ĐẠI HỌC QUỐC GIA HÀ NỘI
ĐẠI HỌC CÔNG NGHỆ

BÙI THỊ LAN HƯƠNG

PHƯƠNG PHÁP MULTICAST TẦNG ỨNG DỤNG DỰA TRÊN CƠ CHẾ KÉO-ĐẨY CHO TRUYỀN VIDEO THỜI GIAN THỰC TRÊN MẠNG P2P

LUẬN VĂN THẠC SĨ

Hanoi – 2011

ĐẠI HỌC QUỐC GIA HÀ NỘI
ĐẠI HỌC CÔNG NGHỆ

BÙI THỊ LAN HƯƠNG

A PUSH-PULL BASED APPLICATION LAYER MULTICAST FOR P2P LIVE VIDEO STREAMING

Chuyên ngành: **Khoa học máy tính**
Mã: **60 48 01**

LUẬN VĂN THẠC SĨ

Giáo viên hướng dẫn: TS. Nguyễn Hoài Sơn

Publication: Huong BUI Thi Lan, Hoai Son NGUYEN(2011). A low-delay push-pull based application layer multicast for P2P live video streaming. *The third International Conference on Knowledge and System Engineering,* 2011 (to appear).

Hanoi – 2011

Tóm tắt

Cùng với sự phát triển của các ứng dụng đa phương tiện trên mạng Internet, streaming video qua Internet đang ngày càng thu được sự quan tâm của nhiều người, đặc biệt là các ứng dụng streaming video thời gian thực. IP Multicast là giải pháp hiệu quả nhất cho yêu cầu này. Tuy nhiên, việc triển khai IP Multicast trên mạng thực gặp nhiều vấn đề về thực tế. Do vậy, nhiều nghiên cứu đã chuyển sang hướng nghiên cứu về các ứng dụng multicast tầng ứng dụng. Nhiều giải pháp multicast tầng ứng dụng đã được đưa ra. Tuy chúng có những ưu điểm riêng, xong vẫn chưa đáp ứng được hết các yêu cầu của P2P streaming thời gian thực ví dụ như sự ra vào của các nút, độ trễ còn lớn. Hơn nữa, các giải pháp này chưa xét đến vấn đề của các nút không đóng góp.

Luận văn này trình bày một phương pháp multicast tầng ứng đụng dựa trên push-pull cho các ứng dụng streaming video trên mạng P2P. Mục tiêu chính của chúng tôi là tối ưu việc truyền dữ liệu trên mạng P2P bằng cách đảm bảo rằng buộc thời gian của streaming video thời gian thực. Chúng tôi đạt được mục tiêu này bằng cách khởi tạo một cây cân bằng cho việc đẩy dữ liệu và tối ưu việc kéo dữ liệu giữa các nút bằng việc làm giảm thời gian truyền giữa đẩy dữ liệu và kéo dữ liệu. Kết quả là phương pháp của chúng tôi có thể làm giảm được độ trễ tại mỗi nút. Chúng tôi cũng đề xuất một cơ chế tit-for-tat để khuyến khích các nút đóng góp nhiều hơn. Để đánh giá hiệu suất hoạt động của phương pháp đề xuất, chúng tôi xây dựng một chương trình thử nghiệm dựa trên bộ lập lịch SMPL. Chúng tôi cũng so sánh hiệu suất hoạt động của phương pháp đề xuất với các phương pháp khác. Các kết quả thử nghiệm đã chứng minh được tính hiệu quả của giải pháp chúng tôi.

Chương 1: Giới thiệu

Cùng với sự phát triển của các ứng dụng đa phương tiện trên Internet, streaming video thời gian thực trên Internet đang ngày càng trở nên hấp dẫn người dùng. Các ứng dụng streaming video thời gian thực thường yêu cầu truyền dữ liệu tới một lượng lớn người sử dụng. IP Multicast [5] là giải pháp hữu hiệu quả nhất cho yêu cầu này. Tuy nhiên, việc triển khai IP Multicast còn gặp nhiều vấn đề về thực tế[6]. Nhiều nhà nghiên cứu đã chuyển sang nghiên cứu về các phương pháp multicast tầng ứng dụng (ALM). ALM tận dụng được khả năng của các thiết bị cuối bằng các mỗi thiết bị cuối cùng lúc đóng vai trò cả người gửi lẫn người nhật. Tuy nhiên, các giải pháp này lại phải đối mặt với các vấn đề như sự ra vào của các nút trong mạng, sự tồn tại của các nút không đóng góp, các nút có băng thông không đồng nhất, ràng buộc về thời gian của các ứng dụng thời gian thực.

Gần đây, phương pháp tiếp cận dựa trên lưới [4,9,11-12,14,16,23-25] đã thu hút rất nhiều sự chú ý vì nó có thể giảm thiểu tác động của việc các nút ra vào và các nút hàng xóm có băng thông thấp bằng cách kéo các dữ liệu cần thiết từ một tập các nút láng giềng thích hợp. Mỗi nút độc lập lựa chọn một tập các nút hàng xóm và kéo dữ liệu từ họ dựa trên một giả định rằng các nút hàng xóm có thể có dữ liệu video cần thiết. Tuy nhiên, có một sự cân đối giữa việc làm giảm độ trễn bằng cách gửi các yêu cầu pull định kỳ với tải của toàn bộ hệ thống [12,14]. Bên cạnh đó, phương pháp này còn gặp phải vấn đề nút cổ chai về dữ liệu. PRIME [4] cải thiện được vấn đề nút cổ chai về băng thông và nút cổ chai về dữ liệu bằng cách kết hợp một phương pháp truyền dữ liệu qua nhiều cây con và một phương pháp kéo dữ liệu từ các nút ở các cây khác. Tuy nhiên, PRIME đã không được giải quyết một số vấn đề như thời gian trễ lớn, sự vào ra của các nút, và các nút không đóng góp cho mạng.

Luận văn trình bày một cơ chế P2P streaming phân tán và có khả năng mở rộng cao. Phương pháp này cũng kết hợp của push và pull giống như PRIME. Các nút được tổ chức vào các cây con, trong đó mỗi nút (ngoại trừ nguồn) chỉ thuộc về một cây. Mỗi nút đồng thời có liên kết tới các nút của các cây con khác. Dữ liệu từ nguồn được chia thành nhiều luồng nhỏ, mỗi luồng sẽ được truyền thông qua một cây con dựa vào cơ chế push. Sau đó, mỗi nút sẽ kéo các luồng con khác từ các nút nằm ở các cây con khác.

