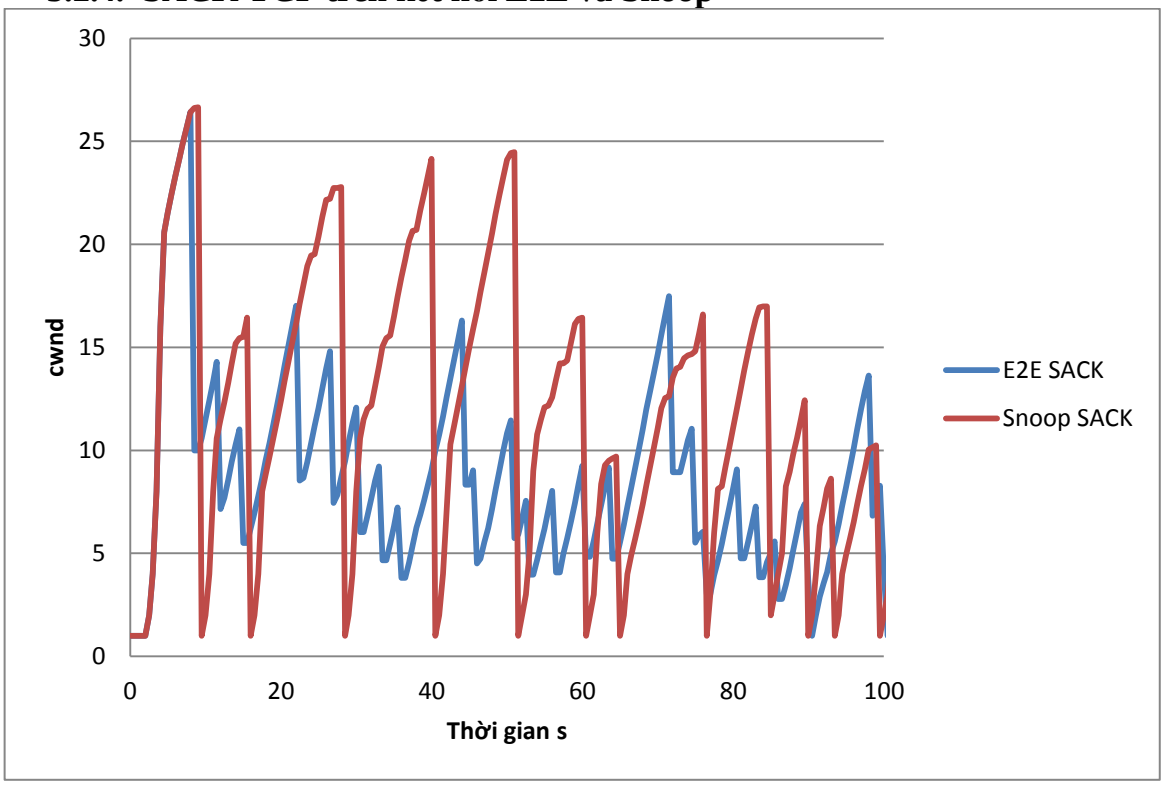


Đồ thị 5.6 Đồ thị cwnd của NewReno trên kết nối E2E và Snoop

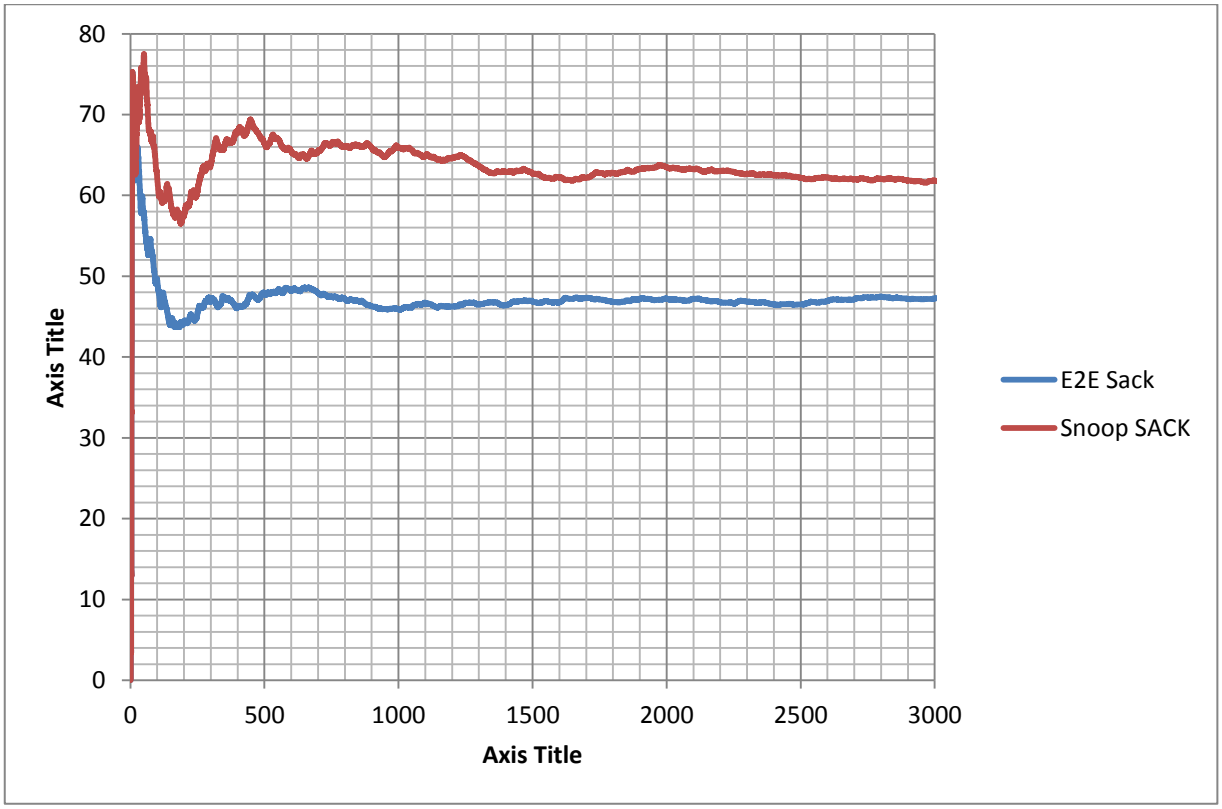


Đồ thị 5.7 Đồ thị thông lượng của NewReno trên kết nối E2E và Snoop

5.2.4. SACK TCP trên kết nối E2E và Snoop



Đồ thị 5.8 Đồ thị cwnd của SACK trên kết nối E2E và Snoop



Đồ thị 5.9 Đồ thị thông lượng của SACK trên kết nối E2E và Snoop

Đánh giá : Dựa vào các Đồ thị trên, có thể thấy :

Theo Đồ thị 5.3, Hiêu năng của Snoop Tahoe được cải thiện rất nhiều so với trên kết nối E2E Tahoe. Thông lượng khi ổn định của Snoop Tahoe là 60kbps trong khi E2E Tahoe là 40 kbps, nghĩa là Snoop đã giúp cải thiện hiệu năng 50% so với E2E cho TCP Tahoe. Mỗi khi xác nhận có mất mát gói tin (nhận được 3 dupACK) thì Tahoe lại thực hiện "Bắt đầu chậm" vì thế cwnd hoạt động ở mức thấp, không đảm bảo tần . Trong khi đó, với bổ sung Snoop cho Tahoe đã khắc phục phần nào nhược điểm này. Bằng chứng là trong 100s đầu của thử nghiệm, khi gặp lỗi, Snoop Tahoe phục hồi rất nhanh và liên tục duy trì cwnd ở mức cao hơn hẳn so với E2E Tahoe.

Ngoài ra, việc sử dụng Snoop trên kênh vệ tinh cũng mang lại những cải thiện hiệu năng đáng kể cho Reno – TCP và NewReno TCP. Mặc dù Reno có cải tiến trong phương pháp phục hồi so với Tahoe, tuy nhiên E2E Reno lại không hiệu quả lắm khi mất gói liên tục, thể hiện ở khoảng 80s đầu, khi cwnd được duy trì ở mức thấp kể từ sau khi xác định được mất mát thông tin. Trong khi đó, tương tự Tahoe, Snoop Reno liên tục duy trì cwnd ở mức cao và ổn định, vì thế mang lại hiệu quả cải tiến hiệu năng so với kết nối E2E.

