ĐẠI HỌC QUỐC GIA HÀ NỘI
KHOA CÔNG NGHỆ

NGUYỄN TUỆ LINH

HIỆU SUẤT HOẠT ĐỘNG TCP TRÊN KÊNH VIỄN THÔNG VỆ TINH

CHUYÊN NGÀNH : CÔNG NGHỆ THÔNG TIN
MÃ SỐ : 01.01.10

LUẬN VĂN THẠC SỸ

HƯỚNG DẪN KHOA HỌC: TS. VŨ DUY LỢI

HÀ NỘI 5-2003

MỤC LỤC

DANH MỤC CÁC BẢNG BIỂU

DANH MỤC CÁC HÌNH VẼ

LỜI CẢM ƠN

Tôi xin trân trọng cảm ơn đến TS. Vũ Duy Lợi - Trung tâm Công nghệ Thông tin Văn phòng Trung ương Đảng - người đã trực tiếp hướng dẫn, định hướng nội dung, tận tình giúp đỡ và đóng góp nhiều ý kiến quan trọng trong quá trình thực hiện luận văn.

Dù không trực tiếp, tôi cũng xin trân trọng bày tỏ sự cảm ơn đến TS. Thomas Ros Hender, người đã cung cấp cho tôi rất nhiều tư liệu và thông tin về hiệu suất TCP trên kênh viễn thông vệ tinh thông qua việc trao đổi thư điện tử.

Tôi xin chân thành cảm ơn đến ThS. Nguyễn Đình Việt, ThS. Nguyễn Hoàng Linh đã có đóng góp ý kiến cho nội dung của bài viết.

Tôi xin trân trọng cảm ơn Giám đốc Công ty Bưu chính Liên tỉnh và Quốc tế đã tạo mọi điều kiện thuận lợi cho tôi được tham gia học tập nghiên cứu tại trường Đại học Quốc gia Hà Nội.

Cuối cùng tôi chân thành cảm ơn các giảng viên, cán bộ nhân viên của trường Đại học Quốc gia Hà Nội đã tham gia giảng dạy và tận tình giúp đỡ trong quá trình học tập tại trường. Cảm ơn các bạn bè, đồng nghiệp trong lớp Công nghệ thông tin K7T khoá 2000-2002, Khoa Công nghệ, Đại học Quốc gia Hà Nội, đã cùng hỗ trợ, giúp đỡ trong quá trình học tập và nghiên cứu.

Một lần nữa tôi xin trân trọng cảm ơn tất cả mọi người đã giúp đỡ tôi trong suốt quá trình học tập, nghiên cứu và hoàn thành bản luận văn tốt nghiệp này.

Nguyễn Tuệ Linh

MỞ ĐẦU

Công nghệ hiện đại cho phép sử dụng hệ thông vệ tinh bao gồm các dải tần dữ liệu rộng và các thiết bị đầu cuối nhỏ. Hệ thống này được nghiên cứu nhằm ứng dụng cung cấp dịch vụ truy cập Internet giá chấp nhận được cho gia đình và cho các doanh nghiệp nhỏ trên khắp thế giới, đặc biệt với địa hình nước ta có vùng lãnh thổ trải dài, nhiều khu vực có mật độ dân cư thưa hoặc địa hình không thuận lợi với các hệ thống viễn thông hữu tuyến.

Nổi bật có hai hệ thống vệ tinh với dải tần rộng hiện đại đang được phát triển đó là: Vệ tinh có quỹ đạo tầng cao - địa tĩnh (GEO) và nhiều vệ tinh ở quỹ đạo thấp hơn LEO. Trong bài viết này chỉ nghiên cứu những vấn đề phát sinh trong quá trình cố gắng sử dụng vệ tinh GEO để cung cấp dịch vụ truy nhập Internet. Đặc biệt là giao thức kiểm soát truyền thông (TCP) bị giảm giá trị do trễ và tỷ lệ lỗi trong khi truyền lớn.

Trong chương I "Cơ sở lý thuyết": Đề cập đến các vấn đề là cơ sở cho các nghiên cứu của luận văn. Phần đầu đề cập các đặc tính của hệ thống vệ tinh trong đó ta chỉ quan tâm đến vệ tinh GEO. Phần tiếp theo đề cập các vấn đề cơ bản của TCP cũng như các vấn đề liên quan đến việc điều khiển lưu lượng và điều khiển tắc nghẽn. Cũng trong phần TCP có trình bày một số giao thức TCP cải tiến nhằm nâng cao hiệu suất làm việc và giải quyết tắc nghẽn.

Cũng trong chương này, bài viết đưa ra phương pháp luận, mô tả các công cụ để thực hiện mô phỏng và phân tích nhằm xác định được hiệu suất làm việc của kênh vệ tinh với các giao thức điều khiển giao vận khác nhau cũng như các kiến trúc khác nhau trong mô hình kết nối.

Chương II: Thực hiện đánh giá hiệu suất của kênh vệ tinh GEO với kết nối TCP đầu cuối – đầu cuối trong đó có thực hiện mô phỏng phân tích số liệu để thấy được một số vấn đề liên quan có ảnh hưởng đến hiệu suất như độ rộng cửa sổ, tốc độ kênh vệ tinh hay kích thước gói tin. Cũng trong phần này triển khai các mô phỏng áp dụng các giao thức tầng giao vận TCP khác nhau nhằm quan sát lối hành xử, so sánh và đánh giá hiệu suất với các giao thức cải tiến khác nhau trên kênh vệ tinh với kết nối TCP đầu cuối – đầu cuối.

Chương III: Để cải tiến hiệu suất trên kênh Viễn thông vệ tinh có một số phương pháp đã được các nhà khoa học nghiên cứu và áp dụng, luận văn chỉ đề cập đến trên phương diện lý thuyết để có được một cái nhìn tổng thể. Quan trong nhất của phần này là việc cố gắng áp dụng kỹ thuật Snoop với trạm mặt đất chiều phát dữ liệu lên kênh vệ tinh nhằm cải tiến hiệu suất làm việc. Bản chất của Snoop là một trạm trung gian nhằm thực hiện việc phát hiện lỗi gói tin và phát lại. Bài viết cũng tiến hành mô phỏng và so sánh hiệu suất làm việc khi các

giao thức điều khiển tầng giao vận TCP mở rộng làm việc khi có trạm Snoop này.

Chương IV: So sánh trực tiếp giữa các giao thức TCP mở rộng làm việc trên kênh vệ tinh GEO với kết nối đầu cuối – đầu cuối và kết nối có trạm Snoop nhằm thấy được hiệu quả của việc áp dụng cải tiến.

Mục đính của Luận văn chính là so sánh hiệu suất của các giao thức TCP mở rộng trên kênh vệ tinh GEO. Với việc áp dụng cải tiến là trạm Snoop vào trạm mặt đất nhằm cải thiện hiệu suất làm việc. Với hình thức thực nghiệm mô phỏng, so sánh kết quả để có phản ánh trực tiếp về kết quả của việc áp dụng cải tiến trong điều kiện lý thuyết.

Vì điều kiện chỉ cho phép thực hiện bằng các mô phỏng và lý thuyết, tôi hy vọng rằng kết quả có thể được tiếp tục nghiên cứu một cách rộng rãi và thử nghiệm bằng thực tiễn để có thể khẳng định tính đúng đắn của kết quả đã được chỉ ra bằng mô phỏng.

CHƯƠNG I: CƠ SỞ LÝ THUYẾT

1. Vệ tinh và Internet

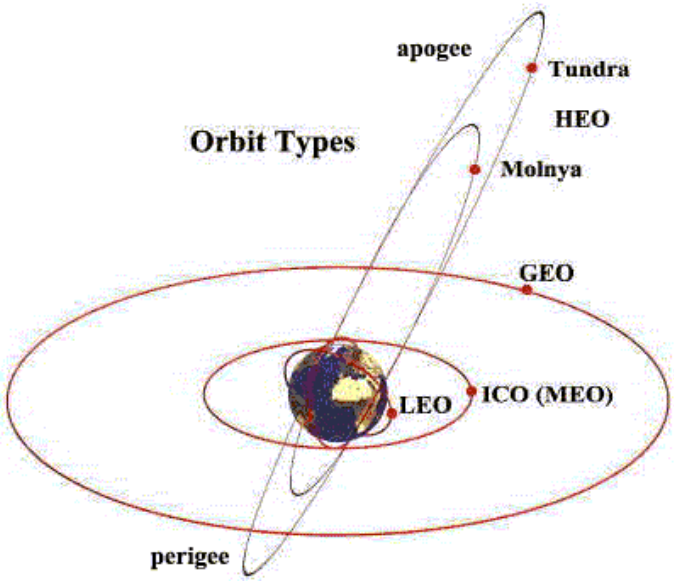
1.1. Giới thiệu về hệ thống vệ tinh

Thông tin vệ tinh là sự kết hợp của 2 thành phần: vệ tinh và trạm mặt đất.

Vệ tinh: được tạo thành bởi 3 thành phần riêng biệt đó là hệ thống đẩy và nhiên liệu, thân vệ tinh và hệ thống cho phép điều khiển từ xa, hệ thống chuyển tiếp thông tin. Hệ thống chuyển tiếp thông tin bao gồm anten để thu nhận tín hiệu từ trạm mặt đất, với cổng thu băng thông rộng, dồn kênh, chuyển đổi tần số, chúng 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. Trạm mặt đất gồm hai chức năng chính: Với chức năng truyền thông tin lên vệ tinh, dữ liệu trên mặt đất dưới hình thức những tín hiệu dải tần cơ sở phù hợp thông qua xử lý dải tần cơ sở, qua bộ chuyển đổi gửi đi, bộ khuếch đại tín hiệu công suất lớn, thông qua anten lòng chảo truyền thông tin đến vệ tinh trên quỹ đạo. Với chức năng nhận thông tin từ vệ tinh, chúng làm việc ngược với quá trình truyền thông tin lên vệ tinh, sau đó chuyển tín hiệu nhận được từ anten lòng chảo thành tín hiệu trên dải tần cơ sở.

Truyền thông vệ tinh được sử dụng rộng rãi trong nhiều lĩnh vực. Một ví dụ, chúng được triển khai cho mạng điện thoại truyền thống, mạng tế bào, truyền tín hiệu truyền hình, truyền thông trong ngành hàng hải, điện thoại đi động vệ tinh, thông tin từ vệ tinh cho thông tin thương mại và dịch vụ thông tin trên diện rộng.



Hình 1: Quỹ đạo một số loại vệ tinh

Một số loại quỹ đạo vệ tinh [3]:

Tên	*Độ cao so với mặt đất*	*Đặc trưng*
LEO	< 2000 Km	
MEO	10.000 Km	Quỹ đạo 6 giờ
GEO	35.786 Km	Vị trí tương đối là cố định so với mặt đất, Quỹ đạo 24 giờ
HEO	Phía dưới cách 500Km Trên cách 50.000Km	Quỹ đạo trên mặt phẳng nghiêng 63.4^o so với mặt phẳng xích đạo

Bảng 1: Một số loại quỹ đạo vệ tinh

1.2. Đặc tính cơ bản của hệ thống vệ tinh

1.2.1. Đặc tính của mạng vệ tinh

Ưu điểm chính của GEO là dải thông lớn, đảm bảo truyền thông liên tục, không đắt, cấu trúc mạng đơn giản có khả năng triển khai các thông tin quảng bá.

Dải thông lớn: Vệ tinh làm việc trên dải tần 3-30Ghz có thông lượng chuyển qua hàng gigabit một giây.

Truyền thông tin liên tục: Người sử dụng có thể thoả mãn với việc không bị mất kết nối với các thông tin di động trong vùng bao phủ rộng của vệ tinh.

Chi phí không cao: hệ thống vệ tinh có chi phí vừa phải bởi không phải mất chi phí cho việc thiết lập hệ thống cáp. Mặt khác vệ tinh cho phép phục vụ vùng che phủ rất lớn.

Cấu trúc liên kết mạng đơn giản: GEO có nhiều đường kết nối, điều này làm cho mạng dễ dàng thiết lập và dễ dàng quản trị.

Thông tin quảng bá và thông tin truy nhập đa điểm: Mạng vệ tinh đương nhiên phù hợp cho các ứng dụng phát triển cho kiểm thông tin quảng bá và thông tin truy nhập đa điểm.

Hệ thống thông tin vệ tinh là một thách thức lớn với hiệu suất của các ứng dụng Internet.

Độ trễ: trong hệ thống thông tin vệ tinh GEO, độ trễ tối thiểu 250ms, đôi khi việc đóng gói số liệu, hàng chờ số liệu, hệ thống chuyển mạch bên trong có thể tạo thêm độ trễ kết nối từ trạm đầu cuối đến trạm đầu cuối làm cho độ trễ lên đến 400ms. Ước lượng độ trễ này cao hơn 10 lần kết nối điểm đến điểm trong kết nối cáp quang xuyên nước Mỹ và

gấp khoảng 25 lần độ trễ kết nối mạng trục Internet Bắc-Nam Việt Nam (khoảng 16ms - số liệu VDC1). Độ trễ này ảnh hưởng đến các ứng dụng có tính tương tác cần đến sự bắt tay làm việc giữa hai phía. Giao thức điều khiển giao vận của Internet (TCP) cần đến sự tương tác trên. Chúng ta sẽ nghiên cứu ảnh hưởng của TCP thông qua mạng có độ trễ lớn và có thể làm gì để giảm bớt những ảnh hưởng tiêu cực đó.

Dải thông giới hạn: phụ thuộc vào giới hạn trải phổ của sóng radio, mà có một số lượng nhất định dải thông chỉ định cho truyền thông của vệ tinh.

Nhiễu: cường độ tín hiệu sóng radio tương xứng với khoảng cách truyền đi. Khoảng không là nơi sinh ra nhiễu, không xác định trước được sự méo tín hiệu do không gian gây nên. Do khoảng cách không gian giữa trạm mặt đất và vệ tinh, tín hiệu nhận được trở nên yếu.

Tính không đối xứng: Với đặc tính của giao thức tầng giao vận, sự mất đối xứng có thể xuất hiện khi mà thông lượng gửi đi đạt được phụ thuộc không chỉ vào đặc điểm của liên kết và mà mức độ truyền trên đường gửi đi mà nó còn phụ thuộc vào luồng thông tin phản hồi [5]. Hệ thống mạng vệ tinh có thể mất đối xứng trên rất nhiều mặt. Một số mạng vệ tinh vốn đã mất đối xứng về dải tần. Ví dụ: như việc dựa trên các vệ tinh quảng bá thông tin trực tiếp (DBS) cho chiều tải xuống và thông tin phản hồi trở về với việc quay số bằng modem. Phụ thuộc vào đường đi, điều này có thể thấy trong trường hợp hệ thống lai ghép hai vệ tinh GEO và LEO; Ví dụ chiều tải xuống bằng quảng bá thông tin trực tiếp từ GEO và chiều phản hồi qua LEO. Hệ thống LEO có thê gây ra cả mất đối xứng về dải tần và trễ. Đối với hệ thống GEO hay LEO về tính thuần tuý sự mất đối xứng về dải tần có thể tồn tại với rất nhiều khách hàng do yếu tố kinh tế. Ví dụ: rất nhiều hệ thống dự kiến sẽ cho phép người sử dụng với thiết bị đầu cuối nhỏ có thể tải dữ liệu với tốc độ 10Mb/s nhưng do kích cỡ của các thiết bị truyền tải liên kết lên trên cũng như giá của các thiết bị lọc công suất không cho phép kết nối lên với tốc độ vượt quá vài trăm Kb/s hay vài Mb/s trừ khi phải mua thiết bị đầu cuối lớn hơn.

Lỗi trong truyền tải (Tỷ lệ lỗi bit (BER)): Sử dụng các thiết bị trễ và rất nhiều thiết bị nhận và phát tín hiệu hiện nay có thể làm giảm tiêu chuẩn truyền dữ liệu tới mức thấp khoảng 10^{-7} và mức trung bình là 10^{-4} trường hợp hỏng. Điều này rất thông thường bởi vì các hệ thống hiện nay được tối ưu hoá cho hệ thống tương tự về âm thanh và hình ảnh. Với hệ thống điều biến, công nghệ mã hoá mới, cùng với các vệ tinh công suất lớn có thể làm giảm sai số bit thông thường xuống mức rất thấp (khoảng 10^{-10}) đối với hệ thống GEO. Đối với hệ thống LEO nhiều đường và nhiều bóng chắn có thể làm tăng BER. Nhưng nhìn chung những hệ thống này có thể hy vọng được cải thiện tốt hơn trước kia.

Vấn đề tắc nghẽn: Với việc sử dụng tần số lớn, dải tần rộng, trong hệ thống liên kết vệ tinh, cổ chai chỉ xuất hiện trong liên kết giữa trái đất và vệ tinh. Những liên kết này về cơ bản có thể giảm thiểu bằng liên kết trải phổ lên và xuống, kết quả là hệ thống mạng vệ tinh có thể giảm tắc nghẽn lớn. Tuy nhiên, tại các cổng giữa vệ tinh và mạng Internet trên mặt đất và có thể trở nên dễ bị tắc nghẽn đặc biệt là trong trường hợp mất kiểm soát.

1.2.2. Kiến trúc hệ thống vệ tinh

Có một vài kiến trúc mạng vệ tinh mà liên kết vệ tinh có thể tích hợp vào mạng Internet.

Mạng vệ tinh phi đối xứng: Một vài mạng vệ tinh cung cấp dải thông phi đối xứng. Đó là: dữ liệu truyền theo một chiều lớn hơn dữ liệu truyền theo chiều ngược lại bởi giới hạn của năng lực truyền thông kích cỡ anten tại mỗi đầu cuối của kết nối. Điều đó có nghĩa, một vài hệ thống vệ tinh khác truyền theo một hướng duy nhất và sử dụng một nhánh kết nối phản hồi không phải là vệ tinh khác (Ví dụ như kết nối thông qua Modem). Bản chất của hầu hết lưu lượng TCP là phi đối xứng với luồng chuyển dữ liệu theo một hướng và trả lời theo hướng ngược lại.

Kết nối vệ tinh là điểm cuối: Kết nối vệ tinh cung cấp dịch vụ trực tiếp đến người sử dụng đầu cuối; ngược với việc kết nối vệ tinh đặt ở khoảng giữa của mạng, có thể cho phép thiết kế một giao thức đặc biệt sử dụng qua điểm cuối. Một vài nhà cung cấp vệ tinh sử dụng kết nối vệ tinh như một kết nối tải xuống tốc độ cao được chia cho những người sử dụng với tốc độ thấp hơn, một kết nối cố định không được chia sẻ kết nối trở lại để giữ các yêu cầu và thông tin phản hồi. Điều này tạo ra mạng không đối xứng được đề cập ở trên.

Mạng vệ tinh lai ghép: Trong nhiều trường hợp tổng quát hơn, những liên kết vệ tinh có thể được đặt tại bất kỳ điểm nào trong kiến trúc liên kết mạng. Trong trường hợp này, liên kết vệ tinh đóng vai chỉ là kết nối khác giữa hai cổng kết nối. Trong môi trường này, một liên kết đưa ra có thể gửi không qua kết nối trên mặt đất (bao gồm cả kết nối trên mặt đất không dây), cũng như kết nối vệ tinh. Mặt khác, việc kết nối cũng có thể truyền tải chỉ qua mạng trên mặt đất hoặc qua phần vệ tinh của mạng.

Mạng vệ tinh điểm-điểm: trong mạng vệ tinh điểm điểm, chỉ có duy nhất chặng kết nối trong mạng và bằng liên kết vệ tinh. Đó là môi trường vệ tinh thuần nhất đặc trưng các vấn đề kết nối giữa vệ tinh.

Liên kết vệ tinh nhiều chặng: trong một số điểm nút, lưu lượng dữ liệu trên mạng có thể được chuyển qua nhiều chặng vệ tinh giữa hai đầu

truyền và nhận. Đó là môi trường làm tăng thêm các đặc tính của môi trường vệ tinh.

1.3. Kết nối Internet qua vệ tinh

Ngày nay, giao thức TCP được sử dụng rộng rãi và chiếm phần lớn trong các ứng dụng Internet. Hiệu suất của giao thức TCP thông qua môi trường mạng có độ trễ lớn sẽ ảnh hưởng lớn trực tiếp đến truy nhập Internet sử dụng vệ tinh GEO. Giao thức TCP xuất hiện làm việc tốt với liên kết vệ tinh bằng cách hoạt động ở tốc độ thấp. Vì thế, làm mất đi đặc tính của hiệu suất làm việc với kênh truyền số liệu tốc độ cao đáng lẽ có được bởi khả năng tiềm tàng của thông tin vệ tinh.

Kết quả hiệu suất trong cơ chế điều khiển của TCP

Kích thước cửa sổ: điều khiển lưu lượng của TCP được bắt đầu với khái niệm "kích thước cửa sổ" trong kết nối TCP. Nó xác định có thể có bao nhiêu dữ liệu có thể đang lưu chuyển mà chưa xử lý. Trong mạng có độ trễ lớn, có thể có nhiều phân đoạn không có trả lời. Trên lý thuyết, dung lượng trên thời gian trễ của kênh truyền (bandwith-delay) cho ta cho ta độ lớn của dữ liệu có thể truyền mà không cần sự trả lời. Nhưng trên thực tế, bộ nhớ và tài nguyên của các hệ thống làm giới hạn của kích thước cửa sổ. Với TCP chuẩn hiện nay, kích thước cửa sổ lớn nhất là 64Kbyte. Trong quá trình triển khai thực tế, kích thước giới hạn của cửa sổ thường là 32Kbyte.

Để có thể sử dụng tối đa dải thông của mạng vệ tinh, giao thức TCP cần kích thước cửa sổ lớn hơn rất nhiều. Ví dụ: Trên liên kết vệ tinh, với tổng thời gian trễn toàn phần là 0.8 giây, dải thông của kết nối vệ tinh là 1.54Mbps, theo lý thuyết, kích thước cửa sổ tối ưu là 154Kbyte; lớn hơn rất nhiều so với kích thước tối đa là 32Kbyte hay 64Kbyte.

Sự thích nghi dải thông: Giao thức TCP có thể thích ứng với với dải thông có thể của mạng bởi việc tăng kích thước cửa sổ, tránh tắc nghẽn và giảm kích thước cửa sổ cho phù hợp. Tốc độ thích nghi tương xứng với độ trễ; Ví dụ: Thời gian trễ toàn phần của thông tin phản hồi. Trong mạng vệ tinh với thời gian trễ lớn, thích nghi dải thông cũng mất nhiều thời gian hơn, và như là giá phải trả, điều khiển chống tắc nghẽn gần như không có hiệu quả.

Phát lại có lựa chọn (Selective Acknowledgement): thứ tự các thông tin phản hồi của TCP chuẩn có tính lũy tích. Nếu một phân đoạn bị mất, TCP phát sẽ truyền lại tất cả dữ liệu kết từ phân đoạn bị mất mà không quan tâm đến các phân đoạn đã truyền thành công sau đó. Giao thức TCP coi như việc mất một phận đoạn là thể hiện sự xung đột và giảm kích thước cửa sổ đi một nửa.

Bắt đầu chậm: Khi kết nối TCP bắt đầu lần đầu tiên hoặc rỗi không truyền số liệu đã lâu, nó cần thiết nhanh chóng xác định khả năng dải thông của mạng. TCP thực hiện bằng cách bắt đầu bằng việc khởi tạo kích thước cửa sổ của một phân đoạn (thông thường 512 Byte), sau đó tăng kích thước cửa sổ mỗi khi gói tin được gửi đi thành công và thông tin phản hồi trở lại, cho đến khi phát hiện trạng thái bão hoà của mạng (thể hiện bởi gói tin truyền đi bị huỷ bỏ). Một mặt, bắt đầu chậm tránh tắc nghẽn mạng trước khi nó xác định được khả năng dải thông làm việc tốt của mạng. Mặt khác, tận dụng dải thông là chức năng tối ưu phụ mà bắt đầu chậm thực hiện. Bắt đầu chậm với liên kết vệ tinh cần đến khoảng 6.5 giây để tìm ra thông lượng lớn nhất. Khi mất một phân đoạn sẽ tạo ra việc chống lại xung đột, kết quả là thông lượng giảm và tiếp tục diễn ra trong một vài phút.

2. Kết nối TCP và điều khiển lưu lượng

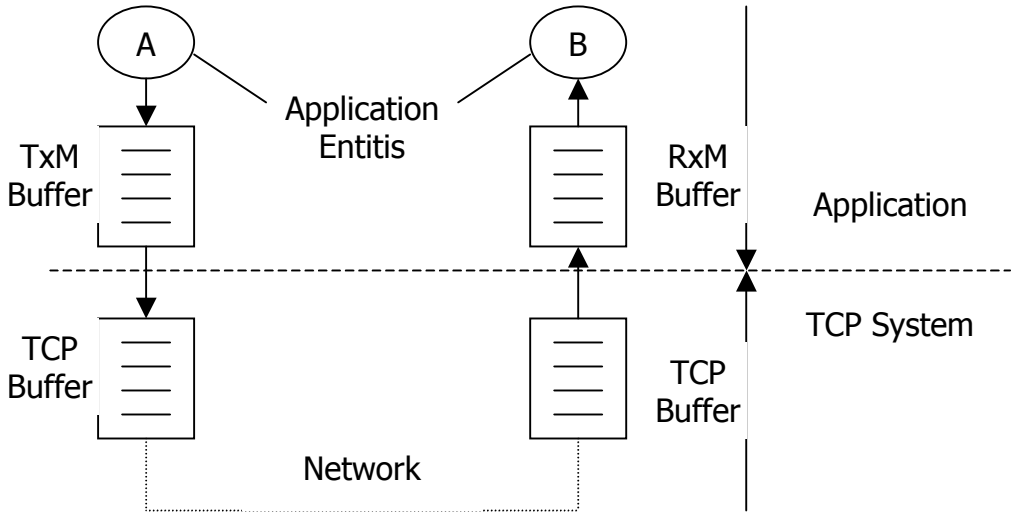
Theo [9], TCP là giao thức trao đổi số liệu, đảm bảo tin cậy và chính xác giữa hai thực thể cuối trong mạng. Việc thiết kế và thực hiện giao thức TCP phức tạp, chính bởi:

- TCP phải quản lý đúng số tuần tự tính theo Byte của dòng số liệu;
- TCP phải tối ưu hoá khả năng dải thông của mạng bằng cách giám sát và điều khiển lưu lượng số liệu từ thực thể gửi đến thực thể nhận.

TCP bảo đảm số liệu tin cậy và chính xác giữa các thực thể cuối trong mạng chính nhờ các yếu tố sau:

- **Đối thoại khi thu phát:** Mỗi khi gửi một gói số liệu, bên nhận phải thông báo nhận đúng sau một khoảng thời gian nhất định. Nếu không, gói số liệu được coi là nhận sai và được phát lại.
- **Kiểm tra số liệu thu, phát:** Số liệu gửi được kiểm tra bằng thuật toán quy định. Byte kiểm tra (checksum) được gửi cùng với số liệu phát và được so sánh với byte kiểm tra tính lại khi thu. Trong trường hợp sai lệch, có nghĩa là lỗi xảy ra trên đường truyền, thực thể thu thông báo kết quả thu cho thực thể phát và yêu cầu gửi lại.
- **Kiểm tra số tuần tự:** Vì các gói TCP được truyền đi bằng các gói IP và các gói IP có thể đến đích không theo thứ tự phát (IP là giao thức không hướng kết nối) nên thực thể TCP nhận phải lập lại trật tự các gói số liệu thu được và huỷ bỏ các gói số liệu trùng lặp khi cần và chuyển các gói số liệu đó theo đúng trật tự phát cho các ứng dụng.
- **Điều khiển lưu lượng:** Mỗi thực thể của kết nối TCP đều có một vùng đệm hạn chế. Thực thể TCP nhận chỉ cho phép thực thể phát gửi một lượng số liệu đủ với vùng đệm thu của mình. Điều này cho

phép ngăn cản thực thể TCP phát nhanh, làm tràn vùng đệm của thực thể TCP thu chậm.

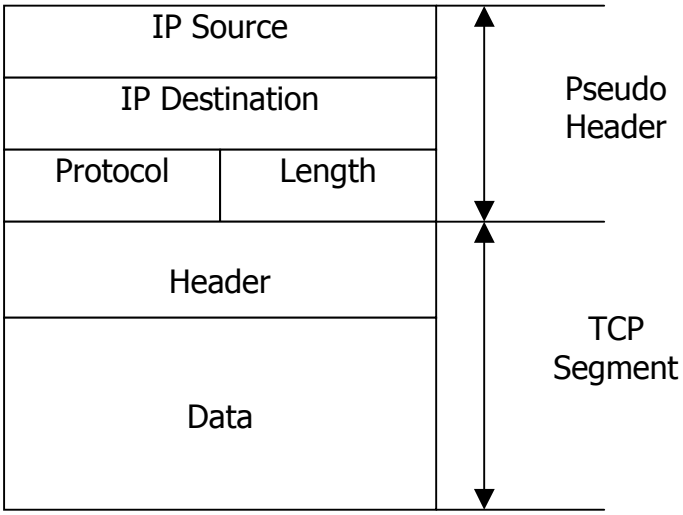


Hình 2: Mô hình hoạt động của giao thức TCP

Mô hình hoạt động của giao thức TCP được mô tả trong hình 2. Các thực thể ứng dụng dịch vụ truyền dẫn tin cậy TCP mô tả ở trên để trao đổi số liệu. Lưu ý rằng, thực thể ứng dụng và thực thể TCP có bộ đệm riêng của mình để lưu trữ tạm thời số liệu trong quá trình xử lý. Cách thức chuyển tiếp số liệu giữa hai bộ đệm trên là yếu tố quyết định hiệu suất chuyển tiếp số liệu của hệ thống TCP. Số liệu có thể được chuyển toàn bộ hoặc một phần từ bộ đệm ứng dụng tới bộ đệm TCP, trước khi quá trình phát được khởi động; Số liệu thu từ kết nối TCP có thể được chuyển tiếp tức thời từ bộ đệm thu TCP tới bộ đệm ứng dụng hoặc chỉ khi tỷ lệ phần bộ đệm bị chiếm dụng so với tổng dung lượng bộ đệm đạt tới một giá trị nào đó. Xin nhắc lại một cách tổng quan ở đây là, ngoài việc quy định về khuôn dạng của gói số liệu, giao thức chỉ quy định về cách thức trao đổi số liệu giữa các thực thể cùng mức chức năng mà không quy định việc thực hiện cụ thể như thế nào.

2.1. Cấu trúc gói số liệu TCP

Cấu trúc gói số liệu TCP bao gồm tiêu đề TCP "giả" (Pseudo header TCP) và gói số liệu TCP "thật", được mô tả trong hình 3. Phần tiêu đề giả cần thiết cho vệc xây dựng gói số liệu IP, bao gồm các thông tin về địa chỉ IP nguồn, địa chỉ IP đích, số liệu thuộc giao thức TCP (trường Protocol có giá trị 0x06) và độ dài của gói số liệu TCP thực.



Hình 3: Gói số liệu TCP với phần tiêu đề giả

Cấu trúc gói số liệu được mô tả trong hình 4, sau đây là ý nghĩa của các trường số liệu:

- Source Port và Destination Port: số hiệu cổng TCP. Cùng với địa chỉ IP nguồn và địa chỉ IP đích trong gói số liệu IP, số hiệu cổng TCP định danh duy nhất hai tiến trình ở hai đầu kết nối TCP.
- Sequenser number: Số tuần tự phát, định danh byte đầu tiên của phần số liệu TCP trong dòng số liệu từ thực thể TCP gửi đến thực thể TCP nhận. Số tuần tự phát là khoảng cách tương đối của byte đầu tiên phần số liệu với phần đầu của dòng byte; Là một số không dấu 32 bit, có giá trị từ 0 đến 2^{32}-1.
    - Nếu ta coi dòng byte là luồng dữ liệu một chiều từ một ứng dụng này tới một ứng dụng kia thì TCP đánh số tất cả các byte với giá trị gọi là số tuần tự Sequenser number.
    - Khi một kết nối được thiết lập, trường số tuần tự chứa giá trị khởi tạo ISN (Initial Sequence Number) được thực thể chọn cho kết nối này.
    - Byte số liệu đầu tiên sẽ có số tuần tự bằng ISN+1.
- Acknowlegement: vị trí tương đối của byte cuối cùng đã nhận đúng bởi thực thể gửi gói ACK cộng thêm 1. Giá trị của trường này còn được gọi là số tuần tự thu. Giá trị này đúng khi bit cờ ACK = 1.
- Data Offset: Khoảng cách tương đối của trường số liệu với phần tiêu đề của TCP (TCP header) tính theo từ 32 bit. Thông thường, trường này có giá trị bằng 5 vì độ dài thông thường của phần tiêu đề TCP là 20 byte.
- Reverved: Luôn đặt là 0, để sử dụng trong tương lai.