Các đóng góp của chúng tôi bao gồm:

- Tối ưu việc truyền các gói tin để giảm thời gian trễ của mỗi nút. Chúng tôi phát hiện ra rằng thời gian trễ của mỗi nút sẽ tùy thuộc vào thời gian đến của gói tin cuối cùng của segment. Do vậy, trong phương pháp của chúng tôi, mỗi nút sẽ cố gắng thiết lập các liên kết pull với các nút ở level dưới hoặc cùng level. Ở đây, level của mỗi nút được định nghĩa là số bước từ nguồn tới nút đó. Do vậy, khoảng cách của thời gian đến của các gói khác nhau sẽ được giảm.
- Đảm bảo tính công bằng giữa các nút và khuyến khích các nút đóng góp nhiều hơn cho mạng. Chúng tôi đề xuất một cơ chế tit-for-tat. Trong đó, các nút đóng góp nhiều hơn sẽ có xác xuất nhận được nhiều dữ liệu hơn. Do vậy, chúng có thể nhận được chất lượng video tốt hơn. Để đạt được điều này, chúng tôi xây dựng một cây cân bằng để chắc chắn rằng mỗi nút ở trong các cây con khác nhau có thể cung cấp được dữ liệu cho nhau.
- Khắc phục lỗi khi có nút gặp sự cố: chúng tôi cố gắng duy trì số lượng nút bị ảnh hưởng của một nút lỗi là nhỏ và tối thiểu hóa thời gian khắc phục lỗi.
- Đảm bảo khả năng mở rộng: phương pháp của chúng tôi hoàn toàn phân tán, do vậy có khả năng mở rộng cao.

Chúng tôi cũng xây dựng một chương trình mô phỏng để đánh giá hoạt động của hệ thống được đề xuất. Kết quả mô phỏng đã chỉ ra rằng phương pháp của chúng tôi cải thiện được thời gian trễ trong khi kiểm soát được sự vào ra của các nút và vẫn đảm bảo được chất lượng video.

Chương 2. Các nghiên cứu liên quan.

Nhiều giao thức multicast tầng ứng dụng đã được đề xuất. Dựa trên topology, các phương pháp multicast tầng ứng dụng hiện nay có thể chia vào làm hai loại: tiếp cận dựa theo cấu trúc cây và tiếp cận dựa theo cấu trúc lưới. Các phương pháp tiếp cận dựa trên cây thường tổ chức các nút vào các cây multicast [8-9,15,24]. Đối với các phương pháp cây đơn, thì có hai vấn đề chính đó là sự bất công bằng giữa các nút trong và các nút lá và ảnh hưởng của việc ra vào của các nút.

Để đối phó với các vấn đề trên, trong SplitStream [3], các nút được tổ chức vào nhiều cây con sao cho nút trong của cây con này sẽ là nút lá ở tất cả các cây còn lại. Luồng video được chia thành các luồng con sử dụng Multiple Description Coding [7] hoặc layered video [16] và mỗi cây con sẽ chuyển một luồng. Tuy nhiên, Splitstream yêu cầu tất cả các nút có băng thông bằng nhau. Một vài nghiên cứu mới hơn [21-22] giải quyết vấn đề này bằng cách đưa vào các giải pháp tối xây dựng cây ngay cả khi các nút không có băng thông bằng nhau. Tuy vậy, các phương pháp này vẫn gặp phải vấn đề thời gian trễ lớn do khoảng cách thời gian đến của các luồng con khác nhau còn lớn. Một vấn đề nữa đó là chi phí duy trì cây và khắc phục lỗi cũng còn cao.

Trong các phương pháp tiếp cận dựa trên cấu trúc lưới [4,9,11-12,14,16,23-25], mỗi nút độc lập lựa chọn một số nút hàng xóm và kéo dữ liệu từ chúng dựa trên giả định rằng các nút hàng xóm này có dữ liệu cần thiết. Cơ chế kéo dữ liệu này cũng chính là điểm khác biệt lớn nhất giữa phương pháp tiếp cận dựa trên lưới và dựa trên cây. Các phương pháp dựa trên lưới có thể giảm thiểu ảnh hưởng của sự ra vào các nút và các nút hàng xóm có băng thông thấp bằng cách kéo dữ liệu từ một tập các nút hàng xóm. Tuy nhiên, có một sự cân bằng giữa thời gian trễ và tải trên toàn hệ thống [12,14]. Hơn nữa, các phương pháp này gặp phải vấn đề của việc thiếu dữ liệu tại các nút nhận được yêu cầu pull. Vấn đề này còn được gọi là nút cổ chai dữ liệu và gây ra thời gian trễ lớn.

PRIME [6và các phương pháp [23,24,25] xây dựng một lưới để truyền dữ liệu thông qua hai pha: pha phân tán và pha kéo. Trong pha phân tán, luồng dữ liệu cũng được chia thành nhiều luồng con và mỗi luồng con được truyền qua một cây con. Trong pha kéo, mỗi nút sẽ kéo dữ liệu của thiếu từ các nút ở các cây khác. Do vậy, các phương pháp này cải thiện được nút cổ chai về băng thông và nút cổ chai về dữ liệu. Tuy nhiên, PRIME không chỉ rõ cách họ xây dựng lưới mà thỏa mãn được các rằng buộc về bậc băng thông mà họ đề ra. [23,25] đề xuất nheiefu chiến thuật để xây dựng lưới khác nhau nhưng nút bootstrap phải lưu dữ liệu về toàn bộ mạng và như vậy khả năng mở rộng của các phương pháp này là hạn chế. [27] chỉ ra phương pháp xây dựng lưới hoàn toàn phân tán nhưng lại quá phức tạp trong triển khai. Thêm vào đó, mọi phương pháp dựa trên lưới này vẫn chưa đưa ra được cách giải quyết hoặc giải quyết chưa hiệu quả cho vấn đề độ trễ dữ liệu lớn (do khoảng cách về thời gian của các gói tin push và pull), các vấn đề về nút không đóng góp và sự vào ra của các nút.

