TCP CUBIC và Ứng dụng trong Lớp truyền vận

Đỗ Quang Huy

Mã số sinh viên: 20172603. Viện Điện tử - Viễn thông, Trường Đại học Bách Khoa Hà Nội, Hà Nội, Việt Nam

Email: huy.dq172603@sis.hust.edu.vn

Abstract - Transmission Control Protocol TCP đã trở thành công nghệ điều khiển luồng và chống tắc nghẽn nổi tiếng và được sử dụng rộng rãi nhất cho đến thời điểm hiện tại. Với phiên bản mới nhất, TCP CUBIC, được tích hợp mặc định trong lõi của những hệ điều hành phổ biến nhất thế giới là Linux và Windows, nó đã giúp ích rất nhiều khi triển khai trên các hệ thống Long Fat Network LFN, các hệ thống đòi hỏi có tốc độ cao (dù đôi khi phải đánh đổi bằng độ trễ). Bản báo cáo này sẽ trình bày những hiểu biết về mối liên hệ của TCP và lớp truyền vận, tìm hiểu về thuật toán nằm trong TCP CUBIC, mô phỏng một bài toán và cuối cùng là đánh giá thông lượng hệ thống đó. Cuối cùng, kết luận ngắn gọn là TCP CUBIC thích hợp với các mạng hữu tuyến, có tốc độ cao; cùng với đó là hiệu suất giảm rõ rệt với các mạng vô tuyến hoặc tốc độ thấp.

Keywords: TCP, TCP CUBIC, điều khiển luồng.

Hà Nội, ngày 20/12/2020

I. INTRODUCTION

Trong thời buổi lưu lượng thông tin ngày một tăng theo cấp số nhân, làm cách nào để có thể bảo trì và nâng cấp hệ thống (về mặt phần cứng) một cách tiết kiệm và hợp lý nhất? Thế giới đã từng chứng kiến sự kiện mạng Internet "sập" hàng loạt vào năm 1988 và kêt tử đó, người ta ngày càng quan tâm đến điều khiển luồng và phòng chống tắc nghẽn. Ngoài các giải pháp thiên về phần cứng thì các giải pháp phần mềm, về thuật toán là vô cùng quan trọng. TCP là một trong số đó. Đây là bộ giao thức điều khiển luồng được sử dụng rộng rãi nhất tính đến thời điểm hiện tại và đã trải qua rất nhiều các phiên bản khác nhau, cung cấp các khả năng truyền tin tốt và đáng tin cậy. Ý tưởng chính của giao thức này là phương pháp cửa sổ trượt được đề cập đến trong nội dung của chương 2.

Phiên bản mới nhất của bộ giao thức TCP kết hợp với giải thuật CUBIC cải thiện đáng kể khả năng "cảm nhận" của hệ thống – là khả năng tăng hay giảm kích thước cửa số trượt theo hàm bậc 3 – một ý tưởng tuy đơn giản nhưng vô cùng độc đáo, sẽ được phân tích trong chương 3.

Về mặt tổng quan, báo cáo gồm 4 chương chính. Phần "Background and Related work" giới thiệu về lớp truyền vận và vấn đề điều khiển luồng. Chương 2 giới thiệu TCP CUBIC trên cả lý thuyết và thực tiễn. Chương 3 phân tích thuật toán TCP CUBIC. Cuối cùng, chương 4 là triển khai (mô phỏng) và đánh giá.

II. BACKGROUND AND RELATED WORK

A. Lớp truyền vận

Mô hình OSI (Open Systems Interconnection Reference Model) tổ chức các giao thức truyền thông thành 7 lớp, mỗi lớp giải quyết một phần hẹp của tiến trình truyền thông. Trong mỗi lớp có thể có nhiều giao thức khác nhau thực hiện các nhu cầu truyền thông cụ thể. Lớp truyền vận hay tầng giao vận là lớp thứ 4 trong 7 lớp của hệ thống OSI.

Lớp truyền vận là nơi các thông điệp được chuyển thành dạng các gói tin nhỏ trước khi gửi đi, đánh số các gói tin và đảm bảo chúng chuyển theo đúng thứ tự, lớp này cũng cung cấp khả năng truyền dữ liệu đáng tin cậy và hiệu quả. Lớp truyền vận kiểm soát độ tin cậy của một kết nối được cho trước bằng cách theo dõi trạng thái truyền của các gói tin và truyền lại các gói bị truyền thất bại. Một ví dụ điển hình của giao thức lớp 4 là TCP (Transmission Control Protocol).

Đây là lớp cao nhất liên có liên quan đến các giao thức trao đổi dữ liệu giữa các hệ thống mở, kiểm soát việc truyền dữ liệu từ nút tới nút (End-to-End). Các giao thức trong 3 lớp dưới (vật lý, liên kết dữ liệu và mạng) chỉ phục vụ việc truyền dữ liệu giữa các lớp kề nhau trong từng hệ thống, trong khi đó, lớp truyền vận điều khiển cả việc truyền thông giữa hệ thống này và hệ thống khác. Các thực thể đồng lớp sẽ trao đổi với nhau và điều chỉnh thông lượng hoặc gửi lại gói tin trong quá trình truyền dữ liệu.

Như vậy, lớp truyền vận có tác dụng vận chuyển thông tin giữa các máy, đồng thời kiểm soát lỗi và luồng dữ liệu.

