ỦY BAN NHÂN DÂN TP. HỒ CHÍ MINH

**TRƯỜNG ĐẠI HỌC SÀI GÒN**

**NGUYỄN VĂN LIẾP**

**HOÀNG CHUNG HIỀN**

**ĐỀ XUẤT THUẬT TOÁN ĐỊNH TUYẾN ĐẢM BẢO CHẤT LƯỢNG DỊCH VỤ**

**KHÓA LUẬN TỐT NGHIỆP**

**NGÀNH: CÔNG NGHỆ THÔNG TIN**

**TRÌNH ĐỘ ĐÀO TẠO: ĐẠI HỌC**

NGƯỜI HƯỚNG DẪN: ThS. CAO THÁI PHƯƠNG THANH

NGƯỜI PHẢN BIỆN: ThS. LÊ NGỌC KIM KHÁNH

**TP. HỒ CHÍ MINH, THÁNG 10 NĂM 2013**

# LỜI CAM ĐOAN

*Chúng tôi xin cam đoan đây là công trình nghiên cứu của riêng chúng tôi, các số liệu và kết quả nghiên cứu nêu trong luận văn là trung thực, được đồng tác giả cho phép sử dụng và chưa từng được công bố trong bất kì một công trình nào khác. Nếu các cam kết trên không đúng chúng tôi xin chịu hoàn mọi hình thức kỷ luật của khoa và nhà trường.*

*Tác giả luận văn*

***Nguyễn Văn Liếp Hoàng Chung Hiền***

# LỜI CẢM ƠN

Lời cảm ơn đầu tiên chúng tôi xin trân trọng gửi đến Ban Giám Hiệu Trường Đại Học Sài Gòn và các thầy cô Khoa Công Nghệ Thông Tin đã tạo điều kiện cho chúng tôi hoàn thành tốt chương trình học tập trong suốt bốn năm qua.

Đặc biệt chúng tôi xin gửi lời tri ân sâu sắc đến thầy Cao Thái Phương Thanh người đã tận tình chỉ bảo, hướng dẫn, giúp đỡ và truyền đạt cho chúng tôi những kinh nghiệm nghiên cứu khoa học quý báu không những trong khóa luận tốt nghiệp này mà còn trong suốt những năm tháng chúng tôi học ở trường đại học Sài Gòn.

Lời cảm ơn sau cùng chúng tôi xin gửi đến bạn bè và những người thân những đã chia sẻ kinh nghiệm, động viên và ủng hộ mọi mặt về vật chất và tinh thần để chúng tôi đạt được thành quả như ngày hôm nay.

# NHẬN XÉT CỦA NGƯỜI HƯỚNG DẪN

# NHẬN XÉT CỦA NGƯỜI PHẢN BIỆN

# NHẬN XÉT CỦA CHỦ TỊCH HỘI ĐỒNG

# **MỤC LỤC**

[LỜI CAM ĐOAN 2](#_Toc369856023)

[LỜI CẢM ƠN 3](#_Toc369856024)

[MỤC LỤC 7](#_Toc369856025)

[DANH SÁCH HÌNH ẢNH 9](#_Toc369856026)

[DANH SÁCH BẢNG 10](#_Toc369856027)

[DANH SÁCH TỪ VIẾT TẮT 11](#_Toc369856028)

[MỞ ĐẦU 12](#_Toc369856029)

[CHƯƠNG 1 GIỚI THIỆU 15](#_Toc369856030)

[1.1 Giới thiệu 15](#_Toc369856031)

[1.2 Các khái niệm cơ bản 15](#_Toc369856032)

[1.2.1 Mô hình đồ thị trọng số 15](#_Toc369856033)

[1.2.2 Kĩ thuật lưu lượng 16](#_Toc369856034)

[1.2.3 Thước đo chất lượng dịch vụ 16](#_Toc369856035)

[1.2.4 Vấn đề định tuyến (Unicast) 17](#_Toc369856036)

[1.2.5 Các chiến lược định tuyến 19](#_Toc369856037)

[CHƯƠNG 2 MÔ TẢ VẤN ĐỀ VÀ CÁC CÔNG TRÌNH LIÊN QUAN 21](#_Toc369856038)

[2.1 Mô tả vấn đề và kí hiệu 21](#_Toc369856039)

[2.2 Các công trình liên quan 22](#_Toc369856040)

[2.2.1 Vấn đề định tuyến ràng buộc băng thông 22](#_Toc369856041)

[2.2.1.1 MHA - Min-Hop Algorithm 22](#_Toc369856042)

[2.2.1.2 BCRA - Bandwidth Constrained Routing Algorithm 23](#_Toc369856043)

[2.2.1.3 MIRA - Minimum Interference Algorithm 25](#_Toc369856044)

[2.2.2 Vấn đề định tuyến thỏa mãn ràng buộc băng thông và độ trễ 27](#_Toc369856045)

[2.2.2.1 Min-Delay (Minimun Delay) 27](#_Toc369856046)

[2.2.2.2 BDCRA – Bandwidth-Delay Constrained Routing Algorithm 28](#_Toc369856047)

[2.2.2.3 MDWCRA – Maximum Delay-Weighted Capacity Routing Algorithm 30](#_Toc369856048)

[CHƯƠNG 3 ĐỀ XUẤT THUẬT TOÁN 35](#_Toc369856049)

[3.1 OBDCRA – Optimized Bandwidth Delay Constrained Routing Algorithm 35](#_Toc369856050)

[3.1.1 Hàm tính toán trọng số - phân phối tải 36](#_Toc369856051)

[3.1.2 Vấn đề tìm đường đi ngắn nhất thỏa mãn một ràng buộc 36](#_Toc369856052)

[3.1.2.1 Xác định vấn đề 36](#_Toc369856053)

[3.1.2.2 Thuật toán heuristic C-Dijkstra 37](#_Toc369856054)

[3.1.3 Thuật toán định tuyến OBDCRA 39](#_Toc369856055)

[3.1.4 Thuật toán định tuyến M-OBDCRA 45](#_Toc369856056)

[CHƯƠNG 4 THỰC NGHIỆM VÀ ĐÁNH GIÁ KẾT QUẢ 53](#_Toc369856057)

[4.1 Môi trường giả lập 53](#_Toc369856058)

[4.1.1 Giới thiệu 53](#_Toc369856059)

[4.1.2 Phân tích thiết kế 53](#_Toc369856060)

[4.2 Thực nghiệm và kết quả 60](#_Toc369856061)

[4.2.1 Thiết lập mạng 60](#_Toc369856062)

[4.2.2 Thực nghiệm 63](#_Toc369856063)

[4.2.3 Kết quả 64](#_Toc369856064)

[CHƯƠNG 5 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 74](#_Toc369856065)

[5.1 Kết quả đạt được 74](#_Toc369856066)

[5.2 Hạn chế 74](#_Toc369856067)

[5.3 Hướng phát triển 74](#_Toc369856068)

[TÀI LIỆU THAM KHẢO 75](#_Toc369856069)

[PHỤ LỤC 77](#_Toc369856070)

# DANH SÁCH HÌNH ẢNH

[Ảnh 1.1 Trạng thái mạng 16](#_Toc369509845)

[Ảnh 1.2 Các vấn đề định tuyến Unicast 18](#_Toc369509846)

[Ảnh 3.1 Các bước thực hiện của thuật toán định tuyến OBDCRA. 45](#_Toc369509847)

[Ảnh 3.2 Các bước thực hiện của thuật toán định tuyến OBDCRA trong trường hợp không tìm được đường đi tối ưu. 50](#_Toc369509848)

[Ảnh 3.3 Kết quả đường đi tìm được của thuật toán định tuyến M-OBDCRA. 51](#_Toc369509849)

[Ảnh 4.1 Sơ đồ tổng thể của bộ giả lập – SRAS 54](#_Toc369509850)

[Ảnh 4.2 Các thành phần của bộ giả lập định tuyến. 55](#_Toc369509851)

[Ảnh 4.3 Sơ đồ lớp của bộ giả lập định tuyến 56](file:///C:\Users\Liep%20Nguyen\Documents\Visual%20Studio%202012\Projects\NetworkSimulator_Delay\NetworkSimulator\NetworkSimulator\Documents\report.docx#_Toc369509852)

[Ảnh 4.4 Lược đồ tuần tự của chức năng giả lập định tuyến 59](file:///C:\Users\Liep%20Nguyen\Documents\Visual%20Studio%202012\Projects\NetworkSimulator_Delay\NetworkSimulator\NetworkSimulator\Documents\report.docx#_Toc369509853)

[Ảnh 4.5 Luồng con thực hiện việc định tuyến của bộ giả lập. 59](file:///C:\Users\Liep%20Nguyen\Documents\Visual%20Studio%202012\Projects\NetworkSimulator_Delay\NetworkSimulator\NetworkSimulator\Documents\report.docx#_Toc369509854)

[Ảnh 4.6 Luồng con thực hiện việc giải phóng băng thông của bộ giả lập. 59](file:///C:\Users\Liep%20Nguyen\Documents\Visual%20Studio%202012\Projects\NetworkSimulator_Delay\NetworkSimulator\NetworkSimulator\Documents\report.docx#_Toc369509855)

[Ảnh 4.7 Sơ đồ mạng MIRA 60](#_Toc369509856)

[Ảnh 4.8 Sơ đồ mạng ANSNET mở rộng 61](#_Toc369509857)

[Ảnh 4.9 Sơ đồ mạng NET1 với 50 nút được tạo ra từ bộ phát sinh sơ đồ mạng của Waxman với thông số ⍺ = 0.13, β = 0.99. 62](#_Toc369509858)

[Ảnh 4.10 Đồ thị tỉ lệ từ chối yêu cầu (mạng MIRA, băng băng  {10, 20, 30, 40}, độ trễ  [35, 45] ms) 65](#_Toc369509859)

[Ảnh 4.11 Đồ thị tỉ lệ từ chối yêu cầu (mạng ANSNET mở rộng, băng băng  {10, 20, 30, 40, 50}, độ trễ  [40, 50] ms) 66](#_Toc369509860)

[Ảnh 4.12 Đồ thị tỉ lệ từ chối yêu cầu (mạng NET1 mở rộng, băng băng  {10, 20, 30, 40, 50}, độ trễ  [15, 25] ms) 67](#_Toc369509861)

[Ảnh 4.13 Đồ thị thời gian tính toán trung bình (mạng MIRA, băng băng  {10, 20, 30, 40}, độ trễ  [35, 45] ms) 70](#_Toc369509862)

[Ảnh 4.14 Đồ thị thời gian tính toán trung bình (mạng ANSNET mở rộng, băng băng  {10, 20, 30, 40, 50}, độ trễ  [40, 50] ms) 71](#_Toc369509863)

[Ảnh 4.15 Đồ thị thời gian tính toán trung bình (mạng NET1 mở rộng, băng băng  {10, 20, 30, 40, 50}, độ trễ  [15, 25] ms) 72](#_Toc369509864)

# DANH SÁCH BẢNG

[Bảng 4.1 Tỉ lệ từ chối yêu cầu (mạng MIRA, băng băng  {10, 20, 30, 40}, độ trễ  [35, 45] ms) 65](#_Toc369509865)

[Bảng 4.2 Tỉ lệ từ chối yêu cầu (mạng ANSNET mở rộng, băng băng  {10, 20, 30, 40, 50}, độ trễ  [40, 50] ms) 66](#_Toc369509866)

[Bảng 4.3 Tỉ lệ từ chối yêu cầu (mạng NET1 mở rộng, băng băng  {10, 20, 30, 40, 50}, độ trễ  [15, 25] ms) 68](#_Toc369509867)

[Bảng 4.4 Số liệu thời gian tính toán trung bình (ms) (mạng MIRA, băng băng  {10, 20, 30, 40}, độ trễ  [35, 45] ms) 70](#_Toc369509868)

[Bảng 4.5 Số liệu thời gian tính toán trung bình (ms) (mạng ANSNET mở rộng, băng băng  {10, 20, 30, 40, 50}, độ trễ  [40, 50] ms) 71](#_Toc369509869)

[Bảng 4.6 Số liệu thời gian tính toán trung bình (ms) (mạng NET1 mở rộng, băng băng  {10, 20, 30, 40, 50}, độ trễ  [15, 25] ms) 73](#_Toc369509870)

# DANH SÁCH TỪ VIẾT TẮT

|  |  |  |
| --- | --- | --- |
| VoD | : | Video on Demand |
| QoS | : | Quality of Service |
| IP | : | Internet Protocol |
| MPLS | : | Multiprotocol Lable Switching |
| TE | : | Traffic Engineering |
| PCPO | : | Path-Constrained Path-Optimization |
| MPC | : | Multi-Path-Constrained |
| MHA | : | Min Hop Algorithm |
| BFS | : | Breath first search |
| BCRA | : | Bandwidth Constrained Routing Algorithm |
| MIRA | : | Minimum Interference Algorithm |
| BDCRA | : | Bandwidth-Delay Constrained Routing Algorithm |
| LDP | : | Least Delay Path |
| MDWCRA | : | Maximum Delay-Weighted Capacity Routing Algorithm |
| DWC | : | Delay-Weighted Capacity |
| EDSP | : | Extended Dijkstra Shortest Path |
| EBF | : | Extended Bellman-Ford |
| M-MDWDCRA | : | Modified-Maximum Delay-Weighted Capacity Routing Algorithm |
| LWP | : | Least Weight Path |
| OBDCRA | : | Optimized Bandwidth Delay Constrained Routing Algorithm |
| M-OBDCRA | : | Modified-Optimized Bandwidth Delay Constrained Routing Algorithm |
| SRAS | : | Simple Routing Algorithm Simulator |
| LDPRA | : | Least Delay Path Routing Algorithm |

# MỞ ĐẦU

Chất lượng dịch vụ (QoS – Quality of Service) ngày càng quan trọng trong các hệ thống mạng ngày nay. Các hệ thống mạng này truyền tải vô số các ứng dụng và dữ liệu, bao gồm truyền hình chất lượng cao (high-quality video), dữ liệu yêu cầu độ trễ thấp (delay-sensitive data) như nói chuyện thời gian thực (real-time voice) và các ứng dụng có cường độ băng thông lớn (bandwidth-intensive applications). Một trong những vấn đề chính là việc định tuyến QoS cho phép lựa chọn các tuyến đường mạng với tài nguyên thỏa mãn các thông số QoS theo yêu cầu như băng thông (bandwidth), các ràng buộc như độ trễ (delay), biến thiên độ trễ (delay variation – jitter), mất gói tin (packet loss) trên mạng. Mục đích của các giải pháp định tuyến QoS là để tìm đường đi cho mỗi kết nối đến thỏa mãn các yêu cầu QoS của ứng dụng một cách hiệu quả bằng việc sử dụng tài nguyên và các thuật toán định tuyến cơ bản để tìm đường đi với chi phí thấp (low-cost) thỏa mãn các thông số QoS đã cho. Mặc dù nhiều công trình nghiên cứu được thực hiện để tối ưu hóa hiệu năng của vấn đề định tuyến nhưng đa số tập trung vào việc thỏa mãn yêu cầu băng thông mà khá ít nghiên cứu được thực hiện để tìm đường đi thỏa mãn cả băng thông và ràng buộc độ trễ. Vấn đề tìm đường đi chi phí thấp thỏa mãn một hoặc nhiều ràng buộc được biết là *NP-đầy đủ* và vài thuật toán đã được đề xuất để tìm kiếm giải pháp cận tối ưu. Nhưng những thuật toán này đều lợi dụng vào các mối liên hệ của các số liệu trên liên kết nhằm hạn chế độ phức tạp mà từ đó có thể làm giới hạn tính ứng dụng chung hoặc quá tốn kém về thời gian thực thi có thể chấp nhận với các hệ thống mạng lớn.