Chương 3. Giải pháp của chúng tôi cho P2P streaming video thời gian thực

3.1. Khái quát

Phương pháp của chúng tôi là một phương pháp tiếp cận dựa trên push-pull. Chúng tôi xây dựng k cây con riêng biệt trong đó mỗi nút ngoại trừ nút nguồn thuộc về chỉ một cây con. Các cây con có chung gốc là nút nguồn. Mỗi nút cũng có các liên kết tới các cây con khác.

Luồng video từ nút nguồn được chia thành k luồng con sử dụng MDC [7]. Cứ sau Δ giây, nút nguồn lại tạo ra một segment dữ liệu mới. Segment này sẽ được mã hóa thành k phần nhỏ dùng MDC và mỗi phần thuộc về một luồng con. K luồng con sẽ được truyền đi thông qua k cây riêng biệt trong pha đẩy. Kết thúc mỗi pha đẩy, mỗi nút sẽ có một phần dữ liệu để hiển thị. Để cải thiện chất lượng video, mỗi nút sẽ kéo dữ liệu còn thieus từ các nút trong các cây khác thông qua pha kéo. Nút kéo càng được nhiều dữ liệu thì nó càng có chất lượng tốt hơn. Như vậy, pha đẩy và kéo được thực hiện đồng thời cùng lúc.

Vấn đề ở đây là làm thế nào để xây dựng mạng lưới bên dưới, làm thế nào mỗi nút chọn một nút để kéo dữ liệu và làm thế nào để đối phó với việc các nút lỗi. Những vấn đề này rất cần xem xét. Bên cạnh đó, chúng tôi cũng xem xét việc các nút ích kỷ chỉ muốn nhận được dữ liệu nhưng không muốn đóng góp băng thông cho mạng. Vì vậy, chúng tôi thiết kế một chính sách hợp lý để đảm bảo rằng các nút đóng góp nhiều hơn sẽ nhận được nhiều dữ liệu hơn và có thời gian trễ nhỏ hơn.

3.2. Xây dựng mạng lưới

Trong giải pháp của chúng tôi, mỗi nút duy trì hai loại kết nối, kết nối trong cây con (tức là kết nối đến một nút cha và một số nút con) và kết nối đến các nút của các cây con khác. Mỗi nút duy trì các thông tin về danh sách các nút con của nó, số lượng và level cao nhất của các nút hậu duệ. Thông tin này được sử dụng để giữ cho xây dựng các cây con được cân bằng. Sau một khoảng thời gian, mỗi nút ở level i sẽ cập nhật các giá chị này cho các nút cha của nó.

Mạng lưới được xây dựng bằng cách xây dựng một cây cân bằng với gốc là nút nguồn sao cho số nút và bậc của các cây tương đồng với nhau. Tính chất này quan trọng để cho các nút ở các cây con khác nhau có thể trao đổi được dữ liệu với nhau. Hình 8 chỉ ra ví dụ về mạng lưới của chúng tôi với 15 nút tham gia.

Khi một nút muốn tham gia vào mạng và nhận dữ liệu, nó sẽ gửi yêu cầu tới nút nguồn. Nút nguồn sẽ lựa chọn một nút tại level 1 dựa trên các ưu tiên sau:

- Chọn nút có số lượng nút hậu duệ là nhỏ nhất
- Chọn nút sao cho giá trị level của các nút hậu duệ là nhỏ nhất.

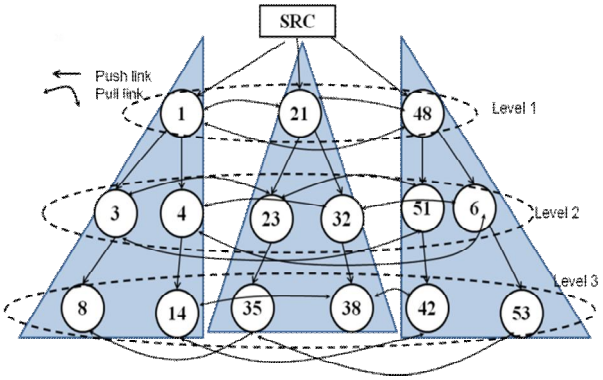
Sau đó, nút nguồn gửi thông tin về các nút này tới nút tham gia. Khi một nút nhận được yêu cầu tham gia, nó sẽ kiểm tra xem còn đủ băng thông để chấp nhận thêm kết nối không. Nếu có, nó sẽ trả lời chấp nhận. Nếu không, nó sẽ tìm một nút con thích hợp và gửi thông tin đi cho nút yêu cầu tham gia. Điều này sẽ được lặp lại cho đến khi nút muốn tham gia tìm được vị trí thích hợp trên cây. Và kết quả là, cây được cân bằng.

Hiển nhiên người dùng muốn giữ lại một lượng băng thông để có thể duy trì chất lượng video của họ trước khi đóng góp cho mạng. Giả sử rằng băng thông của mỗi luồng con là *bwpf*. Nếu băng thông của mỗi nút là *b*bwpf* và số lượng cây con là is k $(k \leq b)$.

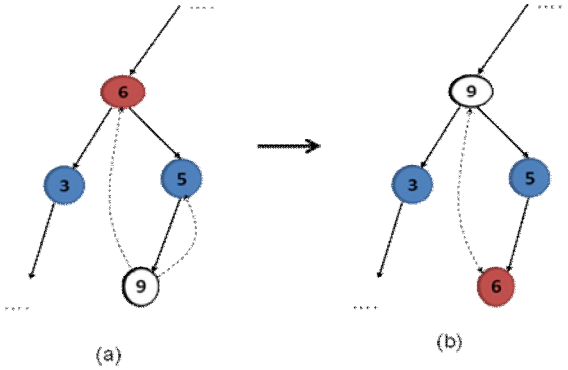
Mỗi nút sẽ sử dụng băng thông $(k-1)*bwpf$ để gửi dữ liệu trong kéo với các nút trong các cây khác theo cơ chế tit-for-tat. Do vậy, mỗi nút sẽ sử dụng tối đa băng thông $(b-(k-1))*bwpf$ để đẩy dữ liệu đi ở các nút con trong pha phân tán. Như vậy, mỗi nút sẽ có tối đa $b-(k-1)$ nút con. Nếu nút có băng

thông nhỏ hơn *k*bwpf*, nó sẽ là các nút lá và không có đủ băng thông để trao đổi dữ liệu với các nút khác. Do vậy theo cơ chế tit-for- tat, nút này sẽ có chất lượng video thấp.