B. Điều khiển luồng và chống tắc nghẽn trong lớp truyền vận

Bài toán cần giải quyết của mọi hệ thống truyền thông nói chung là cân bằng giữa độ tin cậy và giá thành, có nghĩa là đảm bảo hệ thống đạt hiệu suất cao nhất có thể trong tài nguyên giới hạn. Do đó, việc xảy ra tắc nghẽn, mất gói tin là không thể tránh khỏi. Nói cách khác, bài toán cần giải quyết là trên một đường truyền có hạn, làm sao để phân bổ tài nguyên hợp lý giữa các user. Vì vậy, chúng ta cần những giải pháp để điều khiển luồng và chống tắc nghẽn.

Chống tắc nghẽn là cơ chế kiểm soát thông tin đi vào mạng nhằm đảm bảo tổng lượng thông tin trong mạng không vượt quá khả năng xử lý của toàn mạng. Như vậy, đối tượng trọng tâm của điều khiển luồng là bên phát – bên thu (cụ thể) còn đối tượng của chống tắc nghẽn là toàn mạng. Do có nhiều điểm chung và gắn bó hữu cơ mật thiết với nhau, trong các phần sau, chúng được sử dụng chung dưới tên là điều khiển luồng.

Mục đích chính của các cơ chế điều khiển luồng là tối ưu hóa thông lượng của mạng và giảm độ trễ khi đi qua mạng và để thực hiện được điều đó thì phương pháp hiện tại được sử dụng rộng rãi nhất tính tới thời điểm hiện tại là phương pháp cửa sổ.

Theo phương pháp cửa sổ, bên phát sẽ không phát tin chừng nào bên thu chưa xử lý xong các gói tin trước đó. Khi bên thu xử lý xong gói tin, nó sẽ báo cho bên phát biết và

lúc này, bên phát sẽ tiếp tục gửi các gói tin tiếp theo. Cơ chế này đảm bảo việc truyền tin không bao giờ vượt quá khả năng xử lý của bên thu do có sự phối hợp nhịp nhàng giữa cả hai bên. Như vậy, tổng số gói tin trong mạng cũng được đặt trong một giới hạn nhất định, đảm bảo khả năng hoạt động tốt của toàn hệ thống.

Nhờ có các giải pháp điều khiển luồng trên mà mạng Internet đã phát triển vượt bậc, trở thành một trong những nền tảng của mọi khía cạnh đời sống con người. Không chỉ vậy, chúng đã giải quyết phần nào bài toán hóc búa dành cho các nhà phát triển trong việc phân bổ nguồn lực viễn thông một cách tối ưu và hiệu quả trên một đường truyền hữu hạn.

III. TCP CUBIC

A. TCP

Để các máy tính (hoặc các node) có thể liên lạc với nhau trong mạng, chúng cần sử dụng chung một "ngôn ngữ" hay một loại giao thức (protocol). Có thể nói, giao thức là một hệ thống các quy tắc nhằm giúp các node có thể liên lạc với nhau qua lớp truyền vận. Hiện nay, giao thức TCP/IP (Transmission Control Protocol/Internet Protocol Suite) đang được sử dụng cho hầu hết các hệ thống mạng. Thực tế thì TCP/IP không chỉ hai mà là một tập hợp của nhiều giao thức khác nhau, chúng tạo thành bộ giao thức (Suite of Protocol) TCP/IP. Trong đó, TCP là thành phần cốt lõi của bộ giao thức trên, hoạt động hướng tới điều khiển luồng.

TCP cung cấp các kênh truyền thông hướng kết nối và đảm bảo dữ liệu truyền tin cậy. Như đã nói trong phần "Điều khiển luồng và chống tắc nghẽn" (mục B, II. Background and Related work), các thuật toán điều khiển luồng không chỉ phân chia thông điệp thành các gói tin (mà trong thực tế ở TCP là các đoạn hay segment, tuy nhiên trong báo cáo thống nhất gọi là gói tin), điều khiển khi nào bên phát gửi gói tin, xử lý lỗi, đánh số thứ tự mà còn điều chỉnh cả thông lượng (là số lượng gói tin gửi trong một giây), TCP không chỉ có cơ chế kiểm soát khả năng sẵn sàng của bên thu (thông qua bản tin phúc đáp ACK) mà còn có các thuật toán để điều khiển thông lượng hệ thống (như Slow – start kết hợp AIMD - Additive-increase/multiplicative-decrease hay CUBIC). TCP thường dùng để truyền các gói tin có kích thước lớn và yêu cầu xác nhận mỗi khi bên thu nhận được gói tin.

Về một số đặc điểm khác:

- TCP cho độ tin cậy cao: Do TCP đánh số thứ tự các gói tin và kiểm soát lỗi bằng cách yêu cầu bên thu gửi phản hồi cho bên phát [1].
- Là hệ thống end-to-end (thông qua mạng internet): Đây là nhiệm vụ được thiết đặt cho tầng giao vận – truyền thông trực tiếp mà không phụ thuộc vào các lớp bên dưới [2].
- Tốc độ không cao: so sánh với UDP (User Datagram Protocol), TCP cho thông lượng hệ thống thấp hơn rõ rệt. Nguyên nhân là để đáp ứng mục tiêu truyền thông tin cậy, gói tin TCP có cấu trúc phức tạp nên cần nhiều thời gian xử lý hơn, và đặc biệt do yêu cầu bên thu gửi phản hồi cho bên phát, các gói tin sẽ không được gửi cho đến khi được xác nhận đã truyền thành công. Điều này tạo ra nhiều khoảng thời gian chết trong khi truyền, dẫn đến thông lượng hệ thống giảm rõ rệt.