Đặc điểm của NewReno là phát hiện nhiều mất mát xảy ra trên cùng một cửa sổ, tuy nhiên việc phát lại các gói tin bị mất của NewReno lại khá chậm chạm. Khi kết hợp với trạm Snoop, nhờ khả năng chuyển tiếp thông tin của Snoop mà hiệu quả phục hồi của NewReno được cải thiện đáng kể. Nhờ đó hiệu năng được cải thiện tốt hơn so với khi không có Snoop

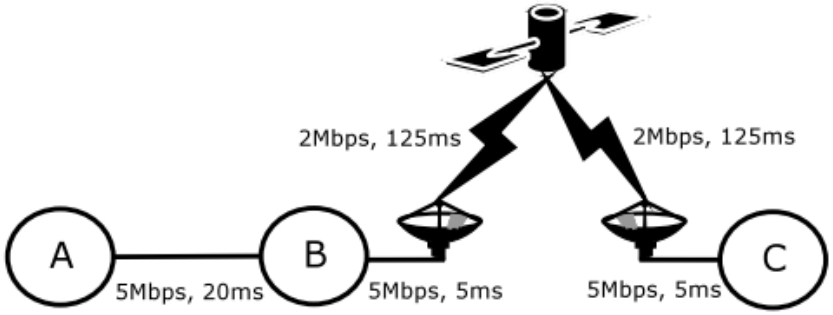
Dựa vào các đồ thị 5.8 và 5.9, có thể thấy Snoop SACK hoạt động hiệu quả hơn E2E SACK, khi luôn duy trì kích thước cwnd ở mức cao và đều đặn hơn so với E2E SACK. Nguyên nhân là do có trạm Snoop hỗ trợ cải thiện đường truyền nên quá trình truyền lại mỗi khi SACK phát hiện mất gói tin được thực hiện rất nhanh và hiệu quả.

Từ các kết quả trên có thể kết luận : *Việc sử dụng Snoop cho kết nối vệ tinh rõ ràng giúp cải thiện hiệu năng TCP ở tất cả các biến thể so với sử dụng kết nối End-to-end thông thường.*

5.3. So sánh kết quả với một số công trình nghiên cứu trước đây

Tôi sẽ so sánh kết quả trên của bài đồ án với kết quả bài nghiên cứu "On the use of Snoop with Geostationary Satellite Links" của Joel Sing và Ben Soh thuộc bộ môn Khoa học máy tính và Kỹ thuật máy tính của Đại học La Trobe, Úc [10]. Văn bản này mô tả phương pháp sử dụng Snoop trên kênh vệ tinh địa tĩnh và sử dụng mô phỏng NS-2 để kết luận đánh giá.

Mô hình mô phỏng sử dụng trong văn bản này có nhiều điểm tương đồng nhất định so với mô hình trong bài đồ án :



Hình 5.1 Mô hình mô phỏng trong văn bản

Ở đây họ cũng sử dụng đặt trạm Snoop bên cạnh tram phát (mặt đất) và độ trễ toàn phần là RTT = 560ms, không khác quá nhiều so với đồ trễ toàn phần của bài đồ án.

Dưới đây là bảng kết quả mô phỏng của nghiên cứu "On the use of Snoop with Geostationary Satellite Links". Từ bảng kết quả có thể nhận xét như sau :

- Hiệu năng Tahoe trên Snoop vệ tinh được cải thiện rất nhiều so với trường hợp không có Snoop. Hiệu quả đạt được là 183% và 20% đối với tỉ lệ bit lỗi tương ứng 10^{-7} và 10^{-8}.
- Mức độ cải thiện hiệu năng Reno và NewReno là tương đối đồng đều nhau, trong trường hợp tỉ lệ lỗi 10^{-7} là 130% với 138%, còn trong trường hợp tỉ lệ lỗi là 10^{-8} là 7%.
- Trong điều kiện không có Snoop, hiệu năng SACK là kém hơn so với NewReno. Tuy nhiên ở một số tỷ lệ lỗi, sau khi áp dụng Snoop, hiệu năng SACK được cải thiện và hiệu quả hơn so với Snoop NewReno trong một số tỉ lệ lỗi. Điều này được thể hiện qua số seq và thông lượng của các TCP.
- Ngoài ra, trong điều kiện tỉ lệ lỗi cực thấp, gần như điều kiện lý tưởng (10^{-9} và 10^{-10}) thì hoạt động của các biến thể TCP trong kết nối có và không có Snoop là như nhau. Nói cách khác, trong điều kiện gần như lý tưởng thì Snoop TCP và E2E TCP có hiệu quả như nhau.