Sau khi tham gia và các cây con, nút sẽ thiết lập các liên kết kéo với *k-1* nút trong *k-1* cây con khác để kéo dữ liệu trong pha kéo. Để làm được điều đó, nút sẽ liên hệ với nút cha của nó để lấy về danh sách *k-1* nút đang liên kết với nút cha trong pha kéo. Sau đó nó sẽ gửi yêu cầu kéo tới các nút này. Một nút nhận được yêu cầu kéo từ nút khác sẽ kiểm tra xem nó có trả lời yêu cầu đó không theo cơ chế tit-for-tat mà chúng tôi sẽ mô tả rõ hơn ở phần sau. Nếu không chấp nhận, nó sẽ gửi tới cho nút yêu cầu một tập các nút thích hợp. Và việc gửi yêu cầu lại được tiếp tục. Mỗi nút sẽ duy trì *k-1* liên kết tới các nút thuộc *k-1* cây con khác.



Hình 8. Cơ chế kéo đẩy của hệ thống



Hình 9. Ví dụ về việc thay đổi vị trí của các nút

Chúng tôi cũng đề xuất một cơ chế để thay thế các nút trong có băng thông thấp hơn bằng các nút có level cao hơn nhưng có băng thông lớn hơn. Mục tiêu của cơ chế này đó là làm giảm độ sau của cây. Rõ ràng là độ sâu của cây quyết định thời gian truyền tin. Do vậy, nút càng có level thấp thì nó càng nhận được dữ liệu sớm hơn.

Cơ chế này được thực hiện như sau. Sau khi một nút có băng thông cao, giả sử là n_h, tham gia vào mạng, nó sẽ kiểm tra các nút tổ tiên của nó để tìm ra nút có thể không cung cấp đủ băng thông. Nếu nó tìm ra được nút như vậy, giả sử là n_l, nút n_h sẽ gửi yêu cầu thay đổi vị trí tới nút n_l. Nếu yêu cầu được chấp nhận, hai nút sẽ trao đổi danh sách liên kết của chúng. Cơ chế này bao gồm cả việc dữ liệu không bị gián đoạn khi đang truyền.

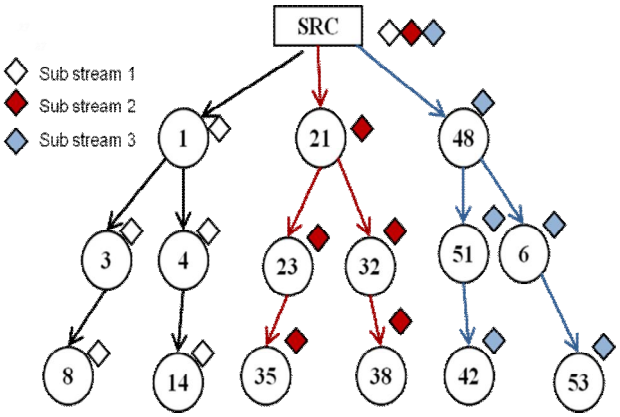
Hình 9 miêu tả một ví dụ của việc thay thế nút. Nút 6 có level thấp hơn nhưng lại có băng thông thấp hơn sẽ bị thay thế bởi nút có băng thông cao hơn 9. Kết quả là 9 từ vị trí là con cháu trở thành tổ tiên của nút 6.

3.3. Phân bổ dữ liệu

Trong pha phân tán, nút nguồn sẽ gửi mỗi luồng con tới mỗi con của nó. Mỗi nút nhận được dữ liệu từ nguồn sẽ tiếp tục đẩy dữ liệu đi tới các nút con có level cao hơn thông qua các cây multicast con. Cuối cùng, thông qua pha kéo, mỗi nút đều có một lượng dữ liệu tối thiểu để hiển thị (ở đây là một sub-stream)

Hình 10 chỉ ra một ví dụ về của pha phân tán với k = 3. Luồng dữ liệu được chia thành 3 luồng con và nút nguồn gửi các luồng này lần lượt tớ các nút 1, 21 và 48. Các nút này sẽ tiếp tục chuyển dữ liệu tới các hậu duệ của nó.

Trong pha kéo, nút sẽ kéo dữ liệu từ các nút ở các cây con khác. Chúng gửi yêu cầu kéo thông qua các liên kết kéo. Nếu một nút có dữ liệu, nó sẽ ngay lập tức trả lời dữ liệu tới cho các nút kéo.



Hình 10. Ví dụ về pha đẩy với k = 3

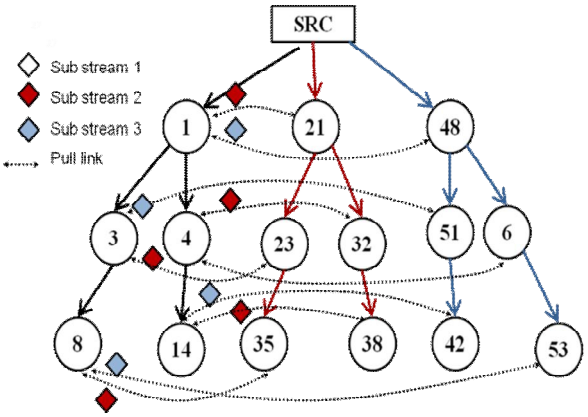


Figure 11. Ví dụ về pha kéo k =3

Hình 11 chỉ ra ví dụ của pha kéo, các nút kéo dữ liệu còn thiếu từ các nút khác. Nút 3 kéo luồng thư s2 từ nút 23 và luồng 3 từ nút 52.

Do chúng ta luôn duy trì một cây cân bằng, một nút luôn có thể kéo dữ liệu từ một nút con ở cây khác mà có cùng level hoặc level thấp hơn với xác suất cao. Điều này có nghĩa là phương pháp của chúng tôi có thể làm giảm chênh lệch thời gian đến của các luồng dữ liệu khác nhau (cụ thể là xuống còn 1 bước). Kết quả là, phương pháp của chúng tôi giảm được thời gian buffer khi mà thời gian hiển thị của mỗi segment phụ thuộc vào thời gian đến của phần cuối cùng của segment đó.