Trong khóa luận này, chúng tôi sẽ đề xuất một thuật toán định tuyến thỏa mãn băng thông-độ trễ và tối ưu hóa việc sử dụng tài nguyên mạng với thời gian thực thi có thể chấp nhận với các hệ thống mạng lớn.

**Nội dung khóa luận gồm 5 chương:**

* Chương 1: Giới thiệu

*Giới thiệu một cách tổng quan về vấn đề định tuyến QoS và một số kiến thức liên quan.*

* Chương 2: Mô tả vấn đề và các công trình liên quan

*Xác định vấn đề, quy ước – kí hiệu, trình bày một số thuật toán định tuyến liên quan thuộc hai nhóm vấn đề là vấn đề định tuyến đảm bảo băng thông và vấn đề định tuyến đảm bảo băng thông ràng buộc độ trễ.*

* Chương 3: Đề xuất thuật toán

*Đề xuất một thuật toán định tuyến có tên là OBDCRA – Optimized Bandwidth-Delay Constrained Routing Algorithm và một cải tiến của thuật toán này là M-OBDCRA (Modified-OBDCRA).*

* Chương 4: Thực nghiệm và đánh giá kết quả thuật toán

*Trình bày môi trường giả lập và phương pháp thực nghiệm để đánh giá hiệu quả của các thuật toán khác nhau.*

* Chương 5: Kết luận và hướng phát triển

*Trình bày kết quả đã đạt được, hạn chế, và hướng phát triển trong tương lai.*

**Đối tượng và phương pháp nghiên cứu**

* *Đối tượng:*
* Các thuật toán định tuyến đảm bảo băng thông
* Các thuật toán định tuyến thỏa mãn băng thông và độ trễ
* Phương pháp giả lập và đánh giá kết quả
* Các vấn đề liên quan đến việc định tuyến đảm bảo chất lượng dịch vụ
* *Phương pháp nghiên cứu:*

Xác định làm rõ vấn đề, đọc và tìm hiểu các công trình nghiên cứu liên quan (báo khoa học, luận văn ...) để rút kết kiến thức. Phân tích và đánh giá ưu khuyết điểm của từng đề xuất, kế thừa ưu điểm và cố gắng hạn chế các khuyết điểm có thể mắc phải khi đề xuất thuật toán mới. Tìm hiểu phương pháp giả lập và đánh giá kết quả thuật toán.

**Sản phẩm dự kiến và ý nghĩa của đề tài**

**Sản phẩm dự kiến**

Sản phẩm dự kiến trong luận văn này mà chúng tôi mong muốn đạt được là một thuật toán định tuyến mới thỏa mãn ràng buộc băng thông-độ trễ và hiệu quả trong việc sử dụng tài nguyên mạng với thời gian tính toán có thể chấp nhận được với các hệ thống mạng lớn. Ngoài ra chúng tôi sẽ xây dựng một môi trường giả lập đơn giản để cài đặt cũng như đánh giá hiệu năng của các thuật toán định tuyến khác nhau.

**Ý nghĩa**

*Về mặt khoa học*

Đóng góp cho nền khoa học về truyền thông máy tính một thuật toán mới để giải quyết vấn đề định tuyến thỏa mãn băng thông và độ trễ với hiệu quả và thời gian tính toán có thể được chấp nhận trên các hệ thống mạng lớn. Từ đó, tạo tiền đề để phát triển tiếp các thuật toán khác hiệu quả hơn.

*Về mặt thực tiễn*

Trong nhiều năm gần đây, nhu cầu ngày càng cao về việc sử dụng các dịch vụ thời gian thực và băng thông rộng như hội nghị truyền hình, truyền hình chất lượng cao, đối thoại thời gian thực, trò chơi trực tuyến, ... đòi hỏi các nhà cung cấp phải tìm ra một giải pháp mới để sử dụng tốt hạ tầng mạng hiện có mà vẫn đáp ứng tốt những nhu cầu trên. OBDCRA là một lựa chọn tốt để giải quyết các vấn đề đó.

# GIỚI THIỆU

## Giới thiệu

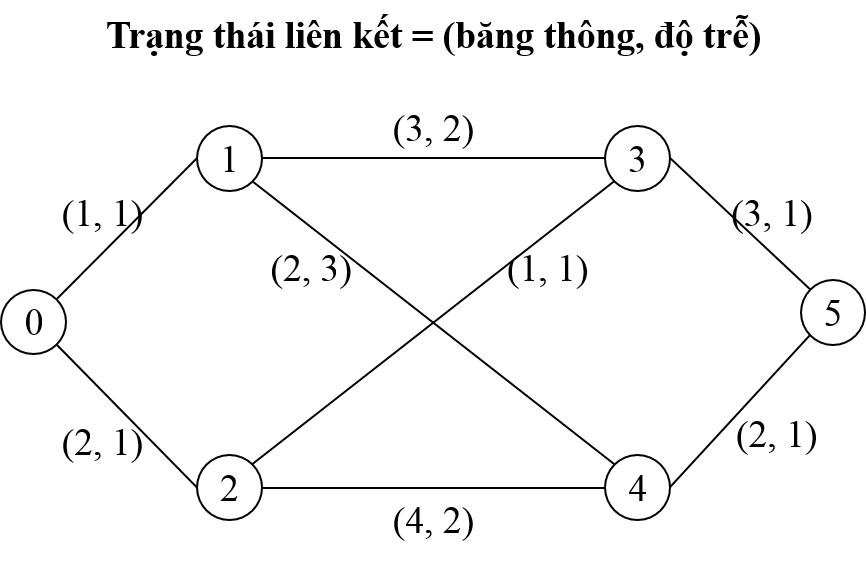
Cùng với sự phát triển của mạng máy tính, các tiến bộ gần đây trong công nghệ mạng tốc độ cao (high-speed network) đã tạo nên nhiều thuận lợi cho việc phát triển các ứng dụng đa phương tiện như là văn bản, hình ảnh, âm thanh và truyền hình. Nhiều ứng dụng thời gian thực như truyền hình theo yêu cầu (VoD – video on demand), hội nghị truyền hình (video conferencing), nói chuyện thời gian thực (real-time voice) đã được triển khai trên internet. Những ứng dụng này yêu cầu hạ tầng mạng cung cấp dịch vụ đảm bảo cho người nhận về các thông số QoS. Các ứng dụng khác nhau thì có những yêu cầu QoS khác nhau để đạt được sự thỏa mãn của người dùng. Các yêu cầu QoS được đo bằng các thông số như: bandwidth, delay, jitter, packet loss và cost. Tuy nhiên, đa số các hệ thống mạng vẫn dựa vào IP. Từ lâu IP là một giao thức phi kết nối (connectionless protocol), các gói tin IP không sử dụng đường đi cụ thể giữa hai điểm kết nối hay nói cách khác chúng có thể đi theo các con đường khác nhau tới đích. Điều này dẫn đến việc không thể dự đoán được QoS trong mạng best-effort (như mạng IP). Ngược lại, chuẩn kết nối có định hướng (connection oriented) như mạng chuyển nhãn đa giao thức (MPLS – Multiprotocol Label Switching) cho phép kiểm soát tốt hơn trong việc định tuyến lưu lượng và kĩ thuật lưu lượng.

## Các khái niệm cơ bản

### Mô hình đồ thị trọng số

Một hệ thống mạng có thể được mô hình hóa như một đồ thị *G=(N, L)*. Với các nút (*N*) của đồ thị đại diện cho các bộ định tuyến, bộ chuyển mạch, ... và các cạnh (*L*) đại diện cho các liên kết truyền thông. Các cạnh là vô hướng nếu các liên kết truyền thông là đối xứng. Một liên kết đối xứng có cùng khả năng thông qua, độ trễ truyền tải, … và cùng lưu lượng mạng ở cả hai chiều. Đa số các hệ thống mạng thực tế, các liên kết là bất đối xứng và do đó mỗi liên kết được biểu diễn bởi cạnh có hai chiều đối ngược nhau. Trong bài báo cáo này chúng tôi đã trình bày các thuật toán định tuyến được thiết kế cho mạng bất đối xứng.

Mỗi liên kết có một trạng thái được đo bằng các thước đo QoS liên quan. Như Ảnh 1.1, trạng thái liên kết là một cặp bao gồm băng thông, độ trễ.



Ảnh 1.1 Trạng thái mạng

### Kĩ thuật lưu lượng

Kĩ thuật lưu lượng (TE - Traffic Engineering) là kĩ thuật điều khiển đường truyền chứa lưu lượng qua mạng. Mục đích để cải thiện việc sử dụng tài nguyên mạng, tránh trường hợp một phần tử mạng bị nghẽn trong khi các phần tử khác chưa được dùng hết. Ngoài ra, còn để đảm bảo đường truyền có các thuộc tính nhất định, tài nguyên truyền dẫn có sẵn trên một đường truyền cụ thể hay xác định luồng lưu lượng nào được ưu tiên lúc xảy ra tranh chấp tài nguyên.

### Thước đo chất lượng dịch vụ

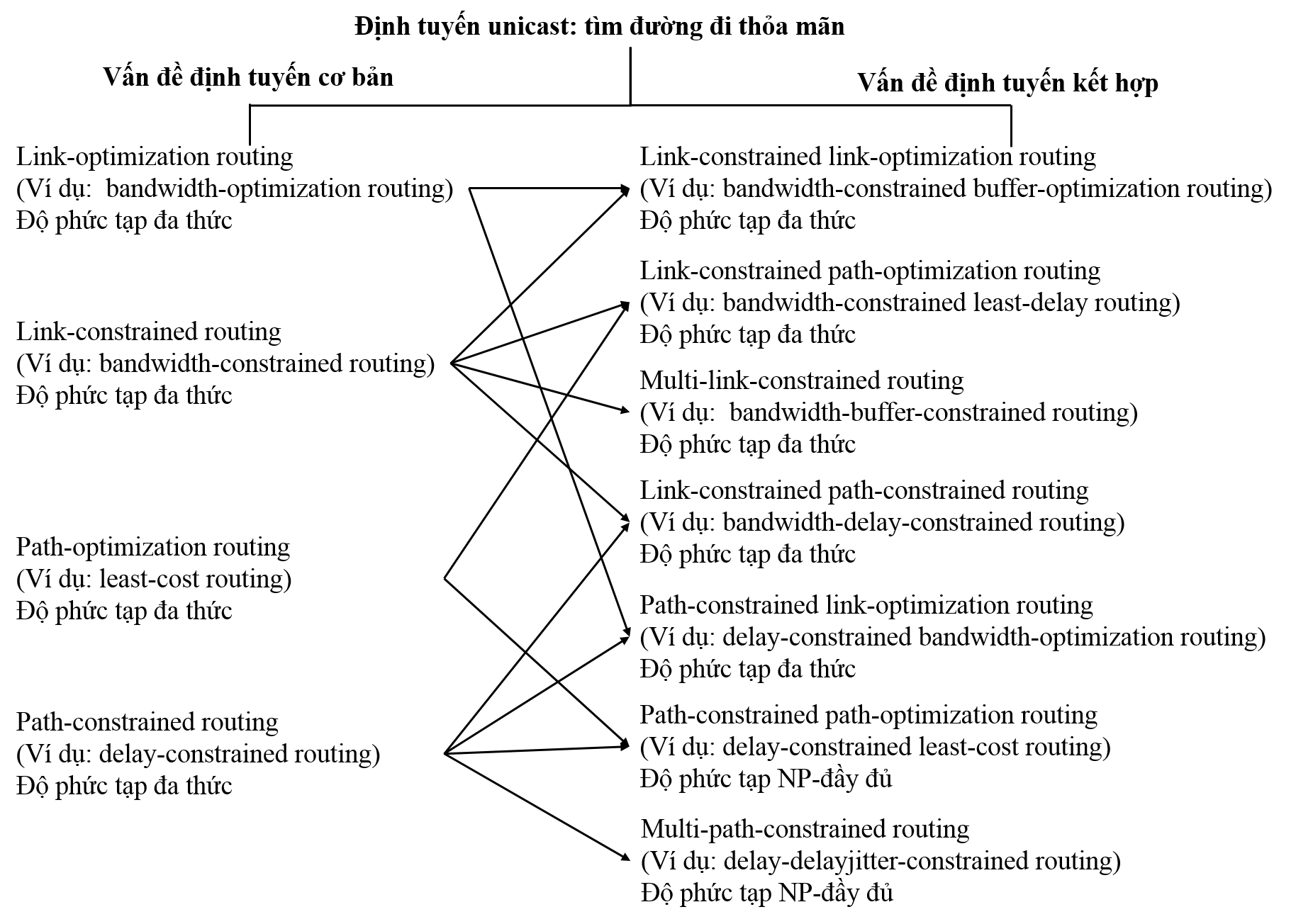
Các yêu cầu QoS được đo bằng các thông số như: bandwidth, delay, jitter, packet loss và cost. Các thước đo QoS có thể được chia thành ba loại: cộng dồn, nhân dồn và lõm [1, tr. 6]. Cho *m(li)* là thước đo cho liên kết *li*. Với bất kì đường đi *P = (l1, …, li)*, thước đo m là:

* Cộng dồn, nếu *m(P) =* *m(l1) + m(l2) + … + m(li)* (ví dụ là delay, jitter, cost, hop-count)
* Nhân đồn, nếu *m(P) = m(l1) \* m(l2) \* … \* m(li)* (ví dụ là độ tin cậy, trong trường hợp này *0 ≤ m(li) ≤ 1*)
* Lõm, nếu *m(P) = min{m(l1), m(l2), …, m(li)}* (ví dụ như băng thông, có nghĩa là băng thông của con đường như một tổng thể được xác định bởi liên kết với băng thông sẵn sàng nhỏ nhất)