- FLAGs: Có 6 bit cờ trong phần tiêu đề TCP. Một hay nhiều cờ có thể được thiết lập tại cùng một thời điểm.
  - URK = 1: Thông báo giá trị trường Urgent Pointer đúng.
  - ACK = 1: Thông báo giá trị trường Acknowlegement đúng.
  - PSH = 1: Thực thể nhận phải chuyển số liệu này cho ứng dụng tức thời.
  - RST = 1: Tái khởi tạo kết nối, dùng để kết thúc kết nối.
  - SYN = 1: Đồng bộ trường số thứ tự, dùng để thiết lập kết nối TCP.
  - FIN = 1: Thông báo thực thể gửi đã kết thúc gửi số liệu.
- Windows size: độ lớn của cửa sổ, quy định tổng số byte số liệu mà thực thể thu có thể nhận được (đồng nghĩa với độ lớn của bộ đệm thu), tính khởi đầu từ giá trị trường số tuần tự thu (Acknowledgement Number).
- Checksum: Byte kiểm tra, là giá trị bù 1 của tổng các 16bit trong phần đầu và phần số liệu TCP. Giá trị này tính cả 12 byte tiêu đề giả của TCP.
- Urgent Pointer: Vị trí tương đối của byte trong trường số liệu TCP cần được xử lý đầu tiên. Giá trị trường này đúng khi bit cờ URG = 1.
- Option: Tuỳ chọn. Tuỳ chọn duy nhất được dùng hiện nay là quy định về độ dài lớn nhất MSS (Maximum Segment Size) của một gói số liệu TCP.
- Pad: Phần vá thêm để phần tiêu đề của gói TCP có độ lớn là bội của 4 byte.
- Data: Phần số liệu của ứng dụng TCP.

<table>
<tr><td colspan="8">16 bit Source Port Number</td><td colspan="4">16 bit Desstination Port Number</td></tr>
<tr><td colspan="12">32 bit Sequenser Number</td></tr>
<tr><td colspan="12">32 bit Acknowledgement Number</td></tr>
<tr><td>4 bit Offset</td><td colspan="2">6 bit Reserverd</td><td>U</td><td></td><td>P</td><td></td><td>S</td><td>F</td><td colspan="4">16 bit Windwos Size</td></tr>
<tr><td colspan="8">16 bit TCP Checksum</td><td colspan="4">16 bit Urgen Pointer</td></tr>
<tr><td colspan="10">Option (If any)</td><td colspan="2">Pad</td></tr>
<tr><td colspan="12">TCP data</td></tr>
</table>

Hình 4: Cấu trúc gói số liệu TCP

2.2. Điều khiển lưu lượng và điều khiển tắc nghẽn

2.2.1. Cơ chế cửa sổ động

Cơ chế cửa sổ động là một trong các phương pháp điều khiển lưu lượng số liệu trong mạng thông tin máy tính. Độ lớn cửa sổ chính bằng số gói số liệu được gửi kèm mà không cần chờ thông báo trả lời về kết quả nhận các gói số liệu đó. Ví dụ: nếu cửa sổ W=3 thì sau khi gửi 3 gói số liệu liên tiếp nhau, thực thể phát phải chờ trả lời về kết quả nhận 3 gói số liệu nói trên trước khi gửi 3 gói số liệu tiếp theo.

Độ lớn của cửa sổ quyết định hiệu suất trao đổi trong mạng. Nếu chọn độ lớn của cửa sổ cao thì có thể gửi được nhiều dữ liệu trong cùng một đơn vị thời gian. Tuy nhiên, một khi việc truyền số liệu gặp lỗi, trong trường hợp này, rõ ràng số liệu phải gửi lại lớn và vì vậy hiệu quả sử dụng đường truyền thấp (tỷ lệ giữa số liệu được chuyển thực sự trên tổng số số liệu được trao đổi trên mạng).

Giao thức TCP cho phép thay đổi độ lớn cửa sổ một cách thụ động, phụ thuộc vào độ lớn bộ đệm thu của thực thể TCP nhận. Để thực hiện cơ chế cửa sổ động, TCP bảo đảm:

- Trả lời về số tuần tự thu tiếp theo trong trường ACK, nghĩa là khẳng định tổng số Byte nhận đúng cho đến thời điểm gửi gói trả lời này.
- Thông báo về tổng số byte nhận được, tương ứng với độ lớn bộ nhớ đệm thu.

2.2.2. Cơ chế phát lại thích nghi

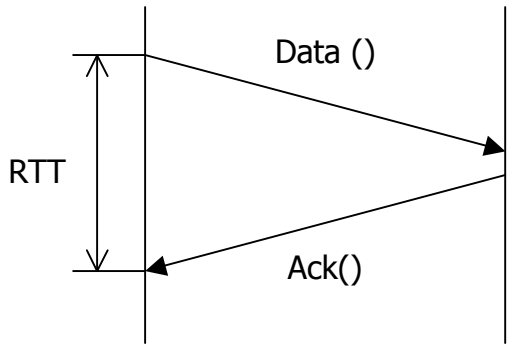
Để bảo đảm kiểm tra, phát hiện và khắc phục lỗi trong việc trao đổi số liệu qua mạng diện rộng, được kết nối từ nhiều mạng khác nhau, TCP phải có cơ chế đồng hồ kiểm tra phát (time-out) và cơ chế phát lại (retransmission) mềm dẻo, thay đổi phụ thuộc vào thời gian trễ thực của môi trường truyền dẫn cụ thể. Thời gian trễ toàn phần RTT (Round Trip Time), được xác định từ thời điểm bắt đầu phát gói số liệu cho đến khi nhận được trả lời về kết quả nhận của thực thể đối tác (Hình 5), là yếu tố quyết định giá trị của đồng hồ kiểm tra phát T_{out}. Rõ ràng T_{out} phải lớn hơn hoặc bằng RTT.

Cơ chế phát lại thích nghi được dựa trên việc xác định thời gian trễ toàn phần RTT theo thời gian. Bằng việc sử dụng các hàm xác định thời gian của hệ điều hành, hoàn toàn có thể xác định được thời gian trễ toàn phần của một kết nối TCP tại các mốc thời gian nhất định [14]. Có thể tính thời gian trễ toàn phần RTT theo công thức sau:

$$RTT = \alpha \times old\_RTT + (1 - \alpha) \times new\_RTT, \text{ trong đó } 0 \le \alpha < 1$$

Từ đó ta có thể tính thời gian kiểm tra phát T_{out} theo công thức:

$$T_{out} = \beta \times RTT, \text{ với } \beta = 2$$



Hình 5: Xác định thời gian trễ toàn phần RTT

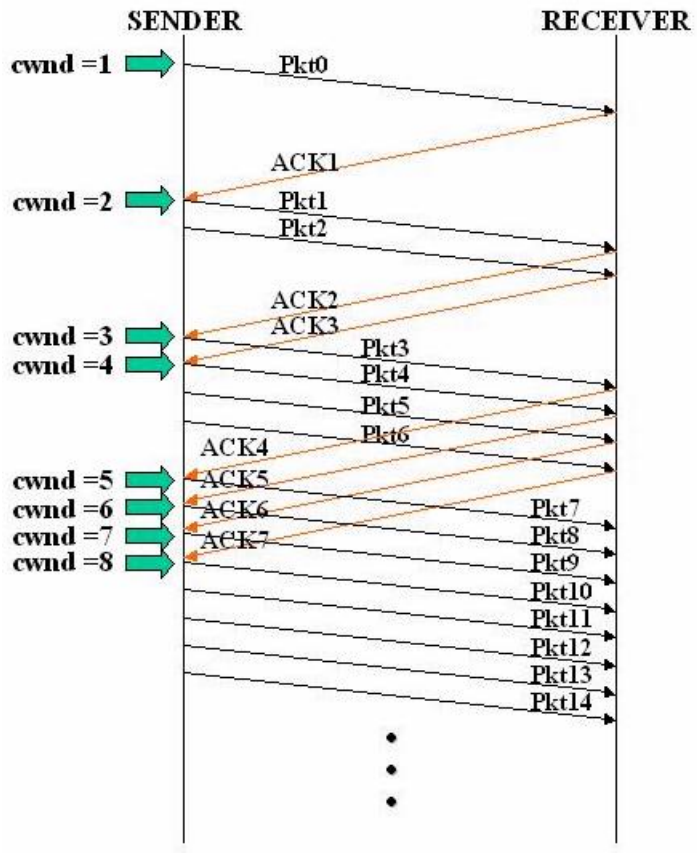
2.2.3. Cơ chế điều khiển tắc nghẽn số liệu

a. Thuật toán "bắt đầu chậm":

Thuật toán "bắt đầu chậm" (slow start) định nghĩa thêm khái niệm "cửa sổ tắc nghẽn" (congestion windows), gọi là cwnd, cho thực thể phát. Một khi kết nối TCP được thiết lập, cwnd được gán giá trị cwnd = 1 đơn vị gói số liệu có độ dài được thông báo bởi thực thể TCP đối tác, hoặc độ dài mặc định là 536 Byte hoặc 512 Byte.

Thực thể TCP bắt đầu phát với cwnd = 1. Mỗi khi nhận được thông báo trả lời ACK, giá trị cửa sổ cwnd được tăng thêm 1 (nói một cách khác giá trị cwnd tăng lên gấp đôi), cho đến khi xuất hiện tình trạng tắc nghẽn số liệu thể hiện qua việc gia tăng của thời gian trễ

toàn phần RTT. Đến đây, thực thể phát cần biết rằng độ lớn cửa sổ cwnd đã quá lớn, và cần phải có điều chỉnh phù hợp.



Hình 6: Lược đồ thời gian thuật toán bắt đầu chậm

b. Thuật toán "tránh tắc nghẽn" số liệu

Đặc trưng của hiện tượng tắc nghẽn số liệu là thời gian trễ toàn phần RTT tăng. Những nguyên nhân chính dẫn đến sự gia tăng giá trị RTT là:

- Hiện tượng mất gói số liệu tại một hệ định tuyến nào đó trong mạng (do thiếu bộ nhớ chẳng hạn), dẫn đến time-out và thực thể phát lại gói số liệu bị mất. Hoặc;
- Thực thể phát nhận được nhiều lần các thông báo trả lời ACK cho biết tại thực thể nhận, các gói số liệu đã đến không đúng theo thứ tự phát (out-of-order).

Biện pháp tránh tắc nghẽn số liệu chính là giảm lưu lượng số liệu phát vào kênh TCP được thiết lập, từ đó hạn chế tình trạng mất gói số liệu. Thuật toán "bắt đầu chậm" (slow start) và "tránh tắc nghẽn" (congestion Avoidance) là những kỹ thuật điều khiển lưu lượng, tránh tắc nghẽn hoạt động độc lập, song có quan hệ mật thiết với nhau và thường được cài đặt cùng nhau. Hoạt động của hai giải pháp kết hợp này như sau:

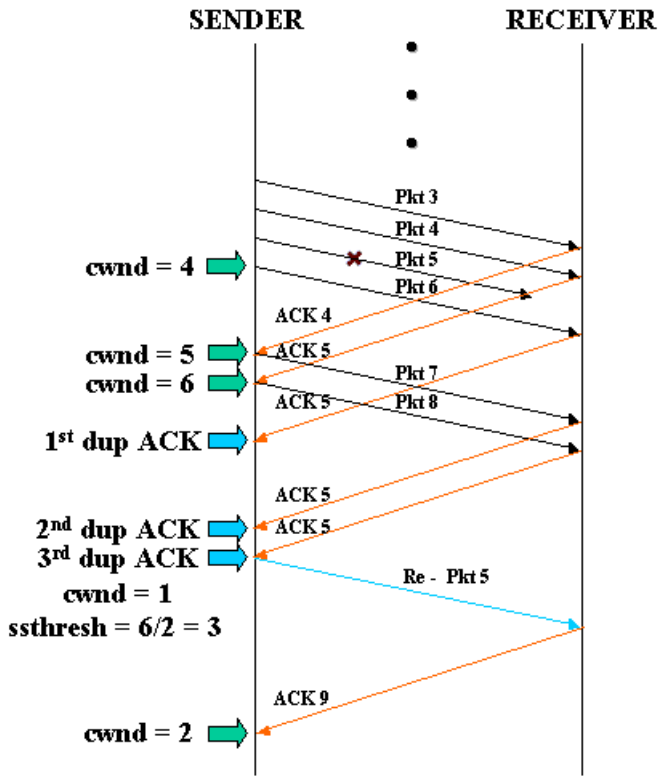
- Đặt giá trị ban đầu cwnd = 1 đơn vị gói số liệu và ssthresh = 65.535 Byte, trong đó ssthresh là ngưỡng trên của thuật toán "bắt đầu chậm".
- Thực thể TCP không bao giờ phát phát nhiều hơn giá trị cửa sổ tối thiểu cwnd và độ lớn cửa sổ thu rcnd được thực thể thu thông báo trước đó: W = min {cwnd, rcnd}.
- Khi hiện tượng tắc nghẽn số liệu xuất hiện, đặt giá trị cửa sổ W/2 (không nhỏ hơn 2 đơn vị gói số liệu) lưu trữ trong trường ssthresh và đặt cwnd = 1 đơn vị gói số 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:
    - Nếu cwnd < ssthresh, thuật toán bắt đầu chậm được thực hiện: giá trị cwnd được tăng 1 đơn vị gói số liệu với mỗi thông báo ACK nhận được.
    - Ngược lại, nếu cwnd = ssthresh, thuật toán tránh tắc nghẽn được thực hiện: giá trị cwnd được tăng 1/cwnd với mỗi thông báo ACK nhận được.

Có thể thấy độ gia tăng giá trị cửa sổ phát trong trường hợp điều khiển tránh tắc nghẽn là hàm tuyến tính, trong khi đó thuật toán bắt đầu chậm (không tắc nghẽn) tăng theo hàm số mũ. Điều này đảm bảo tận dụng băng thông được cấp phát, tăng hiệu suất trao đổi số liệu trên kết nối TCP khi áp dụng thuật toán bắt đầu chậm và vẫn phòng tránh có hiệu quả khi hiện tượng tắc nghẽn số liệu xuất hiện.

c. Thuật toán "phát lại nhanh"

Thông thường, thực thể TCP thực hiện phát lại một gói số liệu khi nhận được thông báo NAK (thu sai) hoặc được đồng hồ quản lý phát lại kích hoạt (time-out). Thuật toán phát lại nhanh (Fast Retransmision) cho phép thực thể phát thực hiện phát lại không cần chờ đồng hồ time-out, trong trường hợp nhận được nhiều hơn ba thông báo ACK lặp lại (duplicated ACK).

Thông báo ACK lặp lại được tạo ra trong trường hợp thực thể nhận thông báo gói số liệu đến không đúng theo thứ tự phát (out-of-order) và nêu số thứ tự gói số liệu chờ nhận. Trong thực tế, thường chỉ cần từ một đến hai thông báo ACK lặp lại là đủ để gói số liệu "bị lạc" đến và và được xử lý đúng thứ tự. Nếu thực thể phát nhận được nhiều hơn ba thông báo ACK lặp lại thì điều đó đồng nghĩa với việc gói số liệu bị mất (packet loss) và trong trường hợp này, gói số liệu đó cần được phát lại nhanh.



Hình 7: Lược đồ thời gian thuật toán phát lại nhanh

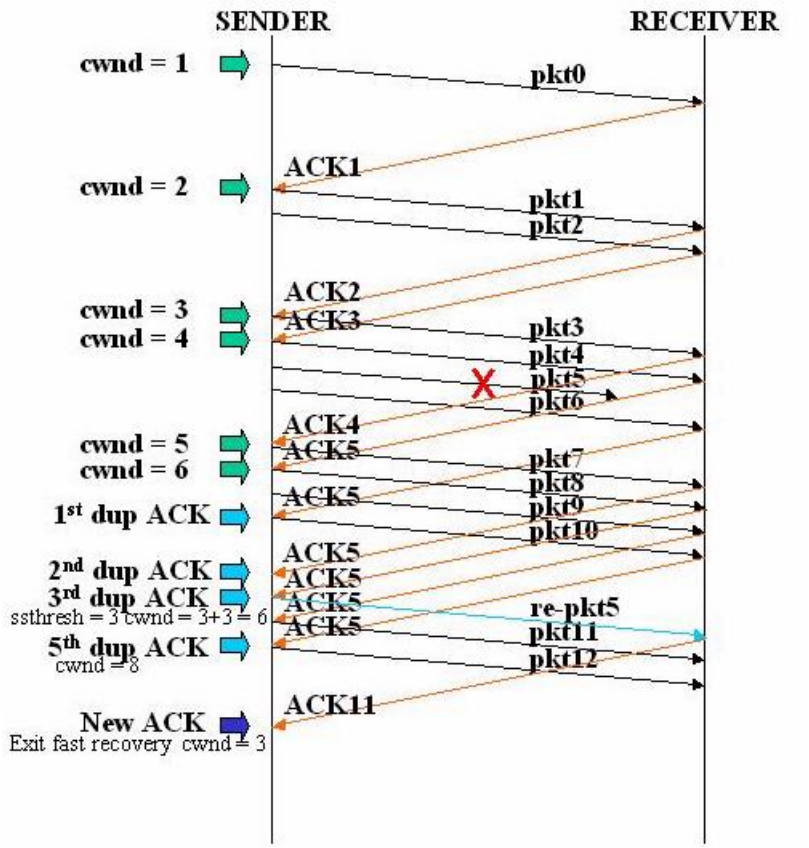
d. Thuật toán "khôi phục nhanh"

Thuật toán khôi phục nhanh (Fast recovery) quy định việc thực hiện thuật toán tránh tắc nghẽn ngay sau khi thực hiện thuật toán phát lại nhanh. Điều này tránh cho lưu lượng số liệu trong kết nối TCP không bị thay đổi đột ngột - nếu thực hiện thuật toán bắt đầu chậm - và bảo đảm thông lượng số liệu đạt được phù hợp với bối cảnh thực tế: việc phát, thu có lỗi.

Thuật toán phát lại nhanh và khôi phục nhanh thường được cài đặt cùng nhau như sau:

- Sau khi nhận được ba thông báo ACK lặp lại liên tiếp nhau, thực thể phát thiết lập ssthresh = wncd/2 (không nhỏ hơn hai đơn vị gói số liệu smss) và phát lại gói số liệu mất; tăng cwnd = cwnd + 3 * smss. Điều này cho phép tăng "nhân tạo" cửa sổ phát cwnd tương ứng với ba gói số liệu đã "rời" khỏi mạng, hay nói cách khác: chúng không còn chiếm giữ tài nguyên của mạng (thực chất là: ba gói số liệu đã được nhận trong bộ đệm thu của thực thể nhận, tương ứng với và thông báo ACK lặp lại).
- Với mỗi thông báo ACK lặp lại, tăng cwnd = cwnd +1 * smss, tương ứng với một gói số liệu được phát vào mạng (và như được giữ trong bộ đệm thu của thực thể nhận).

- Sau khi nhận được thông báo trả lời ACK không bị lặp lại, nghĩa là thông báo về gói số liệu được nhận đúng trong khoảng thời gian cho đến thời điểm đó - bao gồm thời gian chờ xử lý lỗi (phát lại thông báo ACK cho đến khi thực thể phát phát lại gói số liệu bị mất) và thời gian sau khi phát lại, cho đến khi gói số liệu phát lại được nhận đúng - thực thể TCP thiết lập lại giá trị cửa sổ cwnd được giữ trong trường ssthresh và thực hiện phát bình thường với quy tắc W = min {cwnd, rcnd}.



Hình 8: Lược đồ thời gian thuật toán khôi phục nhanh

2.3. Một số thuật toán TCP mở rộng

2.3.1. TCP Tahoe

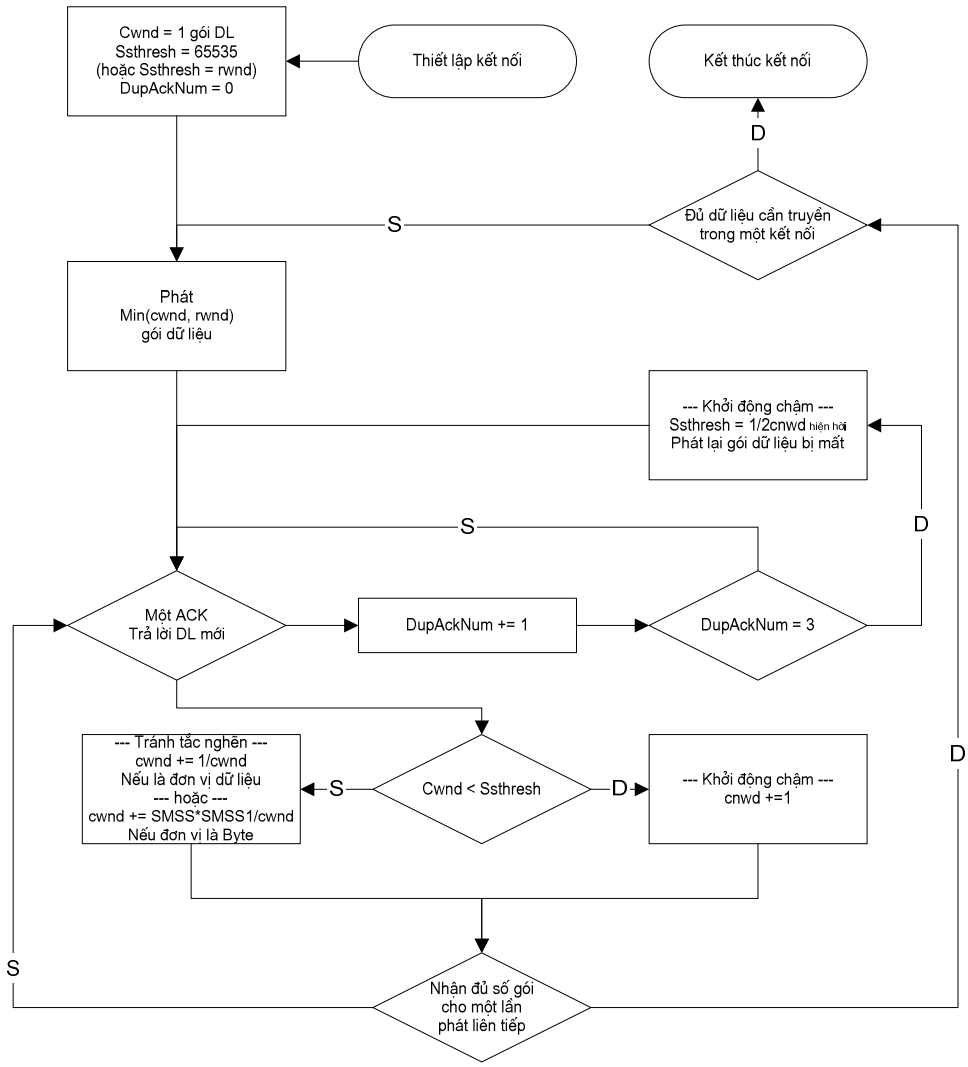
Các phiên bản TCP hiện tại bao gồm một số thuật toán hướng tập trung vào điều khiển tắc nghẽn trên mạng nhưng vẫn duy trì (đảm bảo) thông lượng cho người sử dụng. Ban đầu những sự thực hiện TCP đã theo kiểu go-back-n có sử dụng ACK tích luỹ (cumulative positive ACK) và thông qua một đồng hồ phát lại để phát lại dữ liệu bị mất trong thời gian truyền. Các TCP này cũng làm giảm một phần tắc nghẽn trên mạng.

Phiên bản Tahoe-TCP có thêm một số các thuật toán mới và cải tiến các phiên bản trước đây. Các thuật toán mới bao gồm "Khởi động chậm", "Tránh tắc nghẽn" và "Phát lại nhanh" [14] mà đã được mô tả trong phần trên của bài luận văn này. Những sự cải tiến bao gồm một

sự sửa đổi đối với bộ ước lượng RTT đã được sử dụng để đặt các giá trị thời gian chờ phát lại. Tất cả các thay đổi này đã được mô tả trong [14, 15].

Thuật toán "Phát lại nhanh" là một lưu ý đặc biệt bởi vì nó được thay đổi trong các phiên bản tiếp theo của TCP. Với "Phát lại nhanh", sau khi nhận được một số lượng nhỏ các ACK trùng cho gói dữ liệu cùng TCP, thực thể gửi dữ liệu phỏng đoán rằng một gói dữ liệu đã bị mất và truyền lại gói dữ liệu này không cần đợi thời gian chờ hết hạn, dẫn đến việc sử dụng các kênh cao hơn và cải thiện thông lượng kết nối.

Thuật toán: Thuật toán điều khiển tắc nghẽn Tahoe-TCP là kết hợp của ba thuật toán cơ sở "Khởi động chậm", "Tránh tắc nghẽn" và "Phát lại nhanh" được mô tả ở trên. Đặc trưng của Tahoe-TCP là khi phát hiện thấy mất dữ liệu thông qua 3 ACK trùng thực thể gửi phát lại gói dữ liệu bị mất, đặt cwnd bằng 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 cho mỗi thời gian trễ toàn phần (RTT) và Tahoe có thể phát lại các gói dữ liệu mà các gói này có thể đã được truyền thành công.



Hình 9: Sơ đồ thuật toán Tahoe-TCP

2.3.2. TCP Reno

Sự thực hiện Reno-TCP vẫn giữ lại sự ưu việt được hợp nhất trong 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" [17]. Thuật toán mới loại bỏ đường truyền "rỗng" sau khi "Phát lại nhanh", bằng cách đó tránh việc cần thiết phải "Khởi động chậm" để lấp đầy chỗ trống sau khi một gói dữ liệu bị mất. "Phục hồi nhanh" hoạt động với giả thiết mỗi một ACK trùng nhận được đại diện cho một gói dữ liệu (đơn) đã dời khỏi khỏi mạng. Như vậy trong thời gian "Phục hồi nhanh" thực thể gửi TCP có khả năng làm các ước lượng (đánh giá) thông minh về tổng số dữ liệu còn tồn đọng trên mạng.

"Phục hồi nhanh" được bắt đầu bởi thực thể gửi TCP sau việc nhận một ngưỡng khởi tạo của các gói ACK trùng. Ngưỡng này thông thường được đặt giá trị 3. Một khi ngưỡng của của các ACK trùng được nhận, thực thể gửi phát lại một gói dữ liệu và giảm cửa sổ tắc nghẽn của nó đi một nửa. Thay vì "Khởi động chậm" như đã được thực hiện bởi một thực thể gửi Tahoe-TCP, thực thể gửi Reno-TCP sử dụng các ACK trùng bổ sung vừa đến để làm "clock" (làm nhịp) cho các gói dữ liệu được gửi đi tiếp sau.

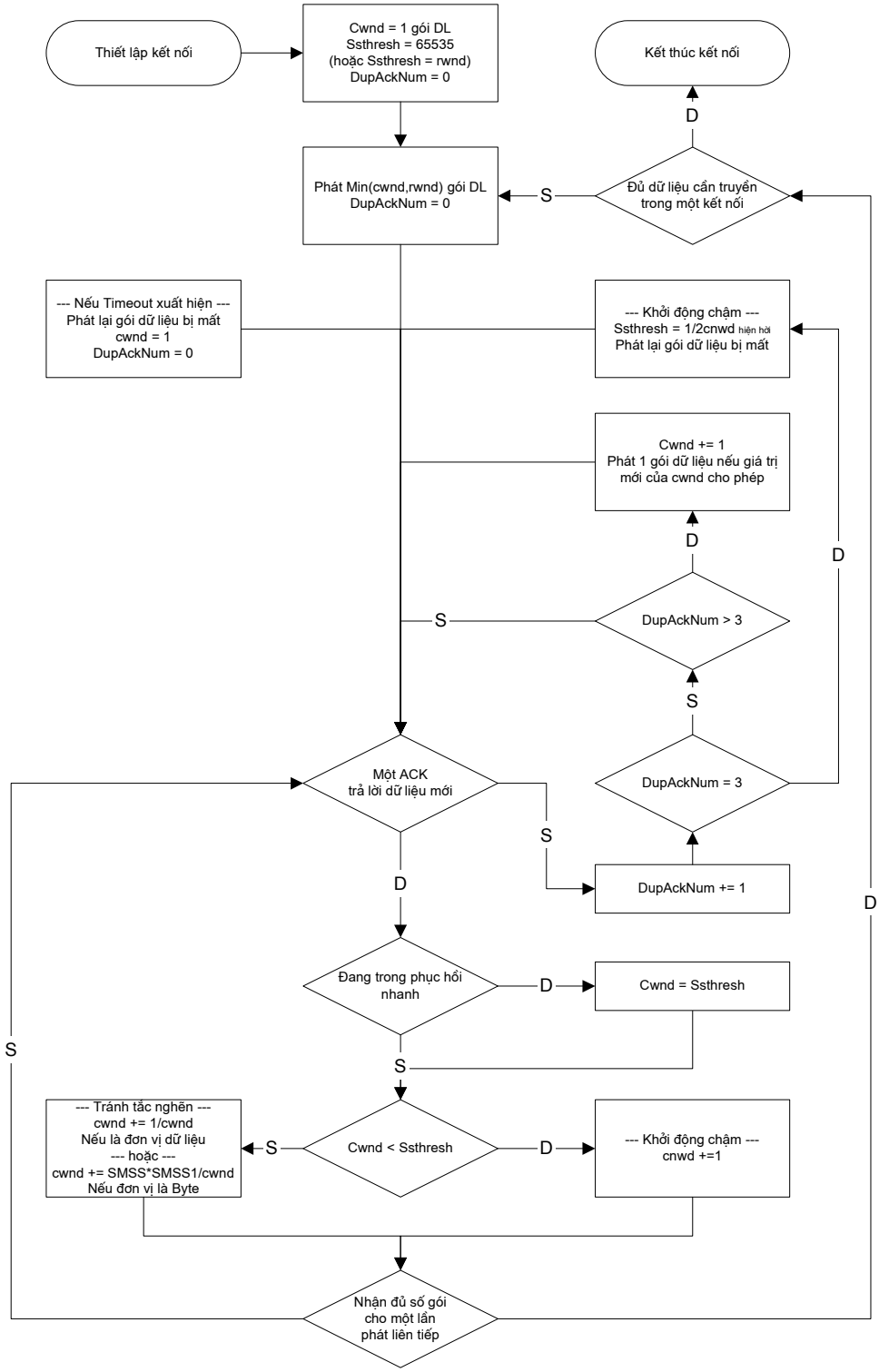
Trong Reno, cửa sổ có-thể-sử-dụng-được của thực thể gửi là min(rwnd, cwnd + ndup) trong đó rwnd là cửa sổ đã được thông báo của thực thể nhận, cwnd là cửa sổ tắc nghẽn của thực thể gửi, và ndup được duy trì bằng 0 cho đến khi số lượng các ACK trùng đạt đến ngưỡng (tức bằng 3), và sau đó nó bằng số các ACK trùng. Do đó, trong thời gian "Phục hồi nhanh" thực thể gửi tăng một cách giả tạo cửa sổ của nó một giá trị đúng bằng số lượng ACK trùng mà nó đã nhận, theo sự quan sát mà mỗi một ACK trùng cho biết một gói nào đó đã rời khỏi mạng và được lưu giữ trong bộ nhớ đệm thực thể nhận. Sau khi vào quá trình "Phục hồi nhanh" và phát lại một gói dữ liệu (đơn), thực thể gửi đợi cho đến khi có hiệu lực một nửa kích thước cửa sổ của các ACK trùng đã nhận, và gửi một gói mới cho mỗi ACK trùng bổ sung được nhận. Vào lúc nhận được một ACK cho dữ liệu mới (được gọi là một sự "khôi phục ACK"), thực thể gửi ra khỏi quá trình "Phục hồi nhanh" với việc đặt ndup bằng 0.