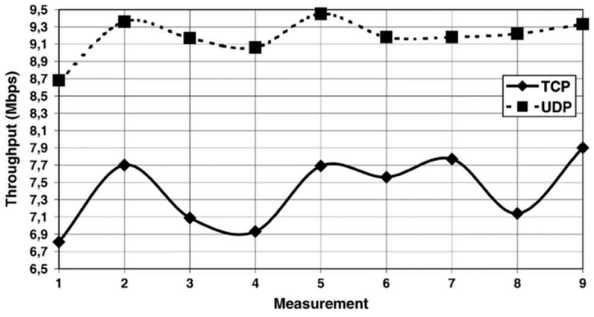

Fig. 1. So sánh thông lượng TCP và UDP [3]

- Một điểm quan trọng là đa số các thuật toán TCP dựa trên thuật toán Slow – start. Slow – start là thuật toán giúp cân bằng lượng dữ liệu bên phát có thể phát và bên thu có thể thubằng cách thiết lập kích thước cửa sổ trượt tăng theo một qui tắc nào đó, ví dụ như hàm mũ hoặc cấp số cộng (AIMD) sau mỗi thời gian trễ trọn vòng (RTT – Round trip time), cho đến khi đạt tới giới hạn của bên thu. Đặc điểm của thuật toán này là tốc độ truyền tin đáp ứng theo điều kiện của bên thu và cả hệ thống.

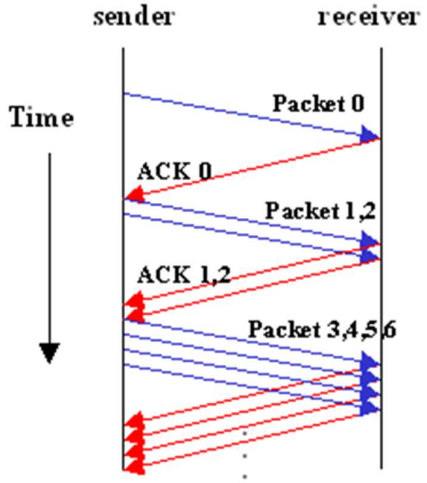

Fig. 2. Mô tả thuật toán Slow – start trong TCP. [5]

- Cuối cùng, về mặt gói tin, TCP cung cấp khẳn năng truyền tin theo thứ tự tốt tự động loại bỏ các gói lặp.

B. Một số định nghĩa quan trọng

Trước khi phân tích sâu hơn vào thuật toán điều khiển luồng của TCP CUBIC, ta cần cụ thể hóa và phân tích một số khái niệm của TCP và TCP CUBIC về mặt kí hiệu và ứng dụng.

1) Cửa sổ tắc nghẽn - cwnd

cwnd (congestion window) là ý tưởng quan trọng nhất của trong cơ chế điều khiển luồng của TCP nói chung và TCP CUBIC nói riêng. Đây là một biến có tác dụng giới hạn lượng dữ liệu bên phát có thể đưa vào mạng trước khi nhận được bản tin ACK. Giá trị của biến này được thiết đặt bởi bên phát. Ngoài ra, trong TCP còn có rwnd – Receiver window là biến đại diện cho lượng dữ liệu bên thu có thể nhận [6].

Hai biến trên được sử dụng cùng nhau để điều khiển luồng và cải thiện hiệu suất cho mạng. Tuy nhiên, trong

phạm vi của đề tài này chủ yếu phân tích yếu tố cwnd và ảnh hưởng của nó đến hiệu năng mạng. Trong các phần sau, cwnd có thể được kí hiệu là W khi đặt trong công thức.

2) ssthresh

ssthresh (Slow – start threshold) xác định khi nào hệ thống mạng bị quá tải và cần sự can thiệp điều khiển luồng để giảm lượng dữ liệu gửi đi và đây không phải là một giá trị cố định. Giá trị của ssthesh giảm và tăng tùy thuộc vào trạng thái tắc nghẽn, và thường là trung bình của kích thước cửa sổ tối thiểu và kích thước tới hạn [7].

3) Mối liên hệ giữa cwnd và ssthresh

Ta có thể coi RTT như một chu kỳ, ở TCP nguyên bản, trong mỗi chu kỳ, thuật toán Slow – start được áp dụng khi cwnd < ssthesh. Khi xảy ra tắc nghẽn, tức cwnd > ssthresh, các thuật toán điều khiển luồng như CUBIC được áp dụng.

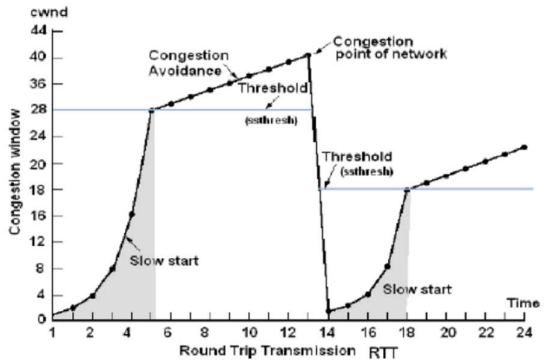

Fig. 3. Liên hệ cwnd và ssthresh [8].

4) Phương thức hoạt động TCP

Pha *: Thiết lập kết nối giữa hai máy, mô tả trong hình…:

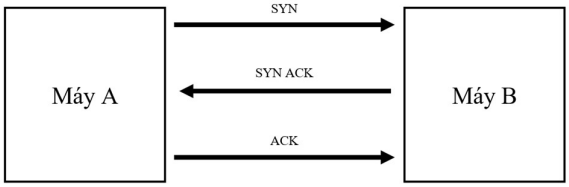

Fig. 4. Mô tả hoạt động thiết lập kết nối giữa hai máy A và B thông qua bộ 3 bản tin SYN – SYN ACK – ACK.