### Vấn đề định tuyến (Unicast)

Vấn đề định tuyến có thể được chia thành hai lớp lớn: *định tuyến Unicast* và *định tuyến Multicast*. Vấn đề định tuyến Unicast được định nghĩa như sau: cho một nút nguồn *s*, một nút đích *d*, một tập các ràng buộc *C* và có thể là một mục tiêu tối ưu nào đó, tìm đường đi từ *s* đến *d* thỏa mãn ràng buộc trong *C*. Vấn đề định tuyến Multicast được định nghĩa như sau: cho một nút nguồn *s*, một tập các nút đích *R*, một tập các ràng buộc *C* và có thể là một mục tiêu tối ưu nào đó, tìm kiếm cây thỏa mãn bao phủ từ *s* đến tất cả các nút trong *R* thỏa mãn ràng buộc *C*. Hai lớp vấn đề định tuyến này liên quan chặt chẽ với nhau. Trong nhiều trường hợp, định tuyến Multicast có thể được xem như trường hợp tổng quát của định tuyến Unicast. Trong bài báo cáo này, chúng tôi chỉ tập trung vào vấn đề định tuyến Unicast*.* Vấn đề định tuyến *Unicast* có thể được chia nhỏ thành các lớp con như sau:

* Với thước đo lõm, chẳng hạn như băng thông khả dụng và không gian bộ đệm khả dụng, trạng thái của đường đi được xác định bởi trạng thái của liên kết hẹp (bottleneck link). Ví dụ, trong Ảnh 1.1 băng thông của đường đi 0 → 1 → 4 → 5 là 1 và cũng là băng thông của liên kết hẹp (0, 1). Với những thước đo QoS này, hai vấn đề định tuyến được định nghĩa như sau: vấn đề thứ nhất được gọi là *định tuyến tối ưu-liên kết (link-optimization routing).* Ví dụ như định tuyến tối ưu-băng thông, tìm đường đi có băng thông rộng nhất trên liên kết hẹp. Như vậy đường đi này được gọi là đường đi rộng nhất *(widest path)* [2]. Vấn đề thứ hai được gọi là *định tuyến ràng buộc-liên kết (link-constrained routing).* Ví dụ như định tuyến ràng buộc-băng thông, tìm đường đi có băng thông hẹp thỏa mãn băng thông yêu cầu. Vấn đề định tuyến link-optimization có thể được giải quyết bằng cách chỉnh sửa giải thuật của Dijkstra hoặc giải thuật Bellman-Ford. Vấn đề định tuyến link-constrained có thể dễ dàng chuyển hóa về vấn đề link-optimization.



Ảnh 1.2 Các vấn đề định tuyến Unicast

* Với các thước đo QoS cộng dồn như delay, delay jitter và cost, trạng thái của đường đi được xác định bởi trạng thái kết hợp qua tất cả các liên kết trên đường đi. Ví dụ, như Ảnh 1.1 độ trễ của đường đi 0 → 1 → 4 → 5 là 5, là tổng độ trễ của tất cả các liên kết trên đường đi. Hai vấn đề định tuyến cơ bản được định nghĩa cho loại thước đo QoS này: vấn đề thứ nhất được gọi là *định tuyến tối ưu-đường đi (path-optimization routing)*. Ví dụ như định tuyến chi phí-thấp nhất (least-cost routing), tìm đường đi có tổng chi phí là cực tiểu. Vấn đề thứ hai được gọi là *định tuyến ràng buộc-đường đi (path-constrained routing).* Ví dụ như định tuyến ràng buộc-độ trễ (delay-constrained routing), tìm đường đi mà độ trễ bị giới hạn bởi một giá trị yêu cầu. Cả hai vấn đề này có thể được giải quyết trực tiếp bằng giải thuật của Dijkstra (hoặc giải thuật Bellman-Ford).

Nhiều vấn đề định tuyến kết hợp có thể được chuyển hóa từ bốn vấn đề cơ bản như Ảnh 1.1. Ví dụ, vấn đề ràng buộc-băng thông độ trễ-thấp nhất (bandwidth-constrained least-delay routing) thuộc vào lớp vấn đề *ràng buộc-liên kết tối ưu-đường đi (link-constrained path-optimization routing)* với đòi hỏi tìm ra đường đi có độ trễ thấp nhất mà vẫn thỏa mãn băng thông yêu cầu. Vấn đề này có thể được giải quyết bằng thuật toán tìm đường đi ngắn nhất trên đồ thị với các liên kết có băng thông không thỏa bị loại bỏ. Có bốn lớp vấn đề có thể giải quyết trong thời gian đa thức bằng cách chỉnh sửa giải thuật tìm đường đi ngắn nhất đó là *định tuyến ràng buộc-liên kết tối ưu-liên kết (link-constrained link-optimization routing)*, *định tuyến đa ràng buộc-liên kết (multi-link-constrained routing)*, *định tuyến ràng buộc-liên kết ràng buộc-đường đi* *(link-constrained path-constrained routing)* và *định tuyến ràng buộc-đường đi tối ưu-liên kết (path-constrained link-optimization routing)*.

Có hai lớp vấn đề NP-đầy đủ mà đặc biệt được chú ý là *định tuyến ràng buộc-đường đi tối ưu-đường đi (path-constrained path-optimization routing)* (PCPO) và *định tuyến nhiều ràng buộc-đường đi (multi-path-constrained routing)* (MPC). Một ví dụ của PCPO là định tuyến ràng buộc – độ trễ chi phí – thấp nhất đòi hỏi tìm ra đường đi có chi phí thấp nhất mà vẫn thỏa mãn ràng buộc độ trễ. Một ví dụ của MPC là định tuyến ràng buộc – độ trễ – biến thiên độ trễ với đòi hỏi tìm ra đường đi sao cho thỏa mãn cả giới hạn độ trễ và giới hạn biến thiên độ trễ. Những vấn đề trên là *NP-đầy đủ*, nếu tất cả các thước đo ngoại trừ một thước đo được giới hạn trong tập các giá trị nguyên thì các vấn đề trên có thể được giải quyết được trong thời gian đa thức bằng thuật toán mở rộng của Dijkstra (Bellman-Ford). Nếu tất cả thước đo phụ thuộc vào một thước đo chung thì các vấn đề có thể giải quyết trong thời gian đa thức.

### Các chiến lược định tuyến

Việc định tuyến bao gồm hai tác vụ, tác vụ thu thập thông tin trạng thái luôn được cập nhật và tác vụ tìm kiếm thông tin trạng thái cho các đường đi thỏa mãn. Để việc tìm kiếm đường đi tối ưu thỏa mãn các ràng buộc thì thông tin trạng thái về các liên kết trung gian giữa nguồn và đích phải được xác định. Việc tìm kiếm đường đi thỏa mãn phụ thuộc vào cách mà thông tin trạng thái được thu thập và lưu trữ. Có ba chiến lược định tuyến: *định tuyến nguồn (source routing)*, *định tuyến phân phối (distributed routing)*, *định tuyến thứ bậc (hierarchical routing)* [1, tr. 15]. Tất cả được phân chia theo cách mà thông tin trạng thái được duy trì và cách mà việc tìm kiếm đường đi thỏa mãn được thi hành. Trong định tuyến nguồn, mỗi nút duy trì trạng thái toàn cục một cách hoàn chỉnh và đường đi thỏa mãn được tính toán cục bộ tại nút nguồn. Thông điệp điều khiển được phát ra dọc theo đường đi đã chọn để cung cấp thông tin cho các nút trung gian của các nút trước và kế tiếp chúng. Giao thức trạng thái liên kết được sử dụng để cập nhật thông tin toàn cục cho mỗi nút. Trong định tuyến phân phối, một đường đi được tính toán bằng việc tính toán phân phối. Thông điệp điều khiển được hoán đổi giữa các nút và thông tin trạng thái được giữ tại mỗi nút được sử dụng cho việc tìm kiếm đường đi một cách tập trung. Đa số các thuật toán định tuyến phân phối cần giao thức vector khoảng cách (hoặc giao thức trang thái liên kết) để duy trì trạng thái toàn cục ở dạng các vector khoảng cách ở mỗi nút. Dựa vào vector khoảng cách, việc định tuyến được thực hiện trên từng nút cơ sở. trong định tuyến thứ bậc, các nút được gộp thành từng nhóm. Mỗi nút vật lí duy trì một trạng thái toàn cục tập trung. Trạng thái này chứa thông tin trạng thái chi tiết về các nút trong cùng một nhóm và thông tin trạng thái tập trung về các nhóm khác. Định tuyến nguồn được sử dụng để tìm đường đi thỏa mãn trên nút mà là nút đại diện cho nhóm. Thông điệp điều khiển sau đó được gửi dọc theo đường đi này để thiết lập kết nối. Khi nút biên của một nhóm được đại diện bởi một nút logical nhận được thông điệp thì nút này sẽ sử dụng định tuyến nguồn để mở rộng đường đi thông qua nhóm.

# MÔ TẢ VẤN ĐỀ VÀ CÁC CÔNG TRÌNH LIÊN QUAN

## Mô tả vấn đề và kí hiệu

Cho một sơ đồ mạng với n nút. Một tập con của những nút này được giả sử là các cặp nút vào-ra (ingress-egress) mà giữa đó các đường đi có thể được thiết lập. Chúng tôi giả định rằng các nút vào-ra đã biết và hiếm khi thay đổi. Mỗi liên kết trong mạng có các thuộc tính: khả năng thông qua, băng thông còn lại, độ trễ. Băng thông còn lại được định nghĩa như là hiệu của khả năng thông qua với tổng băng thông của tất cả các đường đi đã ấn định vào liên kết đó. Độ trễ của một liên kết bao gồm độ trễ lan truyền và độ trễ hàng đợi tại nút bắt đầu.

Một vài kí hiệu được sử dụng trong này được định nghĩa như sau, mạng được mô hình hóa như một đồ thị có hướng *G = (N, L)* bao gồm một tập *n* nút *N = {n1, n2, …, nn}* và một tập *m* liên kết *L = {l1, l2, …, lm}* gọi  là liên kết có nút đầu *u* và nút *v* với  và D được xem như tập các cặp vào-ra. Với mỗi liên kết có khả năng thông qua *c(l)*, băng thông còn lại *r(l)* và độ trễ *dl(l)* như vậy:

*c(l), r(l), dl(l) : L →* *ℝ+*

Cho *R = (r0, …, ri, …)* là một chuỗi (vô hạn) các yêu cầu kết nối đến mạng, với mỗi yêu cầu kết nối *ri = (s, d, b, ∆d)* được gửi từ nút nguồn nào đó *s ∈ N* ,xác định cụ thể địa chỉ của nút đích *d ∈ N,* băng thông yêu cầu *b* *∈ ℝ+* và giới hạn độ trễ tối đa *∆d ∈ ℝ+.* Với *Psd* là tập tất cả các đường đi có thể từ nút *s* đến nút *d*, thuật toán định tuyến QoS phải tìm đường đi thích hợp *psd = (l1, l2, …, lk)* với *p ∈ Psd* và đáp ứng các yêu cầu QoS mong muốn là:

1. Băng thông yêu cầu:

|  |  |  |
| --- | --- | --- |
|  |  | (2.1) |

1. Ràng buộc độ trễ tối đa:

|  |  |  |
| --- | --- | --- |
|  |  | (2.2) |

Nếu đường đi không thỏa mãn các ràng buộc này thì yêu cầu kết nối bị chặn và loại bỏ. Khi các yêu cầu đều thỏa mãn, kết nối được thiết lập và băng thông được giữ để phục vụ cho yêu cầu cho đến khi kết nối được giải phóng (release).

## Các công trình liên quan

Định tuyến QoS có thể được xem như tổ hợp của bốn lớp cơ bản, có tên là: *link-optimization routing*, *link-constrained routing*, *path-optimization routing* và *path-constrained routing* [1, tr. 10]. Link-optimization routing và link-constrained routing được định nghĩa cho các thước đo lõm như băng thông và không gian bộ đệm. Path-optimization routingvà path-constrained routingthì được định nghĩa cho thước đo cộng đồn và nhân đồn như delay và delay jitter. Tổ hợp các vấn đề định tuyến có thể được chuyển hóa từ bốn vấn đề định tuyến cơ bản này.

### Vấn đề định tuyến ràng buộc băng thông

Vấn đề định tuyến ràng buộc băng thông thuộc về vấn đề link-constrained routing được giải quyết nhiều nhất [3] [4] [5].

#### MHA - Min-Hop Algorithm

**Mô tả**

Thuật toán MHA [3] lựa chọn đường đi *psd* thỏa mãn băng thông và với số hop hay số liên kết ít nhất có thể.

Khi yêu cầu định tuyến được gửi đến, MHA loại bỏ những kiên kết có băng thông còn lại nhỏ hơn băng thông yêu cầu, sau đó sử dụng thuật toán tìm đường đi ngắn nhất như: thuật toán Dijkstra, Bellman-Ford với trọng số liên kết là 1 hay thuật toán duyệt rộng BFS …

**Mã giả**

|  |
| --- |
| **MIN-HOP ALGORITHM (MHA)**  **INPUT:**   * Đồ thị *G = (N, L, D)* và yêu cầu định tuyến *r(s, d, b)* từ nút *s* đến nút *d* với băng thông yêu cầu là *b*.   **OUTPUT:**   * Một tuyến đường từ s đến d thỏa mãn băng thông yêu cầu b.   **ALGORTHIM:**   1. Duyệt tất cả liên kết *l* và gán *w(l) = 1* 2. Loại bỏ các liên kết có băng thông còn lại nhỏ hơn băng thông yêu cầu b 3. Sử dụng thuật toán Dijkstra tìm đường đi ngắn nhất sử dụng *w(l)* như trọng số trên liên kết *l*. 4. Định tuyến yêu cầu với *b* đơn vị băng thông từ *s* đến *d* dọc theo đường đi vừa tìm được và cập nhật băng thông còn lại. |

**Độ phức tạp**

Độ phức tạp của MHA phụ thuộc vào độ phức tạp của giải thuật tìm đường đi ngắn nhất, nếu sử dụng thuật toán Dijsktra thì độ phức tạp là *O(m + nlogn)* khi cài đặt với cấu trúc hàng đợi ưu tiên Fibonacci Heap.

**Ưu khuyết điểm**

Thuật toán MHA khá đơn giản và có độ phức tạp thấp. Nhưng thuật toán chỉ tìm đường đi cho các yêu cầu định tuyến hiện tại mà không quan tâm đến các nhu cầu trong tương lai nên việc định tuyến trong tương lai sẽ gặp khó khăn.Thuật toán tìm ra cùng một đường đi cho những yêu cầu định tuyến có cùng cặp nguồn đích cho đến lúc đường đi đó bão hòa rồi mới chuyển sang tuyến đường khác có mức tải thấp hơn, do đó dễ bị tắc nghẽn. Việc này cũng sẽ gây mất cân bằng tải do một số liên kết có tải lưu lượng cao trong khi đó một số liên kết khác chưa được sử dụng.Thuật toán chỉ dựa vào số hop để tìm đường đi mà không quan tâm đếm băng thông còn lại.