Thuật toán "Phục hồi nhanh" của Reno được tối ưu hoá cho trường hợp khi một gói dữ liệu đơn bị mất từ một cửa sổ dữ liệu. Thực thể gửi phát lại nhiều nhất một gói dữ liệu bị mất cho mỗi RTT. Reno cải thiện một cách đáng kể việc xử lý của Tahoe khi một gói dữ liệu đơn bị mất, nhưng có thể có vấn đề về hiệu suất khi nhiều gói dữ liệu bị mất từ một cửa sổ dữ liệu. Vấn đề được xây dựng đơn giản trong bộ mô phỏng của chúng ta khi một kết nối Reno-TCP với một cửa sổ tắc

nghẽn lớn chịu đựng một sự tăng đột ngột số lượng của các gói dữ liệu bị mất sau quá trình "Khởi động chậm" trong một mạng với các cổng drop-tail (hoặc với các cổng khác thiếu sự giám sát độ lớn hàng đợi trung bình).

Thuật toán: Thuật toán điều khiển tắc nghẽn Reno-TCP là kết hợp 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 ACK trùng thực thể gửi 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 bị mất trong một cửa sổ dữ liệu (số gói mất > 2) Reno vẫn phải nhờ vào đồng hồ phát lại để phục hồi các gói dữ liệu bị mất đó.

Sơ đồ thuật toán:



Hình 10: Sơ đồ thuật toán Reno-TCP

2.3.3. TCP NewReno

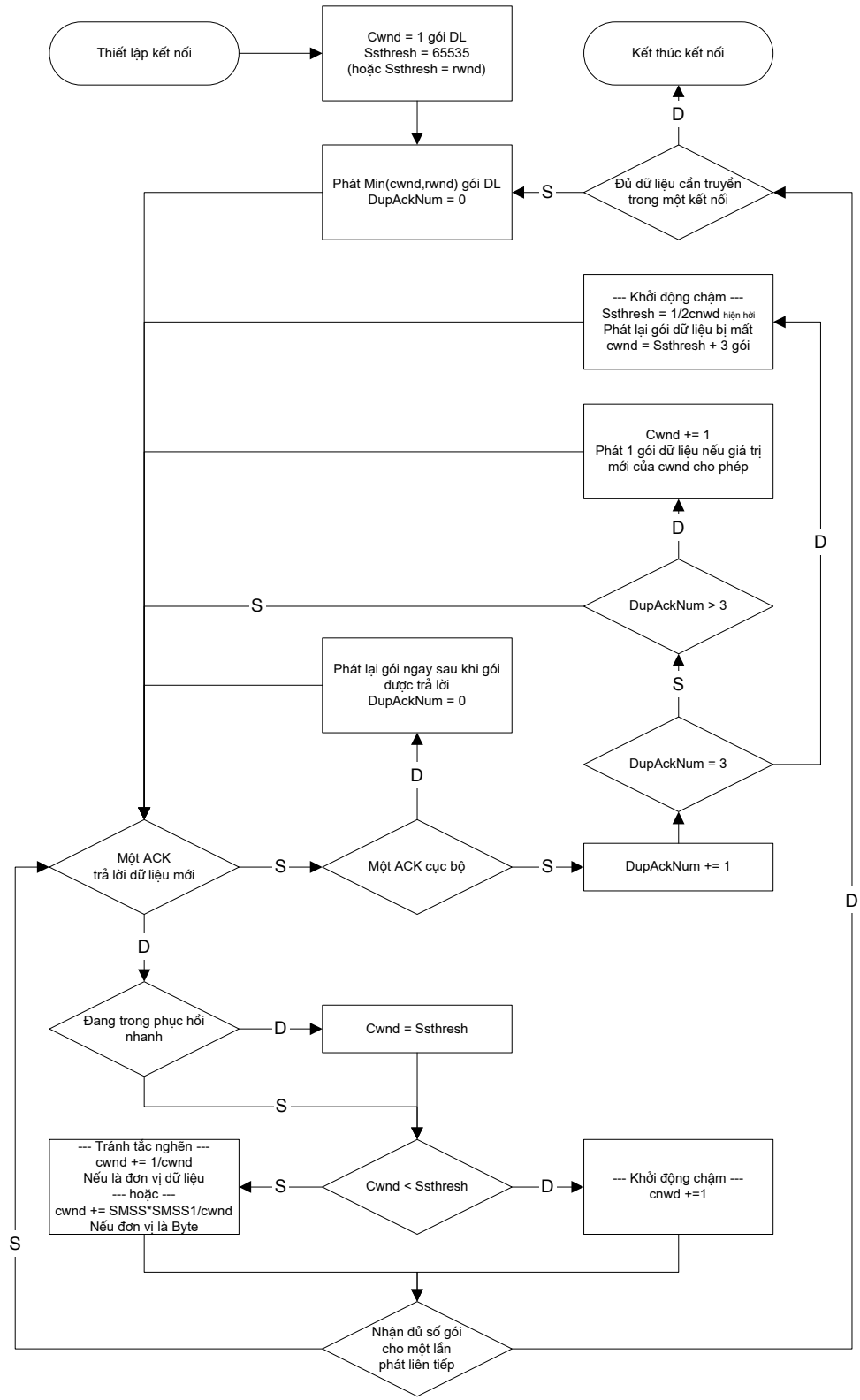
Chúng ta đưa New-Reno TCP vào trong chương này để cho thấy một sự thay đổi đơn giản đối với TCP để có thể tránh một số vấn đề hiệu suất của Reno-TCP khi không có SACK. Vào cùng thời gian đó, chúng ta sử dụng New-Reno TCP để khảo sát những nhược điểm cơ bản của hiệu suất TCP khi không có SACK.

New-Reno TCP có tính đến một sự thay đổi nhỏ đối với thuật toán Reno ở thực thể gửi là loại bỏ việc chờ của Reno thông qua một đồng hồ phát lại khi nhiều gói dữ liệu bị mất trong một cửa sổ dữ liệu [21, 22]. Sự thay đổi này liên quan đến hành vi của thực thể gửi trong thời gian "Phục hồi nhanh" khi một ACK cục-bộ được nhận mà nó trả lời một số (nhưng không phải tất cả) các gói dữ liệu còn tồn đọng tại thời điểm bắt đầu "Phục hồi nhanh". Trong New-Reno, các ACK cục-bộ không đưa TCP ra khỏi quá trình "Phục hồi nhanh". Thay vào đó, các ACK cục-bộ nhận được trong thời gian "Phục hồi nhanh" được xem như là một dấu hiệu chỉ báo rằng gói dữ liệu ngay tiếp sau gói dữ liệu được trả lời (xác nhận) trong khoảng trống tuần tự đã bị mất, và phải được truyền lại. Như vậy, khi nhiều gói dữ liệu bị mất từ một cửa sổ dữ liệu đơn, New-Reno có thể phục hồi không cần thời gian chờ phát lại, phát lại một gói dữ liệu đã mất cho mỗi RTT cho đến khi tất cả các gói dữ liệu bị mất đã được truyền lại. New-Reno giữ nguyên trong "Phục hồi nhanh" cho đến khi tất cả dữ liệu đang tồn đọng khi "Phục hồi nhanh" được bắt đầu báo đã nhận được.

Theo [4], đã đề nghị một thay đổi bổ sung đối với các thuật toán "Phục hồi nhanh" của TCP. Họ đề xuất thực thể gửi dữ liệu gửi một gói dữ liệu mới cho 2 ACK trùng nhận được trong thời gian "Phục hồi nhanh", để giữ "bánh đà" của ACK và tốc độ vận chuyển của các gói dữ liệu. Cái đó không được thực hiện trong "New-Reno" vì chúng ta muốn lưu ý đến một số thay đổi đối với Reno cần thiết để ngăn ngừa thời gian đợi phát lại không cần thiết.

Thuật toán: Thuật toán điều khiển tắc nghẽn New-Reno TCP là kết hợp bốn thuật toán cơ sở "Khởi động chậm", "Tránh tắc nghẽn", "Phát lại nhanh" và "Phục hồi nhanh" được mô tả ở trên. New-Reno có tính đến một sự thay đổi nhỏ đối với Reno chủ yếu là nằm trong hai thuật toán "Phát lại nhanh" và "Phục hồi nhanh" là loại bỏ việc chờ của Reno nhờ vào một đồng hồ phát lại khi nhiều gói dữ liệu bị mất trong một cửa sổ dữ liệu. Cũng như Reno, chiến lược phục hồi của New-Reno là có thể phát lại nhiều nhất một gói dữ liệu cho mỗi thời gian trễ toàn phần (RTT). Trong New-Reno có sử dụng ACK cục-bộ, là ACK mà nó trả lời một số nhưng không phải tất cả các gói dữ liệu đã được phát đi nhưng chưa nhận được trả lời tại thời điểm bắt đầu "Phục hồi nhanh". Khi nhận được ACK này có nghĩa gói dữ liệu ngay sau gói dữ liệu được trả lời bị mất và nó phải được truyền lại tức thì mà không cần phải chờ đồng hồ phát lại hết hạn (timeout xuất hiện).

Sau đây là sơ đồ thuật toán của New-Reno TCP:



Hình 11: Sơ đồ thuật toán Newreno-TCP

2.3.4. TCP Sack

Sự thực hiện SACK TCP trong chương này được thảo luận trong [19, 20]. Tuỳ chọn SACK theo định dạng được mô tả trong [18]. Theo [18], trường tuỳ chọn SACK chứa đựng một số các SACK-block, ở đó mỗi SACK-block báo cáo một tập không tiếp giáp nhau của dữ liệu đã

được nhận và xếp hàng đợi. Block đầu tiên trong một tuỳ chọn SACK được yêu cầu để báo cáo về gói dữ liệu được nhận mới nhất của thực thể nhận dữ liệu, và các SACK-block bổ sung nhắc lại các SACK-block được báo cáo gần đây nhất [18]. Trong các mô phỏng này mỗi tuỳ chọn SACK được giả thiết là có chỗ để cho 3 SACK-block.

Các thuật toán điều khiển tắc nghẽn đã thực hiện trong SACK-TCP của chúng ta là một sự mở rộng thận trọng điều khiển tắc nghẽn Reno-TCP, trong đó chúng sử dụng các thuật toán giống nhau để tăng hay giảm độ lớn cửa sổ tắc nghẽn, và làm những thay đổi tối thiểu đối với các thuật toán điều khiển tắc nghẽn khác. Bổ sung SACK cho TCP không làm thay đổi về cơ bản các thuật toán cơ sở. Sự thực hiện SACK-TCP bảo toàn tính chất của Tahoe và Reno TCP trong sự có mặt các gói dữ liệu không đúng thứ tự, và việc sử dụng thời gian chờ phát lại như là một phương sách cuối cùng để phục hồi dữ liệu. Sự khác nhau chủ yếu giữa sự thực hiện SACK-TCP và sự thực hiện Reno-TCP là ở trong hành vi khi nhiều gói dữ liệu bị mất trong một cửa sổ dữ liệu.

Như trong Reno, sự thực hiện SACK-TCP đi vào "Phục hồi nhanh" khi thực thể gửi dữ liệu nhận được số các ACK trùng liên tiếp bằng 3. Thực thể gửi phát lại một gói dữ liệu và giảm cửa sổ tắc nghẽn một nửa. Trong thời gian "Phục hồi nhanh", SACK duy trì một biến được gọi là pipe, biến này lưu giữ số lượng ước đoán các gói dữ liệu còn tồn đọng trên đường truyền (Điều này không giống cơ chế trong sự thực hiện Reno). Thực thể gửi chỉ gửi dữ liệu mới hay truyền lại dữ liệu khi số lượng ước tính các gói dữ liệu còn tồn đọng trên đường truyền nhỏ hơn cửa sổ tắc nghẽn. Biến pipe được tăng thêm 1 gói khi thực thể gửi phát 1 gói dữ liệu mới hoặc phát lại 1 gói dữ liệu cũ. Nó giảm bớt đi 1 gói khi thực thể gửi nhận được 1 gói ACK trùng với một tuỳ chọn SACK và báo rằng dữ liệu mới đã được nhận tại thực thể nhận.

Việc sử dụng biến pipe tách riêng sự quyết định "khi nào" thì gửi một gói dữ liệu với sự quyết định "gói dữ liệu nào" được gửi. Thực thể gửi duy trì một cấu trúc dữ liệu scoreboard (bảng kết quả) mà nó ghi lại những sự phản hồi từ các tuỳ chọn SACK trước đó. Khi thực thể gửi được phép gửi một gói dữ liệu, nó truyền lại gói dữ liệu tiếp theo từ danh sách các gói dữ liệu được phỏng đoán bị mất tại thực thể nhận. Nếu không có những gói như vậy và cửa sổ thông báo của thực thể nhận (rwnd) đủ lớn thì thực thể gửi phát 1 gói dữ liệu mới.

Nếu một gói dữ liệu được phát lại lại bị mất, sự thực hiện SACK phát hiện việc mất đó thông qua thời gian chờ phát lại (đồng hồ phát lại), khi đó nó phát lại gói dữ liệu bị mất đó và sau đó thực hiện quá trình "Khởi động chậm".

Thực thể gửi ra khỏi quá trình "Phục hồi nhanh" khi một ACK phục-hồi (recovery acknowledgement) được nhận và báo cho biết đã nhận tất cả dữ liệu mà đã tồn đọng khi đi vào "Phục hồi nhanh".

Thực thể gửi SACK có quá trình xử lý riêng biệt cho các ACK cục-bộ (các ACK nhận được trong thời gian "Phục hồi nhanh" mà nó đưa ra trường số ACK của tiêu đề TCP (TCP header), nhưng không bắt thực thể gửi ra khỏi quá trình "Phục hồi nhanh"). Để cho các ACK cục-bộ, thực thể gửi giảm pipe 2 gói dữ liệu đúng hơn là 1 gói, như sau. Khi "Phát lại nhanh" được bắt đầu thì pipe có hiệu lực giảm đi 1 cho gói dữ liệu mà nó được cho là đã bị mất, và sau đó tăng thêm 1 cho gói dữ liệu mà nó đã được phát lại. Như vậy, việc giảm pipe 2 gói dữ liệu khi ACK cục-bộ đầu tiên được nhận thì theo một nghĩa nào đó được cho là "gian lận", khi ACK cục bộ này chỉ đại diện cho một gói dữ liệu đã rời khỏi "pipe". Tuy nhiên, cho các ACK cục-bộ tiếp theo nào đó, pipe đã được tăng khi gói dữ liệu được phát lại đã được đưa vào "pipe", nhưng chưa bao giờ được giảm cho gói dữ liệu được giả thiết là đã mất. Vì vậy, khi ACK cục bộ tiếp theo đến nơi, trong thực tế nó đại diện cho 2 gói dữ liệu đã rời khỏi "pipe": gói nguyên bản gốc (giả thiết là đã mất), và gói đã được truyền lại. Vì thực thể gửi giảm pipe 2 gói dữ liệu đúng hơn là 1 gói để cho các ACK cục bộ, thực thể gửi SACK-TCP không bao giờ phục hồi chậm hơn một "Khởi động chậm".

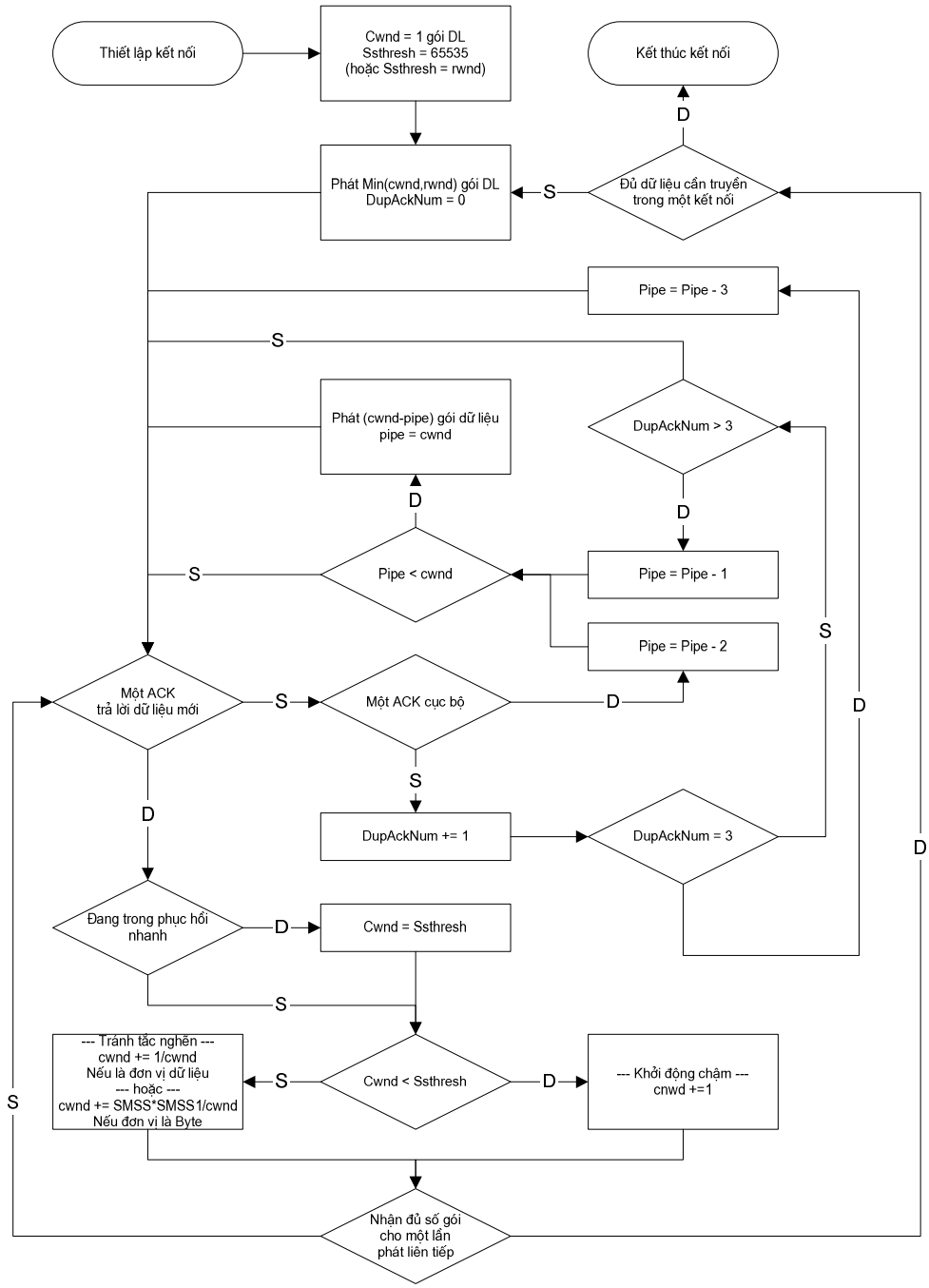
Có một số đề nghị khác cho các thuật toán điều khiển tắc nghẽn TCP có sử dụng SACK [23, 6]. Sự thực hiện SACK trong bộ mô phỏng của chúng ta được thiết kế cho sự mở rộng thận trọng nhất đối với các thuật toán điều khiển tắc nghẽn Reno, mà trong đó nó làm một sự thay đổi tối thiểu đối với các thuật toán điều khiển tắc nghẽn đang tồn tại của Reno.

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" được mô tả ở trên. SACK TCP có một sự cải tiến so với Reno và New-Reno là có sử dụng "Selective ACK" (ACK có lựa chọn) để thông tin kịp thời những gói dữ liệu bị mất thông qua SACK-Option (tuỳ chọn SACK) được chứa đựng trong nó.Và qua đó thực thể gửi có thể sử dụng chiến lược phát lại có lựa chọn để phục hồi khi nhiều gói dữ liệu bị mất.

Trong thời gian "Phục hồi nhanh", SACK duy trì một biến có tên là Pipe, biến này lưu giữ số lượng ước tính các gói dữ liệu còn tồn đọng trên đường truyền. Thực thể gửi chỉ gửi dữ liệu mới hay truyền lại dữ liệu khi số lượng ước tính các gói dữ liệu còn tồn đọng trên đường truyền nhỏ hơn cửa sổ tắc nghẽn. Biến pipe được tăng thêm 1 gói khi thực thể gửi phát 1 gói dữ liệu mới hoặc phát lại 1 gói dữ liệu cũ. Nó giảm bớt đi 1 gói khi thực thể gửi nhận được 1 gói ACK trùng với một

tuỳ chọn SACK và báo rằng dữ liệu mới đã được nhận tại thực thể nhận.

Sau đây là sơ đồ thuật toán của SACK-TCP:



Hình 12: Sơ đồ thuật toán Sack-TCP

3. Đánh giá hiệu suất

3.1. Mô phỏng máy tính

Để phân tích và đánh giá hiệu suất các thuật toán điều khiển lưu lượng và tắc nghẽn số liệu TCP, nói chung có nhiều cách đánh giá. Một trong những cách mà các trường đại học, các viện nghiên cứu cũng như một số tổ chức và cá nhân vẫn dùng để nghiên cứu là phương

pháp mô phỏng bằng một chương trình máy tính. Với một hệ chương trình mô phỏng, một tổ chức hay cá nhân không cần phải đầu tư nhiều lắm vào một hệ thống thiết bị tốn kém mà vẫn có khả năng trắc nghiệm và đánh giá hiệu quả công trình nghiên cứu của mình xem có khả thi với thực tế hay không. Trong bài luận văn này, chúng ta sẽ sử dụng một hệ mô phỏng mạng máy tính có tên NS, sẽ được giới thiệu ở phần sau. Với đặc tính của hệ thống vệ tinh GEO được mô phỏng trên vệ tinh cùng với các trạm làm việc đầu cuối các trạm mặt đất và các đặc tả đường truyền kết hợp với bốn thuật toán cơ sở điều khiển tắc nghẽn số liệu TCP được nghiên cứu và tìm hiểu Tahoe-TCP, Reno-TCP, NewReno-TCP, và SACK-TCP. Các thuật toán này được mô phỏng trên NS và hệ thống sẽ đưa ra các đồ thị mô phỏng tương ứng. Với các đồ thị của bốn thuật toán trên, chúng ta sẽ so sánh và đánh giá hiệu suất của chúng trên kênh vệ tinh GEO.

Trong việc xây dựng một hệ thống mạng thông tin máy tính đòi hỏi rất nhiều tài nguyên, cho nên không phải tổ chức hoặc các nhân nào cũng có thể xây dựng một hệ thống như thế để phục vụ cho việc nghiên cứu. Để tiếp cận với công việc nghiên cứu, đánh giá hiệu suất, tạo các giao thức phù hợp với thực tiễn,... Một trong những cách đơn giản nhưng không kém phần hiệu quả là dùng một phần mềm để mô phỏng hệ thống mạng TTMT. Việc tìm hiểu và ứng dụng một phần mềm mô phỏng hệ thống mạng là rất cần thiết và thực tế đối với việc nghiên cứu về mạng máy tính, là một công cụ hỗ trợ đắc lực trong giai đoạn phân tích và thiết kế hệ thống. Nó giúp chúng ta hiểu rõ và khảo sát được cơ sở lý thuyết, chứng minh được tính đúng đắn của các phương án giả thiết, những mô hình mới được đề ra, không cần chi phí tốn kém mà vẫn có thể tiến hành nghiên cứu một cách độc lập và hiệu quả. Việc xây dựng các chương trình mô phỏng đã được nhiều trường đại học và các tổ chức trên thế giới thực hiện. Hệ mô phỏng máy tính NS mà ta sử dụng trong phạm vi bài luận văn này là một công cụ tốt và hiệu quả để thực hiện trắc nghiệm và đánh giá một số thuật toán điều khiển tắc nghẽn số liệu TCP được đề cập ở phần trên.

3.1.1. Hệ thống mô phỏng NS

NS là hệ mô phỏng máy tính được xây dựng và phát triển bởi dự án VINT của phòng thí nghiệm quốc gia Lawrence Berkeley National Laboratory. NS là một hệ mô phỏng có cấu trúc hướng đối tượng. Nó được xây dựng trên hai ngôn ngữ C^{++} và Otcl và có thể được mở rộng bởi người sử dụng (người sử dụng có thể lập trình được trên nền của hệ mô phỏng NS). Với cách tiếp cận này thì ta có thể coi mỗi một mô phỏng như một chương trình hơn là các mô hình cứng nhắc, tĩnh, không thể thay đổi. Một mô phỏng gồm các đối tượng có thể cấu hình tuỳ ý để có thể đạt được mục đích mô phỏng đề ra.

Do vậy những người thiết kế NS đã không chọn một ngôn ngữ duy nhất để xây dựng môi trường mô phỏng, bởi có những đòi hỏi khác nhau về mục đích của việc mô phỏng. Khi cần mô phỏng những tầng thấp của mạng máy tính, xử lý ở mức độ byte hay tiêu đề (header) các gói tin thì cần đòi hỏi một hiệu quả tính toán cao, do đó nhân của hệ mô phỏng được viết bằng C^{++}. Còn phần định nghĩa, cấu hình và điều khiển mô phỏng được viết bằng ngôn ngữ script Tcl. Cách tiếp cận này mang lại hiệu quả rất cao vì nó chia mục đích của việc mô phỏng thành những phần nhỏ và giải quyết chúng bằng những ngôn ngữ phù hợp. Nó cung cấp cho người lập trình khả năng sử dụng dễ dàng hơn, mềm dẻo và linh hoạt hơn.

Vì C^{++} là ngôn ngữ hướng đối tượng mà Tcl thì không, nên người ta phải xây dựng những đối tượng macro bậc cao cho phù hợp với các lớp của C^{++}. Sau này để đạt được hệ thống mềm dẻo và mạnh hơn, trong phiên bản thứ hai, NS đã thay Tcl bằng Otcl (một Tcl hướng đối tượng). Và để có sự gắn kết giữa hai ngôn ngữ đó, người ta cần có một ngôn ngữ làm cầu nối là TclCl (ngôn ngữ mở rộng của Tcl). Với sự thay đổi đó thì các đối tượng macro đã được thay bằng các lớp Otcl và trở nên dễ sử dụng hơn nhiều.

NS 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 hỗ trợ các khía cạnh khác nhau trong mạng máy tính. Từ các lớp mô phỏng cơ sở như các Node, các liên kết tạo thành Topology của mạng, các gói tin (hay gói dữ liệu là một), 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. Rồi 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, nâng cao hiệu suất truyền tin. Trong NS người dùng có thể mô phỏng các mạng LAN, mạng không giây, thông tin vệ tinh,... rồi các ứng dụng như Ftp, Http, Webcache, Telnet, ...dựa trên các Agent của tầng vận chuyển, trên các mô phỏng về kỹ thuật chọn đường, các giao thức tầng vận chuyển...

Chi tiết về Hệ thống mô phỏng NS, tham khảo tài liệu [16].

3.1.2. Trực quan với NAM

NAM (Network Animator) là công cụ trực quan dựa trên Tcl/TK cho phép hiển thị trực tiếp và quan sát mô hình mô phỏng và quan sát các gói tin truyền qua trên mạng, quan sát trạng thái và các thông tin liên quan trong suốt thời gian thực hiện mô phỏng.

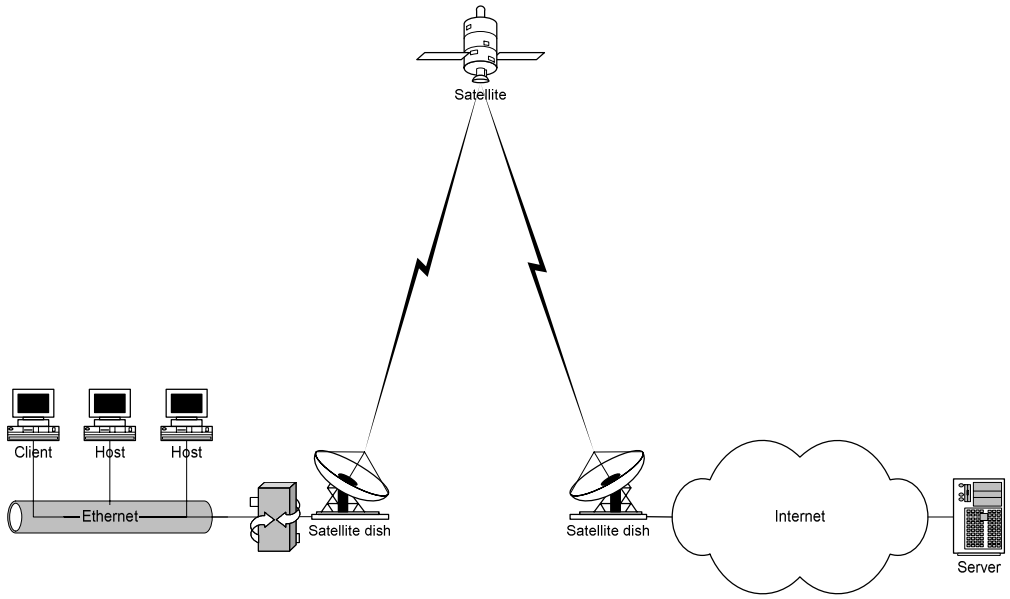
NAM được bắt đầu tại LBNL (Lawrence Berkeley National Laboratory), và đã được phát triển nhiều năm lại đây. Đến nay NAM đang được phát triển bởi ISI Mỹ (Information Sciences Institute).

NAM được thiết kế để cho phép trực quan với việc phân tích một khối lượng dữ liệu lớn được tạo bởi định vị các thông tin do NS thực hiện mô phỏng tạo ra. Với công cụ này, người thực hiện mô phỏng dễ dàng phân tích luồng luân chuyển dữ liệu qua lại cũng như các thông số được ghi lại bởi NS thay vì phải đọc và phân tích một khối lượng lớn các số liệu trong file dữ liệu định vị của NS. NAM có thể đọc và phân tích hầu hết các thông tin trong file định vị này như: các node, các kết nối, các thông tin quan sát như số tuần tự, độ rộng cửa sổ, kích thước gói tin, ... và cũng như định vị trạng thái gói tin khi truyền trong mô phỏng.

Phiên bản NAM sử dụng để phân tích trong luận văn này là phiên bản 1.0a11a với hai platfom khác trên Windows và Linux để dễ dàng quan sát và phân tích số liệu. NAM cũng là phần mềm miễn phí phục vụ công tác nghiên cứu. NAM được kết hợp với NS và thường được phân phối trong một gói cài đặt chung. Có thể tìm thấy tại địa chỉ http://www.isi.edu/nsnam/.

CHƯƠNG II: HIỆU SUẤT KẾT NỐI TCP ĐẦU CUỐI – ĐẦU CUỐI QUA KÊNH VIỄN THÔNG VỆ TINH

1. Mô hình mô phỏng



Hình 13: Mô hình mô phỏng

Mô hình mô phỏng trên được giả thuyết với một mạng LAN gồm một số máy trạm kết nối đến một máy chủ Internet. Trong môi trường truyền dẫn có kênh vệ tinh GEO:

Thông qua cổng kết nối được kết nối với trạm mặt đất, mạng LAN được kết nối với kênh vệ tinh GEO. Cổng kết nối này được triển khai như một trạm lặp. Tuy nhiên với từng bài thực hiện mô phỏng khác nhau, trạm lặp này được cải tiến nhằm cải thiện hiệu suất. Trong mạng LAN, tại mỗi thời điểm chỉ có duy nhất 01 trạm truyền số liệu thông qua kênh vệ tinh để thấy được hiệu suất làm việc của kênh vệ tinh. Trong thực nghiệm cũng loại bỏ các ảnh hưởng của mạng LAN đến hiệu suất làm việc của kênh vệ tinh.

Kênh vệ tinh GEO được triển khai trong với độ rộng băng thông có thể thay đổi và có giới hạn 2048Kbps. Tuy nhiên, độ trễ của liên kết là không thay đổi với các bài thực nghiệm với giá trị RTT = 1000ms.