3.4. Chính sách cân bằng

Chúng tôi thiết kế một cơ chế đảm bảo công bằng giữa các nút để khuyến khích các nút đóng góp nhiều hơn cho mạng. Cơ chế này đảm bảo rằng các nút đóng góp nhiều cho mạng (cung cấp nhiều dữ liệu streaming cho các nút khác) sẽ nhận được chất lượng video tốt hơn.

Đầu tiên, dựa vào cơ chế thay đổi vị trí giữa các nút có level thấp nhưng băng thông thấp và nút có level cao nhưng băng thông cao. Các nút có băng thông thấp do vậy có đóng góp ít hơn cho hệ thống sẽ bị đẩy xuống các level cao hơn. Nút có băng thông cao do vậy sẽ đóng góp nhiều hơn cho mạng và được đẩy lên các level thấp. Như vậy, các nút có băng thông cao sẽ nhận được dữ liệu sớm hơn và thời gian buffer sẽ nhỏ hơn.

Thứ hai, khi nhận được nhiều yêu cầu kéo, nút sẽ sử dụng cơ chế tit-for-tat để lựa chọn nút mà nó sẽ đáp ứng yêu cầu. Việc lựa chọn này sẽ được diễn ra định kỳ, tại mỗi khoảng thời gian, được gọi là round. Mỗi round được tính bằng thời gian truyền của một vài gói. Cơ chế được thực hiện như sau.

Tại round r-1, nếu nút có thể kéo dữ liệu từ một nút n thì trong danh sách của nó, nó cập nhật bảng ưu tiên của nút tại tại round r theo công thức sau:

$$Pri_n^r = Pri_n^{r-1} + \alpha.d_n^{r-1}$$

với

- Pri_n^r là độ ưu tiên của nút n tại round thứ r
- Pri_n^{r-1} là độ ưu tiên của nút n tại round thứ r -1
- d_n^{r-1} là lượng dữ liệu kéo được từ nút n tại round thứ r -1
- α là tham số độ ưu tiên của hệ thống

Trong trường hợp nút có băng thông trống đủ rộng, nút sẽ đáp ứng mọi yêu cầu. Nếu số lượng nút yêu cầu vượt quá khả năng của nút, nút sẽ lựa chọn các nút để đáp ứng yêu cầu dựa theo thứ tự yêu tiên

để đáp ứng. Trong trường hợp hai nút có cùng độ ưu tiên, nút sẽ đáp ứng yêu cầu của nút có level thấp hơn. Bởi vì, nếu nút cung cấp dữ liệu cho nút có level thấp hơn, thì tại round sau nút có thể nhận được dữ liệu với độ trễ nhỏ hơn với xác suất cao hơn.

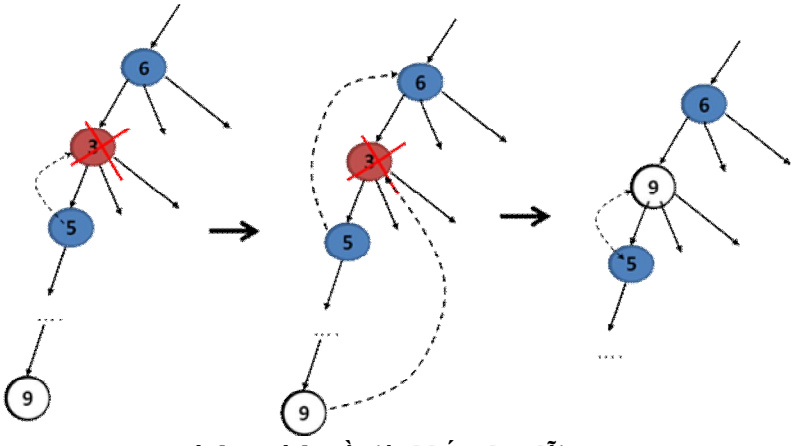
Các nút không cung cấp dữ liệu cho nút khác, thì xác suất mà nó nhận được dữ liệu cũng nhỏ đi. Do vậy, các nút không đóng góp cho mạng sẽ có chất lượng thấp đi.

3.5. Khôi phục lỗi

Khi một nút gặp lỗi hoặc rời khỏi hệ thống mà không có báo trước, các nút hậu duệ của nó trong cây con sẽ bị ảnh hưởng và các nút đang kéo dữ liệu từ nó cũng gặp vấn đề gián đoạn về dữ liệu. Chúng tôi đề xuất một cơ chế khắc phục lỗi nhanh chóng.

Đầu tiên, mỗi nút sẽ duy trì danh sách các tổ tiên của nó trên cây con, bắt đầu từ nguồn. Để làm được điều này, các sẽ gửi đi thông điệp thông báo tới các con của nó theo chu kỳ. Nút con sẽ thêm thông tin này vào khi nhận được thông điệp thông báo và tiếp tục gửi thông điệp thông báo tới các con của nó. Quá trình này tiếp tục cho đến khi nút lá nhận được thông điệp thông báo.

Sau đó, mỗi nút sẽ duy trì danh sách các nút lá mà là các ứng cử viên có thể thay thế nút con của nó, trong trường hợp nút con của nó có lỗi. Các nút lá được lựa chọn dựa vào băng thông của chúng và thời gian sống. Theo lý thuyết được chỉ ra trong [30-31], các nút đã tham gia vào mạng lâu thì cũng có xu hướng ở trong mạng lâu hơn. Quá trình lựa chọn danh sách này được thực hiện ngược lại so với quá trình trên.



Hình 1 Ví dụ về việc khắc phục lỗi

Một nút sẽ coi nút cha của nó trong cây con bị lỗi khi mà sau một khoảng thời gian timeout nó không nhận được thêm thông tin nào từ cha. Trong trường hợp này, nó sẽ kiểm tra nút cha còn sống hay đã chết. Nếu cha còn sống thì nó sẽ chờ cho đến khi tổ tiên của nó sửa được các lỗi phía trên. Nếu nút cha không còn sống, nó sẽ gửi thông điệp yêu cầu khôi phục lỗi lên nút tổ tiên phía trên (ví dụ là nút ông, nút cha của cha). Nút tổ tiên sẽ lựa chọn một nút trong số các nút nằm trong danh sách của nó để thay thế cho nút bị lỗi. Phương pháp này cho phép giảm thời gian khắc phục lỗi trước khi cây cần thiết phải được tổ chức lại.