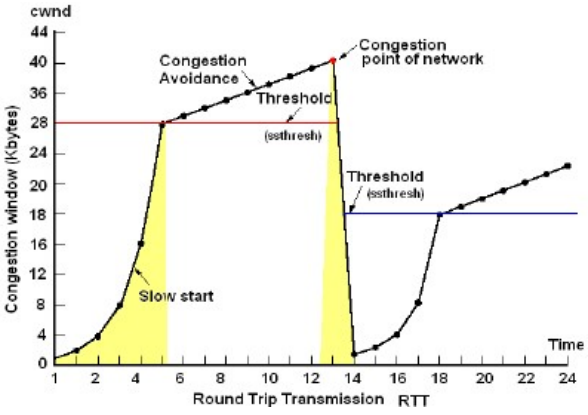

Fig. 5. Mô tả các pha của TCP [8]

Pha 1: Khi truyền dữ liệu:

Trong phần lớn các công nghệ TCP, quá trình bắt đầu với Slow – start. Ngay khi kết nối được thiết lập, kích thước cửa sổ trượt (cwnd) tại A được thiết lập bằng 1, sau mỗi bản tin ACK từ B phản hồi lại, kích thước cửa sổ trượt được thiết lập cwnd = cwnd + 1 và tăng lên gấp đôi sau mỗi RTT.

Pha 2: Phòng chống tắc nghẽn:

Trong giai đoạn này, khi cwnd > ssthresh, cwnd tăng chậm hơn so với quá trình trước, công thức độ tăng tùy thuộc vào loại công nghệ TCP. Ở TCP nguyên bản, sau mỗi RTT, cwnd = cwnd +1 cho đến khi xảy ra mất gói. Khi đó, chuyển sang pha 3, truyền lại nhanh chóng (Fast Retransmit)

Pha 3: Truyền lại nhanh chóng (Fast Retransmit).

Ta tự hỏi làm thế nào để 2 hệ thống thông tin độc lập có thể xác nhận mất gói tin dù không cần thông báo "Gói tin thứ i bị mất"? Đó là nhờ giải thuật Duplicate ACK. Sau khi nhận được một gói tin ACK(i) lặp, hệ thống không thể kết luận gói tin i bị mất, nhưng sau một số ACK lặp (cụ thể là 3) thì có cơ sở để khẳng định đã xảy ra mất gói – thuật toán này gọi là Three Duplicate ACK. Nói cách khác, ngay khi nhận thấy Three Duplicate ACK, không cần chờ gói tin bị timeout, hệ thống tự động gửi lại gói tin thứ i bị lỗi và đặt cwnd = 1 với TCP nguyên bản.

Sau khi nhận ra lỗi gói và truyền lại, hệ thống chuyển nhanh sang pha Nhanh chóng phục hồi (Fast Recovery), tuy nhiên đây chỉ là một tùy chọn mở rộng.

Pha 4: Phát hiện tắc nghẽn (Congestion Detection)

Với TCP nguyên bản, khi xảy ra mất gói, cwnd được tự động trở về giá trị 1 (Fast Retransmit), là bước đầu tiên của slow - start, câu hỏi là có cần thiết phải giảm đột ngột tốc độ gửi về mức tối thiểu không trong khi kích thước cửa sổ của TCP là rất lớn và hiện tượng mất gói ít khi xảy ra? Điều đó là không cần thiết và làm giảm hiệu suất hệ thống khi mất thời gian tăng trở lại lên giá trị ssthresh. Các công nghệ mở rộng của TCP cho phép khi gặp lỗi gói, chu trình không quay về điểm bắt đầu mà chuyển về pha Phòng chống tắc nghẽn.

Để làm được điều đó, ngay từ pha Truyền lại nhanh chóng (Fast Retransmit), ngay khi gói tin i bị lỗi được A gửi lại, nó sẽ ngay lập tức thiết lập lại giá trị: ssthresh = cwnd / 2, cwnd = ssthresh. Như vậy, hệ thống đã trở lại pha Phóng chống tắc nghẽn.

Pha **: Delayed ACK

Càng về sau, với lượng thông tin được gửi đi rất lớn, người ta thấy rằng không nhất thiết phải có bản tin ACK cho mọi gói tin gửi đi. Cho nên, để tiết kiệm tài nguyên hệ thống, sau một số gói gửi đi mới có 1 ACK phúc đáp. Bản tin ACK loại này gọi là Delayed ACK.

Pha ***: Ngắt kết nối

Quá trình ngắt kết nối tương tự quá trình kết nối, được biểu diễn trong Fig6.

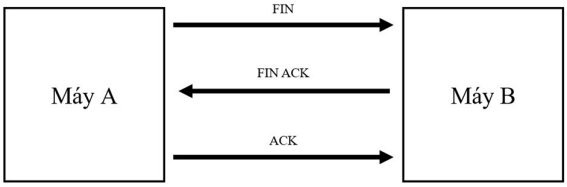

Fig. 6. Mô tả hoạt động thiết lập kết nối giữa hai máy A và B thông qua bộ 3 bản tin FIN – FIN ACK – ACK

5) Phân loại các thuật toán TCP

Như đã nói ở các mục trước, TCP được phân chia dựa theo thuật toán điều khiển tắc nghẽn, được gọi là các tiêu chuẩn TCP (TCP standards) [10] cơ bản chia làm 4 loại như sau:

Phân loại	Các công nghệ TCP tương ứng
Dựa trên mất gói tin (Loss-based), phụ thuộc bản tin ACK (Acknowledge – bản tin phúc đáp).	TCP Reno, TCP Cubic
Dựa trên trễ gói tin (Delay-based), phụ thuộc thời gian trễ trọn vòng (RTT).	TCP CDG (CAIA Delay-Gradient)
Phương pháp kết hợp với đánh giá băng thông.	Westwood +
Phương pháp kết hợp nhưng không sử dụng đánh giá băng thông.	Illinois

Fig. 7. Phân loại một số thuật toán sử dụng trong TCP [4].