Để xác định được hiệu suất làm việc của các bài thực nghiệm, mô phỏng sử dụng giao thức truyền số liệu FTP làm việc trên các giao thức tầng giao vận TCP và TCP mở rộng.

Mô hình mô phỏng này được áp dụng cho tất cả các thực nghiệm trong bài luận văn này. Cụ thể việc tuỳ biến các tham số cho thực nghiệm sẽ được đề cập trước mỗi thực nghiệm.

2. Kết nối TCP đầu cuối – đầu cuối

Kết nối TCP đầu cuối – đầu cuối là hình thức kết nối TCP truyền thống mà trong đó, tại các trạm đầu cuối giao thức TCP bắt tay và làm việc với nhau trực tiếp: Thông tin từ trạm phát thông qua môi trường mạng đến trực tiếp trạm thu và thông tin trả lời trực tiếp từ trạm thu qua môi trường mạng đến trạm phát.

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

3. Ảnh hưởng đến hiệu suất của kết nối đầu cuối – đầu cuối

3.1. Thực nghiệm

Nhằm mục đích khám phá mối quan hệ giữa kích thước cửa sổ, dải thông của mạng, sự trễ trong hiệu suất kết nối đầu cuối - đầu cuối, chương trình mô phỏng được chạy trên hệ NS2 nhằm giả lập môi trường liên kết vệ tinh như đã trình bày phần trên.

Trong thực nghiệm này: Để xác định mối quan hệ giữa cửa sổ, dải thông và kích thước gói tin đến hiệu suất của kênh vệ tinh, thực nghiệm sử dụng giao thức TCP chuẩn. Trên tầng TCP, giao thức truyền tải File FTP được gắn vào để thực hiện truyền tải file.

Kịch bản thử nghiệm 1: Với khối lượng dữ liệu truyền là không đổi cho các lần thí nghiệm là 1Mbyte, thí nghiệm được tiến hành với việc lần lượt thay đổi:

- Kích thước gói tin truyền trên mạng: 512 byte, 1024 byte và 2048 byte.
- Tốc độ kênh truyền (kết nối giữa Vệ tinh và trạm mặt đất): 128Kbps, 256Kbps, 512Kbps, 1024Kbps và 204 Kbps.
- Kích thước cửa sổ: 2Kbyte, 4Kbyte, 8Kbyte, 16Kbyte, 32Kbyte.

Kết quả thu được:

	Win size / Pck Size	2Kbyte	4Kbyte	8Kbyte	16Kbyte	32Kbyte
128 Kbps	512	326.591	211.235	178.220	177.528	177.528
	1024	310.043	177.389	118.305	111.084	110.735
	2048	333.024	174.990	105.750	87.190	87.515
256 Kbps	512	314.877	211.235	170.210	169.502	169.502
	1024	291.824	166.103	109.378	95.911	95.515
	2048	299.500	156.548	93.723	66.987	62.689
512 Kbps	512	308.512	197.988	166.155	165.447	165.447
	1024	282.165	160.260	104.964	91.895	91.488
	2048	282.436	147.577	87.910	61.989	56.824
1024 Kbps	512	305.763	195.750	164.025	163.518	163.518
	1024	227.442	157.506	102.810	89.735	89.229
	2048	277.442	157.506	102.810	89.735	89.229
2048 Kbps	512	304.363	194.628	163.112	162.405	162.405
	1024	274.674	155.962	101.680	88.707	88.201
	2048	269.014	140.526	83.385	58.171	53.156

Bảng 2: Quan hệ giữa thời gian truyền với kích thước cửa sổ và gói tin

3.2. Đánh giá

Độ rộng kênh truyền số liệu:

Lần lượt sử dụng một vài tốc độ truyền số liệu: 128Kbps, 256Kbps, 512Kbps, 1Mbps và 2Mbps để kiểm chứng có đúng tốc độ truyền cao hơn sẽ cải thiện hiệu suất TCP/IP thông qua liên kết vệ tinh. Thí nghiệm được thực hiện trên cùng một cơ sở với việc dữ liệu có thể được truyền qua một dải thông lớn hơn.

Sử dụng thời gian nhanh nhất từ bảng kết quả; Cho thấy tăng 28% lần khi thực hiện nâng dải thông lên 256Kbps từ dải thông 128Kbps.. So sánh kết quả giữa 128Kbps và 256Kbps, cho thấy tăng dải thông lên 9% lần. Tương tự 4% lần khi nâng từ 512Kbps lên 1Mbps và 2% lần khi nâng từ 1Mbps lên 2Mbps.

Rõ ràng khi thực hiện tăng độ rộng kênh truyền cũng có ảnh hưởng đến hiệu suất, tuy nhiên tỷ lệ hiệu suất được cải thiện là rất nhỏ, không phù hợp với việc tăng độ rộng kênh truyền.

Độ rộng của cửa sổ:

Khi thực hiện thay đổi với kích thước cửa sổ lớn hơn, trạm thu dữ liệu có khả năng nhận nhiều dữ liệu hơn trước gửi thông tin

phản hồi trở lại trạm phát. Cửa sổ lớn hơn trên thực tế làm tăng cơ hội xảy ra sự quá ngưỡng thời gian chờ (timeout) của trạm thu. Nếu sau thời gian "timeout" và gói tin không đến đích, trạm thu có thể yêu cầu trạm phát gửi lại dữ liệu, theo cách đó sự tắc nghẽn sẽ xảy ra. So sánh bảng số liệu trên, có sự khác nhau về thời gian giữa kích thước cửa sổ là 2Kbyte và 16Kbyte. Trong bảng thứ nhất, sự cải thiện khoảng 74% với việc tăng kích thước cửa sổ từ 2Kbyte lên 16Kbyte. Tuy nhiên, so sánh việc tăng kích thước cửa sổ 16Kbyte lên 32Kbyte, không có sự cải thiện đáng kể về thời gian. Lý do là kênh truyền đã đạt đến ngưỡng bão hoà. Cũng sử dụng kích thước cửa sổ như trên với các kênh truyền tốc độ cao hơn, sự tăng về thời gian lần lượt là 77% (Kênh tốc độ 256Kbps), 78% (kênh 512Kbps). Sự tăng thời gian lớn nhất là kênh 2Mbps đồng bộ với mức tăng khoảng 79%.

Kích thước gói tin:

Trong việc biến đổi kích thước gói tin, hy vọng đạt được tốc độ truyền dữ liệu cao hơn bởi có thể gửi nhiều byte dữ liệu hơn trong mỗi gói dữ liệu. Dù bằng cách nào, hiệu quả cũng không đạt được, nếu chất lượng đường kết nối vệ tinh tồi, sẽ có nhiều lỗi xảy ra, vì thế sẽ làm giảm hiệu suất nếu triển khai gói tin lớn. So sánh kích thước các gói tin đã thực hiện trong bài thí nghiệm, ta thấy rằng gói tin có kích thước 1024 Byte là tối ưu nhất khi tôi sử dụng với kích thước cửa sổ 32Kbps.

Trong thí nghiệm này, có thể tổng kết: Với các kích thước cửa sổ khác nhau, kích thước gói dữ liệu khác nhau và tốc độ kênh truyền khác nhau, chúng ta có thể đạt được cải thiện được hiệu suất.

4. Hiệu suất các biến thể của TCP với kết nối đầu cuối đầu cuối

4.1. Thực nghiệm

Tiếp tục sử dụng chương trình NS2 để thực hiện việc mô phỏng các thí nghiệm với các mô hình và thay đổi các thiết lập cho phù hợp.

Chương trình mô phỏng được cài đặt trên hệ điều hành RedHat Linux phiên bản 8.0 trong đó phiên bản NS2 (2.1b9) được dịch trên bộ dịch GCC3.2.

Máy tính chạy chương trình mô phỏng là máy tính Dell Optiplex GX240 có cấu hình P4-1.6GHz, 256MB RAM, 20GB HDD.

Mô hình mô phỏng được thiết lập như hình 13, trong đó mô phỏng một mạng LAN kết nối với máy chủ Internet qua hệ thống vệ tinh:

- Mạng LAN trong mô phỏng này là mạng Ethernet có tốc độ 10Mbps và có độ trễ 2ms. Trong mô hình thí nghiệm, giả thuyết

mạng LAN làm việc không có lỗi, tức là trong một thời điểm chỉ có 1 trạm truyền dữ liệu trên mạng và truyền song công, vì vậy loại bỏ các khả năng tác động đến hiệu suất truyền trên kênh vệ tinh.

- Từ mạng LAN kết nối với một Gateway để liên kết với trạm mặt đất. Trạm này trong bài mô phỏng coi như trạm lặp (Repeater) kết nối với trạm mặt đất. Trạm lặp này kết nối trực tiếp với trạm mặt đất với tốc độ 100Mbps và có độ trễ 0ms. Với thiết lập này, giả thuyết xây dựng cổng kết nối với vệ tinh. Ta cũng giả thuyết rằng không có lỗi khi truyền giữa kết nối trạm lặp và trạm mặt đất.
- Để mô phỏng hệ thống vệ tinh GEO liên kết đối xứng, song công, trong đó độ trễ kết nối từ trạm mặt đất lên Vệ tinh là 250ms. Giả thiết rằng tại vệ tinh, tốc độ chuyển tiếp là tức thì, như vậy tổng thời gian từ kết nối trạm mặt đất bên phát truyền đến trạm mặt đất bên thu là 500ms (đây là độ trễ thông thường của các hệ thống vệ tinh GEO). Hệ thống mô phỏng liên kết vệ tinh này có tốc độ 2Mbps. Trên kênh truyền Vệ tinh có thể thay đổi tham số về khả năng hư hỏng và mất gói dữ liệu để có thể quan sát được cách ứng xử của các giao thức điều kiển giao vận (TCP) và hiệu suất của chúng được phân tích sau này.
- Từ trạm mặt đất bên thu kết nối đến trạm thu với tốc độ 10 Mbps, độ trễ 2ms, không có lỗi khi kết nối. Với kết nối này có thể là kết nối Internet qua cáp quang hoặc cáp đồng có tốc độ cao. Tương tự như trong LAN, giả thuyết ở đây không phát sinh các lỗi truyền gói tin vì muốn loại bỏ ảnh hưởng của các kết nối không phải kết nối vệ tinh.
- Trong mạng LAN, chúng ta thực hiện triển khai trạm phát. Trạm phát được cài đặt làm việc với giao thức TCP và có thể triển khai các biến thể của giao thức TCP như Tahoe, Reno, NewReno, Sack...
- Trạm thu thực hiện triển khai giao thức TCP và có khả năng đáp ứng với TCP thông thường hoặc SACK.
- Để tạo ra lưu lượng lưu thông trên mạng, ta sử dụng giao thức FTP (giao thức truyền file truyền thống trên Internet). Giao thức FTP được đặt trên tầng TCP của trạm phát và có thể truyền số liệu liên tục trong suốt thời gian thực nghiệm.
- Trong mô phỏng thí nghiệm, thiết lập độ rộng cửa sổ có kích thước cửa sổ phát 32Kbyte và kích thước mỗi gói tin là 1024Byte là không đổi trong tất cả các thí nghiệm sau này để dễ dàng quan sát ứng xử của cửa sổ chống tắc nghẽn và phương

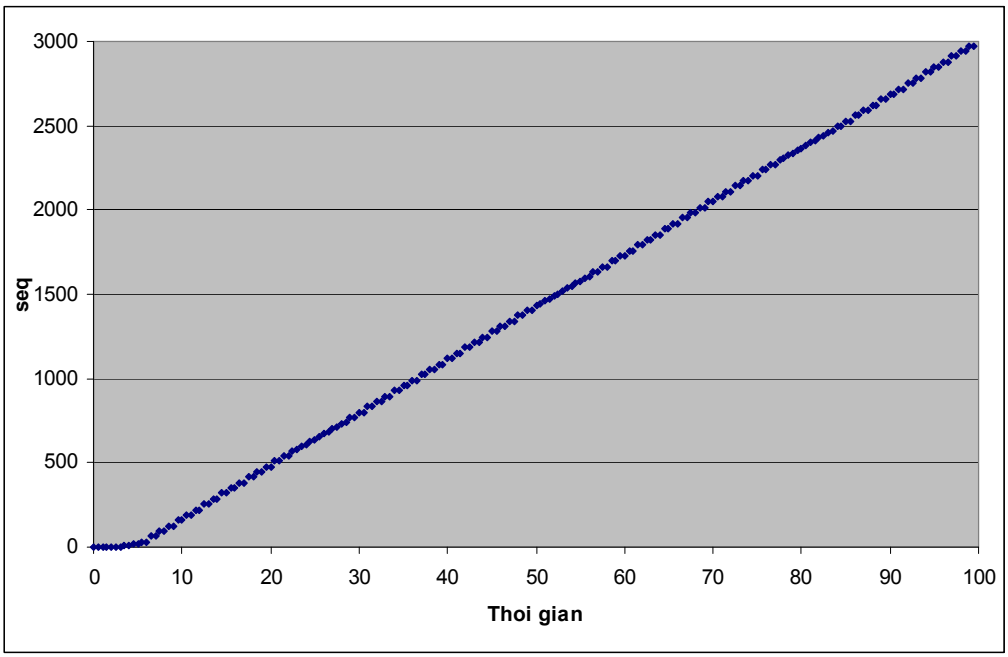
pháp làm việc của các thuật toán TCP và các biện pháp nâng cao hiệu suất.

- Tại tầng TCP của trạm phát, thiết lập hệ thống quan sát quá trình làm việc của TCP trong đó chúng ta thực hiện quan sát và ghi lại các giá trị số tuần tự (Seq) và cửa sổ chống tắc nghẽn (cwnd). Chu kỳ ghi là 0.5s. Từ các tham số ghi được có thể tính toán để xác định được đặc tính của đường truyền.

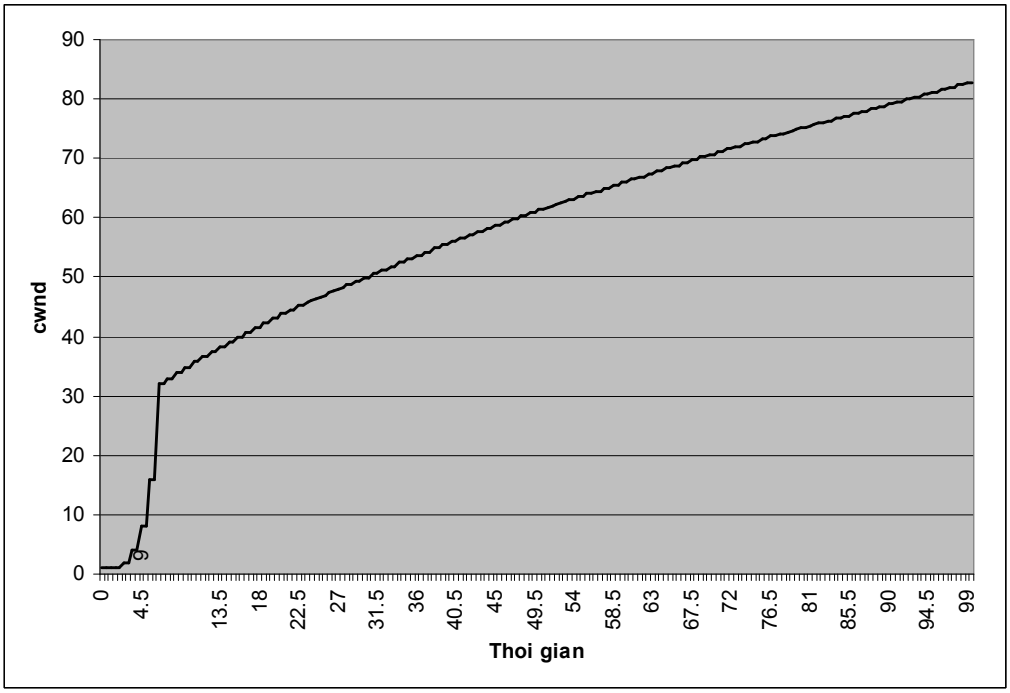
Như vậy, trong mô hình mô phỏng này đã thiết lập một hệ thống truyền thông vệ tinh với một số yếu tố có thể thay đổi đáng lưu tâm để có thể quan sát được hiệu suất truyền trên hệ thống thông tin vệ tinh đó là: Giao thức TCP với các phiên bản mở rộng: Tahoe, Reno, NewReno, Sack. Có thể thay đổi khả năng gây mất mát thông tin của kênh Vệ tinh để thấy được hiệu suất làm việc của các biên thể TCP.

a. Kịch bản thử nghiệm 2: Với các biến thể Tahoe, Reno, NewReno, Sack của TCP. Không có sự mất mát liên kết kênh vệ tinh:

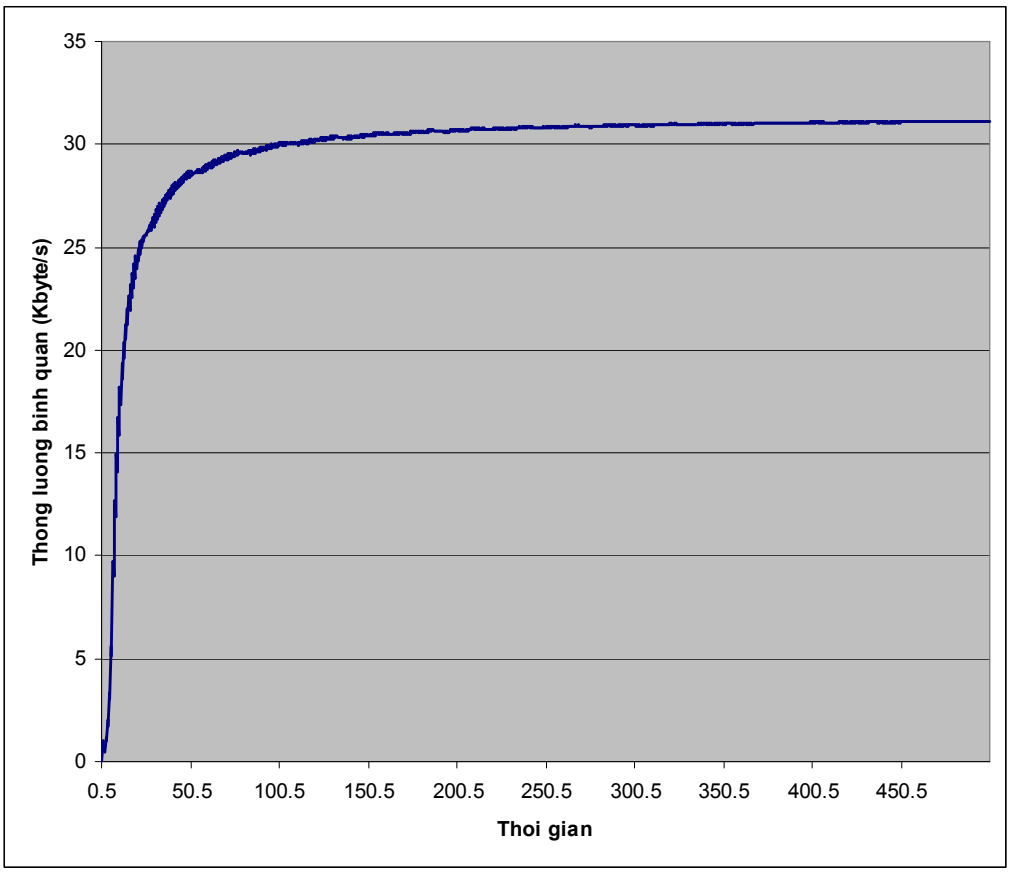
   Thực hiện với thời gian 100s và ghi lại các thông tin. Kết quả thu được vẽ trên đồ thị:



Hình 14: Biểu đồ so sánh số tuần tự của Tahoe, Reno, Newreno, Sack khi không có sự mất mát thông tin trên kết nối E2E

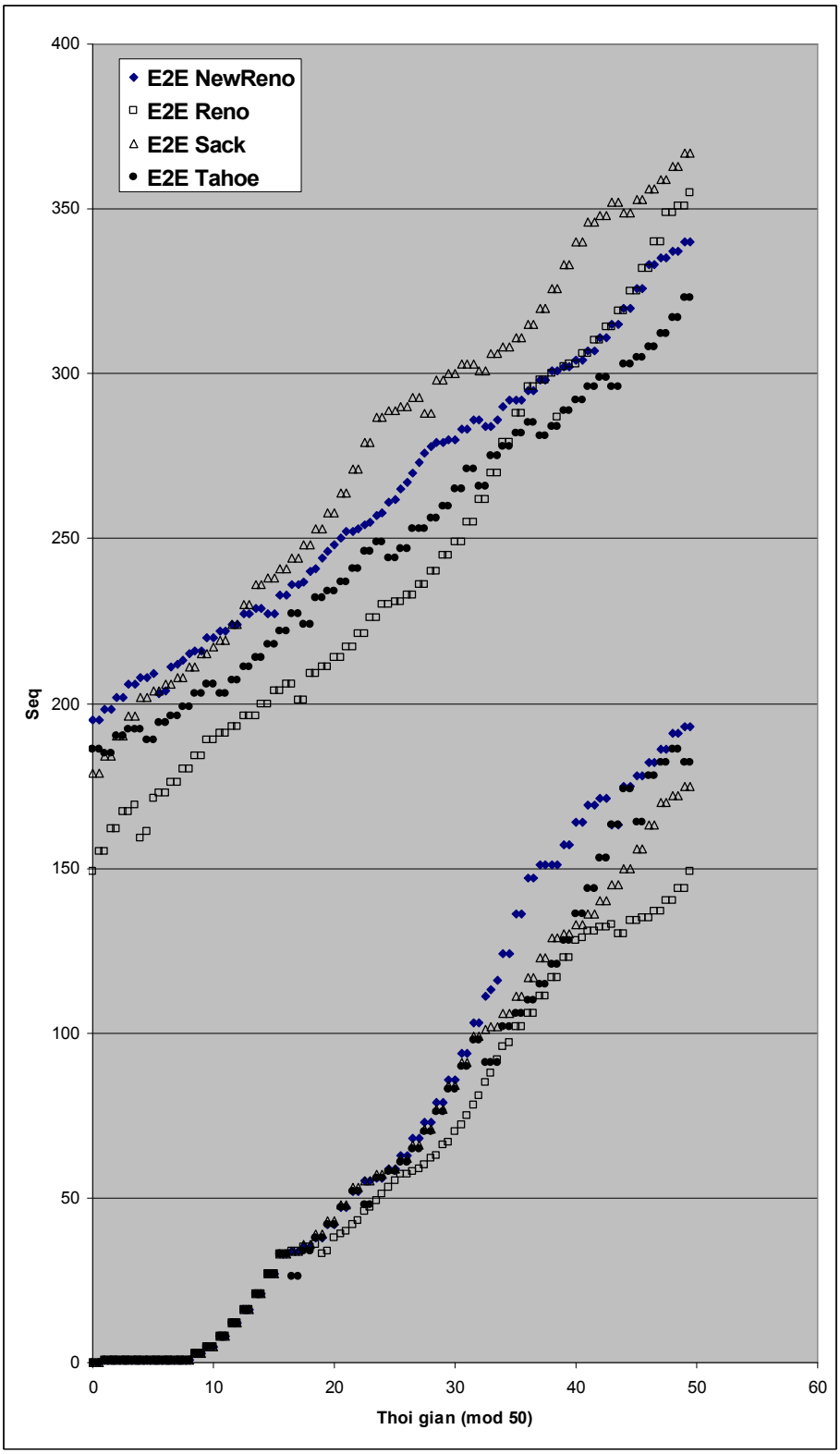


Hình 15: Biểu đồ so sánh sự thay đổi cửa sổ tắc nghẽn của Tahoe, Reno, Newreno, Sack khi không có sự mất mát thông tin trên kết nối E2E

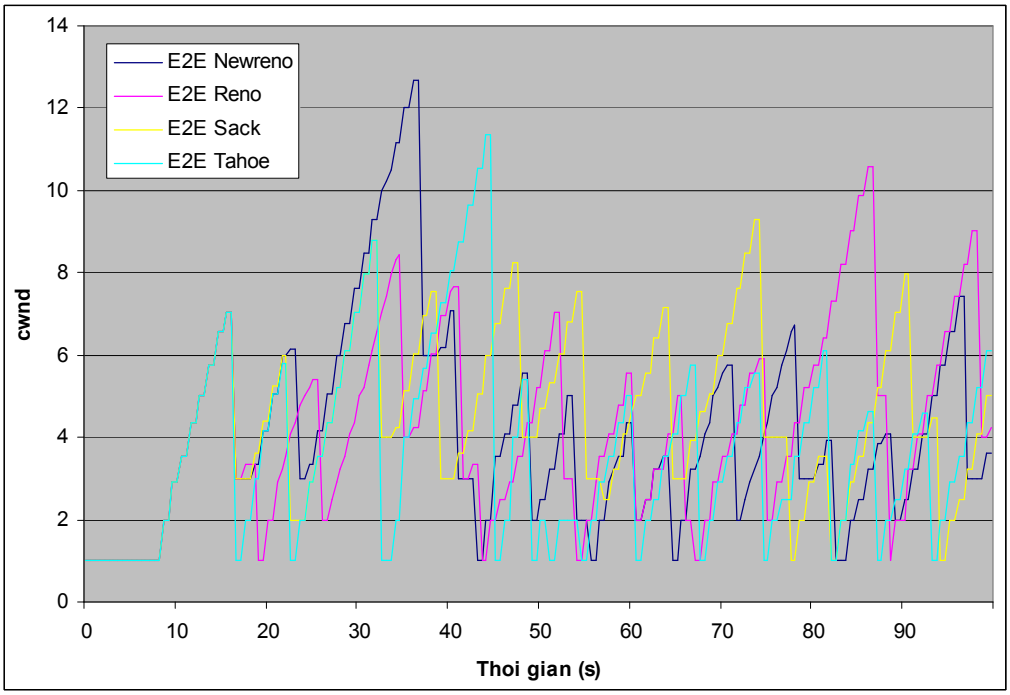


Hình 16: Thông lượng bình quân của Tahoe, Reno, Newreno, Sack khi không có sự mất mát thông tin trên kết nối E2E

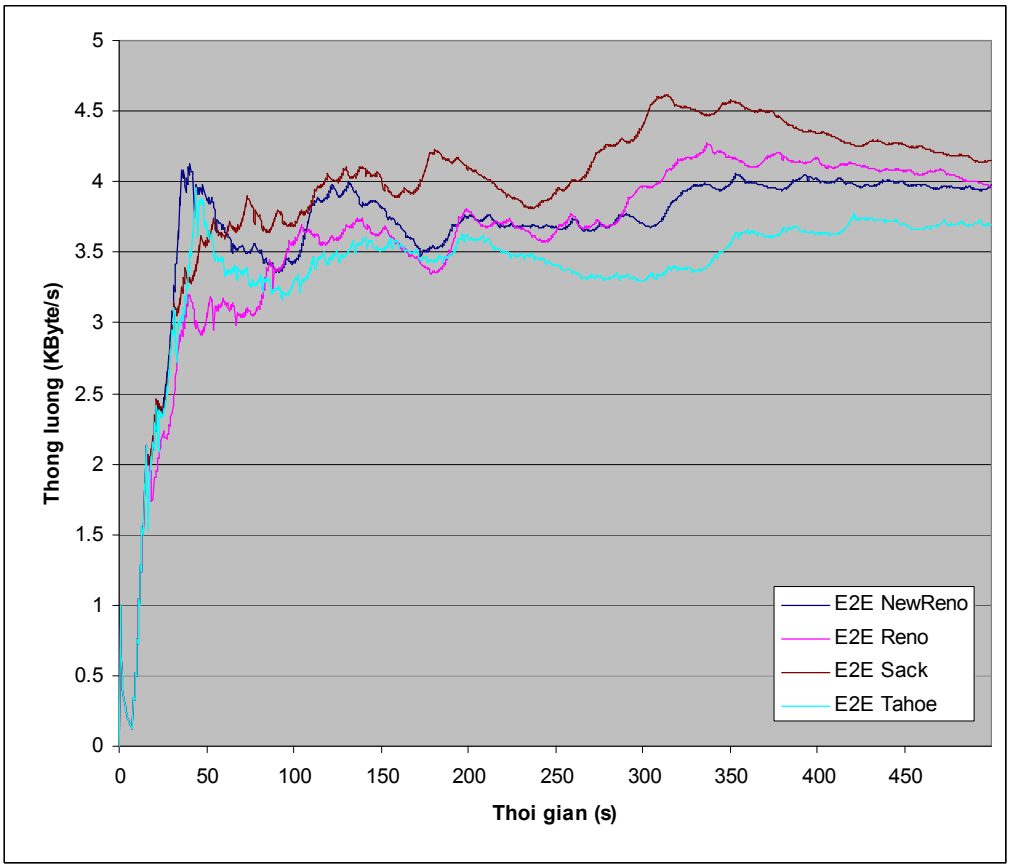
b. Kịch bản thử nghiệm 3: Với các biến thể khác Tahoe, Reno, NewReno, Sack. Sự mất mát trên mỗi chặng liên kết vệ tinh là 2% các gói tin truyền qua.



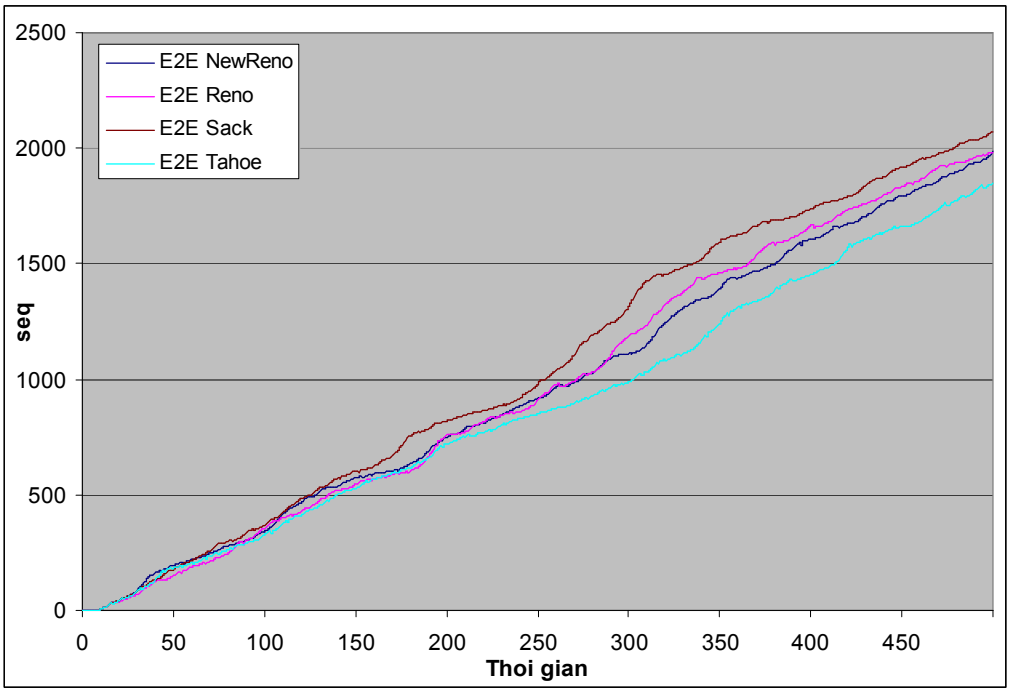
Hình 17: Biểu đồ seq của Tahoe, Reno, Newreno, Sack khi có sự mất mát thông tin trên kết nối E2E



Hình 18: Biểu đồ so sánh sự thay đổi cửa sổ tắc nghẽn của Tahoe, Reno, Newreno, Sack khi có sự mất mát thông tin trên kết nối E2E



Hình 19: Biểu đồ so sánh thông lượng bình quân của Tahoe, Reno, Newreno, Sack khi có sự mất mát thông tin trên kết nối E2E



Hình 20: Biểu đồ so sánh số tuần tự của Tahoe, Reno, Newreno, Sack khi có sự mất mát thông tin trên kết nối E2E

4.2. Đánh giá

Trong kịch bản thử nghiệm 2:

Với biểu đồ hình 14,15,16 cho ta thấy, khi trên kênh vệ tinh không có sự mất mát thông tin (coi như điều kiện lý tưởng) thì các thuật toán mở rộng TCP bao gồm Tahoe, Reno, NewReno và Sack có ứng xử giống nhau kể cả về số gói phát, hiệu suất và sự điều chỉnh, thay đổi cửa sổ tắc nghẽn.