Tuy nhiên, giữa hai kết quả vẫn có một số điểm khác nhau như sự khác nhau về hiệu quả hoạt động của các TCP sử dụng ACK cục bộ khi có sử dụng Snoop, hiểu quả hoạt động của Snoop Reno cao hơn so với Snoop SACK trong một số tỉ lệ lỗi khác. Những điểm khác nhau này có thể là do sự khác nhau giữa các mô

hình mô phỏng, sự sai khác về độ trễ giữa hai mô hình hoặc do sai lệch giữa các phiên bản mô phỏng.

Mặc dù vậy, nếu chỉ xét trên một vài tỉ lệ lỗi riêng biệt, thì kết quả mô phỏng là hoàn toàn giống với kết quả của [10]. Vì thế có thể kết luận : Nhìn chung những kết quả thu được từ kịch bản mô phỏng Snoop vệ tinh với các biến thể TCP và kết quả của công trình [10] có nhiều điểm tương đồng. Qua đó phần nào phản ánh ***kết quả mô phỏng thu được là chính xác.***

BER / Variant	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}	10^{-10}
Tahoe	1553	5730	18585	95135	142076	142076
Reno	1772	6749	18624	89687	142076	142076
New Reno	1924	6597	18624	89687	142076	142076
SACK	1800	6160	18368	78575	142076	142076
FACK	1619	7834	20515	79358	142076	142076
Vegas	2760	9018	27812	88771	112549	112549

Table 1. TCP Sequence Numbers without Snoop

BER / Variant	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}	10^{-10}
Tahoe	3745	13911	52528	114528	142076	142076
Reno	6329	23694	42893	96195	142076	142076
New Reno	3906	27668	44414	96111	142076	142076
SACK	3335	30936	44702	84769	142076	142076
FACK	3201	17021	48249	83818	142076	142076
Vegas	3085	13094	42874	86113	112549	112549

Table 2. TCP Sequence Numbers with Snoop

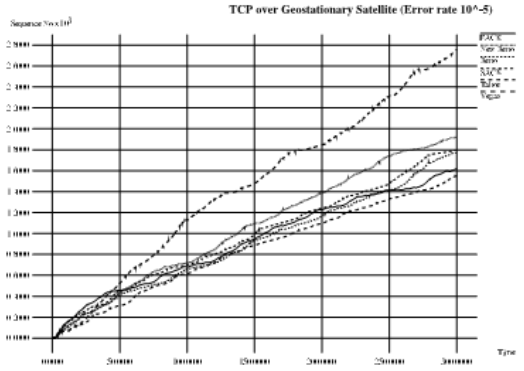
BER / Variant	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}	10^{-10}
Tahoe	2192	8181	33943	19393	0	0
Reno	4557	16945	24269	6508	0	0
New Reno	1982	21071	25790	6424	0	0
SACK	1535	24776	26334	6194	0	0
FACK	1582	9187	27734	4460	0	0
Vegas	325	4076	15062	-2658	0	0

Table 3. Difference in Sequence Numbers using Snoop

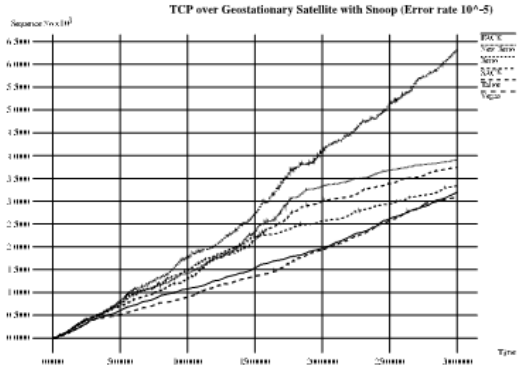
BER / Variant	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}	10^{-10}
Tahoe	141%	143%	183%	20%	0	0
Reno	257%	251%	130%	7%	0	0
New Reno	103%	319%	138%	7%	0	0
SACK	85%	402%	143%	8%	0	0
FACK	98%	117%	135%	6%	0	0
Vegas	12%	45%	54%	-1%	0	0