Nhìn vào Fig7, ta thấy rằng đặc điểm chung của của cách phân loại nêu trên là chọn điểm dừng (exit point) dựa vào bản tin ACK hay RTT. Trong đó, điểm quan trọng của TCP Cubic chính là dựa vào thông tin của bản tin ACK để tăng hay giảm tốc độ phát gói tin của bên phát.

Thấy rằng, chính vì TCP CUBIC độc lập với RTT là khoảng thời gian cố định, TCP CUBIC cho phép hệ thống mở rộng tốt hơn rất nhiều so với các công nghệ khác. Tuy nhiên, điều này mang đến một nhược điểm lớn là với các đường truyền vô tuyến, với nhiễu và các yếu tố môi trường làm cho việc gửi các gói tin dài gặp nhiều bất lợi, đã hạn chế đáng kể hiệu năng TCP CUBIC với các mạng vô tuyến.

6) TCP Cubic

a) Cơ chế

Theo mục II.4. Phương thức hoạt động của TCP, ta đã nắm được các pha hoạt động của TCP và TCP có nhiều tiêu chuẩn TCP để tối ưu hóa hoạt động trên, một trong số đó là TCP CUBIC. TCP CUBIC phát huy tác dụng của mình ở pha "Truyền lại nhanh chóng".

TCP CUBIC thuộc vào loại thuật toán TCP dựa trên mất gói tin (Loss-based) với cơ chế cốt lõi là tăng hay giảm cwnd theo mô hình hàm bậc 3 thay cho mô hình tuyến tính của TCP nguyên bản: với chiều tăng theo một hàm lồi và giảm theo một hàm lõm. Nhờ CUBIC, với tốc độ tăng lớn hơn TCP nguyên bản (do đạo hàm bậc 3 lớn hơn đạo hàm hàm tuyến tính của TCP nguyên bản) nên với TCP CUBIC đã cải thiện đáng kể khả năng áp dụng cho những mạng lớn và trên đường truyền dài, tốc độ cao (mạng Long Fat Network – LFN).

LFN là một trường hợp có Bandwidth Delay Product (BDP) cao. Trong bất kỳ hệ thống nào cũng có độ trễ kênh truyền nhất định, là khoảng thời gian cần thiết để thông tin đi từ bên phát sang bên thu. Như vậy, có nghĩa là trên toàn kênh truyền cũng có các bit chưa đến đích [9]. Lượng tin này được gọi là BDP, công thức tính là:

$$BDP = R * RTT \quad (2.1)$$

Với R là băng thông hay thông lượng (Mbps) là lượng tin lớn nhất mà kênh truyền có thể đáp ứng trong khoảng thời gian 1s và RTT là thời gian trễ trọn vòng (ms) hay thời gian để gói tin đến đích và nhận bản tin phản hồi ACK. Tại sao điều này lại quan trọng? Vì với các giao thức truyền tin đáng tin cậy như TCP, khi chưa có bản tin ACK thì không thể kết luận gói tin đã được truyền thành công. Các mạng có BDP > 1 x 10^5 bit hay 12500 byte được gọi là mạng LFN.

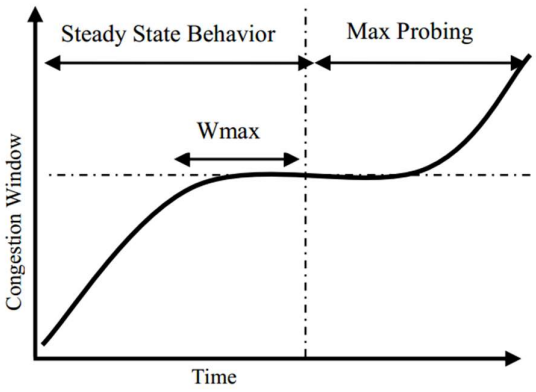

Fig. 8. Mô tả sự biến đổi cwnd trong TCP CUBIC.

b) BIC và CUBIC

Trước khi tìm hiểu sâu hơn về TCP CUBIC, ta hay tìm hiểu phiên bản tiền nhiệm của CUBIC là CUBIC. Thuật toán BIC: Đây là một thuật toán có cách triển khai độc đáo so với các các thuật toán khác được áp dụng trong TCP với sự kết hợp của Slow – Start, và tìm kiếm nhị phân (Binary seach), trong đó pha tìm kiếm nhị phân là quan trọng nhất.

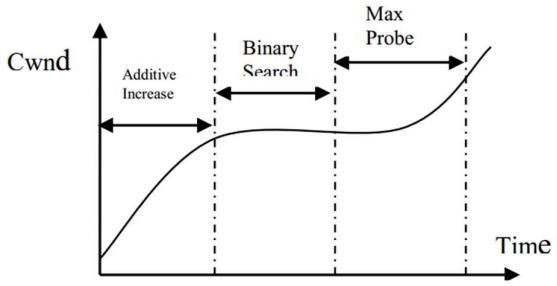

Fig. 9. Mô tả sự biến đổi cwnd trong TCP CUBIC.