#### BCRA - Bandwidth Constrained Routing Algorithm

**Mô tả**

BCRA [5] là thuật toán định tuyến thỏa mãn ràng buộc về băng thông với tiêu chí là cải thiện việc sử dụng tài nguyên mạng bằng cách chọn đường đường đi ngắn nhất với trọng số liên kết được tính bằng:

|  |  |  |
| --- | --- | --- |
|  | *w(l) = cost(l) \* load(l) + 1* | (2.3) |
|  | với và |  |

*cost(l)* được BCRA định nghĩa như chi phí tĩnh trên liên kết *l*, chi phí của liên kết *l* càng lớn khi khả năng thông qua của liên kết *l* càng nhỏ và ngược lại. Còn *load(l)* là giá trị để đánh giá mức độ tải trên liên kết *l* được tính bằng tỉ số của băng thông đã chiếm dụng trên tổng băng thông có thể sử dụng trên liên kết *l*, giá trị *load(l)* càng lớn khi liên kết *l* tải càng nặngvà ngược lại, 1 là giá trị trọng số mặc định trên mỗi liên kết *l* khi *load(l) = 0*. Và kết quả thực nghiệm đã chứng thực rằng việc kết hợp của những yếu tố trên cải thiện được hiệu năng của việc định tuyến.

Khi yêu cầu định tuyến được gửi đến, đầu tiên BCRA loại bỏ những liên kết có khả năng thông qua còn lại nhỏ hơn băng thông yêu cầu, sau đó sử dụng thuật toán tìm đường đi ngắn nhất với trọng số được tính bằng công thức (2.3). Đường đi tìm được sẽ thỏa mãn yêu cầu băng thông và việc định tuyến qua đường đi tìm được này sẽ giúp cho việc sử dụng tài nguyên mạng được cải thiện hơn.

**Mã giả**

|  |
| --- |
| **BANDWIDTH CONSTRAINED ROUTING ALGORITHM (BCRA)**  **INPUT:**   * Đồ thị *G = (N, L, D)* và yêu cầu định tuyến *r(s, d, b)* từ nút *s* đến nút *d* với băng thông yêu cầu là *b*.   **OUTPUT:**   * Một tuyến đường từ *s* đến *d* thỏa mãn băng thông yêu cầu *b*.   **ALGORTHIM:**   1. Duyệt tất cả liên kết, tính toán trọng số liên kết bằng công thức (2.3) 2. Loại bỏ các liên kết có băng thông còn lại nhỏ hơn băng thông yêu cầu b 3. Sử dụng thuật toán Dijkstra tìm đường đi ngắn nhất với *w(l)* như trọng số trên liên kết vừa tính được. 4. Định tuyến yêu cầu với *b* đơn vị băng thông từ *s* đến *d* dọc theo đường đi vừa tìm được và cập nhật băng thông còn lại. |

**Độ phức tạp**

Độ phức tạp BCRA phụ thuộc vào độ phức tạp của giải thuật tìm đường đi ngắn nhất, nếu sử dụng thuật toán Dijsktra thì độ phức tạp là *O(m + nlogn)* khi cài đặt với cấu trúc hàng đợi ưu tiên Fibonacci Heap.

**Ưu khuyết điểm**

BCRA thật sự đơn giản và hiệu quả, với khả năng cân bằng tải tốt và thời gian tính toán thấp. Nhưng mức độ hiệu quả của thuật toán khác nhau đối với từng sơ đồ mạng khác nhau do bị hạn chế bởi hàm số heuristic (cụ thể là con số 108). Vì thế tính ứng dụng chung của BCRA thấp.

#### MIRA - Minimum Interference Algorithm

**Mô tả**

MIRA [4] – thuật toán ảnh hưởng tối thiểu, sử dụng các đặc trưng của maxflow-mincut. Theo lý thuyết về maxflow-mincut việc giảm băng thông trên một liên kết thuộc tập mincut sẽ dẫn đến giảm giá trị luồng cực đại trên liên kết đó. Có nghĩa là nếu các liên kết này được chọn để định tuyến thì sẽ ảnh hưởng đến các nhu cầu định tuyến trong tương lai. MIRA gọi các liên kết là liên kết tới hạn và gán thêm trọng số vào chúng:

|  |  |  |
| --- | --- | --- |
|  |  | (2.4) |

*⍺s’d’*: Phản ánh tầm quan trọng của cặp nguồn-đích (s’, d’)

Cs’d’: Là tập các liên kết tới hạn của cặp nguồn đích (s’, d’)

**Mã giả**

|  |
| --- |
| **MINIMUM INTERFERENCE ROUTING ALGORITHM (MIRA)**  **INPUT:**   * Đồ thị *G = (N, L, D)* và yêu cầu định tuyến *r(s, d, b)* từ nút *s* đến nút *d* với băng thông yêu cầu là *b*.   **OUTPUT:**   * Một tuyến đường từ *s* đến *d* thỏa mãn băng thông yêu cầu *b*.   **ALGORITHM:**   1. Tính toán giá trị maxflow cho tất cả . 2. Tính toán tập các liên kết tới hạn cho tất cả . 3. Tính toán các trọng số  . 4. Loại bỏ các liên kết có băng thông còn lại nhỏ hơn băng thông yêu cầu *b* 5. Sử dụng thuật toán của Dijkstra tính toán đường đi ngắn ngắn nhất sử dụng *w(l)* như trọng số trên liên kết *l*. 6. Định tuyến yêu cầu với *b* đơn vị băng thông từ *s* đến *d* dọc theo đường đi vừa tìm được và cập nhật băng thông còn lại. |

**Ưu khuyết điểm**

Thuật toán tìm đường đi dựa vào việc tránh định tuyến trên các liên kết tới hạn. Đáp ứng yêu cầu kĩ thuật lưu lượng và giảm nguy cơ tắc nghẽn của các liên kết trong tương lai. Thuật toán sử dụng phương pháp tính nhằm đạt được tối đa số yêu cầu. Vì vậy có thể chọn đường đi dài thay vì các đường đi ngắn nhưng dễ tắc nghẽn, điều đó sẽ dẫn đến tải tổng thể mạng tăng lên. Nhược điểm của thuật toán này là thời gian định tuyến lâu do phải tính toán cho tất cả các cặp vào-ra ngoại trừ cặp hiện tại.

### Vấn đề định tuyến thỏa mãn ràng buộc băng thông và độ trễ

#### Min-Delay (Minimun Delay)

**Mô tả**

MIN-DELAY - Minimum Delay (hoặc LDPRA – Least Delay Path Routing Algorithm) [6] [7] là thuật toán định tuyến tìm đường đi *psd* thỏa mãn cả băng thông yêu cầu (2.1) và độ trễ (2.2), Thuật toán loại bỏ những liên kết có băng thông còn lại nhỏ hơn băng thông yêu cầu và sau đó tìm kiếm đường đi ngắn nhất với trọng số dựa vào độ trễ.

**Mã giả**

|  |
| --- |
| **MIN-DELAY ALGORITHM**  **INPUT:**   * Đồ thị *G = (N, L, D)* và yêu cầu định tuyến từ nút s đến nút d với băng thông yêu cầu là *b* và giới hạn độ trễ là .   **OUTPUT:**   * Một tuyến đường từ *psd* thỏa mãn cả băng thông yêu cầu (2.1) và độ trễ (2.2).   **ALGORTHIM:**   1. Loại bỏ các liên kết có băng thông còn lại nhỏ hơn băng thông yêu cầu *b* 2. Sử dụng thuật toán Dijkstra tìm đường đi ngắn nhất sử dụng *w(l) = d(l)* như trọng số trên liên kết *l*. 3. Nếu đường đi *psd* vừa tìm được thỏa mãn độ trễ (2.2) thì định tuyến yêu cầu với *b* đơn vị băng thông từ *s* đến *d* dọc theo đường đi *psd* và cập nhật băng thông còn lại, ngược lại yêu cầu bị loại bỏ. |

**Độ phức tạp**

Độ phức tạp MIN-DELAY phụ thuộc vào độ phức tạp của giải thuật tìm đường đi ngắn nhất, nếu sử dụng thuật toán Dijsktra thì độ phức tạp là *O(m + nlogn)* khi cài đặt với cấu trúc hàng đợi ưu tiên Fibonacci Heap.

**Ưu khuyết điểm**

Mặc dù đơn giản và hiệu quả thuật toán nhưng MIN-DELAY có thể gây ra tắc nghẽn cho việc định tuyến trong tương lai (các yêu cầu tiếp theo trong tương lai) và khiến cho không thể tối đa hóa việc sử dụng tài nguyên mạng.

#### BDCRA – Bandwidth-Delay Constrained Routing Algorithm

**Mô tả**

BDCRA – Bandwidth-Delay Constrained Routing Algorithm [7] là thuật toán được thiết kế để tối ưu hóa việc định tuyến theo các tiêu chí sau: giảm chi phí định tuyến, giảm thiểu độ trễ định tuyến và phân phối tải thông qua mạng. Để đạt được những tiêu chí đó, BDCRA đã định nghĩa một thước đo TE trên mỗi liên kết *l* thuộc đồ thị G như sau:

|  |  |  |
| --- | --- | --- |
|  |  | (2.5) |

*Với và *

*cost(l)* được BDCRA định nghĩa như chi phí tĩnh trên liên kết *l*, chi phí của liên kết *l* càng lớn khi khả năng thông qua của liên kết *l* càng nhỏ và ngược lại. Còn *load(l)* là giá trị để đánh giá mức độ tải trên liên kết *l* được tính bằng tỉ số của băng thông đã chiếm dụng trên tổng băng thông có thể sử dụng trên liên kết *l*, giá trị *load(l)* càng lớn khi liên kết *l* tải càng nặngvà ngược lại, còn 1 là giá trị mặc định khi *load(l) = 0*.

Khi một yêu cầu kết nối  được gửi đến, BDCRA loại bỏ tất cả các liên kết có băng thông còn lại nhỏ hơn băng thông yêu cầu *b* và duy trì một tập là tập các đường đi có độ trễ thấp nhất, đầu tiên thuật toán tính toán đường đường đi có độ trễ thấp nhất thứ nhất  , đánh dấu những liên kết có giá trị *TEML* cao nhất như là liên kết tới hạng và loại bỏ chúng. Sau đó tính toán đường đi có độ trễ thấp nhất thứ hai từ mạng còn lại. Lập lại quá trình này cho đến khi không còn đường đi có độ trễ thấp nhất thỏa mãn ràng buộc độ trễ  hoặc không còn đường đi nào. Sau quá trình trên có được tập đường đi độ trễ thấp nhất và thỏa mãn ràng buộc độ trễ :

|  |  |  |
| --- | --- | --- |
|  |  | (2.6) |

Từ tập suy ra tập  được định nghĩa như sau:

|  |  |
| --- | --- |
|  | (2.7) |

Tiếp theo, BDCRA chọn ra đường đi đầu tiên trong tập là đường định tuyến *psd*.

**Mã giả**

|  |
| --- |
| **BANDWIDTH CONSTRAINED LEAST DELAY LEAST COST ROUTING ALGORITHM (BDCRA)**  **INPUT:**   * Đồ thị *G = (N, L, D)* và yêu cầu định tuyến từ nút s đến nút d với băng thông yêu cầu là *b* và giới hạn độ trễ là .   **OUTPUT:**   * Một tuyến đường từ *psd* thỏa mãn cả băng thông yêu cầu (2.1) và độ trễ (2.2).   **ALGORTHIM:**   1. Loại bỏ các liên kết có băng thông còn lại nhỏ hơn băng thông yêu cầu *b* 2. Duyệt tất cả liên kết *l* trong *G*, tính toán giá trị *TEML(l)* theo (2.5) 3. Tính tập :    1. Khởi tạo một đồ thị *G’* có cùng băng thông còn lại với đồ thị G để sử dụng trong việc tính toán tập , i = 1.    2. Khi vẫn còn đường đi thỏa mãn độ trễ từ s đến d trong đồ thị G’       1. Tìm đường đi độ trễ thấp nhất thứ i  cho cặp (s, d)       2. Đánh dấu các liên kết có giá trị TE lớn nhất và loại bỏ chúng       3. i = i + 1, đi đến (3.2) 4. Suy ra tập  theo (2.7) 5. *psd* là đường đi đầu tiên trong tập 6. Định tuyến yêu cầu với *b* đơn vị băng thông từ *s* đến *d* dọc theo đường đi *psd* và cập nhật băng thông còn lại, ngược lại yêu cầu bị loại bỏ. |

**Ưu khuyết điểm**

BDCRA thật sự đơn giản và hiệu quả, với khả năng cân bằng tải tốt và thời gian tính toán thấp. Nhưng mức độ hiệu quả của thuật toán khác nhau đối với từng sơ đồ mạng khác nhau do bị hạn chế bởi hàm số heuristic (cụ thể là con số 108). Vì thế khả năng ứng dụng chung của BDCRA thấp.

#### MDWCRA – Maximum Delay-Weighted Capacity Routing Algorithm

**Mô tả**

MDWCRA - Maximum Delay-Weighted Capacity Routing Algorithm [8] là thuật toán định tuyến thỏa mãn ràng buộc băng thông và độ trễ.

Cho cặp vào-ra *(s, d)*. Tác giả của MDWCRA đã hình dung đường đi giữa *s* và *d* là một loại máy móc để đáp ứng các yêu cầu với ràng buộc băng thông và độ trễ. Sức mạnh của cỗ máy này có thể được đo bằng độ trễ từ đầu tới cuối của đường đi. Giá trị này càng nhỏ thì cỗ máy càng mạnh mẽ vì thuật toán có thể xử lí được các yêu cầu có ràng buộc độ trễ khắc khe hơn. Số lượng các yêu cầu mà cỗ máy có thể đáp ứng được gọi là khả năng đáp ứng (capacity), được đo bằng băng thông của đường đi. Có thể dễ dàng thấy rằng cỗ máy mạnh nhất của cặp *(s, d)* là đường đi có độ trễ thấp nhất giữa *s* và *d*. Loại bỏ những liên kết dọc theo đường đi có độ trễ thấp nhất này. Cỗ máy mạnh nhất thứ hai của cặp *(s, d)* là đường đi có độ trễ thấp nhất thứ hai được tính toán từ mạng còn lại. Lập lại quá trình này cho đến khi không còn đường đi nào giữa *s* và *d*, cuối cùng thu được tập các đường đi có độ trễ thấp nhất  với  là đường đi có độ trễ thấp nhất được tính từ mạng mà các liên kết thuộc , …,  bị loại bỏ. Cho  là băng thông và  là độ trễ của đường đi có độ trễ thấp nhất . Tổng số đường đi của  là :

Gái trị DWC của cặp vào-ra *(s, d)* được định nghĩa như sau:

|  |  |  |
| --- | --- | --- |
|  |  | (2.8) |

Băng thông của một đường đi được xác định bởi băng thông của liên kết nhỏ nhất thuộc đường đi đó. Những liên kết mà xác định băng thông đường đi được xem là *liên kết hẹp* của đường đi đó. Nếu định tuyến yêu cầu qua những liên kết hẹp của bất kì đường đi nào trong  thì giá trị DWC của cặp *(s, d)* giảm. Vì thế một liên kết được định nghĩa như một *liên kết tới hạn* cho *(s, d)* khi một đường đi được định tuyến qua đó, giá trị DWC của một hoặc nhiều cặp vào-ra giảm. Tập các liên kết tới hạn được biểu diễn bởi , với  bao gồm tất cả các liên kết hẹp cho đường đi độ trễ thấp nhất .