Ta thấy cửa sổ tắc nghẽn là một đường đồng biến chứng tỏ không có sự xung đột va chạm trong quá trình truyền số liệu.

Tuy nhiên, dễ dàng quan sát trong 6-7 giây đầu tiên của biểu đồ hình 14 cho thấy độ dốc đồng biến thấp: điều này được giải thích bằng nguyên lý của thuật toán bắt đầu chậm. Cũng có thể quan sát thuật toán bắt đầu chậm với đồ thị cửa sổ chống tắc nghẽn tăng nhanh (hình 15) (với hàm số mũ) cho đến khi cwnd = ssthresh (đạt đến trong khoảng 6-7 giây khi thuật toán bắt đầu chậm được thực hiện). Lúc này TCP đã phát huy tối đa thông lượng có thể.

Thuật toán bắt đầu chậm mất đến gần 7 giây mới đạt được thông lượng bão hoà, đây cũng là một điểm yếu của TCP khi làm việc với kênh truyền có độ trễ lớn như kênh viễn thông vệ tinh. Biện pháp để cải thiện vấn đề này cũng được các nhà nghiên cứu đưa ra bằng biện pháp cải tiến với việc áp dụng một số thuật toán thay đổi cwnd không phải là bắt đầu chậm mà thực hiện bằng bước nhảy (không đề cập đến trong bài luận văn này song có thể tham khảo RFC793edu-TCP), hoặc

với tiếp cận khác như buffing làm cho thuật toán bắt đầu chậm tăng tốc nhanh hơn được mô tả trong phần sau.

Khi cwnd = ssthresh, thuật toán tránh tắc nghẽn hoạt động. Với mỗi thông báo Ack nhận được chỉ tăng kích thước cwnd = cwnd + 1/cwnd. Vì vậy đường biểu thị là hàm hypebol (hình 15). Và vì lúc này TCP đã đạt đến thông lượng cực đại, vì vậy đồ thị thể hiện số tuần tự là một đường đồng biến tuyến tính.

Trong kịch bản thử nghiệm này cho ta xác định được:

- Thông lượng tối đa với giao thức điều khiển giao vận mở rộng Tahoe-TCP, Reno-TCP, NewReno-TCP, Sack-TCP là như nhau cực đại là 31 Kbyte/s ~ 248Kbps.
- Thông lượng bình quân khi không có sự mất mát thông tin sẽ là tiệm cận đến điểm bão hoà chính là năng lực tối đa mà giao thức TCP có thể đạt được khi truyền trên kênh truyền có độ trễ lớn như hệ thống vệ tinh GEO.
- Thông lượng này thấp hơn so với dải thông chính là do độ trễ lớn của kênh vệ tinh. Trong mô hình giả lập này có độ trễ toàn phần RTT = 1008ms (chưa kể đến độ trễ tại các node). Để đạt được hiệu suất tối đa tốc độ kênh truyền là Throughput = Bandwith = 2048Kbps thì độ rộng cửa sổ tắc nghẽn theo lý thuyết cwnd = (Throughput * RTT)/1000 = 2084.384 Kbit ~ 258Kbyte. Kích thước cửa sổ cần thiết như vậy lớn hơn rất nhiều với cửa sổ tối đa thực tế của các thuật toán trong thí nghiệm và trên thực tiễn (64Kbyte/32Kbyte).
- Thuật toán Tahoe-TCP, Reno-TCP, NewReno-TCP, Sack-TCP có hiệu suất như nhau khi không có sự tắc nghẽn (hay thực chất ở đây đường kết nối bảo đảm, không bị mất thông tin trên đường truyền và thông lượng tối đa đạt được nhỏ hơn độ rộng của kênh kết nối).
- Thử nghiệm cũng cho ta thấy sự hiện diện của thuật toán bắt đầu chậm và chống tắc nghẽn. Và khi không có tắc nghẽn phải mất đến hơn 6 giây, TCP mới đạt được thông lượng cao nhất có thể truyền qua.

Trong kịch bản thử nghiệm 3:

Với sự mất mát 2% trên kênh truyền thực chất là một tỷ lệ lỗi rất lớn so với các kết nối thông thường. Sở dĩ đưa ra tỷ lệ lỗi lớn như vậy để có thể dễ dàng nhận thấy được sự khác biệt đáng kể trong hiệu suất của các thuật toán.

Đồng thời với tỷ lệ lỗi cao cũng phản ánh được lỗi của hệ thống liên kết vệ tinh trên thực tiễn với khả năng chịu ảnh hưởng rất lớn đến thời

tiết và môi trường. Trong điều kiện thời tiết xấu, tỷ lệ lỗi thậm chí lớn hơn rất nhiều so với tỷ lệ lỗi đưa ra trong bài thí nghiệm.

Rõ ràng ta thấy, khi không mất mát thông tin thì các giao thức có hành xử giống nhau và hiệu suất làm việc giống nhau. Nhưng các gói tin mất mát trên kênh truyền đã làm thay đổi các hành xử của các thuật toán điều khiển tầng giao vận đã đề cập đến phần trước của bài luận văn. Trong đó:

Quan sát biểu đồ hình 18 cho ta thấy cùng một quá trình gặp lỗi gói tin như nhau nhưng hành xử của việc thay đổi cửa sổ tắc nghẽn là khác nhau. Quan sát 100 giây đầu tiên cho ta thấy cửa sổ chống tắc nghẽn đạt đỉnh cao rất thấp và là một đường răng cưa trong đó sườn trái của đồ thị là đường đồng biến tăng và có độ dốc như nhau của các thuật toán Tahoe-TCP, Reno-TCP, Newreno-TCP và Sack-TCP. Sườn phải của các thuật toán có độ dốc thẳng đứng, chính là thời điểm tắc nghẽn xảy ra và khi đó giá trị cwnd đặt bằng 1, kể từ đó thuật toán bắt đầu chậm được thực hiện. Tuy nhiên, đỉnh của các đồ thị thể hiện trạng thái cửa sổ chống tắc nghẽn và thời điểm xảy ra tắc nghẽn khác nhau chính là do hành sử khác nhau của các thuật toán điều khiển tầng giao vận bởi áp dụng các thuật toán "khôi phục nhanh" và "phát lại nhanh" cũng như thuật toán "chống tắc nghẽn".

Quan sát đồ thị hình 17 cho ta thấy các thuật toán số tuần tự không tăng liên tục mà nhiều chỗ số tuần tự có điểm đột biến với số tuần tự thấp. Chính các điểm đó thể hiện các gói tin bị mất và phải thực hiện phát lại.

Sự khác nhau thể hiện bởi đồ thị đầy đủ về số tuần tự (Hình 20) trong toàn bộ 500 giây thí nghiệm và thu thập số liệu cũng như đồ thị so sánh hiệu suất làm việc (hình 19) của các giao thức tầng giao vận Tahoe-TCP, Reno-TCP, Newreno-TCP và Sack-TCP. Rõ ràng kết quả cho ta thấy hiệu suất của các thuật toán là khác nhau. Tuy nhiên hiệu suất giữa các giao thức điều khiển tầng giao vận chênh lệch cũng không lớn. Nếu lấy Tahoe-TCP làm cơ sở so sánh ta có: Reno-TCP có hiệu suất cao hơn 7%, Newreno-TCP cao hơn 8% và Sack-TCP có sự cải thiện cao hơn cả là 12%.

Tuy nhiên đồ thị hình 19 cho ta thấy, kênh vệ tinh GEO có độ trễ ~500ms và tỷ lệ lỗi 2% gói tin có hiệu suất làm việc rất thấp, các giao thức điều khiển giao vận có thông lượng trong khoảng 32Kbps, một tốc độ rất thấp chỉ bằng với kết nối Modem thông thường. Hiệu suất này cho ta thấy yếu điểm của các giao thức điều khiển tầng giao vận với kết nối đầu cuối - đầu cuối trên kênh vệ tinh GEO mặc dù giải thông của kênh vệ tinh rất lớn (2048Kbps). Giải quyết vấn đề này, một số biện pháp nâng cao thông lượng đề cập trong phần sau.

	Tahoe-TCP	Reno-TCP	Newreno-TCP	Sack-TCP
Số gói tin truyền sau 500 giây	1846	1983	1985	2072
Tỷ lệ so sánh với Tahoe-TCP		107%	108%	112%

Bảng 3: So sánh hiệu suất làm việc các giao thức điều kiển giao vận trên kênh vệ tinh GEO kết nối đầu cuối – đầu cuối

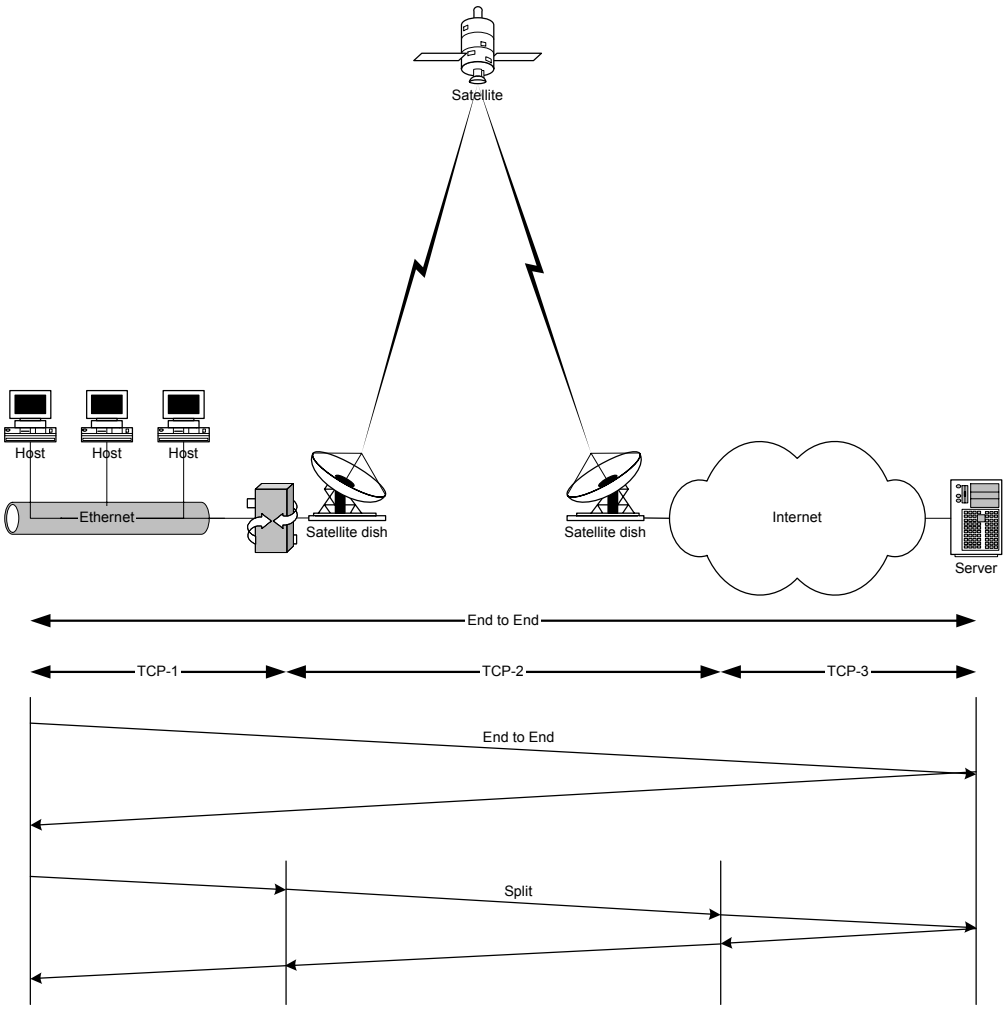
CHƯƠNG III: HIỆU SUẤT KẾT NỐI TCP CHIA CẮT QUA KÊNH VIỄN THÔNG VỆ TINH

1. Giới thiệu chung

1.1. Chia cắt kết nối TCP

Ý tưởng của việc chia cắt kết nối TCP (còn được biết đến với TCP gián tiếp), là chia kết nối đầu cuối - đầu cuối thành hai hoặc ba đoạn. từng đoạn tự nó là kết nối TCP hoàn thiện. Luồng dữ liệu được chuyển tiếp từ đoạn này sang đoạn kia (có thể sử dụng bộ đệm nếu cần thiết). Khi chia cắt kết nối được áp dụng trong mạng vệ tinh, đoạn kết nối giữa vệ tinh là đoạn có độ trễ lớn, hai đoạn từ bộ định tuyến kết nối Internet trên mặt đất với trạm đầu cuối nguyên gốc [1, 2].

Sự chia cắt phân ra các ảnh hưởng của độ trễ lớn. Nếu phân đoạn TCP đầu tiên và phân đoạn TCP cuối cùng là mạng có độ trễ thấp, TCP bắt đầu chậm có thể tăng tốc nhanh chóng và cửa sổ làm việc (không phải có độ rộng cửa sổ lớn - TCP-LW) có thể điều chỉnh làm việc tốt. Đoạn giữa là liên kết vệ tinh, có độ trễ lớn, vì thế có thể triển khai với tính năng đặc biệt, như T/TCP và sử dụng độ rộng cửa sổ lớn để đương đầu với độ trễ lớn. Với cách này, hiệu suất của TCP có thể được cải thiện với thay đổi không đang kể của các phần mềm ứng dụng.



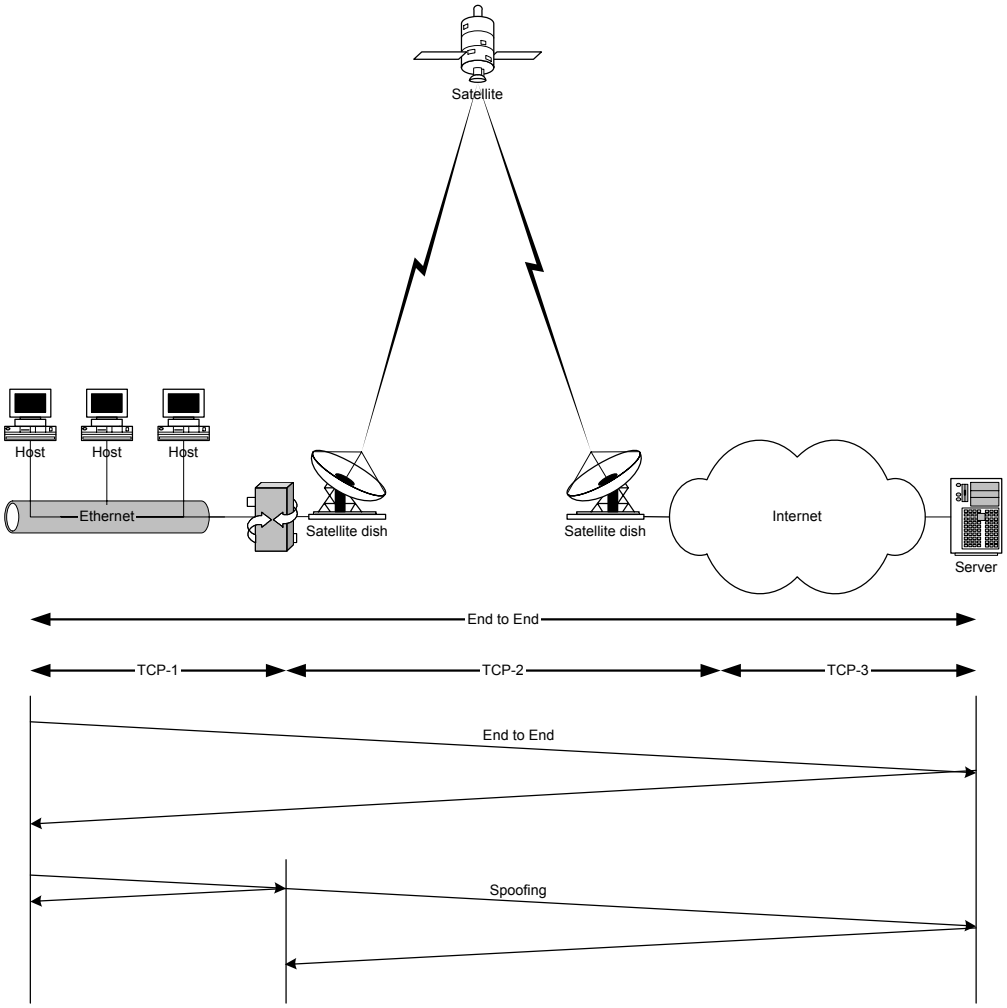
Hình 21: Lược đồ thời gian của chia cắt kết nối TCP

1.2. TCP Spoofing

Trong mô hình giao thức TCP Spoofing, một cổng kết nối trung gian (thường đặt tại cổng kết nối chiều lên của mạng vệ tinh) sớm trả lời ngay cho trạm phát về phân đoạn TCP mà không cần chờ thông tin trả lời thật sự từ trạm thu. Điều này tạo cho trạm phát ảo tưởng là mạng có độ trễ nhỏ và vì thế giai đoạn bắt đầu chậm của TCP tiến tới một cách nhanh chóng hơn. Tại cổng kết nối trung gian làm vùng đệm cho các phân đoan TCP quá giang. Khi thông tin phản hồi thực thực sự từ trạm thu trở lại cổng kết nối này, nó loại bỏ để ngăn ngừa sự trùng lặp thông tin trả lời đến trạm phát. Nếu thông tin phản hồi của trạm nhận không trở lại hoặc quá ngưỡng chờ của cổng kết nối, nó sẽ được phát lại phần thông tin bị mất từ bộ đệm cục bộ [1].

Giống như chia cắt kết nối TCP, TCP Spoofing cũng phá vỡ khái niệm kết nối đầu cuối - đầu cuối về mặt bản chất khi trạm phát nhầm tưởng phân đoạn dữ liệu đã đến đích, trong khi thực chất đang được chuyển tiếp quá giang. Việc này có thể chấp nhận trong nhiều ứng dụng (Ví

dụ WWW có thể duyệt WEB qua máy chủ Proxy), nhưng có thể có vấn đề nếu ứng dụng đã được thiết lập dựa vào kiến trúc đầu cuối - đầu cuối.



Hình 22: Lược đồ thời gian của TCP Spoofing

2. Snoop TCP

Nâng cao hiệu suất bằng uỷ quyền (proxy) là phương pháp sử dụng để làm giảm hiệu suất tại các đường kết nối khác nhau [12]. Giao thức Snoop là một trong các phương pháp đó [13]. Mục đích lớn nhất là cải tiến hiệu suất của truyền thông thông qua liên kết không dây và liên kết vệ tinh mà không cần đến tác động truyền lại thông tin và sự biến đổi cửa sổ tại tầng giao vận.

Thực chất Snoop TCP là một biến thể của TCP Spoofing. Nếu TCP Spoofing tự thực hiện việc đáp trả lời ACK thì Snoop TCP chỉ chuyển tiếp Ack khi nhận được Ack từ trạm thu trả lời. Tương tự TCP Spoofing, Snoop TCP cũng là trạm trung gian sử dụng bộ đệm và đồng hồ để kiểm soát việc phát lại.

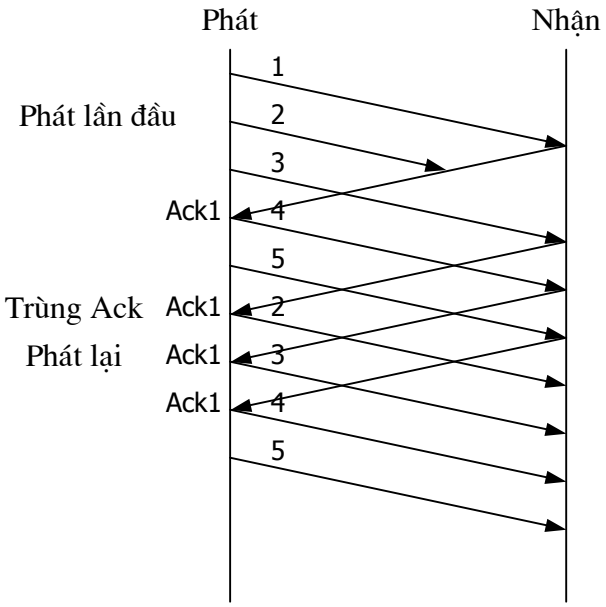
Tuy nhiên sự khác biệt cơ bản: TCP Spoofing được triển khai trên tầng 4 mô hình ISO/OSI vì thế là hình thức chia cắt TCP thực sự, khi triển khai trên thực tế phải tác động đến đầu cuối truyền số liệu, thực thể phát và trạm Spoofing phải triển khai một bộ giao thức làm việc để có thê gián tiếp truy nhập đến thực thể thu. Còn đối với Snoop TCP được triển khai chủ yếu trên tầng 2 của mô hình ISO/OSI. Vì vậy 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 ảnh cần phải triển khai đồng bộ một giao thức đặc biệt nào khác.

2.1. Giao thức Snoop TCP

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

Trong giao thức TCP, số tuần tự (sequence) liên quan đến từng gói tin phản hồi (Ack). Số tuần tự này cho biết thứ tự của byte dữ liệu cuối cùng của thực thể phát được phát thành công đến thực thể thu. Nếu thực thể phát nhận được thông báo phản hồi với số tuần tự giống như trước đó, điều đó có nghĩa dữ liệu được gửi và lần cuối cùng nhận được, các thông tin sau đó bị mất và được xác định bởi số tuần tự phản hồi. Một gói tin phản hồi có chứa số tuần tự nhỏ hơn số tuần tự nhận được lần cuối cùng được gọi là trùng thông tin phản hồi.

Xem xét lược đồ truyền thông tin mà một thực thể TCP phát năm tin gói 1, 2, 3, 4, 5 và chúng được chuyển tiếp bởi trạm mặt đất. Cho rằng gói tin số 2 bị lỗi trong khi truyền, các thông tin phản hồi (ACK) của các gói số 3, 4, 5 đều là các thông tin phản hồi trùng lặp. Khi thực thể phát nhận được ba gói tin phản hồi trùng lặp, thực thể phát truyền lại các gói tin và làm giảm đi kích thước của cửa sổ tắc nghẽn. Cơ chế phát lại của TCP được thể hiện trong hình 23:

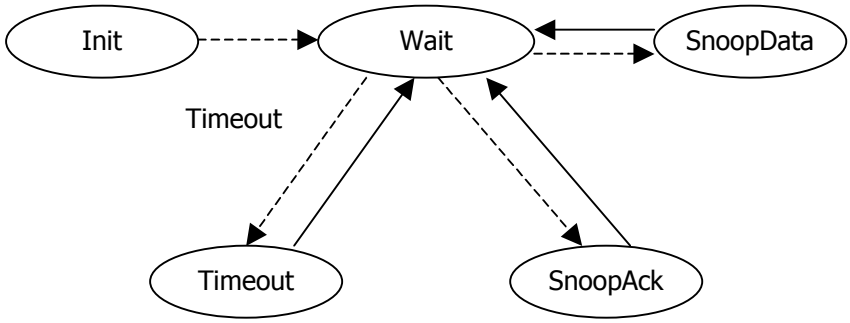


Hình 23: Cơ chế phát lại của TCP

Quy tắc của trạm Snoop là lưu trữ tạm thời các gói tin cho kết nối TCP. Khi dữ liệu bị mất (được thể hiện bằng việc nhận được các gói tin trả lời trùng lặp - duplicate Acks), trạm Snoop sẽ gửi lại các gói tin bị mất đó từ vùng đệm. Vì thế, tầng TCP sẽ không nhận thấy việc mất gói tin, 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 việc phát lại quá thời gian, trạm Snoop sẽ phát lại các gói tin mà không nhận được thông báo trả lời. Đồng hồ này được gọi là đồng hồ ổn định (persist) bởi vì không giống với đồng hồ phát lại TCP, nó có một giá trị cố định.

Giao thức Snoop chặn lại các gói TCP, phân tích chúng, và phát lại chúng nếu cần thiết. Kết quả là, không cần thêm và định dạng dữ liệu để chỉ dẫn cho giao thức hoạt động, và tất cả các gói tin gửi hay nhận đều thực sự không thay đổi và tuân theo giao thức TCP. Hơn nữa, không có sự thay đổi nào ở các tầng khác của tầng giao thức TCP/IP là một vấn đề được lưu ý. Điều này có ý nghĩa vô cùng quan trọng vì giao thức TCP được triển khai phổ biến, không phải dễ dàng đồng loạt triển khai với một phiên bản TCP sửa đổi để phù hợp với kết nối vệ tinh.

Hình 24 mô tả trạng thái làm việc của trạm Snoop



Hình 24: Lược đồ trạng thái xử lý của trạm Snoop

Lưu tạm (Snoop Cache): Snoop cache 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 nội dung thông tin điều khiển giao diện (Interface Control Information - ICI), cũng như nhiều bộ đếm của các gói tin mà đã được lưu tạm.

Bảng lưu kết nối Snoop (snoop connection table): Trạm Snoop luôn duy trì bảng thông tin giữ lại dấu vết của kết nối TCP. Bản thông tin này là cần thiết bởi vì hai trạm kết nối có thể có rất nhiều các liên kết được thiết lập, và mỗi thiết lập kết nối đó cần có một duy trì riêng biệt (bởi vì số tuần tự là không chỉ có một giữa các kết nối khác nhau). Mỗi thông tin của bảng lưu kết nối Snoop xác định một kết nối TCP bởi địa chỉ IP nguồn, địa chỉ IP đích kết hợp với cổng TCP. Chúng luôn giữ lại dấu vết của số tuần tự sau cùng và ghi lại số thông tin phản hồi sau cùng cho các liên kết. Ngoài ra mỗi liên kết sẽ được bổ sung thêm tham số timeout_evt để xác định thời gian truyền lại.

Trạng thái khởi tạo (Init state): Xử lý Snoop bắt đầu bằng Init State khi mà bảng lưu kết nối Snoop và Snoop Cache được khởi tạo.

Trạng thái chờ (Wait State): Sau khi trạng thái khởi tạo được thực hiện, tiến trình xử lý chuyển trạng thái từ "Khởi tạo" sang "chờ". Và tiến trình duy trì trạng thái chờ cho đến khi một trong số các sự kiện sau xảy ra:

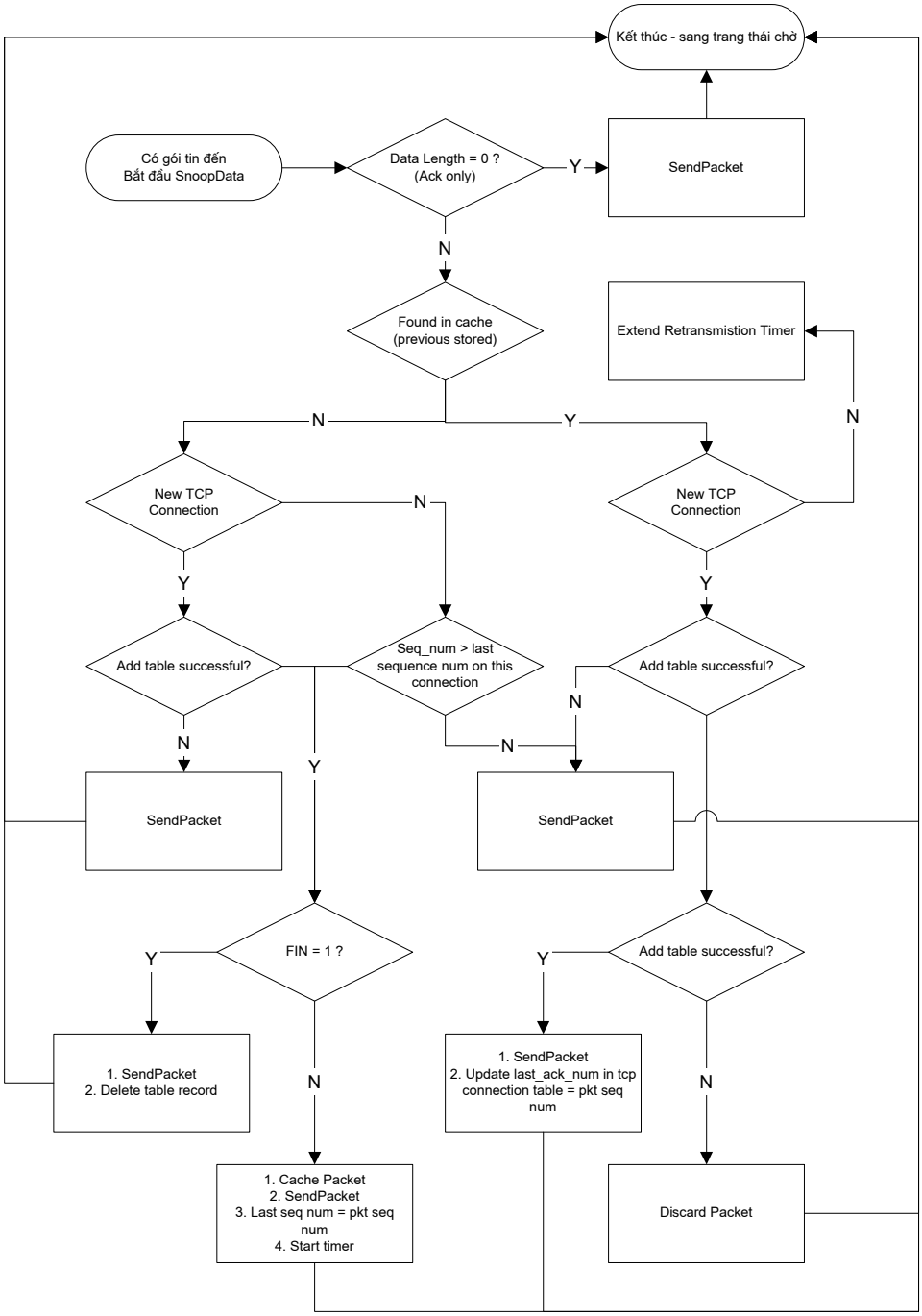
- Một gói tin đến từ tầng MAC
- Một gói tin đến từ tầng trên TCP/IP
- Đồng hồ xác định thời gian truyền lại quá ngưỡng.

Tất cả các sự kiện đến như là một ngắt. Gói tin đến tương ứng như một một ngắt tiến trình. Sự kiện đồng hồ xác định thời gian truyền lại quá ngưỡng được triển khai như một ngắt được tự lập lịch trước.

Trạng thái dữ liệu (SnoopData State): Khi gói tin ở tầng trên TCP/IP đến, tiến trình chuyển trạng thái từ "chờ" sang "dữ liệu". Trạm Snoop lưu vết của số tuần tự cuối cùng xác định từ tầng trên. Gói tin được xử lý tuỳ thuộc vào số tuần tự:

- Một gói tin mới với sự tuần tự TCP bình thường: đây là việc xử lý bình thường khi một gói tin mới với số tuần tự lớn hơn số tuần tự đã nhận. Gói tin được lưu tạm tại trạm Snoop đồng thời được chuyển tiếp đến tầng dưới.
- Một gói tin không đúng tuần tự mà đã được lưu tạm trước đó: điều này xảy ra khi mất gói tin bởi quá ngưỡng thời gian của thực thể phát. Nếu số tuần tự là lớn hơn số tuần tự cuối cùng được phản hồi được xác định bởi trạm Snoop, đó là dấu hiệu mất gói tin. Vì thế gói tin được chuyển tiếp đến thực thể nhận. Nếu số tuần tự nhỏ hơn số tuần tự cuối cùng đã có phản hồi, gói tin sẽ bị loại bỏ.

Sau khi xử lý xong gói tin, tiến trình xử lý Snoop chuyển trạng thái trở về trạng thái "chờ" và chờ đợi trạng thái kế tiếp. Hình 25 mô tả sơ đồ thuật toán của việc xử lý SnoopData.