Table 4. Performance Gained using Snoop

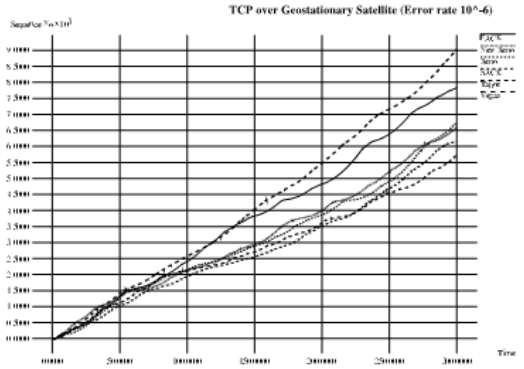
Hình 5.2 Bảng kết quả seq của [10]



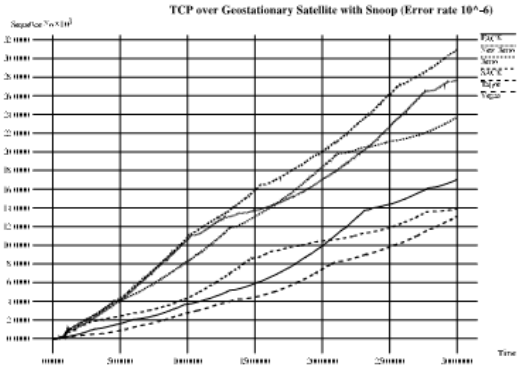
(a) TCP Throughput without Snoop (10**-5)



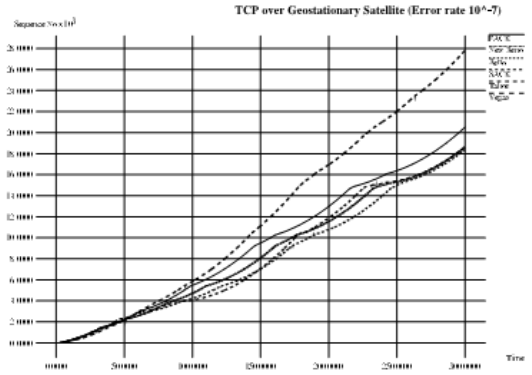
(b) TCP Throughput with Snoop (10**-5)



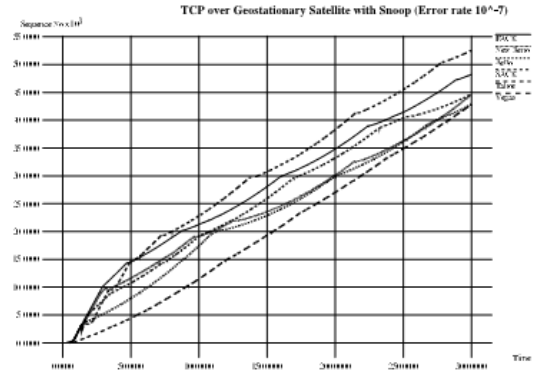
(c) TCP Throughput without Snoop (10**-6)



(d) TCP Throughput with Snoop (10**-6)



(e) TCP Throughput without Snoop (10**-7)



(f) TCP Throughput with Snoop (10**-7)

Figure 2. TCP Throughput Graphs

Hình 5.3 Bảng kết quả thông lượng của [10]

KẾT LUẬN

Đồ án đã trình bày lý thuyết cơ bản và một số phương pháp tối ưu hóa hiệu năng của TCP trong môi trường mạng vệ tinh. Thông qua phương pháp tiếp cận bằng thực nghiệm mô phỏng có thể phần nào thấy được hiệu quả của các giải thuật chống tắc nghẽn TCP và của phương pháp cải tiến khác. Từ kết quả của các kịch bản mô phỏng, đồ án đã thu được những kết luận riêng :

- Snoop cải tiến hiệu năng tất cả các biến thể TCP, tuy nhiên với các TCP khác nhau thì mức độ hiệu quả là khác nhau.
- Trong điều kiện lý tưởng , không có mất mát thông tin. Snoop không có giá trị
- Snoop ở các vị trí khác nhau thì có những tác động đến hiệu năng TCP là khác nhau.