Tác giả của MDWCRA mong muốn cực đại hóa tổng trọng số DWC của mỗi cặp vào-ra sau khi đáp ứng cho yêu cầu hiện tại. Để đạt được điều này MDWCRA hạn chế tối đa việc định tuyến qua những liên kết tới hạn bằng cách gán vào chúng một trọng số thích hợp và định tuyến yêu cầu qua đường đi có tổng trọng số nhỏ nhất. Việc tính toán trọng số cho liên kết được quy về việc xác định các liên kết tới hạn cho tất cả cặp vào-ra. Điều này có thể được giải quyết như một quá trình lặp đi lặp lại của việc tính toán  cho mỗi cặp *(s, d).* Khi tất cả các liên kết tới hạn được xác định, trọng số sẽ được gán vào các liên kết tới hạn và định tuyến theo đường đi có trọng số thấp nhất. Trước khi làm điều này, các liên kết có băng thông còn lại nhỏ hơn băng thông yêu cầu phải được loại bỏ khỏi mạng để đảm bảo bất kì đường đi nào tìm được từ mạng còn lại cũng thỏa mãn yêu cầu băng thông. Để đảm bảo rằng ràng buộc độ trễ được đáp ứng, thuật toán EDSP hoặc EBF [1, tr. 41] được sử dụng để tính toán đường đi có trọng số thấp nhất.

**Mã giả**

|  |
| --- |
| **MAXIMUM DELAY-WEIGHTED CAPACITY ROUTING ALGORITHM (MDWCRA)**  **INPUT:**   * Đồ thị *G = (N, L, D)* và yêu cầu định tuyến từ nút s đến nút d với băng thông yêu cầu là *b* và giới hạn độ trễ là .   **OUTPUT:**   * Một tuyến đường từ *psd* thỏa mãn cả băng thông yêu cầu và độ trễ.   **ALGORTHIM:**   1. Khởi tạo tất cả các trọng số liên kết *w(l) = 0* 2. Với mỗi cặp *(s, d)* thuộc D, tính toán giá trị DWC cho mỗi cặp đó    1. Khởi tạo đồ thị làm việc *G’* có cùng băng thông còn lại như độ thị *G*, *i=1*.    2. Khi vẫn còn đường đi từ *s* đến *d*       1. Tìm đường đi có độ trễ thấp nhất thứ i  cho cặp *(s, d)*       2. Đánh dấu các liên kết hẹp của đường đi có độ trễ thấp nhất như liên kết tới hạn và cho chúng vào       3. Duyệt tất các liên kết tới hạn đã xác định, cập nhật trọng số liên kết       4. Loại bỏ tất cả các liên kết thuộc  trong *G’*       5. *i = i + 1*, đi đến (2.2) 3. Loại bỏ tất cả các liên kết có băng thông còn lại nhỏ hơn băng thông yêu cầu *b* trong đồ thị *G*. 4. Sử dụng thuật toán EDSP hoặc EBF, tính toán đường đi ràng buộc độ trễ trọng số thấp nhất trong mạng còn lại bằng việc sử dụng *w(l)* như trọng số trên liên kết *l*. 5. Định tuyến yêu cầu với *b* đơn vị băng thông từ *s* đến *d* dọc theo đường đi *psd* và cập nhật băng thông còn lại, ngược lại yêu cầu bị loại bỏ. |

Trong thuật toán trên, các trọng số liên kết thật sự được tính toán theo . Bằng việc biến đổi giá trị , các định nghĩa khác nhau của trọng số liên kết có thể thu được như sau:

1. , với . Trọng số của liên kết *l* được biểu diễn bởi số cặp vào-ra mà liên kết *l* là liên kết tới hạn.
2. , với . Trọng số liên kết là nghịch đảo của giá trị độ trễ từ đầu đến cuối của đường đi có độ trễ thấp nhất.
3. , với . Trọng số liên kết là nghịch đảo của băng thông và độ trễ từ đầu tới cuối của đường đi có độ trễ thấp nhất.

**Độ phức tạp**

Theo tác giả của MDWCRA và M-MDWCRA [8], hai thuật toán được đánh giá độ phức tạp như sau:

* MDWCRA có độ phức tạp là O(*np*log*n* + *x*2*n*2) nếu sử dụng thuật toán EDSP hoặc O(*np*log*n* + *xmn*) nếu sử dụng thuật toán EBF.
* M-MDWCRA có độ phức tạp là O(*n*2*p*log*n* + *x*2*n*2) nếu sử dụng thuật toán EDSP hoặc O(*n*2*p*log*n* + *xmn*) nếu sử dụng thuật toán EBF.

Với *p* là số cặp vào-ra, *x* là độ trễ yêu cầu.

**Ưu khuyết điểm**

MDWCRA và M-MDWCRA thật sự hiệu quả về khả năng phân phối tải, nhưng nhược điểm là không có tính ứng dụng chung do phụ thuộc vào hai thuật toán EDSP (EBF) và thời gian tính toán tương đối cao vì thế chúng chỉ chạy được với các hệ thống mạng mạng nhỏ.

# ĐỀ XUẤT THUẬT TOÁN

Đa số các thuật toán định tuyến thỏa mãn ràng buộc băng thông và độ trễ, điều đưa vấn đề về bài toán tìm đường đi thỏa mãn với tổng trọng số trên các liên kết của đường đi là thấp nhất:

|  |  |  |
| --- | --- | --- |
|  | ,  *là tập các đường đi thỏa mãn (2.1) và (2.2).* | (3.1) |

Cả hai thuật toán tìm đường đi ngắn nhất của Bellman-Ford và Dijkstra là thuật toán chính xác và chạy trong thời gian đa thức. Như tên gọi của thuật toán đã cho thấy, thuật toán tìm đường đi ngắn nhất cực tiểu hóa tổng chiều dài của những liên kết trên đường đi từ nguồn tới đích và vì thế chúng chỉ tối ưu một thước đo cộng dồn. Nếu chiều dài của liên kết là thước đo độ trễ trên liên kết đó thì thuật toán tìm đường đi ngắn nhất sẽ tính toán đường đi có độ trễ thấp nhất (LDP); nếu chiều dài liên kết là trọng số trên liên kết đó thì thuật toán tìm đường đi ngắn nhất sẽ tính toán đường đi có tổng trọng số thấp nhất (LWP). Với các ứng dụng thời gian thực, các nhà cung cấp dịch vụ mong muốn tìm đường mà quan tâm cả hai thước đo cùng lúc. Đó là độ trễ dọc theo đường đi phải được chấp nhận bởi ứng dụng và trọng số của đường đi phải thấp nhất có thể để tối ưu việc định tuyến (sử dụng tài nguyên mạng). Vấn đề này thuộc vấn đề định tuyến kết hợp *path-constrained routing* và *path-optimization routing* mà đã được chứng minh là *NP-đầy đủ* [1, tr. 11].

## OBDCRA – Optimized Bandwidth Delay Constrained Routing Algorithm

Chúng tôi sẽ trình bày trong mục này một giải pháp mới cho vấn đề định tuyến QoS*.* Giải pháp của chúng tôi được gọi là *OBDCRA – thuật toán định tuyến tối ưu thỏa mãn ràng buộc băng thông – độ trễ*, một *thuật toán heuristic* được đề xuất từ việc kết hợp giữa một thuật toán mà chúng tôi gọi là C-Dijkstra và một hàm tính toán trọng số, với tiêu chí tìm đường đi thỏa mãn ràng buộc băng thông – độ trễ và tối ưu hóa việc sử dụng tài nguyên mạng (cân bằng tải). Để đạt được điều này, chúng tôi gán vào mỗi liên kết một trọng số *w* được tính toán từ hàm trọng số để đánh giá mức độ tới hạn của liên kết và chuyển vấn đề định tuyến về vấn đề tìm đường đi ngắn nhất theo trọng số *w* thỏa mãn một ràng buộc *c* (trong vấn đề định tuyến, *c* chính là ràng buộc độ trễ). Chúng tôi đã chỉnh sửa giải thuật tìm đường đi ngắn nhất của Dijkstra thành C-Dijkstra để giải quyết vấn đề trên mà sẽ được trình bày chi tiết ở mục sau. OBDCRA thật sự đơn giản và hiệu quả hơn so với các thuật toán đã được trình bày trên với thời gian tính toán và tỉ lệ từ chối yêu cầu thấp hơn đã kiểm chứng được bằng thực nghiệm.

### Hàm tính toán trọng số - phân phối tải

Một trong những tiêu chí quan trọng của việc định tuyến là phân phối tải nhằm duy trì một mức độ cân bằng tải cao trong mạng. Điều đó thật sự quan trọng trong việc lựa chọn các tuyến đường định tuyến tránh đi qua những liên kết có mức độ tải cao. Để phân phối tải, chúng tôi định nghĩa một trọng số cho mỗi liên kết như sau:

|  |  |  |
| --- | --- | --- |
|  |  | (3.2) |

Mục tiêu của việc định tuyến là cực tiểu hóa tổng trọng số của đường đi. Với công thức trên, việc lựa tuyến đường định tuyến sẽ hạn chế đi qua những liên kết có băng thông đang bị chiếm giữ cao hoặc băng thông khả dụng thấp.

### Vấn đề tìm đường đi ngắn nhất thỏa mãn một ràng buộc

#### Xác định vấn đề

Gọi là tập số thực không âm. Chúng tôi tóm tắt vấn đề như sau: cho đồ thị *G=(N, L)* nút nguồn *s*, nút đích *d*, hai hàm trọng số **và **, một ràng buộc , qui ước ** và ** nếu *.* Vấn đề là tìm đường đi *psd* từ *s* đến *d* mà  và  nếu đường đi tồn tại, với *Psd* là tập các đường đi từ *s* đến *d*.

#### Thuật toán heuristic C-Dijkstra

**Ý tưởng**

Gọi *lw*2*(v)* là khoảng cách ngắn nhất theo trọng số *w2* của mọi nút *v* đến nút đích *d*, thuật toán C-Dijkstra sẽ tối ưu hóa dần khoảng cách theo trọng số *w1* mà  từ một đỉnh xuất phát đến tất cả các đỉnh còn lại. C-Dijkstra duy trì một tập cho mọi , với:

|  |  |  |
| --- | --- | --- |
|  |  | (3.3) |

Khi thuật toán bắt đầu, , *D*1(*s*) *= d*1(*s*) *= d*2(*s*) *= 0,* ,  và tính toán *lw*2. Việc tính toán *lw*2 thuộc về vấn đề *single-destination shortest path[[1]](#footnote-1)*. Vấn đề này có thể được quyết như vấn đề *single-source shortest* *path[[2]](#footnote-2)* bằng cách đổi chiều các liên kết trong đồ thị và tìm đường đi ngắn nhất từ nút đích *d* đến tất cả các nút trong đồ thị. Ở mỗi lần lập ta thăm nút  mà *D(u)* là bé nhất – nếu có hai hay nhiều nút thỏa hai điều kiện trên thì ta có thể chọn tùy ý. Với mỗi cạnh , *D(v)* được giảm xuống  và  nếu  và . Sau đó, cũng như *D*1*(v) = d*1*(v)*, *v* có thể được đưa vào tập *S*. Thuật toán kết thúc khi *S = V.*

**Mã giả**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | 1 | | 2 | | 3 | | 4 | | 5 | | 6 | | 7 | | 8 | | 9 | | 10 | | 11 | | 12 | | 13 | | 14 | | 15 | | 16 | | 17 | | 18 | | 19 | | 20 | | 21 | | 22 | | 23 | | 24 | | 25 | | 26 | | 27 | | 28 | | 29 | | 30 | | 31 | | 32 | | 33 | | 34 | | 35 | | 36 | | 37 | | 38 | | 39 | | 40 | | 41 | | 42 | | 43 | | 44 | | 45 | | 46 | | 47 | | Initialize(*G*, *s*)  **begin**  **for** *v* in *N* **do**  *d*1[*s*] := ∞  *d*2[*s*] := ∞  *π*[*v*] := *nil*  *d*1[*s*] := 0  *d*2[*s*] := 0  **end**  Relax(*u*, *v*, *w1*, *w2,* *c, lw2*)  **begin**  **if**  **and**  **then**        **end**  C-Dijkstra(*G*, *s,* *d*, *w1, w2, c*)  **begin**  Initialize(*G*, *s*)  *G’* := Reverse(*G*) */\* đổi chiểu tất cả các liên kết trong đồ thị G \*/*  Dijkstra(*G’*, *d*, *w*2)      **while**  **then**        **for** *v* kề với *u* **then**  Relax(*u*, *v*, *w*1, *w*2, *c*, *lw*2)  **return** Construct(*π*, *s*, *d*)  **end**  Construct(*π*, *s*, *d*)  **begin**    *v* := *d*  **while** *v ≠ NIL* **then**      **return** *psd*  **end** |

**Độ phức tạp**

Độ phức tạp của C-Dijkstra bằng hai lần độ phức tạp của giải thuật Dijkstra O(*n + nlogn*) khi cài đặt với cấu trúc hàng đợi ưu tiên Fibonacci-Heap.

### Thuật toán định tuyến OBDCRA

**Ý tưởng**

Cho mạng *G*=(*N*, *L*), khi yêu cầu định tuyến  được gửi đến. OBDCRA tiến hành loại bỏ những liên kết có băng khả dụng hay băng thông còn lại thấp hơn băng thông yêu cầu *b*, tính toán trọng số *w*1 theo công thức (3.2) và gán  cho mỗi liên kết trong mạng còn lại. Tiếp theo, OBCDRA sử dụng thuật toán C-Dijkstra để tính toán đường đi ngắn nhất từ *s* đến *d* theo trọng số *w*1 và thỏa mãn ràng buộc độ trễ . Nếu tìm được đường đi, tiến hành định tuyến yêu cầu từ *s* đến *d* với *b* đơn vị băng thông dọc theo đường đi vừa tìm được và cập nhật băng thông còn lại, ngược lại yêu cầu bị loại bỏ.