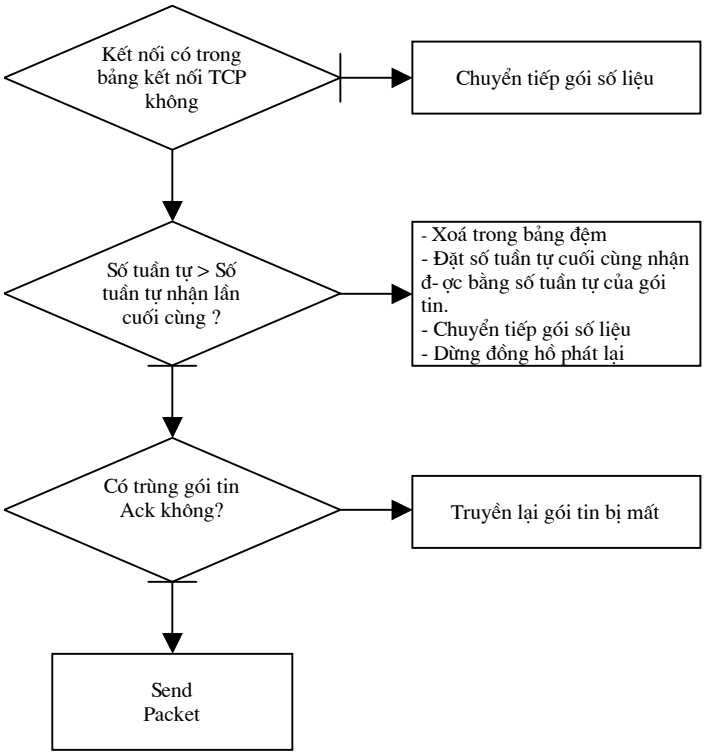


Hình 25: Sơ đồ thuật toán SnoopData

Trạng thái phản hồi thông tin (SnoopAck State): Khi một gói tin được đưa lên từ tầng MAC đến trạm Snoop, tiến trình xử lý Snoop chuyển trạng thái từ "chờ" sang trạng thái "phản hồi thông tin". Gói tin sẽ được xử lý phụ thuộc vào số tuần tự phản hồi:

- Gói tin phản hồi là mới: Là gói tin phản hồi có số tuần tự lớn số tuần tự mà lần cuối cùng đã nhận được. Gói tin phản hồi này khởi đầu cho việc xoá đi bản ghi tương ứng trong vùng đệm. Thông tin phản hồi được chuyển tiếp lên tầng trên là tầng TCP/IP.

- Gói tin phản hồi lỗi: Gói tin phản hồi này có số tuần tự nhỏ hơn số tuần tự lần cuối cùng nhận được. Điều này hiếm khi xảy ra, và gói tin phản hồi này bị huỷ bỏ.
- Trùng gói tin phản hồi: Gói tin phản hồi này giống hệt với gói tin phản hồi nhận được lần cuối cùng. Đó là nguyên nhân để trạm Snoop xác định gói dữ liệu có số tuần tự lớn hơn đã bị hư hỏng hoặc mất khi truyền. Trạm Snoop sẽ phát lại tất cả các gói bắt đầu từ gói tin bị hư hỏng xác định bằng số tuần tự đó.



Hình 26: Sơ đồ thuật toán SnoopAck

Trạng thái quá ngưỡng thời gian chờ (Timeout State): Khi đồng hồ kiểm soát việc phát lại quá ngưỡng thời gian chờ, tiến trình xử lý chuyển trạng thái từ trạng thái "chờ" sang trạng thái "quá ngưỡng thời gian chờ". Quá trình xử lý của trạng thái quá ngưỡng thời gian chờ tương tự với việc xử lý trùng gói tin phản hồi trong trạng thái "Phản hồi thông tin": Các gói tin không có thông tin phản hồi được phát lại.

Đồng hồ phát lại của liên kết được mở rộng thời gian chờ khi một gói dữ liệu mới được nhận từ tầng trên, một gói tin phản hồi nhận từ tầng dưới hay đồng hồ phát lại vừa quá ngưỡng thời gian chờ.

2.2. Kết nối vệ tinh với Snoop

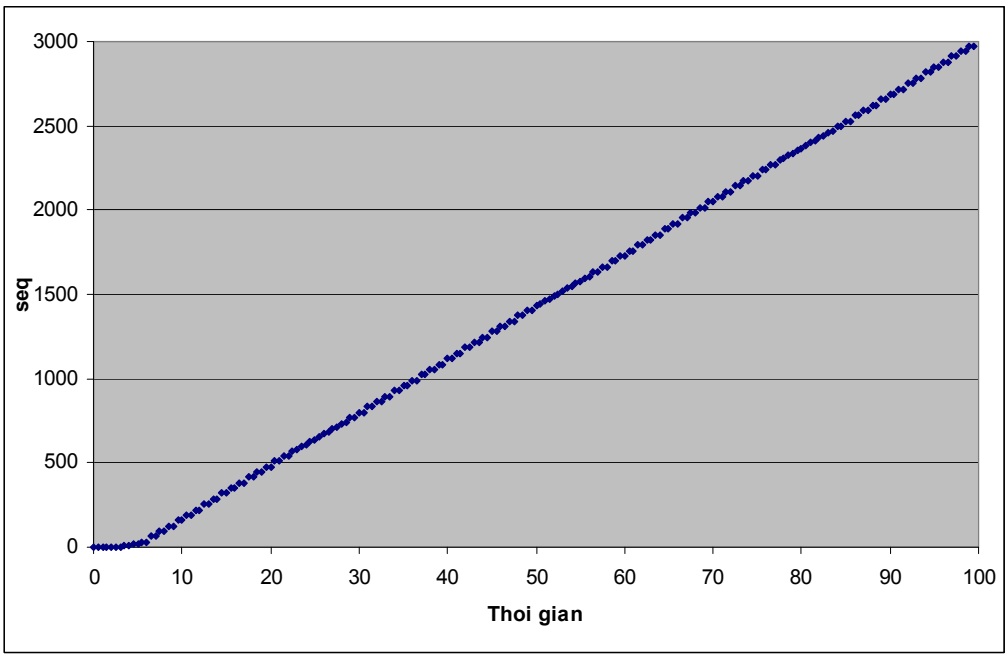
Như đã đề cập ở trên, Snoop được triển khai trên tầng 2 của mô hình ISO/OSI.

Để xem xét và đánh giá kết quả, chúng ta tiếp tục sử dụng chương trình NS2 [16] làm công cụ xây dựng mô phỏng hệ thống.

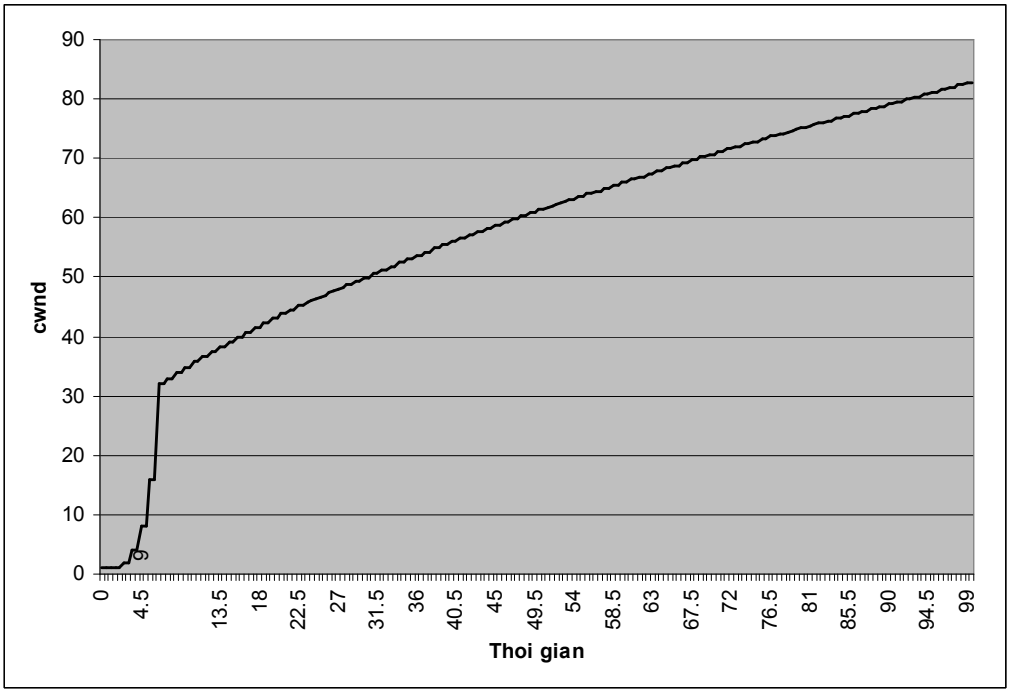
Trong bài thực nghiệm này giữ nguyên mô hình thí nghiệm trong mục 2.3 của phần này với sự thay đổi duy nhất: Trạm lặp (Repeter) được triển khai trong mô phỏng trước có tầng liên kết (Link Layer - LL) nguyên thuỷ, vì vậy trạm sẽ chuyển tiếp các gói tin qua lại và không hề có xử lý gì. Tuy nhiên, trong bài mô phỏng, thực hiện triển khai trạm này là trạm Snoop với tầng liên kết mở rộng đã cài đặt thuật toán Snoop kể trên (Link Layer Snoop - LL/LLSnoop).

Lặp lại các thí nghiệm của các kịch bản thử nghiệm 2 và kịch bản thử nghiệm 3: Sự thay đổi này được thực hiện tại tầng liên kết của cổng kết nối vệ tinh chiều truyền số liệu lên vệ tinh. Mô hình thí nghiệm vẫn được thực hiện với việc truyền số liệu bằng FTP trong thời gian 500s. Giao thức tầng giao vận vẫn sử dụng lần lượt các Tahoe-TCP, Reno-TCP, Newreno-TCP và Sack-TCP. Vẫn thiết lập với 2 kịch bản thử nghiệm khi kênh Vệ tinh lý tưởng không xảy ra lỗi và khi kênh có tỷ lệ lỗi là 2%.

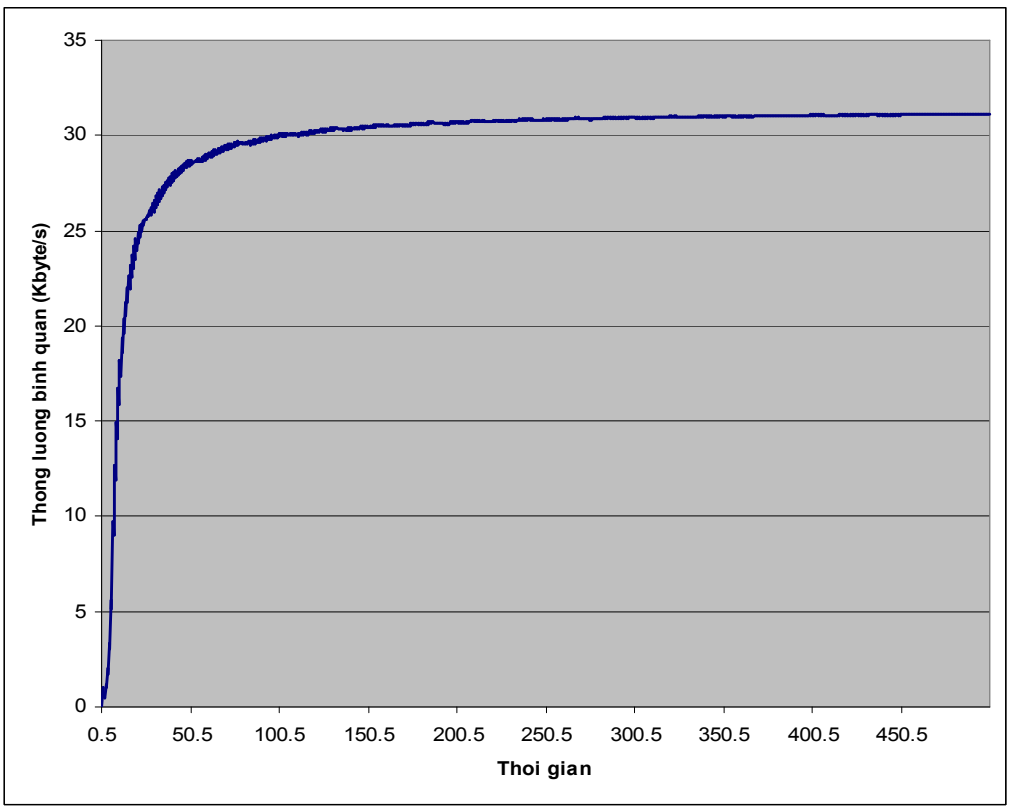
a. Kịch bản thử nghiệm 4: Kết nối qua kênh vệ tinh có trạm Snoop với các biến thể Tahoe, Reno, NewReno, Sack của TCP. Không có sự mất mát trên kênh kết nối:



Hình 27: Biểu đồ so sánh hiệu suất của Tahoe, Reno, Newreno, Sack khi không có sự mất mát thông tin trên kết nối E2E

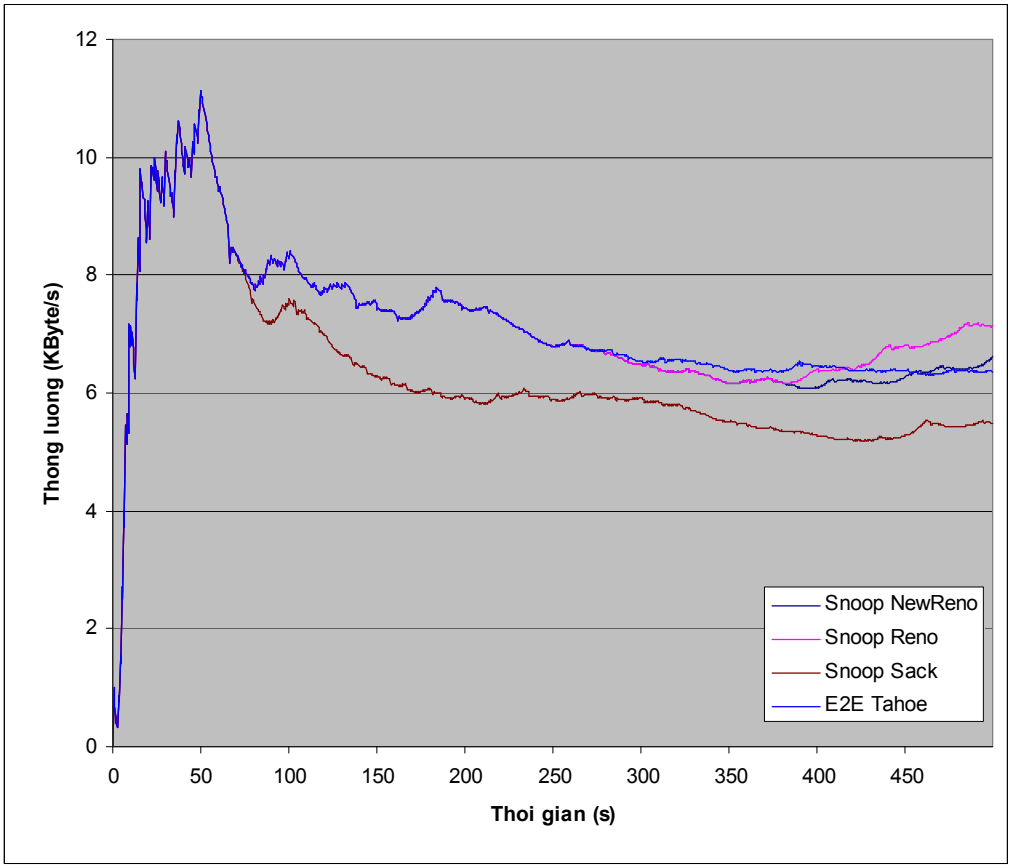


Hình 28: Biểu đồ so sánh sự thay đổi cửa sổ tắc nghẽn của Tahoe, Reno, Newreno, Sack khi không có sự mất mát thông tin trên kết nối Snoop

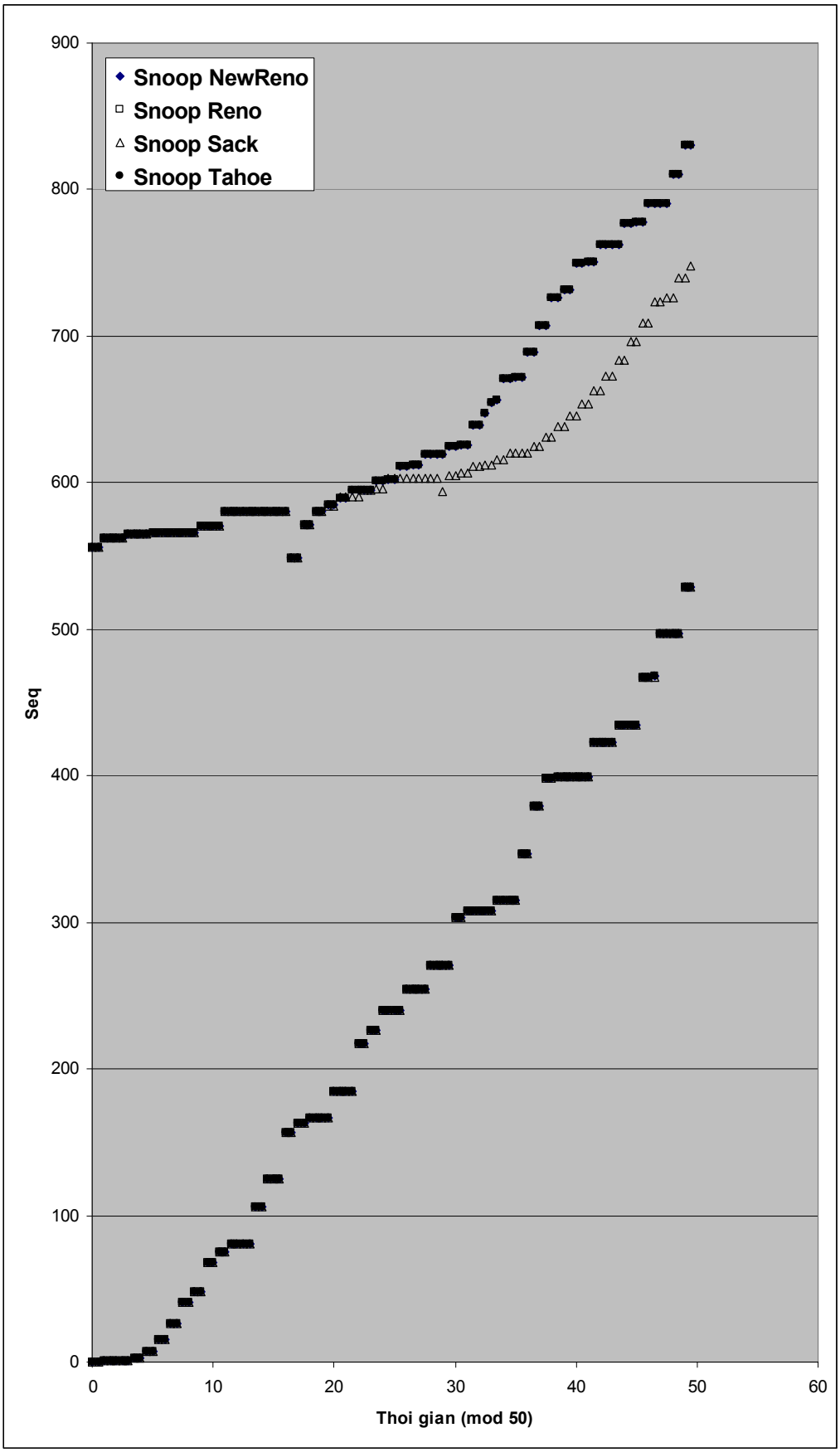


Hình 29: Thông lượng bình quân của Tahoe, Reno, Newreno, Sack khi không có sự mất mát thông tin trên kết nối E2E

b. Kịch bản thử nghiệm 5: Kết nối qua kênh vệ tinh có trạm Snoop với các biến thể khác Tahoe, Reno, NewReno, Sack. Sự mất mát trên mỗi chặng liên kết vệ tinh là 2% các gói tin truyền qua.



Hình 30: Biểu đồ so sánh thông lượng bình quân của Tahoe, Reno, Newreno, Sack khi có sự mất mát thông tin trên kết nối Snoop



Hình 31: Biểu đồ số tuần tự của Tahoe, Reno, Newreno, Sack khi có sự mất mát thông tin trên kết nối Snoop

2.3. Hiệu suất kết nối vệ tinh với Snoop

Kịch bản thử nghiệm 4:

Với việc áp dụng trạm Snoop là cổng kết nối vệ tinh chiều tải dữ liệu lên. Trong điều kiện lý tưởng là không xảy ra hư hỏng và mất mát gói tin trên kênh truyền, kết quả thu được cho ta thấy:

Giống như khi không áp dụng trạm Snoop, các thuật toán TCP mở rộng bao gồm Tahoe, Reno, NewReno và Sack có ứng xử giống nhau kể cả về số gói phát và sự điều chỉnh, thay đổi cửa sổ tắc nghẽn.

Thông lượng tối đa với giao thức điều khiển giao vận mở rộng Tahoe-TCP, Reno-TCP, NewReno-TCP, Sack-TCP là như nhau. Có giá trị cực đại trong thời gian thử nghiệm là 248Kbps.

Thí nghiệm cho ta thấy việc áp dụng trạm Snoop cũng không cải thiện được tốc độ của thuật toán bắt đầu chậm. Thông lượng tối đa vẫn chỉ đạt được sau 6.5 giây và đạt ngưỡng bão hoà. Khi đó thuật toán chống tắc nghẽn được thực thi.

Hiệu suất làm việc khi thuật toán bắt đầu chậm kết thúc và thuật toán chống tắc nghẽn làm việc là một đường đồng biến cho ta thấy khi không có sự mất mát thông tin, kênh làm việc ổn định.

Việc áp dụng trạm Snoop khi không có sự mất mát gói tin không làm thay đổi cách ứng xử của các thuật toán tầng giao vận TCP mở rộng bao gồm Tahoe, Reno, NewReno và Sack.

Kịch bản thử nghiệm 5:

Khác với kết nối đầu cuối - đầu cuối. Khi trạm Snoop được triển khai tại trạm mặt đất trong kết nối vệ tinh GEO, một phần việc phát hiện hư hỏng và mất gói tin được do trạm Snoop trung gian đảm nhận, vì thế trạm TCP đầu cuối không nhận thấy được sự mất mát, và giao thức tầng giao vận không phát hiện thấy sự tắc nghẽn và tiếp tục phát với số gói tin lớn có thể được. Vì mỗi khi phát hiện tắc nghẽn, cửa sổ tắc nghẽn lại đặt lại giá trị bằng 1, và như phân tích trên, nếu không xảy ra tắc nghẽn tiếp theo thì cũng phải mất đến hơn 6 giây thì mới đạt hiệu suất cao nhất của kênh truyền. Điều này làm giảm đáng kể hiệu suất làm việc.

Vì trạm Snoop giải quyết phần lớn các hư hỏng và mất gói tin, vì vậy giao thức điều khiển tầng giao vận không phát hiện tắc nghẽn và vì thế lối hành xử của các giao thức gần giống nhau. Ít nhất quan sát trên biểu đồ số tuần tự của các giao thức tầng giao vận trên kênh vệ tinh ta thấy trong 80 giây đầu, hầu như không có sự khác biệt giữa các giao thức Tahoe-TCP, Reno-TCP, Newreno-TCP.

Trong biểu đồ hình 30 ta cũng thấy hiệu suất trung bình các thuật toán đạt xấp xỉ khoảng 52Kbps chỉ ngang với Modem thoại phổ biến hiện nay.

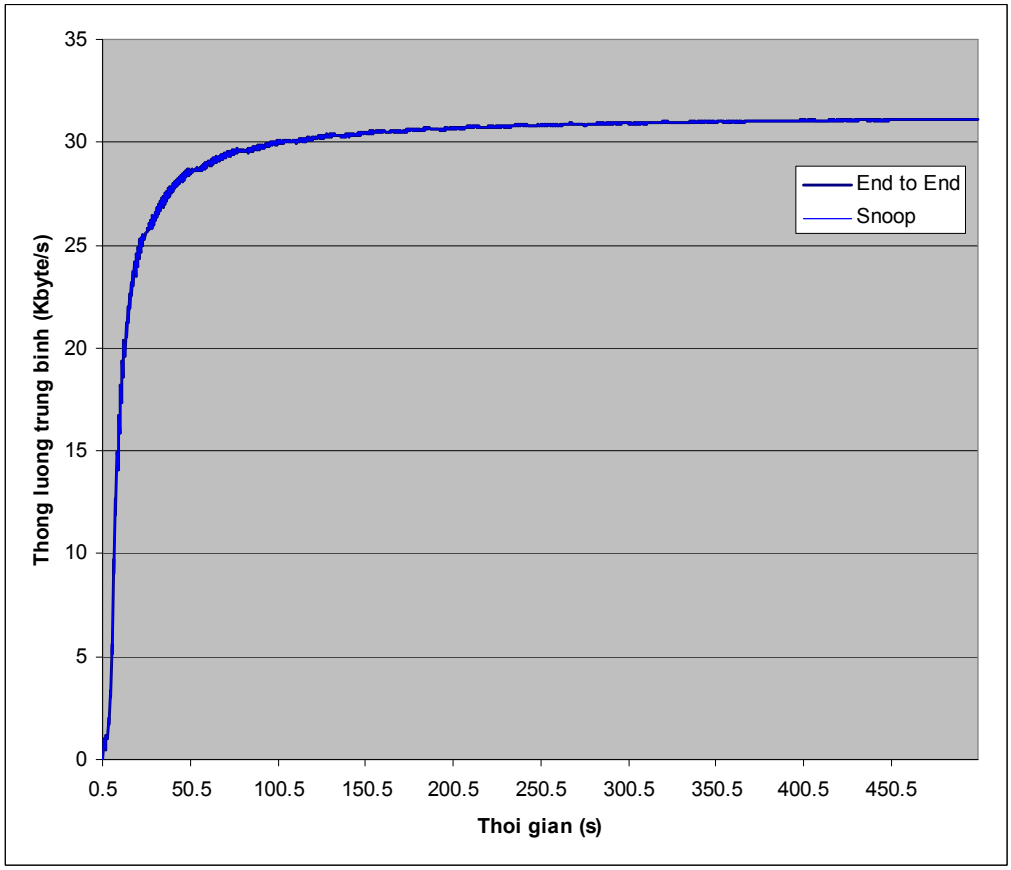
Đánh giá chính xác hơn hiệu suất làm việc của các giao thức điều kiển tầng giao vận này với giao thức Tahoe-TCP làm chuẩn, ta có Reno có hiệu suất cải thiện cao nhất, hơn 12%. Trái ngược với kết nối đầu cuối - đầu cuối khi mà hiệu suất của Sack-TCP cao hơn Tahoe-TCP và là giao thức có hiệu suất cao nhất thì ở đây hiệu suất Sack-TCP kém Tahoe-TCP tới 14% và là giao thức tồi nhất trong kết nối vệ tinh GEO có trạm Snoop. Nguyên nhân chính là do giao thức Sack-TCP với các thông tin phản hồi có sử dụng đến SACK. Tuy nhiên trạm Snoop với thuật toán SnoopAck kể trên không quan tâm đến trường dữ liệu này và chuyển tiếp gói tin ACK đến trạm TCP đầu cuối và ở đó trạm TCP phải thực hiện xử lý việc phát lại có lựa chọn. Trạm Snoop chỉ có thể giải phát lại các gói tin Timeout khi chưa có thông báo SACK của gói tin kế tiếp mà thôi. Điều đó giải thích vì sao giao thức Sack-TCP có hiệu suất kém hơn cả.

	Tahoe-TCP	***Reno-TCP***	***Newreno-TCP***	***Sack-TCP***
Số gói tin truyền sau 500 giây	3169	3561	3303	2735
Tỷ lệ so sánh với Tahoe-TCP		112%	104%	86%

Bảng 4: So sánh hiệu suất làm việc các giao thức điều kiển giao vận trên kênh vệ tinh GEO có trạm Snoop

CHƯƠNG IV: SO SÁNH HIỆU SUẤT CỦA TCP ĐẦU CUỐI – ĐẦU CUỐI VỚI TCP SNOOP

1. Điều kiện lý tưởng



Hình 32: Thông lượng trung bình của liên kết đầu cuối – đầu cuối và Snoop trên kênh vệ tinh

Xét hai kịch bản thí nghiệm 2 và 4 trong đó: Kịch bản 2 được thử nghiệm với kênh vệ tinh không có lỗi khi áp dụng các giao thức tầng giao vận là Tahoe-TCP, Reno-TCP, NewReno-TCP và Sack-TCP. Kịch bản 4 cũng được thử nghiệm với kênh vệ tinh không có lỗi khi áp dụng các giao thức tầng giao vận là Tahoe-TCP, Reno-TCP, NewReno-TCP và Sack-TCP song tại cổng truy nhập vệ tinh có triển khai trạm Snoop tại tầng liên kết dữ liệu.

Như kết luận của kịch bản thử nghiệm 2: các giao thức tầng giao vận Tahoe-TCP, Reno-TCP, NewReno-TCP và Sack-TCP có hiệu suất như nhau và hoạt động tương tự nhau khi không có tắc nghẽn.

Tương tự tại kịch bản 4 cũng có kết luận như vậy.

So sánh hiệu suất làm việc của giao thức tầng giao vận Tahoe-TCP, Reno-TCP, NewReno-TCP và Sack-TCP trong kịch bản thử nghiệm 2 và 4 ta thấy: hiệu suất làm việc hoàn toàn giống nhau. Số tuần tự và ứng xử của cửa sổ tắc nghẽn là hoàn toàn giống nhau. Điều này cho ta thấy việc không ảnh hưởng của Snoop khi không có sự thất lạc hoặc

hư hỏng gói tin. Cũng với việc so sánh này cho ta kết luận Snoop không cải thiện thuật toán làm việc của tầng giao vận.

Áp dụng trạm Snoop không ảnh hưởng đến kênh vệ tinh lý tưởng. Trong thí nghiệm, thông lượng tối đa là 258Kbps.

2. Kết nối trên kênh vệ tinh với tỷ lệ lỗi lớn

2.1. Tahoe TCP trên kết nối E2E và Snoop - kênh vệ tinh

Sử dụng lại kết quả của kịch bản thử nghiệm 3 và kịch bản thử nghiệm 5 với việc sử dụng giao thức Tahoe-TCP trên kênh vệ tinh với kết nối đầu cuối – đầu cuối và kết nối có trạm Snoop.