Các ký hiệu và giá trị

Kí hiệu	Ý nghĩa
S_max	Số gia lớn nhất
S_min	Số gia nhỏ nhất
ssthresh	Mức ngưỡng W
β	Hệ số suy giảm cwnd theo cấp số nhân.
bic_inc	Độ tăng cwnd sau mỗi RTT

Fig. 10. Ý nghĩa các biến liên quan TCP BIC [10]

Cụ thể, TCP BIC được chia làm 3 pha: Pha tăng trưởng, Pha tìm kiếm nhị phân và Pha tối đa hóa khả năng, 3 pha này được gói gọn trong khoảng thời gian RTT. Ý tưởng chính của BIC là trong các trường hợp thuận lợi, độ rộng cửa sổ được thiết lập giá trị tối đa có thể. Sau khi xác nhận lỗi truyền gói xảy ra, ngay lập tức độ rộng cửa sổ trở về vị trí tối thiểu (lớn hơn 0).*

*: Tối đa và tối thiểu phụ thuộc vào khả năng của kênh truyền tại RTT đang xét.

Pha thứ nhất là Pha tăng trưởng. Trong đó, từ giá trị tối thiểu W_min, là giá trị mà tại đó đường truyền không gây ra

bất kỳ lỗi nào, thuật toán tăng W đến ssthresh, giá trị trung bình nằm giữa giá trị tối thiểu và giá trị tới hạn của hệ thống.

Pha 2: tại W = ssthresh áp dụng cây nhị phân tìm kiếm để xác định có thể tăng tốc độ truyền (tương ứng tăng cwnd) thêm nữa hay không. Nếu kết quả tìm kiếm nhị phân là có khả năng gây lỗi, thuật toán sẽ đặt W_max = ssthresh, giảm nhanh cwnd xuống W_min gần nhất, tính toán lại ssthresh và quay lại pha 1. Nếu không có lỗi xảy ra, vẫn thiết đặt W_min = ssthresh và chuyển sang pha 3.

Ở pha thứ 3, sau khi xác định khả năng truyền là thuận lợi, thuật toán sẽ tìm điểm bão hòa tiếp theo gần nhất mà hệ thống có thể đáp ứng, hay tìm W_max, sau đó tính toán ssthresh là trung bình (W_min, W_max).

Quá trình trên được lặp đi lặp lại cho đến khi gói tin bị quá thời gian và bị hủy. Thuật toán TCP BIC cho hiệu suất tốt bởi việc điều chỉnh tốc độ tăng trong các giai đoạn 1 và 3, tăng nhanh khi ở xa ssthresh và "thận trọng" ở gần ssthresh.

TCP CUBIC là sự cải tiến của BIC khi giữa lại các đặc trưng quan trọng của BIC (đặc biệt là tính ổn định vả khả năng mở rộng hệ thống. TCP CUBIC đơn giản hóa hàm cửa sổ, vận dụng các tính chất của hàm bậc 3 nhằm loại bỏ pha Binary search của TCP BIC. Sau khi giảm cwnd sau khi xảy ra mất gói, đặt W_max = W_tại_thời_điểm_mất gói và giảm cwnd thông qua hệ số suy giảm β. Tiếp theo, nhanh chóng khôi phục lại kích thước (fast recovery) theo hàm bậc 3 biến thể (xem hình…). Cách thức tận dụng ưu thế của hàm bậc 3 biến thể với tốc độ điều chỉnh nhanh trong điều kiện bình thường, và rất chậm khi tiệm cận ssthresh đã tạo ra vùng đồ thị tương đối bằng phẳng xung quanh ssthresh, cải thiện tốt độ ổn định của mạng.

Hàm tăng kích thước cửa sổ theo thời gian, kể từ lúc xuất hiện mất gói (packet loss event) của TCP CUBIC được biểu diễn bởi công thức 2.1 sau đây:

$$W\ (t) = \ C\ x\ (t - K)^3 + \ ssthresh\ (2.1)$$

Với:

$$K = \sqrt[3]{\frac{ssthresh\ x\ \beta}{C}}\ (= \sqrt[3]{\frac{ssthresh}{2}})\ (2.2)$$

Trong đó, các kí hiệu được mô tả trong Fig11:

STT	Kí hiệu	Ý nghĩa
1	W	Kích thước cửa sổ cwnd
2	C	Hệ số tỉ lệ. Thông thường: C = 0.4.
3	T	Mốc thời gian đang xét, so sánh với t = 0.
4	K	Thể hiện thời gian để W đạt giá trị W = ssthresh.
5	ssthresh	Mức ngưỡng và là kích thước cửa sổ cuối cùng ngay trước khi suy giảm.

Fig. 11. Mô tả ký hiệu các công thức TCP – CUBIC

Công thức 2.1 cho thấy W là một hàm bậc 3 phụ thuộc biến t. Đồng thời ta xét:

$$W\ (t) = \ C\ x\ (t - K)^3 + \ ssthresh\ (2.1)$$

$$W'\ (t) = \ 1{,}2t^2 - 2{,}4Kt + 1{,}2K^2$$

$$W''\ (t) = \ 2{,}4t - 2{,}4K$$

Như vậy, khi W < ssthresh, tương đương t < K, suy ra $W''(t) < 0$, hàm số là hàm lõm. Tương tự W > ssthresh thì đồ thị là hàm lồi. Điều này nhằm chứng minh dạng đồ thị của thuật toán TCP CUBIC.