Figure 12 chỉ ra ví dụ về việc khắc phục lỗi. Nút 3 rời mạng, sau thời gian timeout nút 5 phát hiện nút 3 đã bị lỗi. Nút 5 gửi yêu cầu khắc phục lỗi lên nút 6. Nút 6 khắc phục lỗi bằng việc chuyển nút 9 lên thay thế vị trí của nút 3.

Một nút không nhận được dữ liệu pull sau thời gian timeout sẽ kiểm tra xem nút mà nó pull còn sống hay không. Nếu nó không nhận được message phản hồi, nó sẽ coi nút mà nó đang pull dữ liệu về đã lỗi và tự thiết lập một kết nối pull khác.

Chương 4. Thực nghiệm và kết quả

4.1. Experimental setup

Chúng tôi xây dựng một chương trình mô phỏng hướng sự kiện dựa trên bộ lập lịch SMPL [2]. Chúng tôi lựa chọn triển khai phương pháp của chúng tôi trên chương trình mô phỏng thay vì triển khai trên một hệ thống thực ví dụ như Planet Lab bởi vì chúng cho phép chúng tôi theo dõi và đánh giá được hoạt động của hệ thống dễ dàng. Đồng thời, chúng tôi cũng có thể đánh giá được hoạt động của hệ thống trên môi trường rộng.

Các hoạt động của hệ thống sẽ được biểu diễn dưới dạng các sự kiện sau:

- *NODE_CHURN:* mô tả sự kiện vào ra của nút khi có một nút rời ra hoặc tham gia vào mạng.
- *MESSAGE_SENT:* chỉ sự kiện gửi một thông điệp đi giữa 2 nút.
- *MESSAGE_RECEIVED:* chỉ sự kiện một nút nhận được một thông điệp.

Để đánh giá hiệu năng hoạt động của hệ thống, chúng tôi cũng xây dựng chương trình mô phỏng PRIME[4] để so sánh hiệu năng của hai hệ thống với nhau. Các tham số và thuật toán được thiết lập cho PRIME, được lựa chọn từ các kết quả tốt nhất trong [4].

Chúng tôi sử dụng GT-ITM Generator [1] để sinh ra topology của mạng với 1500 nút theo đồ thị transit-stub. Trong đó có 5 transit nút, mỗi transit nút được kết nối với trung bình 6 stub-domain. Mỗi stub-domain bao gồm trung bình khoảng 50 nút. Xác suất cạnh giữa các nút transit là 1. Chúng tôi sử dụng hàm random để sinh ra độ trễ giữa các nút. Thời gian trễ giữa các transit nút từ 100ms tới 200ms. Thời gian trễ giữa các transit nút và stub-domain từ 50ms tới 100ms. Thời gian trễ giữa các nút trong cùng stub-domain từ 1ms tới 30ms.

Nút nguồn có băng thông là 400Kbps. Luồng video được chia làm 4 mảnh nhỏ và có cùng tốc độ là 100Kbps. Nguồn sẽ tạo một segment mới sau mỗi khoảng thời gian Δs trong suốt thời gian quan sát là 3000s, với Δ có giá trị là 2s trong mọi kịnh bản. Chúng tôi mô phỏng hệ thống với 1000 nút tham gia được lựa chọn ngẫu nhiên từ 1500 nút trên mạng.

Chúng tôi sẽ so sánh kết quả thực nghiệm của các giá trị sau:

- *Tỷ lệ mất mát gói:* là tỷ lệ số các gói bị mất mát trên tổng số các gói mà nút nên nhận được trong suốt thời gian sống. Nếu nút nhận đủ số lượng các gói, nút có thể hiển thị 100% chất lượng video và do vậy tỷ lệ này sẽ bằng 0
- *Độ chênh lệch thời gian giữa các gói khác nhau của cùng một segment:* để hiện thỉ được video tốt nhất, nút sẽ phải đợi đến khi nhận được hết tất cả gói của segment với mỗi gói thuộc về một luồng con. Do vậy, nếu giá trị độ chênh lệch này cao sẽ dẫn thới thời gian buffer càng dài.
- *Thời gian trễ trung bình của các gói từ nguồn tới nút và thời gian trễ trung bình của segment từ nguồn tới nút:* hai giá trị này thể hiện thời gian buffer cần thiết để nút có thể hiển thị được chất lượng video tốt nhất.

4.2. Kết quả

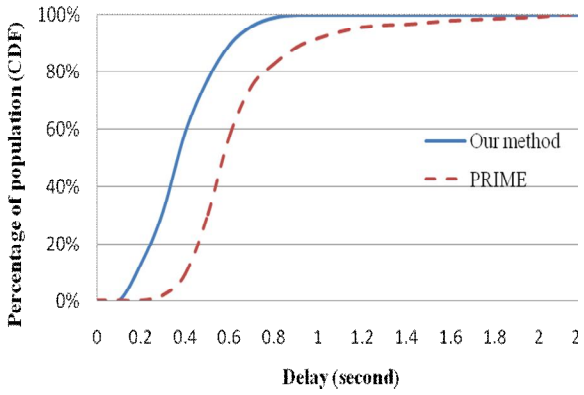
Chúng tôi tập trung vào các kịch bản mà băng thông của nguồn bị giới hạn (chỉ 400Kbps). Cụ thể, chúng tôi thực hiện thí nghiệm trên 3 kịch bản sau:

- Không có nút ra vào trong suốt thời gian mô phỏng
- Có sự ra vào của nút trong suốt thời gian mô phỏng
- Băng thông của các nút trong mạng là không đồng nhất.

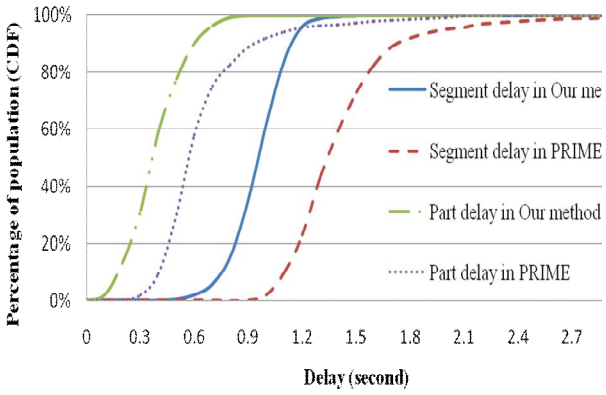
Chúng tôi chỉ so sánh hoạt động của hệ thống của chúng tôi với PRIME trong kịch bản đầu. Bởi lẽ PRIME chỉ hỗ trợ trên kịch bản này, PRIME chưa giải quyết việc ra vào của các nút.