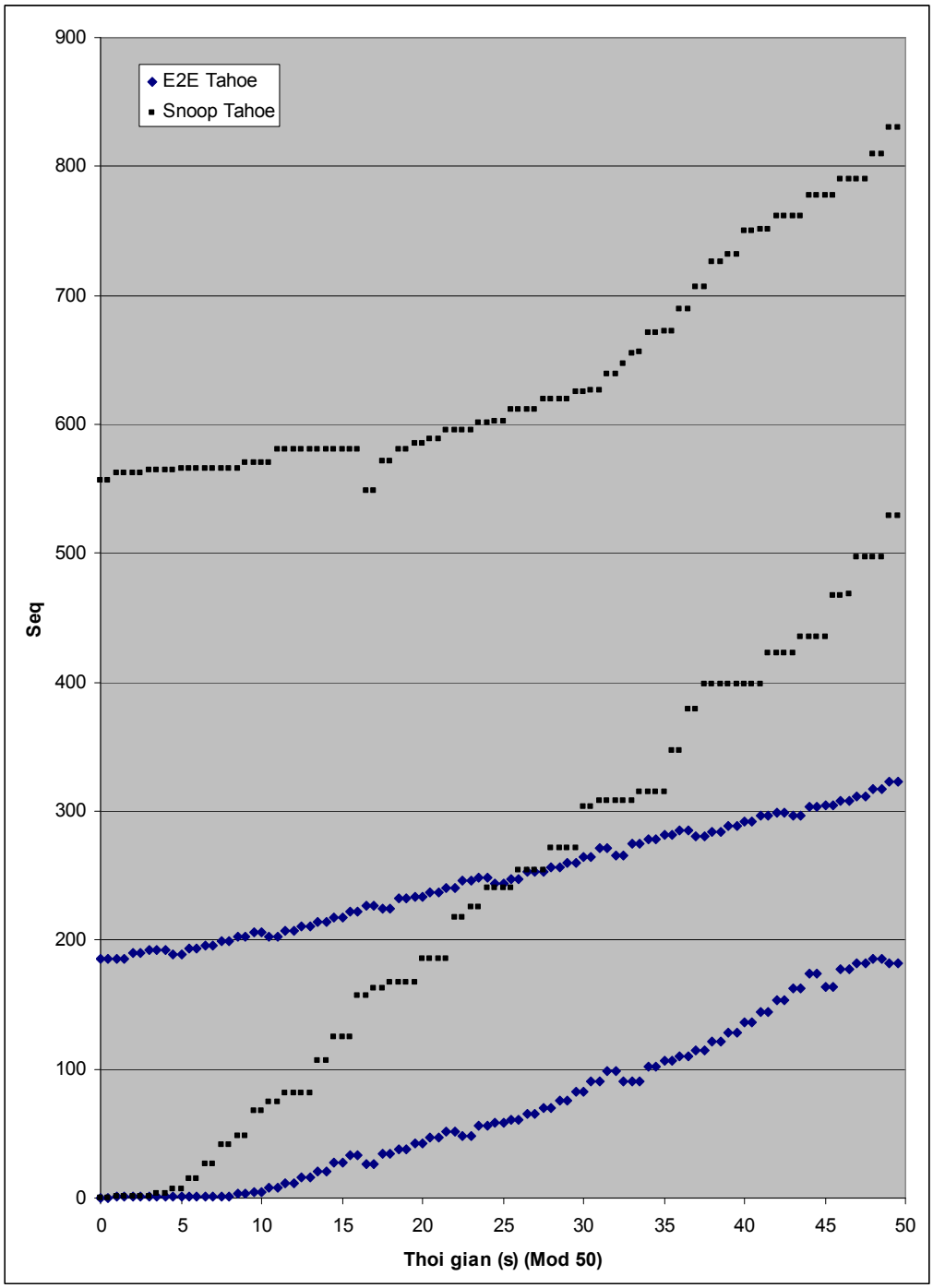
Tiêu chí	*E2E Tahoe*	*Snoop Tahoe*
Hiệu suất so với thuật E2E Tahoe		172%
Hiệu suất so với kênh không có lỗi	12%	20%

Bảng 5: So sánh hiệu suất làm việc Tahoe-TCP qua kết nối vệ tinh GEO

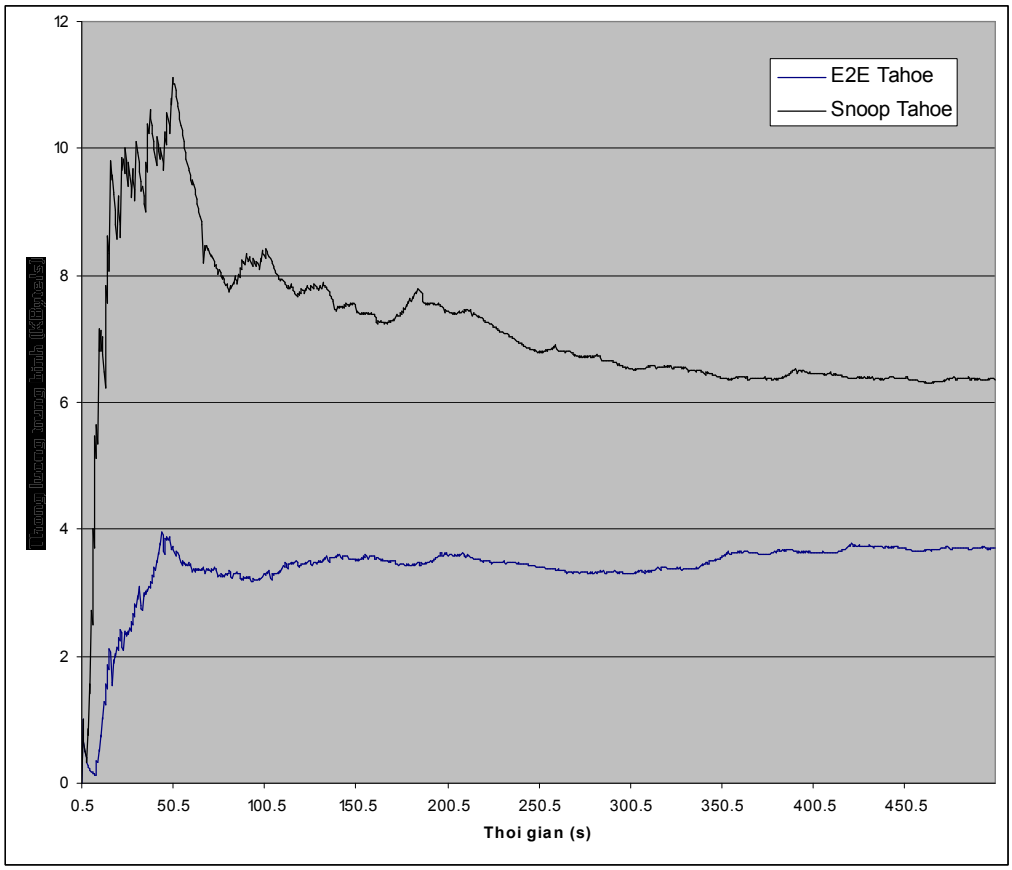
Phân tích sự quan sát của trạm TCP với giao thức Tahoe-TCP bằng công cụ phân tích NAM như đã đề cập ta thấy:

- Với kênh vệ tinh GEO có mô hình kết nối tầng điều khiển giao vận đầu cuối - đầu cuối có thể quan sát được các gói tin bị mất và chính là nguyên nhân gây hiện tượng tắc nghẽn.
- Tuy nhiên với kênh vệ tinh GEO có trạm Snoop, tầng điều khiển giao vận không nhận thấy sự mất mát các gói tin này.

Quan sát biểu đồ so sánh thông lượng trung bình của Tahoe-TCP trên kênh vệ tinh GEO đầu cuối - đầu cuối và kênh vệ tinh GEO có trạm Snoop, cho thấy thông lượng được cải thiện một cách đáng kể tăng hơn 72%.



Hình 33: Biểu đồ so sánh số tuần tự của Tahoe-TCP trên kết nối E2E và Snoop - kênh vệ tinh

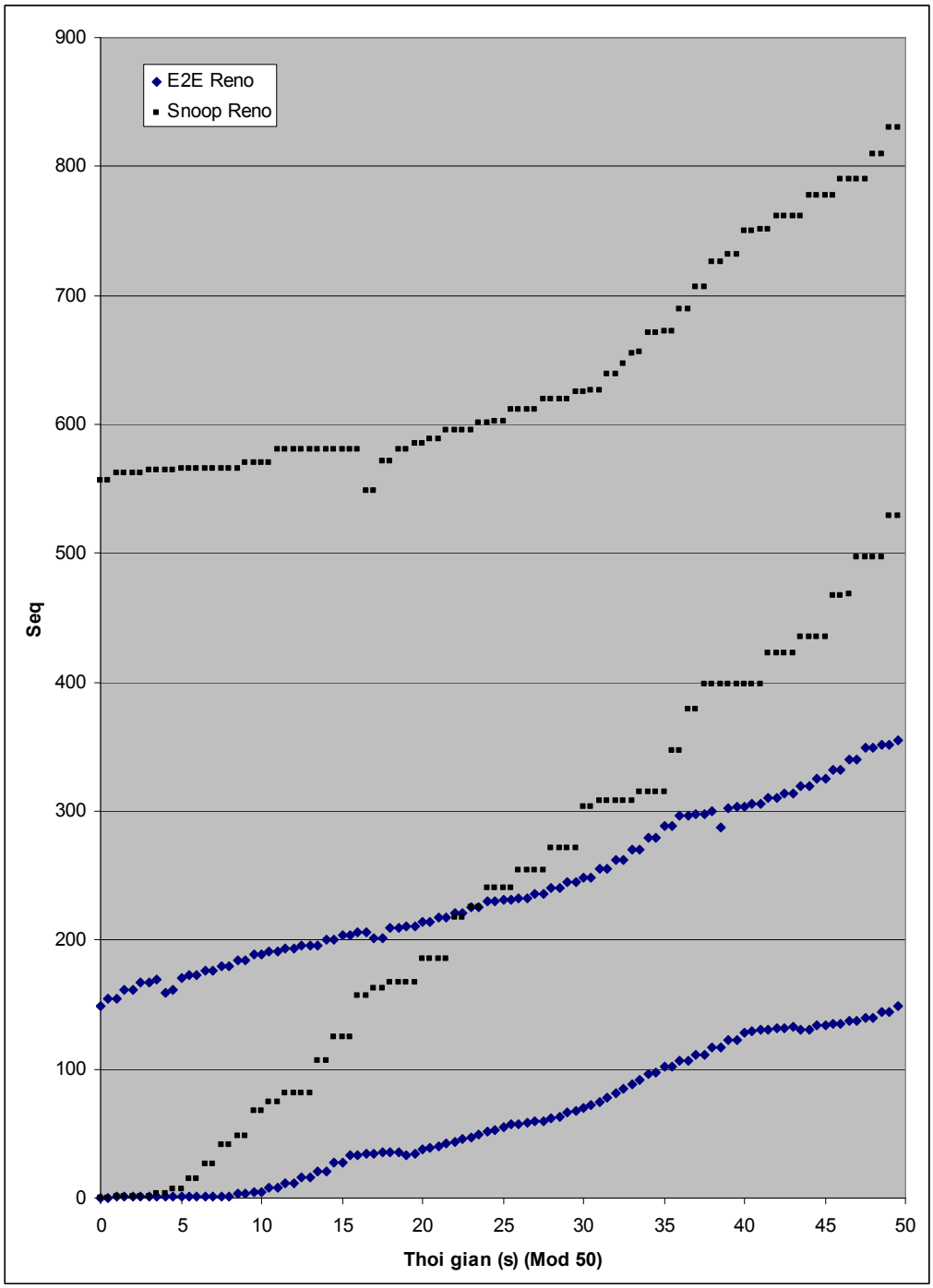


Hình 34: Biểu đồ so sánh thông lượng trung bình của Tahoe-TCP trên kết nối E2E và Snoop - kênh vệ tinh

2.2. Reno TCP trên kết nối E2E và Snoop kênh vệ tinh

Tiêu chí	*E2E Reno*	*Snoop Reno*
Hiệu suất so với thuật E2E Tahoe	107%	193%
Hiệu suất so với kênh không có lỗi	13%	23%

Bảng 6: So sánh hiệu suất làm việc Reno-TCP qua kết nối vệ tinh GEO



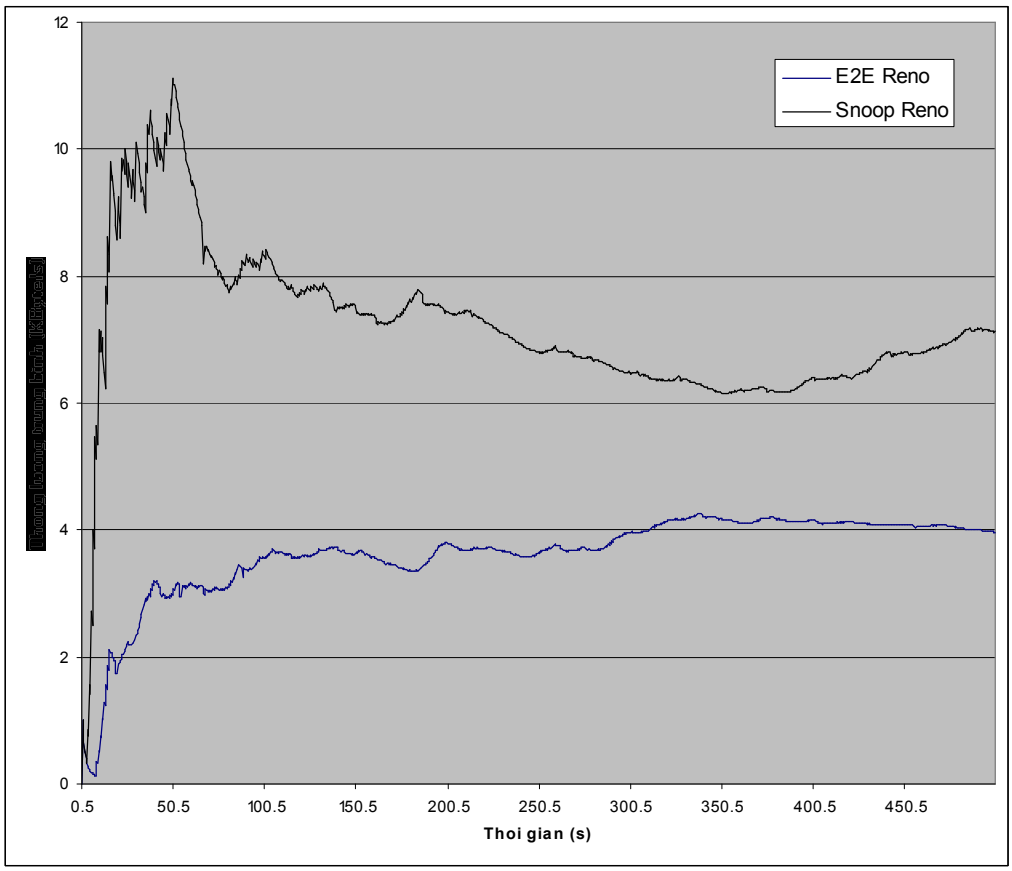
Hình 35: Biểu đồ so sánh số tuần tự của Reno-TCP trên kết nối E2E và Snoop - kênh vệ tinh

Tương tự với Tahoe, ta cũng quan sát giao thức Reno-TCP trên cả hai kênh vệ tinh GEO đầu cuối - đầu cuối và có trạm Snoop. Nhận xét cũng hoàn toàn tương tự như với Reno-TCP khi phân tích bằng NAM. Trong đó:

- Với kênh có trạm Snoop, Reno-TCP không phát hiện được các gói tin bị mất bởi các thông báo NACK hoặc bởi số tuần tự mà thực chất Reno-TCP phát hiện chỉ khi đồng hồ Time Out quá ngưỡng thời gian chờ. Tuy nhiên nếu trạm Snoop phát lại mà thành công và

gói tin Ack trở về bình thường thì khi đó Reno-TCP coi như đã phát thành công

Chính vì thế mà hiệu suất của Reno-TCP được cải thiện tốt với Reno-TCP có trạm Snoop so với kết nối đầu cuối - đầu cuối. Hiệu suất được cải thiện 180%. Một biện pháp cải tiến cho hiệu quả đáng kể.



Hình 36: Biểu đồ so sánh thông lượng trung bình của Reno-TCP trên kết nối E2E và Snoop - kênh vệ tinh

2.3. Newreno TCP trên kết nối E2E và Snoop kênh vệ tinh

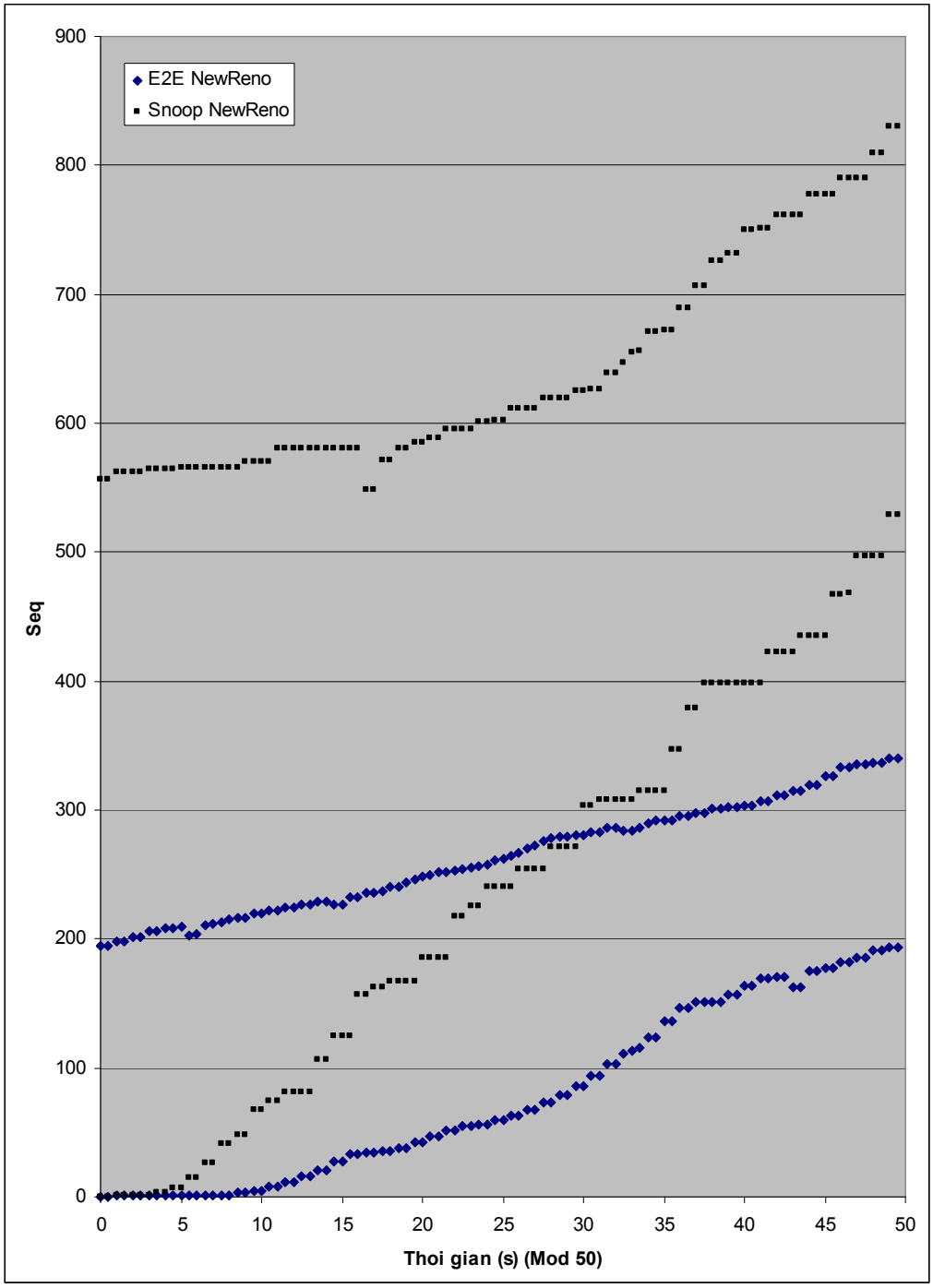
Tiêu chí	*E2E Newreno*	*Snoop Newreno*
Hiệu suất so với thuật E2E Tahoe	108%	179%
Hiệu suất so với kênh không có lỗi	13%	21%

Bảng 7: So sánh hiệu suất làm việc Newreno-TCP qua kết nối vệ tinh GEO

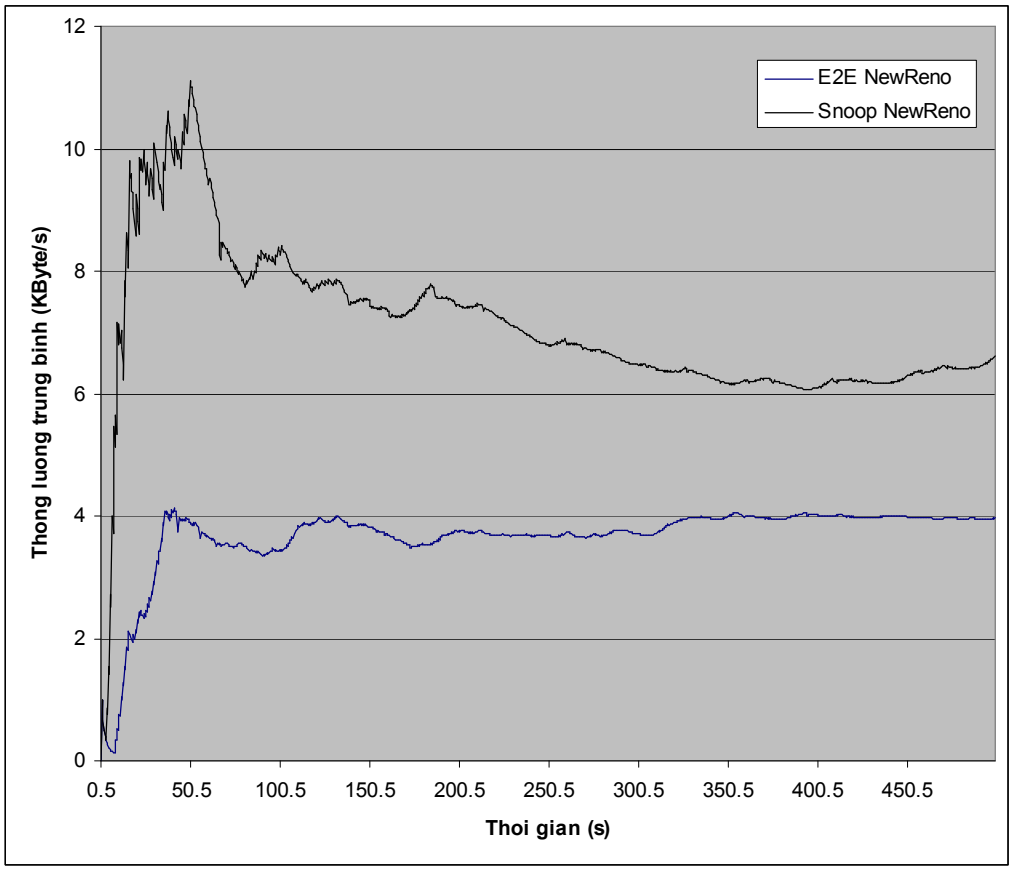
Tiếp tục sử dụng sử dụng kết quả của kịch bản thí nghiệm 3 và thí nghiệm 5 cho việc so sánh và phân tích hiệu suất của giao thức Newreno-TCP trên kênh vệ tinh GEO. So sánh và vẽ đồ thị với kết nối đầu cuối - đầu cuối và kết nối có trạm Snoop.

Rõ ràng so với Reno-TCP, mặc dù có sự cải tiến so với Reno-TCP, song trên thí nghiệm cho thấy mức độ cải thiện của Newreno-TCP không tốt hơn trên kênh vệ tinh GEO có độ trễ lớn và tỷ lệ lỗi lớn khi

áp dụng trạm Snoop. Tuy nhiên, hiệu suất trên kênh có trạm Snoop được cải thiện 166% so với kết nối đầu cuối của giao thức này.



Hình 37: Biểu đồ so sánh số tuần tự của Newreno-TCP trên kết nối E2E và Snoop - kênh vệ tinh

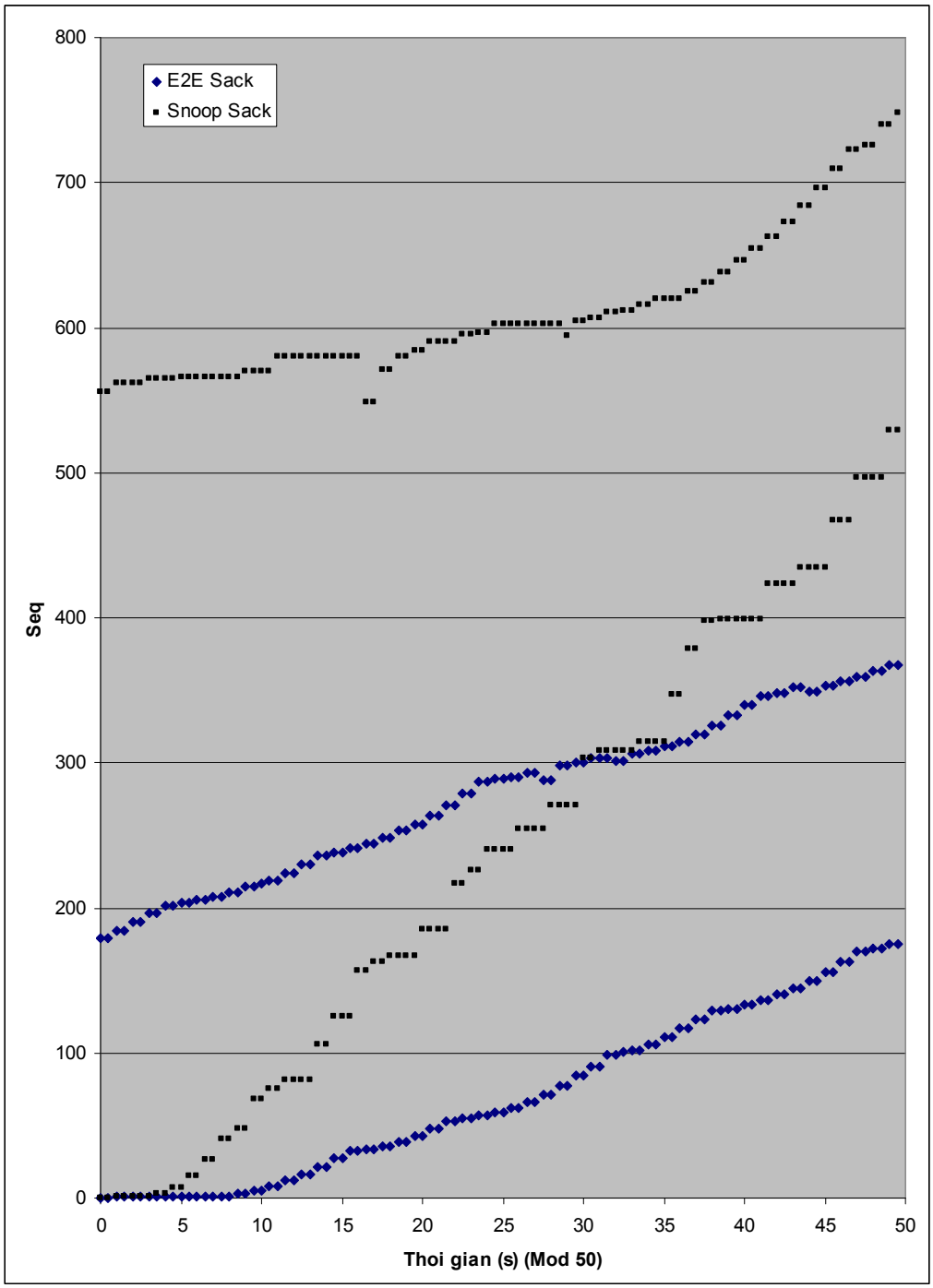


Hình 38: Biểu đồ so sánh thông lượng trung bình của Newreno-TCP trên kết nối E2E và Snoop - kênh vệ tinh

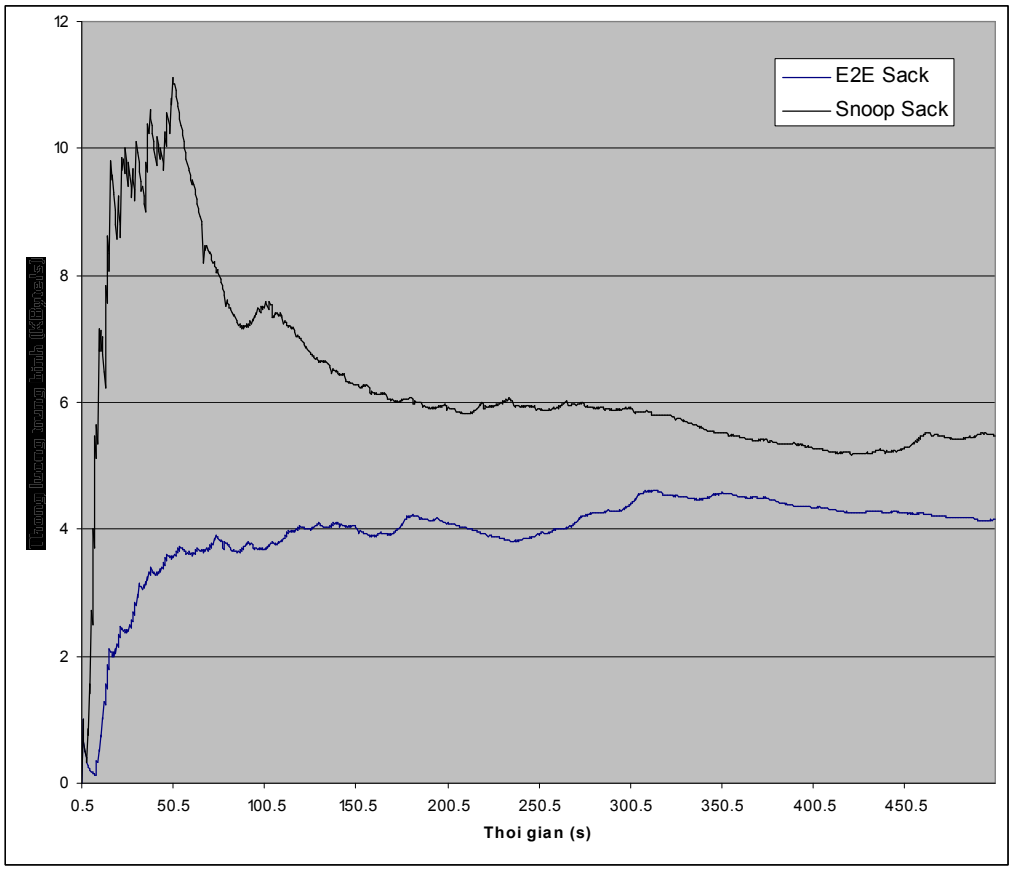
2.4. Sack TCP trên kết nối E2E và Snoop kênh vệ tinh

Tiêu chí	*E2E Sack*	*Snoop Sack*
Hiệu suất so với thuật E2E Tahoe	112%	148%
Hiệu suất so với kênh không có lỗi	13%	18%

Bảng 8: So sánh hiệu suất làm việc Reno-TCP qua kết nối vệ tinh GEO



Hình 39: Biểu đồ so sánh số tuần tự của Sack-TCP trên kết nối E2E và Snoop - kênh vệ tinh



Hình 40: Biểu đồ so sánh thông lượng trung bình của Sack-TCP trên kết nối E2E và Snoop - kênh vệ tinh

So sánh kết quả của giao thức Sack-TCP trên kết nối Snoop và đầu cuối - đầu cuối kênh vệ tinh GEO. Hiệu suất được cải thiện chỉ có 132%.

Thực tế của thí nghiệm cho thấy áp dụng Snoop cũng có kết quả làm tăng hiệu suất, song tỷ lệ cải thiện không cao như so với các giao thức Tahoe-TCP, Reno-TCP, Newreno-TCP. Nguyên nhân chính do thuật toán SACK sử dụng tuỳ chọn SACK song trạm snoop không quan tâm đến giá trị này khi quyết định phát lại hay chuyển tiếp gói tin phản hồi. Thuật toán SnoopAck được thực hiện khi nhận gói tin phản hồi chỉ quan tâm chủ yếu đến số tuần tự trong khi đó SACK sử dụng mô tả [18] để điều khiển việc phát lại nhanh. Chính vì vậy, trạm snoop không phát hiện và thực hiện phát lại các gói tin mà chuyển tiếp gói ACK cho trạm phát vì thế việc phát lại chủ yếu do trạm TCP phát thực hiện. Chỉ các gói tin qua trạm Snoop có đồng hồ thời gian chờ quá ngưỡng mới thực hiện phát lại; vì thế, hiệu quả cải tiến chưa được cao.

Trong bài luận văn này, tôi chưa đủ điều kiện thời gian nghiên cứu để cải tiến việc áp dụng trạm snoop cho phù hợp với giao thức Sack-TCP nhằm nâng cao hiệu quả làm việc. Tôi hy vọng vấn đề này sẽ được quan tâm nghiên cứu và làm sáng tỏ.