IV. PHÂN TÍCH THUẬT TOÁN TCP CUBIC

A. *Thuật toán:*

1) *Pha Slow – start*

Ban đầu thiết lập cwnd = 1
Sau 1 RTT, cwnd = 2^(1) = 2
2 RTT, cwnd = 2^(2) = 4
3 RTT, cwnd = 2^(3) = 8
…

2) *Pha phòng chống tắc nghẽn*

cwnd = i
Sau 1 RTT, cwnd = i+1
2 RTT, cwnd = i+2
3 RTT, cwnd = i+3
…

3) *Truyền lại nhanh chóng (CUBIC)*

(Dựa theo 3 ACK)
ssthresh = ½ * cwnd
cwnd= ssthresh
(Trở lại pha phòng chống tắc nghẽn)
cwnd = C x (t-K) ^ 3 + ssthresh

B. *Nhận xét:*

So sánh với TCP nguyên bản, TCP CUBIC:

- Tránh việc đặt cwnd = 1 tại pha phát hiện tắc nghẽn gây lãng phí tài nguyên.
- Cho khả năng hồi phục đường truyền nhanh hơn rất nhiều trong pha truyền lại nhanh chóng ngay khi nhận thấy mất gói.

V. TRIỂN KHAI VÀ ĐÁNH GIÁ

A. *Đặt bài toán*

- Mô phỏng bài toán trên phần mềm NS2.
- Đặt khung giới hạn 800 x 400, gồm 4 node mô phỏng n0 – n3.
- Thông số các đường truyền:

Tên đường truyền: nude phát – node nhận	Tốc độ (Gbps)	Độ trễ (ms)
01	2.0	10
12	1.5	100
23	2.0	10

- Mô phỏng trong thời gian 300s. Lập biểu đồ truyền dữ liệu từ node 0 đến node 3.
- Mỗi cửa sổ có độ dài 5555 byte = 44440 bit.

B. *Mô hình topo*

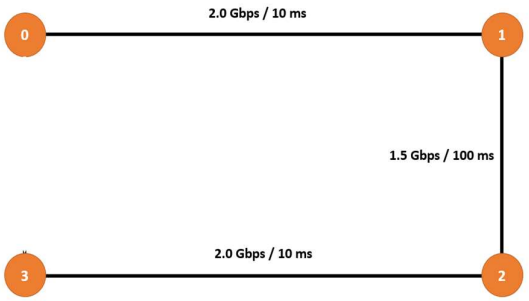

Fig. 12. Mô hình topo của mạng

C. Triển khai code

```
#prepare data
set ns [new Simulator]
set nf [open out.nam w]
$ns namtrace-all $nf
set wf1 [open flow_1.tr w]

# on finish
# flush all trace and open nam
proc finish {} {
    global ns nf
    $ns flush-trace
    close $nf
    exec xgraph flow_1.tr -geometry 800x400 &
    # exec nam out.nam &
    exit 0
}

# create node
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

$ns duplex-link $n0 $n1 2.0Gb 10ms DropTail
$ns duplex-link $n1 $n2 1.5Gb 100ms DropTail
$ns duplex-link $n2 $n3 2.0Gb 10ms DropTail
$ns queue-limit $n1 $n2 10

# setup queue watcher and queue limit bet n2 and n3
$ns duplex-link-op $n1 $n2 queuePos 0.1

# setup nam positions
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n1 $n2 orient right
$ns duplex-link-op $n2 $n3 orient right

# setup simulation colors
$ns color 1 Blue

# setup n0 to n3 connection
set tcp0 [new Agent/TCP/Linux]
$tcp0 set fid_ 1
$tcp0 set class_ 1
$tcp0 set window_ 8000
$tcp0 set packetSize_ 5555
$ns at 0 "$tcp0 select_ca cubic"
$ns attach-agent $n0 $tcp0

set sink0 [new Agent/TCPSink/Sack1]
$sink0 set class_ 1
$sink0 set ts_echo_rfc1323_ true
$ns attach-agent $n3 $sink0

# setup traffic
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ftp0 set type_ FTP

$ns connect $tcp0 $sink0

$ns at 0.2 "$ftp0 start"
$ns at 299.5 "$ftp0 stop"

# setup proc for cwnd plotting
proc plotWindow {tcpSource1 file1} {
  global ns

  set time 0.1
  set now [$ns now]
  set cwnd1 [$tcpSource1 set cwnd_]

  puts $file1 "$now $cwnd1"
  $ns at [expr $now+$time] "plotWindow $tcpSource1 $file1"
}

# setup plotting
$ns at 0.1 "plotWindow $tcp0 $wf1"

# when to stop
$ns at 300.0 "finish"

# starto!
```