Trong kịch bản đầu tiên, mọi nút đều có cùng băng thông là 700Kbps. Các nút trong cả hai phương pháp đều nhận được 100% các gói trong khoảng thời gian timeout là 3s. Do vậy, chúng tôi sẽ không so sánh tỷ lệ mất mát gói tin trong trường hợp này.

Độ chênh lệch thời gian giữa các gói trong kịch bản này được chỉ ra trong hình 14. Chúng ta có thể thấy giá trị này trong phương pháp của chúng tôi nhỏ hơn so với PRIME. Trong phương pháp của chúng tôi 80% nút có độ chênh lệch thời gian nhỏ hơn 0.5s và 100% các nút có độ chênh lệch thời gian giwuax các gói là 0.7s. Trong khi đó, với PRIME chỉ có khoảng 30% nút có độ chênh lệch nhỏ hơn 0.5s, và khoảng chưa đến 80% nút thời gian chênh lệch nhỏ hơn 0.7s.



Hình 14. Đồ thị biến thiên của sự chênh lệch thời gian của các phần khác nhau của cùng segment trong PRIME và phương pháp của chúng tôi

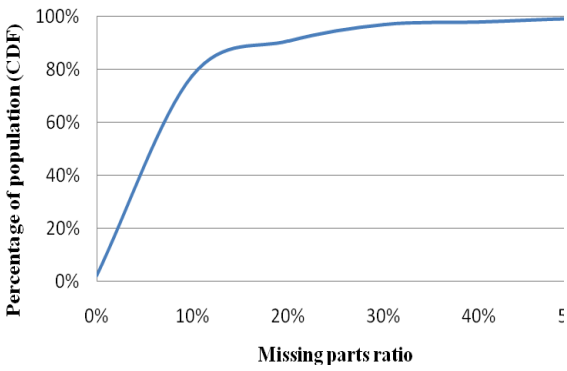


Hình 15. Đồ thị biến thiên của độ trễ của các phần trung bình từ nguồn tới nút trong PRIME và trong phương pháp của chúng tôi

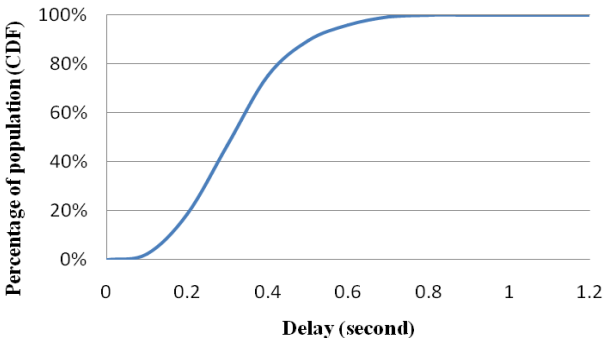
Hình 15 chỉ ra thời gian trễ trung bình của các part và thời gian trễ trung bình của các segment giữa phương pháp của chúng tôi và PRIME. 100% các nút tham gia hệ thống của chúng tôi nhận được gói sau 1s, trong khi đó giá trị này chỉ là 20% ở PRIME. 100% các nút trong hệ thống của chúng tôi nhận đầy đủ các gói sau 1,2s trong khi đó giá trị này chỉ là 40% ở PRIME.

Trong kịch bản thứ 2, chúng tôi đánh giá hoạt động của hệ thống của chúng tôi ngay cả khi có sự vào ra của các nnuts. Chúng tôi sử dụng phân bố Pareto[17] để sinh các giá trị vào ra của các nút. Hình 16 chỉ ra rằng hơn 80% các nút có tỷ lệ mất mát gói nhỏ hơn 10%, và gần 100% các nút có tỷ lệ mất mát gói nhỏ hơn 30%. Điều này chỉ ra là chất lượng của video tại mỗi nút được duy trì ở mức chấp nhận được mặc dù trong hệ thống có sự ra vào của các nút.

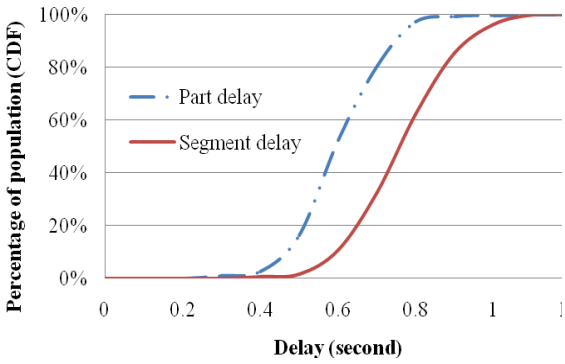
Độ chênh lệch thời gian đến của các gói của tất cả các nút được giữ dưới 0.7s ngay cả khi có sự vào ra của các nút (được chỉ ra trong hình 17). Hình 18 chỉ ra thời gian trễ trung bình của nút nhận được gói và thời gian trễ trung bình khi nút nhận được đầy đủ các gói của một segment. Gần 100% nút có thời gian trễ trung bình nhận được gói nhỏ hơn 0.8s và thời gian trễ trung bình khi nhận được đầy đủ các gói của một segment là 1.2s.