2.5. Đánh giá

Tiêu chí	*Tahoe-TCP*	*Reno-TCP*	*Newreno-TCP*	*Sack-TCP*
Hiệu suất Snoop so với E2E	172%	180%	166%	132%
Hiệu suất Snoop so với kênh không có lỗi	20%	23%	21%	18%
Hiệu suất E2E so với kênh không có lỗi	12%	13%	13%	13%

Bảng 9: So sánh hiệu suất của các thuật toán TCP mở rộng

Tổng hợp với bảng so sánh hiệu suất của các thuật toán TCP mở rộng trên cho ta thấy một cách bao quát về hiệu suất trên kênh Vệ tinh:

- Khi kênh vệ tinh với giả thuyết có tỷ lệ mất gói tin 2% và độ rộng cửa sổ phát cố định 32Kbyte, kích thước dữ liệu trong gói tin là 1024byte. Khi đó hiệu suất thực sự xác định được trên tầng điều khiển giao vận giảm đi rất lớn trong đó với các thuật toán TCP mở rộng, hiệu suất giảm chỉ còn 12 đến 13%. Điển hình với Sack-TCP hiệu suất giảm chỉ còn 12%. Rõ ràng tỷ lệ lỗi lớn trên kênh truyền có độ trễ lớn của vệ tinh GEO có hiệu suất làm việc thấp với kết nối đầu cuối - đầu cuối truyền thống, ngay cả với các thuật toán cải tiến, hiệu suất cũng rất thấp và mức độ cải thiện giữa các thuật toán cũng không đáng kể trong đó với thuật toán Sack có mức độ cải tiến khả quan nhất so với tất cả.
- Tuy nhiên với việc cải tiến bổ sung thêm trạm Snoop, hiệu suất được cải thiện trong đó: Sack-TCP trong kết nối đầu cuối - đầu cuối cải thiện hiệu suất tốt hơn cả thì trong kết nối vệ tinh có trạm Snoop, hiệu suất cải thiện so với các thuật toán khác là thấp nhất song cũng đạt 18% so với kênh truyền không có lỗi (kết nối đầu cuối - đầu cuối cao nhất là 13%). Hiệu suất cải thiện tốt nhất trên kênh vệ tinh có trạm Snoop chính là thuật toán Reno-TCP với hiệu suất cải thiện và bằng 23% so với kênh không có lỗi. Tiếp sau là Newreno-TCP và Tahoe-TCP với 21% và 20%.

Nếu đánh giá trong cùng một thuật toán tầng điều khiển giao vận, giữa hai phương thức kết nối đầu cuối - đầu cuối và kết nối có trạm snoop: Rõ ràng tất cả các thuật toán tầng điều khiển giao vận đều làm việc tốt hơn do một phần các gói tin hư hỏng đã được trạm Snoop phát lại và vì tầng điều khiển giao vận ít gặp phải tắc nghẽn hơn. Hiệu suất trên kênh vệ tinh có trạm Snoop được cải thiện nhiều nhất so với kênh kết nối đầu cuối - đầu cuối cùng một thuật toán là Reno-TCP với mức độ cải thiện 180%. Tiếp đến là Tahoe-TCP với 172%. Tuy rằng có cải thiện hiện suất nhưng Sack-TCP có giá trị cải thiện thấp nhất chỉ 132%.

KẾT LUẬN

Công nghệ thông tin phát triển, mạng máy tính phát triển một cách nhanh chóng đặc biệt Internet đã phát triển một cách vượt bậc về chiều rộng và chiều sâu. Mạng Internet đã vươn rộng không chỉ kết nối và cung cấp thông tin trên máy tính mà còn kết nối truyền dẫn hình ảnh âm thanh và các dữ liệu đa phương tiện khác. Tốc độ Internet cũng không ngừng cải thiện nhằm đáp ứng cao hơn về các ứng dụng yêu cầu băng thông rộng. Song Internet phát triển không những về tốc độ mà mức độ trải rộng ngày càng lớn. Mạng Internet phát triển cũng chính nhờ giao thức TCP/IP, một giao thức hiệu quả và đã trở thành chuẩn phổ biến của tất cả các mạng số liệu hiện nay.

Internet không chỉ phát triển ở các thành phố lớn mà còn phát triển ở cả vùng xa xôi hẻo lánh, núi cao hải đảo,... Góp phần cho sự phát triển đó chính là hệ thống truyền dẫn bằng Vệ tinh. Và cùng với sự phát triển của Internet, hệ thống truyền thông vệ tinh cũng vào cuộc phục vụ nhu cầu kết nối Internet

Ưu điểm nổi bật của Vệ tinh là cho phép cung cấp kết nối với dải thông lớn, vùng phủ rộng trên địa hình phức tạp bởi sự không cần đi dây liên kết. Chính điều này giúp cho phép truy nhập mạng với chi phí không cao. Tuy nhiên nhược điểm của hệ thống vệ tinh chính là do khoảng cách vệ tinh cách xa mặt đất cùng với các yếu tố môi trường tác động khi truyền thông tin làm cho hệ thống truy nhập vệ tinh có độ trễ lớn và tỷ lệ lỗi khá cao so với các hệ thống truyền thông hữu tuyến khác.

Cố gắng sử dụng mạng vệ tinh trong kết nối Internet với giao thức TCP/IP đã được các nhà khoa học trên thế giới nghiên cứu và được áp dụng thực tiễn và thương mại hoá. Trong luận văn nghiên cứu này không có ý định đánh giá lại việc triển khai việc kết nối Internet qua kênh viễn thông vệ tinh mà trên phương diện lý thuyết cũng như việc giả lập và mô phỏng hệ thống để xem xét các đặc trưng của việc điều khiển lưu lượng của TCP trên kênh viễn thông vệ tinh. Dựa vào đặc điểm của kênh viễn thông vệ tinh cũng như có thể áp dụng một số biện pháp cải tiến, bài viết cố gắng đưa ra khả năng cải thiện việc điều khiển truyền số liệu Internet qua kênh vệ tinh. Trong bài viết chỉ quan tâm với kênh vệ tinh GEO.

Để giải quyết vấn đề, trong phần “Cơ sở lý thuyết":

Bài viết đã đề cập về đặc tích của hệ thống vệ tinh với các đặc điểm: dải thông lớn, cung cấp khả năng truyền thông liên tục và có cấu trúc mạng đơn giản. Tuy nhiên kênh vệ tinh cũng có những đặc điểm cần phải đặc biệt lưu tâm khi áp dụng để truyền thông tin Internet: Độ trễ kênh truyền lớn với RTT khoảng 1000ms. Nhiễu do môi trường tác động gây nên tỷ lệ lỗi lớn

trong khi truyền. Và với đặc trưng của kiến trúc mạng vệ tinh cũng tạo nên tính không đối xứng trong truyền thông. Và cuối cùng chính là hiện tượng tắc nghẽn cổ chai tại các điểm nút kết nối vệ tinh có thể xảy ra. Đó là các vấn đề có thể ảnh hưởng khi thực hiện truyền dữ liệu Internet. Tuy nhiên để thu hẹp phạm vi, trong luận văn này chỉ quan tâm đến hai yếu tố chính đó là độ trễ lớn và tỷ lệ lỗi lớn làm ảnh hưởng đến truyền thông.

Để thực hiện việc điều khiển giao vận trên kênh vệ tinh Internet, một loạt các vấn đề liên quan đến việc điều khiển như: kích thước cửa sổ, sự thích nghi dải thông, phát lại có lựa chọn, bắt đầu chậm, tránh tắc nghẽn, phát lại nhanh, khôi phục nhanh được đề cập cũng như cấu trúc và nguyên tắc làm việc của TCP.

Với sự kết hợp của các thuật toán bắt đầu chậm, tránh tắc nghẽn, phát lại nhanh, khôi phục nhanh và sự cải tiến của giao thức TCP trong việc điều khiển và chống tắc nghẽn, luận văn đã đề cập lại nguyên tắc và thuật toán của các giao thức TCP mở rộng bao gồm Tahoe-TCP, Reno-TCP, Newreno-TCP và Sack-TCP. Đó là giao thức điều khiển giao vận trên Interrnet mà đã đưa vào xem xét và đánh trên kênh vệ tinh GEO.

Thay vì không thể có điều kiện thử nghiệm và đo đạc các thông số và quan sát khả năng làm việc trên hệ thống vệ tinh trong điều kiện thực tế, luận văn sử dụng bộ chương trình mô phỏng mạng NS được phát triển bởi Phòng thí nghiệm quốc gia Mỹ, một phần mềm miễn phí phục vụ công tác nghiên cứu và phân tích mạng làm công cụ mô phỏng các thí nghiệm và thu thập kết quả để đánh giá.

Với các vấn đề đã được trình bày trong phần lý thuyết làm cơ sở của "Hiệu suất TCP qua kênh vệ tinh":

Với việc cố định một số tham số của kênh vệ tinh, ta tiến hành quan sát hiệu suất làm việc của giao thức điều khiển giao vận khi thay đổi kích thước gói tin, tốc độ kênh truyền và kích thước cửa sổ.

- Từ đó ta có nhận định với kích thước gói tin khác nhau cũng ảnh hưởng đến hiệu suất làm việc tuy rằng mức độ ảnh hưởng không lớn song với kích thước 1024Byte dữ liệu TCP sẽ là hiệu suất tốt nhất.
- Bằng việc thay đổi độ rộng kênh truyền, cũng có thể thấy được ảnh hưởng của độ rộng kênh truyền với hiệu suất TCP. Tuy rằng hiệu suất cũng tăng khi kênh có độ rộng lớn. Tuy nhiên có thể thấy được mức độ được cải thiện cao nhất khi kênh đạt 256Kbps và với các kênh tốc độ cao hơn, mức độ được cải thiện thêm là không đáng kể.

- Theo tính toán lý thuyết để có thể phát huy tốt kênh vệ tinh GEO với độ rộng kênh truyền số liệu có thể đến 2Mbps và độ trễ rất lớn thì độ rộng cửa sổ phải rất lớn khoảng 256Kbyte tuy nhiên trên thực tế, các giao thức TCP chỉ độ rộng tối đa 64Kbyte và thông thường là 32Kbyte vì vậy không thể làm việc với độ rộng cửa sổ lớn như vậy. Qua thí nghiệm cho thấy với khích thước cửa sổ lớn thì các giao thức TCP thông thường cũng không được cải thiện, thậm chí còn làm ảnh hưởng đến hiệu suất truyền vì có thời gian Timeout quá lớn; qua thí nghiệm với giao thức TCP chuẩn, ta thấy có thể làm việc tốt nhất với độ rộng cửa sổ là 32 Kbyte và chính vì thế mặc dù độ rộng kênh là rất lớn song giới hạn thông lượng chỉ là 256Kbps.

Với việc ứng dụng các giao thức TCP mở rộng: Tahoe-TCP, Reno-TCP, Newreno-TCP và Sack-TCP trên kênh vệ tinh không có lỗi, kích thước gói tin cố định là 1024Byte, độ rộng cửa sổ cố định 32Kbps và kênh truyền là đồng bộ có độ rộng 2Mbps:

- Một lần nữa khẳng định giới hạn thông lượng TCP khi không áp dụng biện pháp để có thể làm việc với cửa sổ phát lớn là 256Kbps.
- Khi không có lỗi trên kênh vệ tinh các giao thức TCP mở rộng kể trên làm việc như nhau. Và cũng qua đó thấy được thời gian để thuật toán bắt đầu chậm làm việc để đạt thông lượng cao nhất phải mất đến >6.5giây. Và như tốc độ sẽ tăng rất chậm mỗi khi tắc nghẽn xảy ra và thuật toán bắt đầu chậm phải thực hiện.

Khi kênh vệ tinh có tỷ lệ lỗi 2% gói tin truyền qua:

- Có thể thấy được hiệu suất làm việc khác nhau của các giao thức TCP mở rộng và có thể so sánh hiệu suất làm việc của các giao thức TCP mở rộng trong điều kiện làm việc cụ thể này.
- Cũng qua đó thấy được khi tắc nghẽn xảy ra, thuật toán bắt đầu chận phải thực hiện để giải quyết tránh tắc nghẽn tiếp theo. Nhưng đồng nghĩa với việc hiệu suất làm việc giảm đi một cách đáng kể; Lúc này hiệu suất chỉ còn khoảng 32Kbps và mức độ cải thiện giữa các giao thức TCP mở rộng không lớn.

Tiếp cận với một số phương pháp cải tiến hiệu suất trên các kênh có độ trễ lớn, một số biện pháp được đề cập đến như chia cắt kết nối đầu cuối - đầu cuối thành nhiều đoạn TCP mà trong đó với đoạn liên kết vệ tinh có thể sử dụng một giao thức TCP

mở rộng làm việc được với cửa sổ lớn. Khi đó hiệu suất trên kênh vệ tinh được cải thiện đáng kể.

Một tiếp cận khác là TCP-Spoofing: có nghĩa một trạm Spoofing được thiết lập trên trạm mặt đất chiều phát nhằm sớm trả lời trạm phát rằng ACK song tự nó tiếp tục quản lý việc phát gói tin đến trạm thu. Với hình thức này sẽ làm cho thuật toán bắt đầu chậm tăng tốc nhanh hơn để đạt được thông lượng cao nhất và vì thế cải thiện được hiệu suất.

Cả hai biện pháp tiếp cận trên đã được nhiều công trình nghiên cứu và công bố kết quả. Trong bài luận văn này chỉ đề cập một cách tóm tắt để thấy được một số biện pháp cải tiến.

Một biện pháp nữa đã được nghiên cứu áp dụng để cải tiến hiệu suất của kênh vệ tinh đó là sử dụng trạm Snoop. Snoop thực chất được nghiên cứu và phát triển để nâng cao chất lượng hiệu suất của mạng không dây (WireLess) nói riêng và mạng có tỷ lệ lỗi kênh truyền lớn nói chung. Trạm Snoop là trạm giải quyết vấn đề thu nhận gói tin từ trạm phát và quản lý việc phát lại nếu phát hiện lỗi, vì thế giao thức TCP sẽ hạn chế phát hiện thấy tắc nghẽn và vì thế thuật toán bắt đầu chậm ít phải thực hiện. Cũng nhờ đó, hiệu suất được cải thiện:

- Trong phần này bài viết có trình bày chi tiết về sự làm việc của thuật toán Snoop và sự triển khai trong mô hình mạng vệ tinh.
- Với trạm Snoop được triển khai cũng thực hiện thử nghiệm với 4 giao thức TCP mở rộng khi kênh vệ tinh có lỗi và không có lỗi: kết quả cho thấy trạm Snoop không ảnh hưởng đến các giao thức khi kênh không có lỗi. Nhưng khi kênh có lỗi thì các giao thức TCP có lối hành xử khác với kênh kết nối đầu cuối - đầu cuối. Trong đó Sack tỏ ra không phù hợp khi làm việc có trạm Snoop mặc dù vẫn được cải thiện hiệu suất.

Cuối cùng, so sánh cùng một giao thức TCP giữa hai mô hình đầu cuối - đầu cuối và có trạm Snoop. Ta quan sát thấy hiệu suất được cải thiện một cách đáng kể trong đó cải thiện cao nhất là Reno-TCP.

Ưu điểm của Snoop khi được áp dụng vào trạm mặt đất chiều phát lên vệ tinh đó chính là:

- Trong suốt với sự làm việc của tầng giao vận TCP: Tại trạm TCP phát và trạm TCP thu không phải thay đổi một chút ứng dụng nào. Điều này đặc biệt quan trọng bởi không phải dễ dàng để tất cả các máy tính khi làm việc với kênh vệ tinh

phải sử dụng một giao thức TCP mở rộng viết riêng nhằm tối ưu hoá hiệu suất. Trong khi đó việc áp dụng trạm Snoop chỉ tăng cường việc xử lý lỗi do kênh vệ tinh gây nên; vì vậy trạm Snoop không làm ảnh hưởng gì đến TCP.

- Quan trọng hơn cả là với mô phỏng thử nghiệm cho thấy tất cả các giao thức TCP đều được cải thiện về hiệu suất. Trong đó cải thiện cao nhất là Reno-TCP là 180% và thấp nhất là Sack-TCP là 132%

Nhược điểm của áp dụng trạm Snoop:

- Với các kết nối được mã hoá (IPSec), mã hoá đến cả tiêu đề TCP thì trạm snoop không thể giải quyết được, thậm chí không thể làm việc bởi bản chất Snoop được triển khai tại tầng Link xong vẫn quan tâm đến giá trị trong tiêu đề TCP. Lúc này tiêu đề TCP bị mã hoá, chỉ có trạm thu hoặc trạm phát mới có thể giải mã thì trạm Snoop không thể giải quyết được vấn đề.

Vấn đề chưa được giải quyết khi áp dụng trạm Snoop đó chính là trong trường hợp với giao thức Sack-TCP:

- Mặc dù vẫn cải thiện được hiệu suất của Sack-TCP song thực chất chỉ giải quyết được một nửa tính chất đó là khi không có gói dữ liệu trả lời và timeout.
- Với gói trả lời yêu cầu phát lại có lựa chọn (SACK) thì trạm Snoop không giải quyết được và chuyển tiếp để TCP xử lý và lúc này hiện tượng tắc nghẽn có thể xảy ra.
- Vì vậy, để giải quyết cần phải xử lý ngay SACK tại trạm Snoop. Tuy nhiên vấn đề này chưa được đề cập đến trong thuật toán Snoop. Tôi hy vọng vấn đề này được xem xét và nghiên cứu khả năng giải quyết vấn đề này.

Trên đây là tóm tắt bài luận văn, kết quả đạt được cũng như hướng để có thể tiếp tục nghiên cứu trong việc cải thiện hiệu suất TCP trên kênh viễn thông vệ tinh.

Đề tài nghiên cứu này là một đề tài có phạm vi rộng, khó có thể kiểm chứng bằng thực tiễn vì hiện nay Việt Nam cũng đang còn trong giai đoạn chưa có Vệ tinh truyền thông riêng. Vì vậy, trong quá trình nghiên cứu, còn nhiều vấn đề chưa đề cập đến cũng như đề cập chưa chi tiết. Tôi mong nhận được sự đóng góp và hợp tác của các chuyên gia để có thể bổ sung chỉnh sửa và hoàn thiện thêm nội dung bài viết, nhằm đưa ra một kết quả có tính thực tiễn cao.

Bài viết chắc chắn còn có nhiều thiếu sót, tôi mong nhận được thông cảm và góp ý để tôi tiếp tục chỉnh sửa và hoàn thiện bài viết này.

Tôi hy vọng bài viết là một phần nghiên cứu đóng góp cho việc điều khiển lưu lượng, tránh tắc nghẽn trên kênh vệ tinh nói riêng và đóng góp vào việc nâng cao hiệu suất các kết nối mạng Internet nói chung.

Một lần nữa, tôi trân trọng cảm ơn sự giúp đỡ và đóng góp ý kiến để tôi hoàn thành bản luận văn này.

TỪ VIẾT TẮT

Từ viết tắt	***Ý nghĩa***
ACK	Acknowledgement
BER	Bits Error Rate
CWND	Congestion Windows
E2E	End to End
GEO	Geostationary Orbit satellites
IP	Internet Protocol
ISI	Information Sciences Institute
ISO/OSI	International Standards Organization / Open Systems Interconnect
LAN	Local Area Network
LBNL	Lawrence Berkeley National Laboratory
LEO	Low Earth Orbits satellites
MAC	Medium access control
MSS	Maximum Segment Size
NAK	No Acknowledgement
NAM	Network Animator
NS	Network Simulator
RTT	Route Trip Time
SACK	Selective Acknowledgement
SEQ	Sequence
TCP	Transport Control Protocol

Bảng 10: Từ viết tắt

TÀI LIỆU THAM KHẢO

[1] Thomas Ross Henderson and Randy H.Katz - TCP Perfomments over Satellite Channnels - 1998

[2] Thomas Ross Henderson - Networking over Neext-Generation Satellite systems - 1999

[3] Geostationary, LEO, MEO, HEO Orbits Including Polar and SunSynchronous Orbits with Example Systems. Truy nhập 5/2003: http://www.geo-orbit.org/sizepgs/geodef.html.

[4] J. Hoe, Startup Dynamics of TCP's Congestion Control and Avoidance Schemes. Master's Thesis, MIT, 1995.

[5] H. Balakrishman, V.Padmanabhan, and R. Katz. The Effects of Asymmetry on TCP Perfomance. Processdings of Third ACM/IEEE MobiCom Conference, page 77-89, September 1997

[6] Matthew Mathis and Jamshid Mahdavi. "Forward Acknowledgement: Refining TCP Congestion Control". SIGCOMMSymposium on Communications Architectures and Protocols, Aug. 1996. to appear.

[7] RFC2760 - Ongoing TCP Research Related to Satellites - 2000

[8] Kenny, Qing Shao, Grace, Hui Zhang - TCP performance analysis over satelite links - 2002

[9] Vũ Duy Lợi - Mạng thông tin máy tính, Kiến trúc, nguyên tắc và hiệu suất hoạt động, 2001.

[10] Yavuz Fatih YAVUZ - Satelite communications and impact of TCP over satelite networks - 2000

[11] Nguyễn Hoàng Linh - "Điều khiển tắc nghẽn TCP" - 2002

[12] Performance Enhancing Proxy (PEP) Request for Comments: http://community.roxen.com/developers/idocs/drafts/draft-ietf-pilc-pep-04.html

[13] Improving TCP/TP Performance over Wireless Networks: http://www2.cs.cmu.edu/~srini/Papers/publications/1995.mobicom/mobicom95.pdf

[14] Jacobson, V., "Congestion Avoidance and Control", Computer Communication Review, vol. 18, no. 4, pp. 314-329, Aug. 1988. ftp://ftp.ee.lbl.gov/papers/congavoid.ps.Z

[15] Stevens, W., "TCP/IP Illustrated, Volume 1: The Protocols", Addison-Wesley, 1994.

[16] Bùi Quang Hưng, "Nghiên cứu công cụ mô phỏng phân tích đánh giá hiệu suất các hệ thống mạng thông tin máy tính NS", 2001.

[17] Jacobson, V., "Modified TCP Congestion Avoidance Algorithm", end2end-interest mailing list, April 30, 1990. ftp://ftp.isi.edu/end2end/end2end-interest-1990.mail.

[18] Mathis, M., Mahdavi, J., Floyd, S. and A. Romanow, "TCP Selective Acknowledgement Options", RFC 2018, October 1996.

[19] S. Floyd. "Issues of TCP with SACK,".Technical report, Mar. 1996. URL ftp://ftp.ee.lbl.gov/papers/issues sa.ps.Z.

[20] S. Floyd. "SACK TCP: The sender's con-gestion control algorithms for the implementation "sack1" in LBNL's "ns" sim-ulator (viewgraphs).,".Technical re-port, Mar. 1996.Presentation to the TCP Large Windows Working Groupof the IETF, March 7, 1996. URL ftp://ftp.ee.lbl.gov/talks/sacks.ps

[21] J. Hoe. "Start-up Dynamics of TCP's Con-gestion Control and Avoidance Schemes"

[22] D.D. Clark and J. Hoe. "Start-up Dynamicsof TCP's Congestion Control and Avoid-ance Schemes,". Technical report, Jun.1995. Presentation to the Internet End-to-End Research Group, cited for acknowl-edgement purposes only.

[23] S. Keshav. "Packet-Pair Flow Control,".Technical report, Nov. 1994. Presenta-tion to the Internet End-to-End ResearchGroup, cited for acknowledgement purposes only

PHỤ LỤC

Mã nguồn chương trình mô phỏng

```
# Nguyen Tue Linh
# K7T Khoa cong nghe thong tin - Dai hoc Quoc gia - Ha Noi
# Dien thoai: 0913505995
# Email: TueLinh@vps.com.vn
#
# Topo thu nghiem End to End
#
# FTP
# TCP/tcpvar       LL                                                          TCPSink
# (nsc) ----------(repeter-|-nsg1) -------------(ngeo)-------------(nsg2)------------(nds)
#      10Mb,2ms          100Mb       2Mb,250ms           2Mb,250ms          10Mb,2ms
#                         0ms      <-> lost 0.02       <-> lost 0.02
#
#
#
# Topo thu nghiem Snoop
#
# FTP
# TCP/tcpvar        LL/LLSnoop                                                  TCPSink
# (nsc) -----------(snoop-|-nsg1) --------------(ngeo)-------------(nsg2)------------(nds)
#      10Mb,2ms          100Mb       2Mb,250ms           2Mb,250ms          10Mb,2ms
#                         0ms      <-> lost 0.02       <-> lost 0.02
#

global opt
set opt(qsize)          100
set opt(bw)             1000Mb
set opt(delay)          0ms
set opt(ll)             LL
set opt(ifq)            Queue/DropTail
set opt(mac)            Mac/802_3
set opt(chan)           Channel


Class TestSimpleRep
TestSimpleRep instproc init {} {
    $self instvar ns_ nsc nsg1 ngeo nsg2 nds tcp1 f0 f1 nf tcpvar tcptyp cwndfile seqfile
    set ns_ [new Simulator]
    # $ns_ rtproto Dummy

    global argc argv argv0

    switch $argc {
      1 {
              set test $argv
              set test1 "E2E"
        }
      2 {
              set test [lindex $argv 0]
              set test1 [lindex $argv 1]
              if {($test1 != "Snoop")} {
                  puts "Error!"
                  exit 0
              }
        }
      default {
        puts "*********************************************************************************"
        puts "                                HUONG DAN SU DUNG"
        puts " ns $argv0 <tcp variant> <tcp type>"
        puts "   <tcp variant>: Reno, Newreno, Sack1, SackRH, Vegas, Fack, Asym, RFC793edu"
        puts "   <tcp type>   : Snoop"
        puts "                  Rong: E2E"
        puts ""
        puts "Nguyen Tue Linh - K7T - Cong nghe thong tin - Dai hoc Quoc gia - Ha Noi"
        puts "*********************************************************************************"
        # "Usage: ns $argv0 <tcp variant>"
      exit 0
       }
    }
```

```
    set tcpvar $test
    set tcptyp $test1

    # Mo cac file luu ket qua
    # Do rong cua so

    set cwndfile "$tcptyp cwnd."
    lappend cwndfile $tcpvar
    set f0 [open $cwndfile w]

    set seqfile "$tcptyp seq."
    lappend seqfile $tcpvar
    set f1 [open $seqfile w]

    set nf [open $tcptyp$tcpvar.nam w]
    $ns_ namtrace-all $nf
}


TestSimpleRep instproc run {} {
$self instvar ns_ nsc nsg1 ngeo nsg2 nds nsnoop tcp1 f0 nf tcpvar tcptyp
puts "Begin of Test!"

global opt

# dinh nghia cac node
set nsc [$ns_ node]
set nsnoop [$ns_ node]
set nsg1 [$ns_ node]
set ngeo [$ns_ node]
set nsg2 [$ns_ node]
set nds [$ns_ node]


# Tao node repeter hoac snoop
lappend nodelist $nsg1
set lan [$ns_ make-lan $nodelist $opt(bw) \
                       $opt(delay) $opt(ll) $opt(ifq) $opt(mac) $opt(chan)]
if {$tcptyp == "Snoop"} {
        set opt(ll) LL/LLSnoop
        set opt(ifq) $opt(ifq)
        $opt(ifq) set limit_ 1000
}
$lan addNode [list $nsnoop] $opt(bw) $opt(delay) $opt(ll) $opt(ifq) $opt(mac)


# Lien ket cac node
$ns_ duplex-link $nsc $nsnoop 10Mb 2ms DropTail
$ns_ duplex-link-op $nsc $nsnoop orient right
$ns_ queue-limit $nsc $nsnoop $opt(qsize)

$ns_ duplex-link $nsg1 $ngeo 2Mb 250ms DropTail
$ns_ duplex-link-op $nsg1 $ngeo orient right
$ns_ queue-limit $nsg1 $ngeo $opt(qsize)

$ns_ duplex-link $ngeo $nsg2 2Mb 250ms DropTail
$ns_ duplex-link-op $ngeo $nsg2 orient right
$ns_ queue-limit $ngeo $nsg2 $opt(qsize)

$ns_ duplex-link $nsg2 $nds 10Mb 2ms DropTail
$ns_ duplex-link-op $nsg2 $nds orient right
$ns_ queue-limit $nsg2 $nds $opt(qsize)

# Tao loi ngau nhien tren kenh truyen
set loss_model(1) [new ErrorModel/Uniform 0.02 pkt]
$ns_ lossmodel $loss_model(1) $nsg1 $ngeo
$loss_model(1) drop-target [new Agent/Null]

set loss_model(2) [new ErrorModel/Uniform 0.02 pkt]
$ns_ lossmodel $loss_model(2) $ngeo $nsg2
$loss_model(2) drop-target [new Agent/Null]

set loss_model(3) [new ErrorModel/Uniform 0.02 pkt]
```

```
$ns_ lossmodel $loss_model(3) $ngeo $nsg1
$loss_model(3) drop-target [new Agent/Null]

set loss_model(4) [new ErrorModel/Uniform 0.02 pkt]
$ns_ lossmodel $loss_model(4) $nsg2 $ngeo
$loss_model(4) drop-target [new Agent/Null]


# Tao TCP, va tao luu luong tren ket noi
if {$tcpvar == "Tahoe"} {
        Agent/TCP set nam_tracevar_ true
        $self set tcp1 [new Agent/TCP]
}
if {$tcpvar != "Tahoe"} {
        Agent/TCP/$tcpvar set nam_tracevar_ true
        $self set tcp1 [new Agent/TCP/$tcpvar]
}

$tcp1 set eln_ 1
$tcp1 set windowInit_ 1

$ns_ attach-agent $nsc $tcp1

$ns_ add-agent-trace $tcp1 tcp
$ns_ monitor-agent-trace $tcp1
$tcp1 tracevar cwnd_
$tcp1 tracevar t_seqno_

set tcpsink1 [new Agent/TCPSink]
set sack "Sack1"
if [string match $tcpvar $sack] {
    puts "sack1"
    set tcpsink1 [new Agent/TCPSink/Sack1]
}

$ns_ attach-agent $nds $tcpsink1

set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1


# set cbr1 [new Application/Traffic/CBR]
# $cbr1 attach-agent $tcp1
$ns_ connect $tcp1 $tcpsink1


puts "at 0.0 record"
$ns_ at 0.0 "$self record"

puts "at 1.1 $ftp1 start"
$ns_ at 1.0 "$ftp1 start"

#puts "at 1.1 $cbr1 start"
#$ns_ at 1.0 "$cbr1 start"

puts "at 100.0 finish"
$ns_ at 100.0 "$self finish"
$ns_ run
# puts "exiting testSimpleRep run"
}

TestSimpleRep instproc record {} {
$self instvar ns_ tcp1 f0 f1
        #Set the time after which the procedure should be called again
        set time 0.5
        # current cwnd
        set bw0 [$tcp1 set cwnd_]
        set bw1 [$tcp1 set t_seqno_]
        #Get the current time
        set now [$ns_ now]

        # puts "$now $bw0"
        puts $f0 "$now $bw0"
```

```
        puts $f1 "$now $bw1"
        #Re-schedule the procedure
        $ns_ at [expr $now+$time] "$self record"
}

TestSimpleRep instproc finish {} {
$self instvar ns_ f0 nf tcpvar tcptyp cwndfile seqfile
        puts "Finishing..."
        $ns_ flush-trace
        close $f0
        close $nf
        puts "Filltering for nam..."
        exec tclsh /study/ns/nam-1.0a11a/bin/namfilter.tcl $tcptyp$tcpvar.nam
        puts "Test Ok, end of test!"
        # puts "running nam..."
        # exec nam out.nam &
        # exec xgraph $cwndfile &
        # exec xgraph $seqfile &
        exit 0
}

set trep [new TestSimpleRep]

$trep run

puts "after runtest"
ns sp.tcl Reno
ns sp.tcl Sack1
ns sp.tcl SackRH
ns sp.tcl Newreno
ns sp.tcl Vegas
ns sp.tcl Fack
ns sp.tcl Asym
ns sp.tcl RFC793edu

# xgraph cwnd.\ SackRH cwnd.\ Sack1 cwnd.\ Reno cwnd.\ Newreno cwnd.\ Vegas cwnd.\ Fack cwnd.\
Asym cwnd.\ RFC793edu &
# xgraph seq.\ SackRH seq.\ Sack1 seq.\ Reno seq.\ Newreno seq.\ Vegas seq.\ Fack seq.\ Asym
seq.\ RFC793edu &
```