So với phương pháp kết nối End-to-end truyền thông, Snoop TCP có nhiều ưu việt hơn trong vấn đề cải thiện hiệu năng TCP. Tuy nhiên đây cũng là một phương pháp này đòi hỏi thiết bị trung gian (là trạm Snoop) phải có năng lực xử lý lớn để đệm gói, và phát lại khi có lỗi xảy ra.

Hiện tại, các phương pháp tối ưu hóa hiệu năng TCP trên kênh vệ tinh vẫn đang được nghiên cứu và phát triển. Việc tối ưu hoạt động TCP trong mạng vệ tinh cho phép hệ thống thông tin vệ tinh và mạng thông tin toàn cầu tiếp tục phát triển và mang lại hiệu quả tốt hơn phục vụ cho con người.

Hạn chế

Hướng tiếp cận của đề tài với bài toán đặt ra chỉ là bằng phương thức thực nghiệm, mà chưa thực hiện được đảm bảo bằng toán học để có thể đảm bảo tính chính xác, tạo cơ sở cho các nghiên cứu khác.

Hệ thống các mô phỏng chưa toàn diện. Hiện tại, đề tài chỉ tiếp cận các mô phỏng End-to-end và Snoop với các TCP truyền thống : Tahoe, Reno, NewReno và SACK mà chưa có những thử nghiệm đổi mới về mặt thuật toán với các TCP hay sử dụng các TCP mới, tối ưu hơn cho kênh vệ tinh.Ngoài ra, mô hình mô phỏng của đề tài mới chỉ thực hiện với vệ tinh địa tĩnh GEO.

Hướng phát triển

Để bổ sung đảm bảo toán học, hướng mở rộng đồ án là tiếp cận bài toán đặt ra theo phương pháp toán học, xem xét các kết quả để đảm bảo tính đúng đắn toán học. Giải quyết bài toán trọn vẹn hơn

Ngoài ra, ngoài các phương pháp đã đề cập trong bài đồ án, vẫn còn nhiều nghiên cứu khác đã và đang thực hiện để tối ưu hóa hiệu năng cho TCP trên kênh vệ tinh. Hướng phát triển tiếp theo của đề tài là tìm hiểu sâu hơn về mạng vệ tinh với các mở rộng TCP mới, sử dụng các thuật toán có hiệu quả hơn các TCP truyền thống. Thêm vào đó là tìm hiểu các phương pháp tối ưu mới cho TCP trong điều kiện kênh viễn thông vệ tinh với độ trễ cao và tỷ lệ lỗi lớn, từ đó đề xuất ra một cách giải quyết vấn đề trọn vẹn hơn.

TÀI LIỆU THAM KHẢO

[1] W. Stevens. "*TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms"* – RFC 2001, 1-1999

[2] Andreas Schilke. *"TCP over Satellite Links".* TUBerlin. June 5, 1997

[3] M. Allman, D. Glover, LSanchez. *"Enhancing TCP over Satellite channels using standard mechanisms"* -RFC 2488. 1-1999.

[4] V. Jacobson and M. J. Karels. *"Congestion Avoidance and Control"*. Proc. of SIGCOMM, 1988

[5] S. Floyd, T. Henderson, A. Gurtov. *"The NewReno Modification to TCP's Fast Recovery Algorithm"* – RFC3872. 4-2004

[6] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow. *"TCP Selective Acknowledgment Options"* - RFC2018 – 1996

[7] Yongguang Zhang, Dante De Lucia, Bo Ryu, Son K. Dao. *"Satellite Communications in the Global Internet: Issues, Pitfalls, and Potential"*

[8] Joseph Ishac, Mark Allman. *"On the performance of TCP Spoofing in Satellite networks"* , 9 – 2001.

[9] Chi Ho Ng, Jack Chow, and Ljiljana Trajkovic, *"Performance Evaluation of TCP over WLAN 802.11 with the Snoop Performance Enhancing Proxy"* . 2002

[10] Joel Sing and Ben Soh.*"On the use of Snoop with Geostationary Satellite Links"*.ICITA'05 – IEEE 2005

[11] Information Sciences Institue, University of Southern California. http://www.isi.edu/nsnam/ns/. Last visited May-2011

[12] Nhóm Adnet - vntelecom.org. *"Mô phỏng trong NS-2"* . 12-2009.