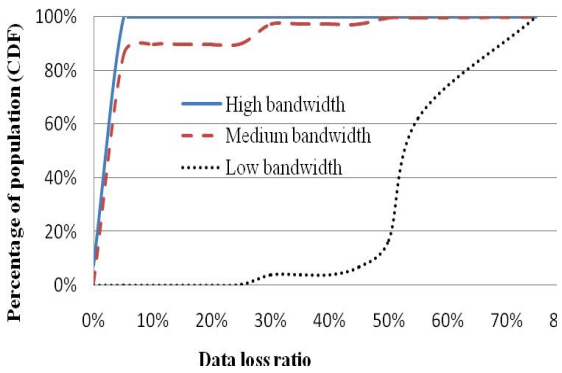
Hình 16. Tỷ lệ mất mát gói của các nút khi hệ thống có sự ra vào của các nút



Hình 17. Độ biễn thiên thời gian giữa thời gian đến của các phần khác nhau khi hệ thống có sự ra vào của các nút



Hình 18. Trung bình độ trễ của phần và trung bình độ trễ của segment từ nguồn tới nút trong hệ thống của chúng tôi khi có sự vào ra của các nút



Hình 19. Tỉ lệ mất mát gói trong trường hợp băng thông của các nút là không đồng nhất

Trong thí nghiệm cuối cùng, chúng tôi đánh giá hoạt động của hệ thống khi mà băng thông của các nút tham gia là không đồng nhất. Có khoảng 35% các nút có băng thông cao với băng thông là 700Kbps, 45% nút có băng thông trung bình với băng thông là 300Kbps, và 20% nút là các nút ích kỷ, chỉ đóng góp băng thông nhỏ hơn hoặc bằng 100Kbps. Trong kịch bản này, chúng tôi sẽ so sánh tỉ lệ giữa chất lượng video, độ trễ và khả năng đóng góp của mỗi nút để kiểm tra tính công bằng.

Hình 19 chỉ ra tỉ mất mát gói của các nút được chia ra thành từng nhóm. 100% các nút có băng thông cao có độ mất mát nhỏ hơn 5%. Khoảng 90% các nút có băng thông trung bình có độ mất mát nhỏ hơn 50%. Trong khi đó, chỉ có 60% các nút không đóng góp là có tỷ lệ lỗi nhỏ hơn 50%. Như vậy các nút đóng góp ít băng thông đã có thể nhận dữ liệu với xác suất cao hơn và với độ trễ nhỏ hơn.

Hình 20 và hình 21 chỉ ra độ trễ nhận được gói trung bình và độ trễ nhận được đầy đủ segment trung bình từ nguồn tới nút. Các nút có băng thông cao có độ trễ nhỏ hơn rất nhiều so với các nút có băng thông thấp trong khi vẫn đảm bảo được độ trễ chấp nhận được (độ trễ của các segment ở các nút tất cả đều nhỏ hơn 1.3s). Như vậy, kết quả thực nghiệm đã chứng tỏ hiệu quả của cơ chế đảm bảo tính công bằng. Các nút đóng góp nhiều có chất lượng video tốt hơn và thời gian trễ nhỏ hơn.

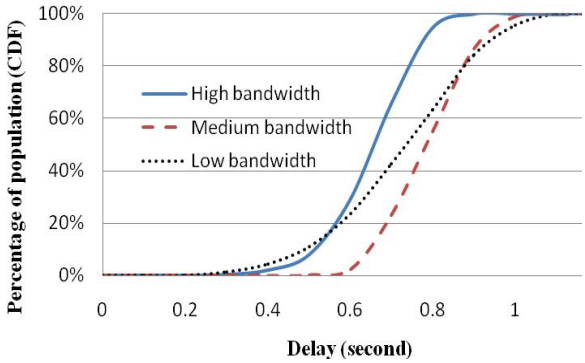


Figure 20. CDF of average parts delay from source to node when participated nodes have different bandwidth

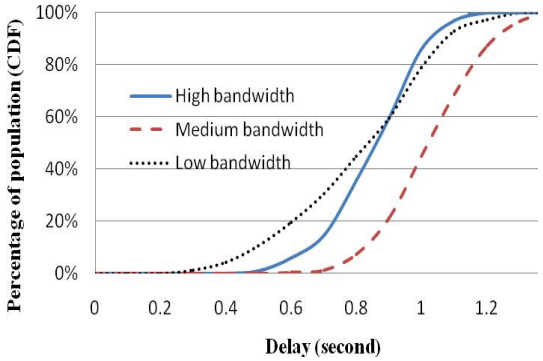


Figure 21. CDF of average segment delay from source to node when participated nodes have different bandwidth

Chương 5. Conclusion

Luận văn này trình bày một cơ chế multicast phân tán và có tính mở rộng cao cho các ứng dụng streaming video thời gian thực trên P2P. Trong hệ thống của chúng tôi, các nút được tổ chức vào các cây con riêng biệt cụ thể mà trong đó ngoại trừ nút nguồn thì mỗi nút chỉ thuộc về một cây con. Mỗi nút còn có liên kết với các nút của các cây con khác. Mỗi luồng con được truyền thông qua một cây con dựa trên cơ chế đẩy. Mỗi nút sẽ ddwuwowjc nhận từ cha của nó ít nhất một luồng dựa trên cơ chế đẩy để đảm bảo lượng dữ liệu tối thiểu. Sau đó, nút sẽ kéo các luồng con khác từ các nút khác để tăng chất lượng video. Chúng tôi tạo ra cơ chế để tạo ra các cây con cây bằng cho phép các nút ở các level gần nhau trong các cây con khác nhau có thể trao đổi dữ liệu với nhau. Do vậy, phương pháp của chúng tôi có thể làm giảng khoảng cách thời gian giữa hai luồng dữ liệu đẩy và kéo. Bên cạnh đó, chúng tôi còn đề xuất được phương pháp có thể khôi phục lỗi của hệt thống một cách nhanh chống và chính sách cân bằng để khuyến khích các nút đóng góp nhiều hơn cho mạng bằng một cơ chế tit-for-tat.

Các kết quả thực nghiệm đã chứng minh được các ưu điểm vượt trội của phương pháp được đề xuất bao gồm: giảm độ trễ hay thời gian buffer cần thiết ở các nút tham gia và khả năng khôi phục lỗi. Ngoài ra, kết quả thực nghiệm cũng chỉ ra rằng phương pháp của chúng tôi cũng hoạt động hiệu quả ngay cả trên môi trường băng thông của các nút không đồng nhất.

Trong kế hoạch sắp tới, chúng tôi sẽ cải tiến phương pháp bằng thêm và các giải pháp làm giảm thời giant ham gia của một nút, giải pháp để tối ưu cây trên toàn mạng… Sau đó chúng tôi sẽ cố gắng triên rkhai giải pahsp của chúng tôi trên môi trường kiểm tra thực ví dụ như Planet Lab để chứng minh tính hiệu quả của giải pháp đưa ra.

Publications

Huong BUI Thi Lan, Hoai Son NGUYEN(2011). A low-delay push-pull based application layer multicast for P2P live video streaming. *The third International Conference on Knowledge and System Engineering*, 2011 (to appear).