**Mã giả**

|  |
| --- |
| **OPTIMIZED BANDWIDTH DELAY CONSTRAINED ROUTING ALGORITHM (OBDCRA)**  **INPUT:**   * Đồ thị *G = (N, L)* và yêu cầu định tuyến từ nút *s* đến nút *d* với băng thông yêu cầu là *b* và giới hạn độ trễ là .   **OUTPUT:**   * Một tuyến đường *psd* thỏa mãn cả băng thông yêu cầu (2.1) và độ trễ (2.2).   **ALGORTHIM:**   1. Loại bỏ các liên kết có băng thông còn lại nhỏ hơn băng thông yêu cầu *b* 2. Tính toán trọng số *w*1 theo công thức (3.2) và gán  cho mỗi liên kết trong mạng còn lại. 3. Sử dụng thuật toán C-Dijkstra để tìm đường đi tối ưu từ *s* đến *d* với ràng buộc là . 4. Định tuyến yêu cầu với *b* đơn vị băng thông từ *s* đến *d* dọc theo đường đi *psd* và cập nhật băng thông còn lại, ngược lại yêu cầu bị loại bỏ. |

**Độ phức tạp**

Độ phức tạp của OBDCRA phụ thuộc vào độ phức tạp của thuật toán C-Dijkstra và bằng hai lần độ phức tạp của giải thuật tìm đường đi ngắn nhất của Dijkstra (*n + nlogn*) khi cài đặt với cấu trúc hàng đợi ưu tiên Fibonacci-Heap.

**Ưu khuyết điểm**

Thuật toán OBDCRA thật sự đơn giản và hiệu quả với thời gian tính toán và tỉ lệ từ chối yêu cầu thấp, nhưng hạn chế của thuật toán này là vì C-Dijkstra là thuật toán heuristic – đường đi tìm được chỉ là đường đi cận tối ưu.

Ví dụ 3‑1 Thực hiện thuật toán định tuyến OBDCRA, tìm đường đi từ nút 0 đến nút 5 với băng thông yêu cầu là 0.5 và độ trễ là 4.

|  |
| --- |
| Cho sơ đồ mạng *G = (N, L)* như sau:    ***Trạng thái liên kết = (băng thông, độ trễ)***  Yêu cầu định tuyến từ nút 0 từ đến nút 5 với băng thông 0.1 là và độ trễ là 4. |
| 1. *Tính toán trọng số w1, w2 theo công thức (3.2) ta được kết quả như sau:*     ***(w*1*, w*2*)*** |
| 1. *Tính toán độ trễ thấp nhất từ tất cả các nút v tới nút đích d (nút 5) ta được kết quả như sau:* |
| 1. *Khởi tạo* *, d1[s] = 0, d2[s] = 0, π[s] = nil và* *: d1[v] = w1(s, v), d2[v] = 0, π[v] = nil.* |
| 1. *: chọn u với d1[u] là bé nhất, như vậy u = 0; lần lượt xét các nút v kề u, với v = 1: d1[1] = d1[0] + w1(0, 1) và d2[1] = d2[0] + w2(0, 1) (vì d1[1]> d1[0] + w1(0, 1) và d2[0] + w2(0, 2) + lw2[2] ≤ delay ); tương tự với v = 2: d1[2] = d1[0] + w1(0, 2) và d2[2] = d2[0] + w2(0, 2) (vì d1[2]> d1[0] + w1(0, 2) và d2[0] + w2(0, 2) + lw2[2] ≤ delay ); .* |
| 1. *: chọn u với d1[u] là bé nhất, như vậy u = 1; lần lượt xét các nút v kề u, với v = 3: d1[3] = d1[1] + w1(1, 3) và d2[3] = d2[1] + w2(1, 3) (vì d1[3]> d1[1] + w1(1, 3) và d2[1] + w2(1, 3) + lw2[3] ≤ delay ); với v = 4: không cập nhật bảng số (vì d2[1] + w2(1, 4) + lw2[4] > delay ); .* |
| 1. *: chọn u với d1[u] là bé nhất, như vậy u = 2; lần lượt xét các nút v kề u, với v = 3: d1[3] = d1[2] + w1(2, 3) và d2[3] = d2[2] + w2(2, 3) (vì d1[3]> d1[2] + w1(2, 3) và d2[2] + w2(2, 3) + lw2[3] ≤ delay ); với v = 4: d1[4] = d1[2] + w1(2, 4) và d2[4] = d2[2] + w2(2, 4) (vì d1[4]> d1[2] + w1(2, 4) và d2[2] + w2(2, 4) + lw2[4] ≤ delay ); .* |
| 1. *: chọn u với d1[u] là bé nhất, như vậy u = 3; lần lượt xét các nút v kề u, với v = 5: d1[5] = d1[3] + w1(3, 5) và d2[5] = d2[3] + w2(3, 5) (vì d1[5]> d1[3] + w1(3, 5) và d2[3] + w2(3, 5) + lw2[5] ≤ delay);* |
| 1. *: chọn u với d1[u] là bé nhất, như vậy u = 4; lần lượt xét các nút v kề u, với v = 5: d1[5] = d1[4] + w1(4, 5) và d2[5] = d2[4] + w2(4, 5) (vì d1[5]> d1[4] + w1(4, 5) và d2[4] + w2(4, 5) + lw2[5] ≤ delay ); .* |
| 1. *: chọn u với d1[u] là bé nhất, như vậy u = 5; không còn nút v nào kề với u; .* |
| 1. *S = V: thuật toán kế thúc; đường đi psd thỏa mãn từ s đến d: 0 → 2 → 3 → 5. Vậy w1(psd) = 6, w2(psd) = 3 ≤ delay.* |

Ảnh 3.1 Các bước thực hiện của thuật toán định tuyến OBDCRA.

### Thuật toán định tuyến M-OBDCRA (Modified-OBDCRA)

**Ý tưởng**

Với thuật toán OBDCRA, việc tìm kiếm đường đi hoàn toàn thuộc vào thuộc toán C-Dijkstra, một thuật toán heuristic – chỉ tìm được lời giải gần đúng. Nhược điểm lớn của thuật toán C-Dijkstra là chỉ quan tâm đến việc giảm khoảng cách của đường đi từ nguồn đến đích theo trọng số *w*1, thỏa mãn ràng buộc *c* và cập nhật ngay vào đường đi ở mỗi lần lập mà không quan tâm về sau đường đi đó có thể sẽ đi qua những liên kết có trọng số cao do bị ràng buộc về độ trễ. Cụ thể hãy xem ví dụ sau:

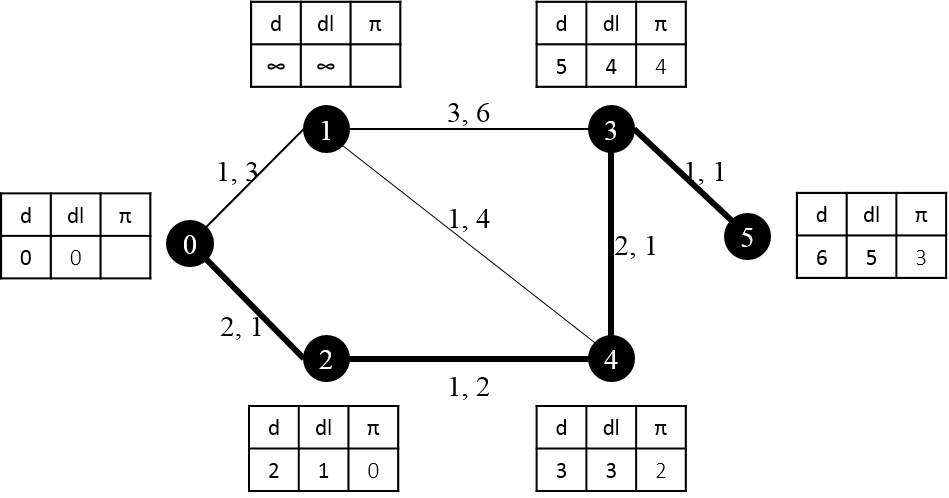
Ví dụ 3‑2 Thực hiện thuật toán định tuyến OBDCRA trong trường hợp không tìm được đường đi tối ưu.

|  |
| --- |
| Cho sơ đồ mạng *G = (N, L)* như sau:    ***Trạng thái liên kết = (băng thông, độ trễ)***  Yêu cầu định tuyến từ nút 0 đến nút 5 với băng thông = 0.1 độ trễ = 8 |
| 1. *Tính toán trọng số w1, w2 theo công thức (3.2) ta được kết quả như sau:*     ***(w*1*, w*2*)*** |
| 1. *Tính toán độ trễ thấp nhất từ tất cả các nút v tới nút đích d (nút 5) ta được kết quả như sau:* |
| 1. *Khởi tạo* *, d1[s] = 0, d2[s] = 0, π[s] = nil và* *: d1[v] = w1(s, v), d2[v] = 0, π[v] = nil.* |
| 1. *: chọn u với d1[u] là bé nhất, như vậy u = 0; lần lượt xét các nút v kề u, với v = 1: d1[1] = d1[0] + w1(0, 1) và d2[1] = d2[0] + w2(0, 1) (vì d1[1]> d1[0] + w1(0, 1) và d2[0] + w2(0, 2) + lw2[2] ≤ delay ); tương tự với v = 2: d1[2] = d1[0] + w1(0, 2) và d2[2] = d2[0] + w2(0, 2) (vì d1[2]> d1[0] + w1(0, 2) và d2[0] + w2(0, 2) + lw2[2] ≤ delay ); .* |
| 1. *: chọn u với d1[u] là bé nhất, như vậy u = 1; lần lượt xét các nút v kề u, với v = 3: không cập nhật bảng số (vì d2[1] + w2(1, 3) + lw2[3] > delay ); với v = 4: d1[4] = d1[1] + w1(1, 4) và d2[4] = d2[1] + w2(1, 4) (vì d1[4]> d1[1] + w1(1, 4) và d2[1] + w2(1, 4) + lw2[4] ≤ delay ); .* |
| 1. *: chọn u với d1[u] là bé nhất, như vậy u = 2; lần lượt xét các nút v kề u, với v = 4: d1[4] = d1[2] + w1(2, 4) và d2[4] = d2[2] + w2(2, 4) (vì d1[4]> d1[2] + w1(2, 4) và d2[2] + w2(2, 4) + lw2[4] ≤ delay ); .* |
| 1. *: chọn u với d1[u] là bé nhất, như vậy u = 4; lần lượt xét các nút v kề u, với v = 5: d1[5] = d1[4] + w1(4, 5) và d2[5] = d2[4] + w2(4, 5) (vì d1[5]> d1[4] + w1(4, 5) và d2[4] + w2(4, 5) + lw2[5] ≤ delay ); .* |
| 1. *: chọn u với d1[u] là bé nhất, như vậy u = 5; lần lượt xét các nút v kề u, với v = 3: không cập nhật bảng số (vì d2[1] + w2(1, 3) + lw2[3] > delay ); .* |
| 1. *: chọn u với d1[u] là nhỏ nhất, như vậy u = 3;không còn nút v nào kề với u; .* |
| 1. *S = V: thuật toán kế thúc; đường đi psd thỏa mãn từ s đến d: 0 → 1 → 4 → 5. Vậy w1(psd) = 12, w2(psd) = 8 ≤ delay.* |

Ảnh 3.2 Các bước thực hiện của thuật toán định tuyến OBDCRA trong trường hợp không tìm được đường đi tối ưu.

Để hạn chế được nhược điểm ở Ví dụ 3-2 chúng tôi đã cãi tiến OBDCRA bằng cách xóa đi những liên kết có trọng số cao nhất trong đường đi tìm được sau mỗi lần chạy thuật toán, cho đến khi không còn tìm được đường đi nào nữa. Kết thúc giai đoạn này, ta có được một tập các đường đi, trong tập đường đi này chúng tôi chọn ra một đường đi có tổng trọng số thấp nhất chính là đường đi tối ưu.

Như Ví dụ 3-2 đường đi tìm được ở lần chạy đầu tiên là 0 → 1 → 4 → 5. Ta thấy liên kết có trọng số *w*1 cao nhất trong đường đi là liên kết (4, 5) có trọng số là 10, xóa liên kết này và thực hiện lại giải thuật ta sẽ thu được kết quả như sau:



Ảnh 3.3 Kết quả đường đi tìm được của thuật toán định tuyến M-OBDCRA.

Đường đi mới tìm được là 0 → 2 → 4 → 3 → 5 có tổng trọng số là 6 thấp hơn đường đi 0 → 1 → 4 → 5 có tổng trọng số là 12.

**Mã giả**

|  |
| --- |
| **MODIFIED-OPTIMIZED BANDWIDTH DELAY CONSTRAINED ROUTING ALGORITHM (M-OBDCRA)**  **INPUT:**   * Đồ thị *G = (N, L)* và yêu cầu định tuyến từ nút *s* đến nút *d* với băng thông yêu cầu là *b* và giới hạn độ trễ là .   **OUTPUT:**   * Một tuyến đường *psd* thỏa mãn cả băng thông yêu cầu (2.1) và độ trễ (2.2).   **ALGORTHIM:**   1. Loại bỏ các liên kết có băng thông còn lại nhỏ hơn băng thông yêu cầu *b* 2. Tính toán trọng số *w*1 theo công thức (3.2) và gán  cho mỗi liên kết trong mạng còn lại. 4. Khi vẫn còn đường đi thỏa mãn độ trễ từ *s* đến *d* trong đồ thị G    1. Sử dụng thuật toán C-Dijkstra để tìm đường đi *ptempsd* từ *s* đến *d* với ràng buộc độ trễ là .    2. Nếu *w*1(*ptempsd*) < *w*1(*psd*) thì *psd = ptempsd*    3. Loại bỏ những liên kết có trọng số *w*1 cao nhất trong đồ thị G    4. Đi đến bước (4) 5. Nếu *psd* được tìm thấy thì định tuyến yêu cầu với *b* đơn vị băng thông từ *s* đến *d* dọc theo đường đi *psd* và cập nhật băng thông còn lại, ngược lại yêu cầu bị loại bỏ. |

**Ưu khuyết điểm**

M-OBDCRA đã khắc phục được nhược điểm của OBDCRA với thời tính toán chấp nhận được với các hệ thống mạng lớn đã kiểm chứng được bằng thực nghiệm.