$ns run

D. Kết quả

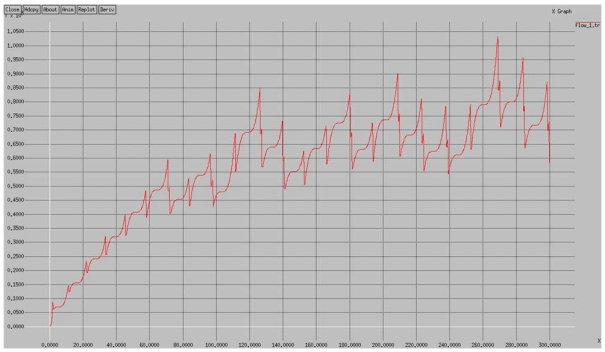

Fig. 13. Sơ đồ toàn thể

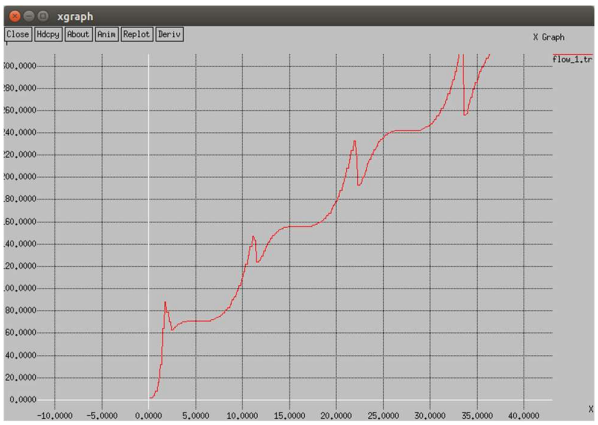

Fig. 14. Mô tả cwnd tăng ở giai đoạn đầu

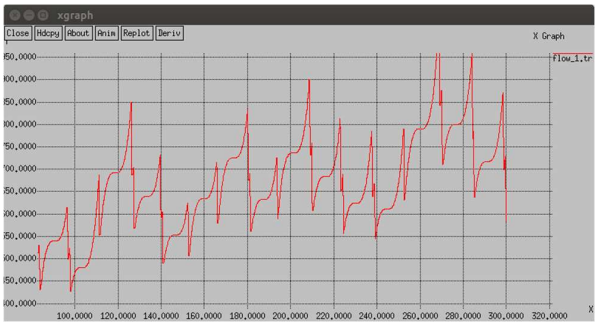

Fig. 15. Mô tả cwnd trong giai đoạn ổn định

E. Nhận xét

1) Nhận xét chung

- Gia tốc tăng lớn nên nhanh chóng tận dụng hết tốc độ đường truyền ở mức tối đa.
- Không giảm cwnd về 1 như với TCP cơ bản.
- Tốc độ còn kém ổn định, cần được nghiên cứu thêm. Các hướng nghiên cứu mới cho phép tốc độ ổn định trong một dải hẹp.

2) Đánh giá thông lượng

Thông lượng trung bình (throughput) = cwnd / RTT (gói/s) = cwnd x (số bit/gói) / RTT (bps) (4.1)

Trạng thái	Cwnd	Throughput
Ban đầu	0	0
Max	1.04	~ 2.7 Mbps
Ổn định	0.6 – 0.7	1.56 Gbps ~ 1.83 Gbps

Fig. 16. Đánh giá thông lượng mạng được mô tả trong mục V.A

VI. Conclusion

Qua đề tài trên, một số khái niệm liên qua đến TCP CUBIC: Lớp truyền vận, TCP, TCP BIC, vv đã được làm sáng tỏ. Bài cũng đã trình bày kịch bản mô phỏng TCP CUBIC và đánh giá một số yếu tố liên quan đến thông lượng của mạng trên.

Hướng nghiên cứu: Đề tài còn nhiều thiếu sót và có khả năng mở rộng thêm ở một số khía cạnh như sau: tối ưu hóa trên đường truyền không phải LFN như đường truyền vô tuyến. Ngoài ra, hướng nghiên cứu khác là tối ưu hóa thuật toán, ổn định tốc độ truyền gói trong một dải hẹp để hiệu suất tổng thể được tốt hơn nữa.

VII. References

[1] Chấn Phong, "Sự khác nhau giữa giao thức TCP và UDP", 07/08/2020. [Trực tuyến], Địa chỉ: https://quantrimang.com/su-khac-nhau-giua-giao-thuc-tcp-va-udp-154559 [Truy cập: 20/12/2020]

[2] "TCP/IP", 30/11/2020. [Trực tuyến], Địa chỉ: https://vi.wikipedia.org/wiki/TCP/IP [Truy cập: 20/12/2020]

[3] Konstantinos A. Gotsis, Sotirios Goudos, "A Test Lab for the Performance Analysis of TCP Over Ethernet LAN on Windows Operating System", 5/2005. [Trực tuyến]. Địa chỉ: https://www.researchgate.net/figure/TCP-UDP-performance-comparison_fig6_3050994 [Truy cập: 20/12/2020]

[4] Ivan Ganchev, R. D. van der Mei, Hán van den Berg, *Autonomous Control for a Reliable Internet of Services*, The Neitherlands, Springer, 1973.

[5] ""Slow Start" in TCP". [Trực tuyến]. https://www.isi.edu/nsnam/DIRECTED_RESEARCH/DR_HYUNAH/D-Research/slow-start-tcp.html [Truy cập: 20/12/2020]

[6] Justin Johnson, "What is CWND and RWND?", 24/05/2017 [Trực tuyến]. Địa chỉ: https://blog.stackpath.com/glossary-cwnd-and-rwnd/ [Truy cập: 20/12/2020]

[7] M. Allman, "TCP Congestion Control", 09/2009 [Trực tuyến]. Địa chỉ: https://tools.ietf.org/html/rfc5681 [Truy cập: 20/12/2020]

[8] Jawad Munir, "TCP in LTE/LTE-A Networks", 08/2014. [Trực tuyến]. Địa chỉ: https://www.researchgate.net/figure/TCP-slow-start-and-Congestion-avoidance-4_fig1_265906000 [Truy cập: 20/12/2020]

[9] Lê Gia Công, "CN_2_1_Tổng quan tầng Physical", 27/08/2013. [Trực tuyến]. http://legiacong.blogspot.com/2013/08/cn21tong-quan-tang-physical.html [Truy cập: 20/12/2020]

[10] [Trực tuyến]. https://drive.google.com/drive/folders/1qpUUcJ_x3O_CKntjNxuPUwaRPGF7q [Truy cập: 20/12/2020]