# THỰC NGHIỆM VÀ ĐÁNH GIÁ KẾT QUẢ

## Môi trường giả lập

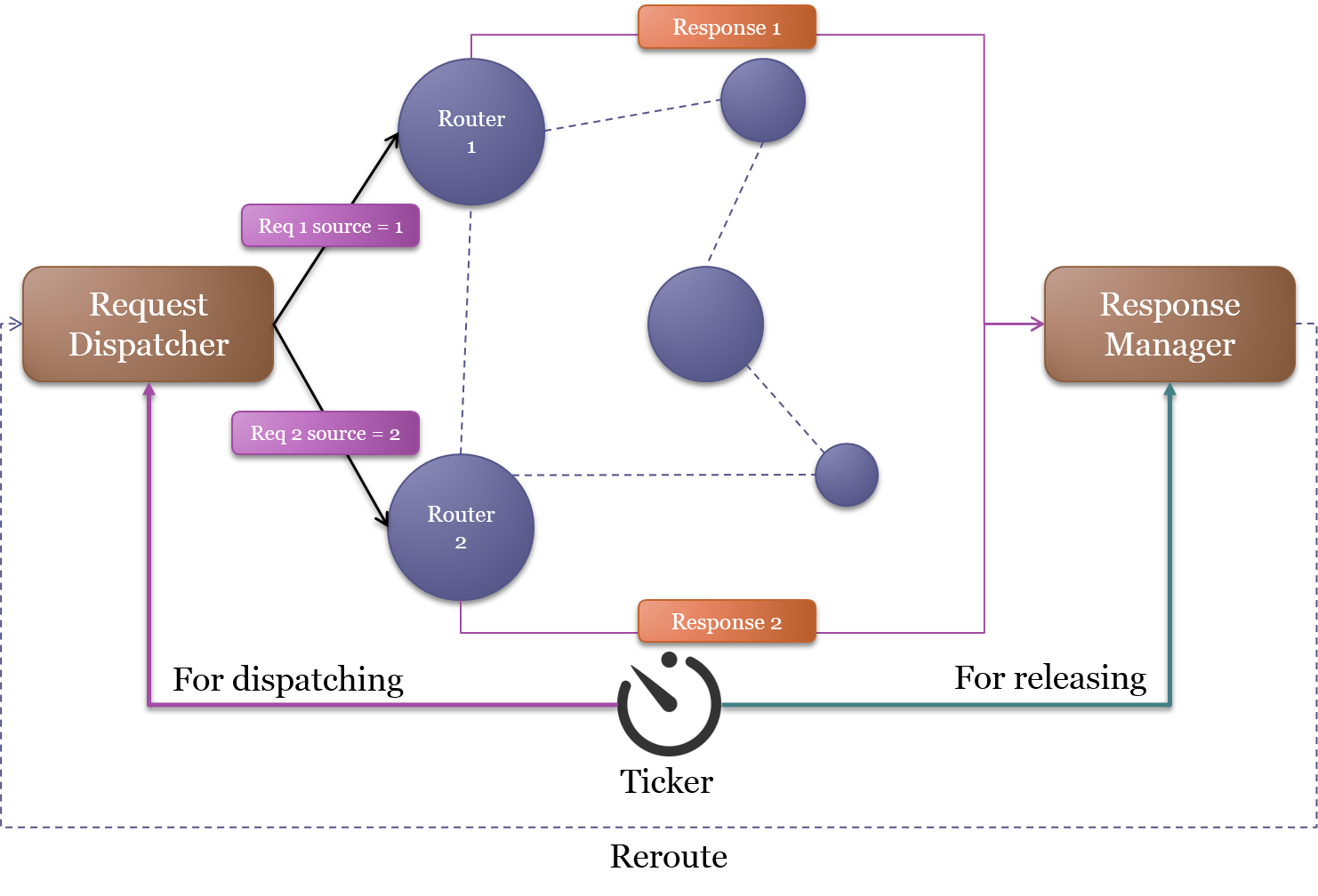
### Giới thiệu

Công việc thực nghiệm là một trọng những công việc quan trọng trong quá trình nghiên cứu. Bằng việc thực nghiệm, chúng ta có thể đánh giá được hiệu năng và các đặc điểm của các thuật toán định tuyến. Trên thực tế, chúng ta không thể dể dàng cài đặt và thực nghiệm các thuật toán định tuyến trên mô hình vật lí được, mà hầu hết các nhà nghiên cứu điều thực nghiệm trên một môi trường giả lập nào đó như NS2, OPNET, NetSim, vv… Nhưng hầu hết các bộ giả lập này đều rất cồng kềnh và phức tạp chỉ để giải quyết một số yêu cầu đơn giản cho bài toán đồ thị. Chính vì vậy, chúng tôi quyết định xây dựng một công cụ được gọi là *SimpleRoutingAlgorithmSimulator-SRAS* để tạo môi trường thực nghiệm và đánh giá hiệu năng của các thuật toán định tuyến một cách đơn giản hơn.

### Phân tích thiết kế

Giả sử chúng tôi có một hệ thống mạng *G* có *n* nút và *m* liên kết. Vấn đề mà thuật toán định tuyến như đã xác định chỉ giải quyết việc tìm đường đi định tuyến cho yêu cầu định tuyến từ nút nguồn *s* đến nút đích *d* với băng thông yêu cầu *b* và độ trễ từ đầu tới cuối đường đi . Nhưng trên thực tế có rất nhiều yêu cầu định tuyến được gửi đến các nút nguồn mạng tại một thời điểm nào đó, giữ băng thông yêu cầu trong một khoảng thời gian và giải phóng băng thông yêu cầu. Vì vậy, để kiểm tra được hiệu năng của các thuật toán định tuyến, chúng tôi đã xây dựng bộ giả lập định tuyến để thực hiện việc giả lập các yêu cầu định tuyến gửi đến nút nguồn s tại thời điểm *t*1, tìm đường đi định tuyến thỏa mãn đến nút *d*, giữ băng thông yêu cầu trong một khoảng thời gian *t*2 và giải phóng băng thông băng thông yêu cầu. Với *s*, *d*, *b,* , *t*1 và *t*2 được phát sinh từ bộ phát sinh yêu cầu định tuyến theo kịch bản cần giả lập.

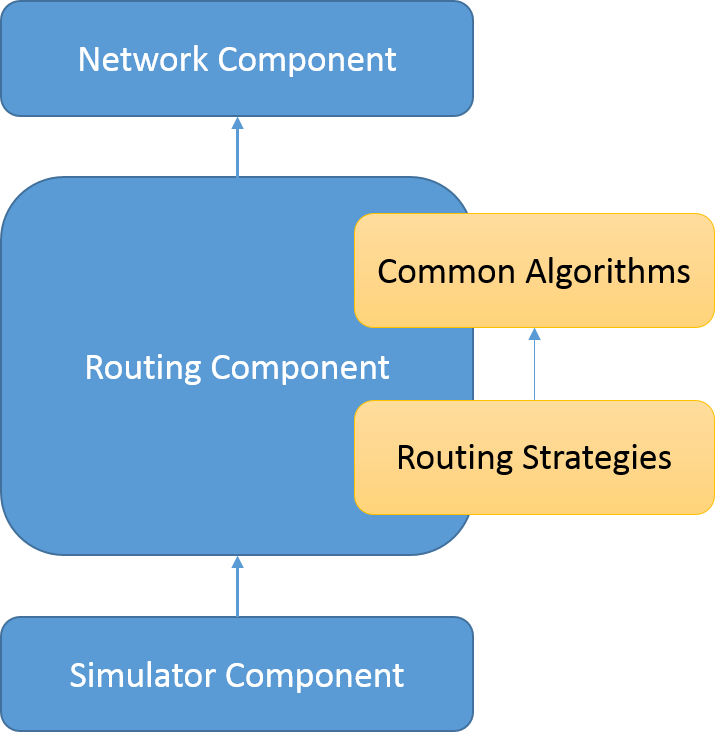
Dựa vào những nhu cầu cho bộ giả lập định tuyến chúng tôi đã phân tích và phát thảo nên một mô hình tổng thể cho bộ giả lập này như Ảnh 4.1 .



Ảnh 4.1 Sơ đồ tổng thể của bộ giả lập – SRAS

Trong sơ đồ này chúng tôi xem mỗi nút trong trong mô hình mạng như một bộ định tuyến có một luồng xử lí và không gian bộ nhớ riêng. Việc phân phối các yêu cầu định tuyến sẽ được bộ phân phối yêu cầu phân phối đến các bộ định tuyến, khi đó các bộ định tuyến này sẽ tranh chấp và tính toán trên cùng một mô hình mạng, đường đi tìm thấy sẽ được gửi đến bộ quản lí phản hồi để giữ và chờ đến thời gian giải phóng. Cả hai bộ phân phối và quản lí phản hồi đều lắng nghe sự kiện từ bộ đếm thời gian.

Bước đầu của việc phần tích, chúng tôi chia hệ thống thành ba thành phần là: Network Component, Routing Component, Simulation Component như Ảnh 4.2



Ảnh 4.2 Các thành phần của bộ giả lập định tuyến.

* Network component là thành phần tổ chức nên mô hình mạng từ các đối tượng cơ bản như link, node, topology để nắm giữ các thông tin trang thái.
* Routing component được chia thành hai thành phần con là Routing Strategies và Common Algorithms. Routing Strategies bao gồm tất cả các thuật toán định tuyến được phát triển mà sử dụng chung những thuật toán khác như: Dijkstra, BFS, Ford-Fukerson … tất cả những thuật toán dùng chung này được chứa trong Common Algorithms.
* Simulation Component là thành phần giả lập cung cấp một cơ chế vận hành cho việc thực thi của các thuật toán định tuyến ở Routing Component.



Ảnh 4.3 Sơ đồ lớp của bộ giả lập định tuyến

Thành phần Network Component gồm có ba lớp: Topology, Link, Node.

* Lớp Topology: là lớp đối tượng đại diện cho một mô hình mạng nắm giữ toàn bộ thông tin trạng thái của mô hình mạng đó. Trong mỗi Topology luôn duy trì một tập các Node – nút mạng và Link – liên kết mạng.
* Lớp Node: là lớp đối tượng đại diện cho nút mạng, trong mỗi đối tượng Node có một tập các Link là các liên kết đi ra từ nút đó.
* Lớp Link: là lớp đối tượng đại diện cho liên kết mạng, với mỗi liên kết mạng có 2 đối tượng là Node nguồn và Node đích.

Thành phần Routing Component gồm có 2 thành phần con là: Routing Strategies và Common Algorithms.

* Common Algorithms là thành phần chứa các lớp đối tượng thuật toán dùng chung như lớp thuật toán Dijkstra – tìm đường đi ngắn nhất với theo trọng số nào đó, FordFukerson – thuật toán tính maxflow-mincut, BreathFirstSearch – thuật toán tìm đường đi theo duyệt rộng, cũng có thể dùng để tìm đường đi qua số nút (liên kết) ít nhất, EDSP (EBF) – tìm đường đi ngắn nhất với trọng số nào đó mà thỏa một các ràng buộc. Các lớp đối tượng thuật toán này được sử dụng chung trong thành phần định tuyến để hỗ trợ cho việc thực thi các thuật toán định tuyến và lớp thuật toán Dijkstra là một ví dụ điển hình trong việc thực thi thuật toán MHA – thuật toán tìm đường đi qua số hop ít nhất.
* Routing Strategies là thành phần chứa các lớp thuật toán định tuyến được đặt tên theo tên viết tắt của các thuật toán định tuyến như MHA, BCRA, MDWCRA … Những thuật lớp thuật toán định tuyến này đều thực thi cùng một *Interface* là RoutingStrategy để thành phần Simulation Component có thể yêu cầu định tuyến từ các đối tượng định khác nhau trên.

Thành phần Simulation Component là thành phần chứa các lớp đối tượng mà những đối tượng này được phối hợp với nhau để tạo nên một cơ chế giả lập, như việc phân phối yêu cầu theo thời gian đến, đáp ứng yêu cầu theo băng thông và độ trễ, giữ băng thông trong một khoảng thời gian yêu cầu và giải phóng băng thông. Thành phần này gồm những lớp đối tượng sau:

* Lớp Request: là lớp đối tượng đại diện cho yêu cầu định tuyến nắm giữ các thông tin như địa chỉ nút nguồn, đích, thời điểm vào, thời gian giữ băng thông, băng thông yêu cầu, độ trễ.
* Lớp Ticker: là lớp đối tượng để phát sinh sự kiện sau mỗi lần gõ nhịp (trong một khoảng thời gian được gọi là *interval time*), trong bộ giả lập có 2 lớp lắng nghe sự kiện từ lớp này là Request Dispatcher và Response Manager.
* Lớp Request Dispatcher: là lớp đối tượng đại diện cho một bộ phân phối các yêu cầu định tuyến đến các bộ định tuyến theo thời gian đến của yêu cầu.
* Lớp Response Manager: là lớp đối tượng đại diện cho một bộ quản lý tất cả các kết quả định tuyến được gửi đến từ các bộ định tuyến, khi nhận được kết quả định tuyến, ngay lập tức Response Manager giữ băng thông yêu cầu và giải phóng băng thông sau một khoảng thời gian (giữ) kể từ lúc bất đầu giữ.
* Lớp Router: là lớp đối tượng đại diện bộ định tuyến trong hệ thống, mỗi nút trong mô hình mạng tương ứng sẽ có một đối tượng Router có luồng xử lí, không gian trạng thái và đối tượng thuật toán định tuyến riêng. Khi yêu cầu định tuyến được gửi đến từ bộ phân phối yêu cầu – Request Dispatcher, ngay lập tức yêu cầu sẽ được xử lí, tính toán đường đi, sau đó đóng gói đường đi này lại trong một đối tượng Reponse và gửi đến đối tượng Response Manager để thực hiện tiếp việc giữ và giải phóng băng thông.
* Lớp Response: là lớp đối tượng đại diện cho kết quả định tuyến được sử dụng để mang các thông tin như đường đi định tuyến cho Request nào, thời gian tính toán bao nhiêu, giải phóng vào thời điểm nào …



Ảnh 4.4 Lược đồ tuần tự của chức năng giả lập định tuyến



Ảnh 4.5 Luồng con thực hiện việc định tuyến của bộ giả lập.

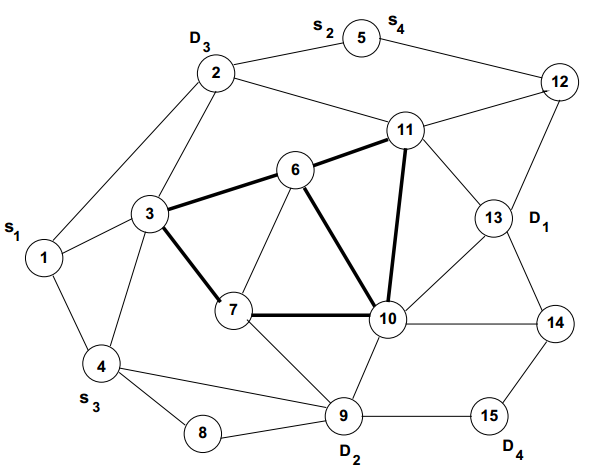


Ảnh 4.6 Luồng con thực hiện việc giải phóng băng thông của bộ giả lập.

## Thực nghiệm và kết quả

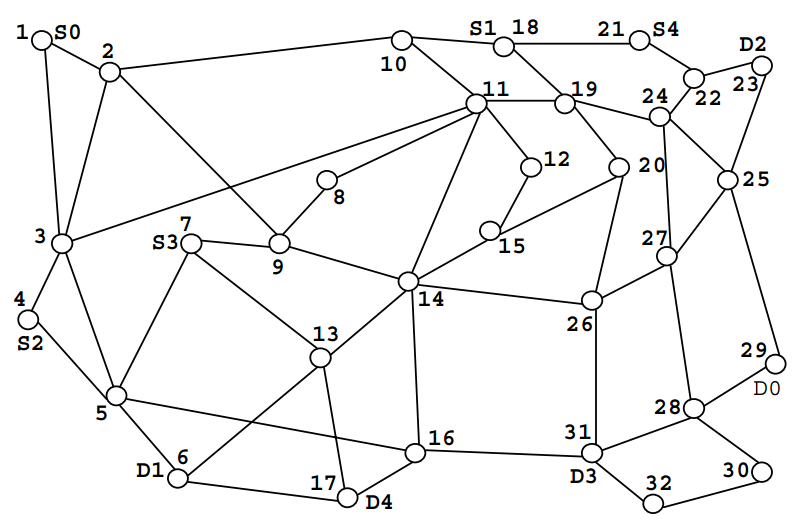
### Thiết lập mạng

Chúng tôi đã sử dụng hai mô hình mạng để so sánh các thuật toán định tuyến với nhau. Thứ nhất là sơ đồ mạng theo [9] được gọi là sơ đồ MIRA và bao gồm 15 nút như Ảnh 4.7. Tất cả các liên kết là hai chiều. Có hai loại liên kết trong mạng: liên kết mảnh có khả năng thông qua là 1200 đơn vị băng thông và liên kết đậm có khả năng thông qua là 4800 đơn vị băng thông. Trong sơ đồ mạng MIRA có bốn cặp vào-ra là: (1, 13), (5, 9), (4, 2), (5, 15).



Ảnh 4.7 Sơ đồ mạng MIRA

Sơ đồ mạng thứ hai là theo [1] [8] được gọi là sơ đồ mạng ANSNET mở rộng bao gồm 32 nút như Ảnh 4.8, mỗi liên kết trong sơ đồ mạng này có 1200 đơn vị băng thông. Có năm cặp vào-ra là (1, 29), (18, 6), (4, 23), (7, 31) và (21, 17).



Ảnh 4.8 Sơ đồ mạng ANSNET mở rộng

Một trong những phần quan trọng của môi trường giả lập là việc phân phối độ trễ liên kết. Chú ý rằng độ trễ liên kết bao gồm độ trễ hàng đợi cũng như độ trễ lang truyền. Các phương pháp tiếp cận phổ biến trong các tài liệu của việc mô hình hóa độ trễ liên kết được tóm tắt bên dưới:

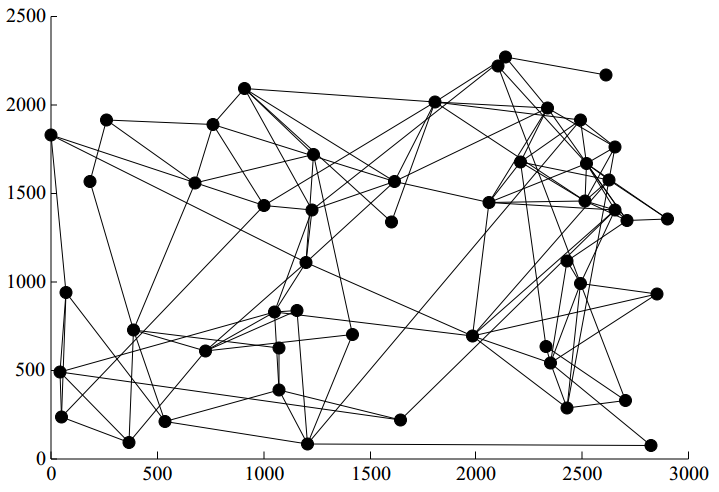
1. Giá trị độ trễ của liên kết được giả định là phân phối đều trong khoảng [5,11]. Một biến thể khác của phương pháp này là nhóm các liên kết lại thành các liên kết cục bộ ngắn (1-5ms), các liên kết cục bộ dài hơn (5-8ms), các liên kết xuyên lục địa (20-30ms).
2. Vị trí của các nút mạng được giữ trong trong một hình chữ nhật có kích thước là 3000 \* 2400 Km2, tương đương diện tích của nước Mỹ. Một bộ phát sinh mạng dựa vào bộ phát sinh của Waxman [10] được sử dụng để tạo ra hệ thống mạng. Đầu tiên bộ phát sinh tạo ra một danh sách các nút (ở vị trí ngẩu nhiên) và sau đó tạo các liên kết giữa những nút này. Xác suất của các liên kết xuất hiện giữ hai nút u và v được cho bởi:

|  |  |  |
| --- | --- | --- |
|  |  | (4.1) |

Với  là khoảng cách giữa *u* và *v*, L là khoảng cách dài nhất của hai nút bất kì,  và  là tham số trong khoảng (0,1]. Giá trị  càng lớn thì càng tăng tỉ lệ số liên kết dài với số liên kết ngắn và giá trị  càng lớn thì càng tăng số bậc trung bình của mạng. Các mạng giả lập đều liên thông và có bậc trung bình là 4 thông qua các thông số đều chỉnh. Mỗi liên kết được giả định là có bộ đệm đầu ra nhỏ. Tốc độ truyền thông qua liên kết được lấy là hai phần ba vận tốc ánh sáng. Phương pháp tiếp cận này được sử dụng trong [11] [12].

Chúng tôi đã áp dụng phương pháp tiếp cận thứ nhất cho hai sơ đồ mạng MIRA và ANSNET mở rộng, độ trễ của các liên kết trong hai sơ đồ mạng này được phân phối đều trong khoảng [5, 11].

Ngoài ra, với phương pháp tiếp cận thứ hai chúng tôi đã sử dụng bộ phát sinh của Waxman [phụ lục 1] để tạo ra sơ đồ mạng mà chúng tôi gọi là NET1 như sau:



Ảnh 4.9 Sơ đồ mạng NET1 với 50 nút được tạo ra từ bộ phát sinh sơ đồ mạng của Waxman với thông số ⍺ = 0.13, β = 0.99.

Như vậy, Có 3 sơ đồ mạng: MIRA, ANSNET mở rộng và NET1, tương ứng với mỗi sơ đồ mạng này chúng tôi đã dựng các kịch bản giả lập khác nhau mà chi tiết được trình bày ở mục sau.

### Thực nghiệm

Theo [13] [4] [14] các yêu cầu định tuyến được gửi đến giữa mỗi cặp vào-ra theo tiến trình Poisson với tần suất trung bình là λ và thời gian giữ được phân phối mũ với giá trịnh trung bình là 1/μ.

Trong phần thực nghiệm, chúng tôi đã xây dựng các kịch bản khác nhau cho mỗi sơ mạng (MIRA, ANSET mở rộng, NET1) như sau:

* Đối với đồ mạng MIRA: phát sinh 2000 yêu cầu định tuyến động với thông số λ = 40, μ = 10, băng thông yêu cầu được phân phối đều trong {10, 20, 30, 40} và độ trễ yêu cầu được phân phối đều trong khoảng [35, 45].
* Đối với đồ mạng ANSET mở rộng: phát sinh 2000 yêu cầu định tuyến động với thông số λ = 40, μ = 10, băng thông yêu cầu được phân phối đều trong {10, 20, 30, 40, 50} và độ trễ yêu cầu được phân phối đều trong khoảng [40, 50].
* Đối với đồ mạng NET1: phát sinh 4000 yêu cầu định tuyến động với thông số λ = 80, μ = 30, băng thông yêu cầu được phân phối đều trong {10, 20, 30, 40, 50} và độ trễ yêu cầu được phân phối đều trong khoảng [15, 25].

Với mỗi kịch bản trên chúng tôi sử dụng các thước hiệu năng sau để so sánh các thuật toán khác nhau như sau:

1. *Tỉ lệ từ chối yêu cầu*:

|  |  |  |  |
| --- | --- | --- | --- |
|  | *Tỉ lệ từ chối yêu cầu =* | *Số yêu cầu bị từ chối* | (4.2) |
| *Tổng số yêu cầu* |

mục tiêu tối ưu là cực tiểu hóa tỉ lệ từ chối yêu cầu, do đó sẽ tối đa hóa tỉ lệ chấp nhận yêu cầu.

1. *Thời gian tính toán trung bình*: là thời gian tính toán trung bình (trong mili giây) mà thuật toán mất phải để xác định đường đi cho yêu cầu.

### Kết quả

Kết quả thực nghiệm cho thước đo *tỉ lệ từ chối yêu* cầu được trình bày ở Ảnh 4.10 đến Ảnh 4.12. Trục *x* biểu diễn số yêu cầu và trục *y* biểu diễn *tỉ lệ từ chối yêu cầu.* Với thước đo này, chúng tôi đã so sánh 5 thuật toán định tuyến khác nhau gồm có: LDPRA, BDCRA, MDWCRA, OBDCRA, M-OBDCRA, tương ứng được kí hiệu như trong đồ thị. Từ Ảnh 4.10 đến Ảnh 4.12 chúng tôi thấy rằng, thuật toán LDPRA luôn có tỉ lệ từ chối yêu cầu cao nhất. Nhược điểm của thuật toán định tuyến này như đã xác định là luôn tìm đường đi có độ trễ thấp nhất cho mỗi yêu cầu, mà không tâm đến việc cân bằng tải. Đều này có thể làm cho các liên kết nhanh chóng bị tắc nghẽn và dẫn đến việc giảm tỉ lệ chấp nhận yêu cầu. Đối với thuật toán định tuyến BDCRA, kết quả ở Ảnh 4.10 cho thấy rất hiệu quả với tỉ lệ từ chối yêu cầu gần như thấp nhất chỉ cao hơn M-OBDCRA, nhưng kết quả từ Ảnh 4.11 đến Ảnh 4.12 thì BDCRA có tỉ lệ từ chối yêu cầu tương đối cao hơn so vởi các thuật toán định tuyến còn lại trừ LDPRA. Điều này thật sự phù hợp với đặc điểm của BDCRA mà chúng tôi đã xác định, đó là khả năng ứng dụng chung thấp do phụ thuộc vào hằng số 108 trong hàm số heuristic và cách mà thuật toán tìm đường đi thỏa mãn băng thông-độ trễ vì vậy mà hiệu quả của thuật toán có thể tốt với sơ đồ mạng này nhưng không tốt với sơ đồ mạng khác. Tiếp đến thuật toán MDWCRA, tỉ lệ từ chối yêu cầu của thuật toán này luôn ở vị trí ổn định, tức là thấp hơn LDPRA và cao hơn OBDCRA, M-OBDCRA. Thuật toán OBDCRA được thiết kế với tiêu chí cải thiện việc cân bằng tải nhờ đó tỉ lệ từ chối yêu cầu thấp hơn các thuật toán còn lại trừ thuật toán M-OBDCRA, một thuật toán được cải tiến để khắc phục nhược điểm của OBDCRA nên tỉ lệ từ chối yêu cầu được giữ ở mức thấp nhất.

Ảnh 4.10 Đồ thị tỉ lệ từ chối yêu cầu (mạng MIRA, băng thông  {10, 20, 30, 40}, độ trễ  [35, 45] ms)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| NR | LDPRA | BDCRA | MDWCRA | OBDCRA | M-OBDCRA |
| 100 | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% |
| 200 | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% |
| 300 | 0.67% | 0.00% | 0.00% | 0.00% | 0.00% |
| 400 | 2.00% | 0.25% | 0.00% | 0.00% | 0.00% |
| 500 | 3.20% | 0.60% | 1.00% | 1.00% | 0.60% |
| 600 | 5.00% | 1.83% | 2.33% | 2.00% | 1.83% |
| 700 | 5.00% | 2.29% | 3.43% | 2.71% | 2.29% |
| 800 | 7.63% | 5.38% | 6.88% | 6.00% | 5.50% |
| 900 | 8.67% | 5.89% | 7.44% | 6.44% | 6.22% |
| 1000 | 9.70% | 6.00% | 8.30% | 6.70% | 6.40% |
| 1100 | 11.09% | 7.18% | 9.09% | 7.45% | 7.18% |
| 1200 | 11.67% | 7.67% | 9.67% | 8.17% | 7.67% |
| 1300 | 11.62% | 7.92% | 9.31% | 8.00% | 7.62% |
| 1400 | 12.14% | 8.43% | 9.71% | 8.50% | 8.29% |
| 1500 | 12.53% | 8.73% | 10.00% | 8.73% | 8.73% |
| 1600 | 12.88% | 8.94% | 10.56% | 9.06% | 9.25% |
| 1700 | 12.88% | 9.29% | 10.88% | 9.59% | 9.71% |
| 1800 | 13.56% | 10.17% | 11.61% | 10.61% | 10.28% |
| 1900 | 13.21% | 9.84% | 11.11% | 10.32% | 9.95% |
| 2000 | 13.10% | 9.75% | 11.05% | 9.95% | 9.65% |

Bảng 4.1 Tỉ lệ từ chối yêu cầu (mạng MIRA, băng băng  {10, 20, 30, 40}, độ trễ  [35, 45] ms)

Ảnh 4.11 Đồ thị tỉ lệ từ chối yêu cầu (mạng ANSNET mở rộng, băng thông  {10, 20, 30, 40, 50}, độ trễ  [40, 50] ms)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| NR | LDPRA | BDCRA | MDWCRA | OBDCRA | M-OBDCRA |
| 100 | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% |
| 200 | 15.50% | 13.50% | 7.00% | 0.00% | 0.00% |
| 300 | 20.67% | 20.67% | 14.67% | 10.67% | 10.67% |
| 400 | 19.75% | 21.00% | 16.25% | 11.75% | 11.00% |
| 500 | 23.40% | 24.20% | 19.60% | 16.60% | 14.60% |
| 600 | 24.33% | 25.00% | 20.67% | 17.67% | 16.00% |
| 700 | 26.14% | 26.00% | 22.57% | 19.57% | 18.00% |
| 800 | 26.50% | 26.38% | 23.13% | 20.13% | 18.88% |
| 900 | 25.56% | 25.56% | 22.22% | 19.11% | 18.22% |
| 1000 | 24.80% | 24.70% | 21.00% | 18.50% | 17.50% |
| 1100 | 24.64% | 24.45% | 20.64% | 18.45% | 17.55% |
| 1200 | 25.25% | 24.50% | 20.50% | 18.50% | 18.00% |
| 1300 | 25.08% | 24.62% | 20.54% | 18.62% | 18.08% |
| 1400 | 25.36% | 24.93% | 21.64% | 19.36% | 18.57% |
| 1500 | 25.00% | 24.40% | 21.27% | 19.20% | 18.33% |
| 1600 | 24.88% | 24.19% | 21.31% | 19.00% | 18.25% |
| 1700 | 24.47% | 23.88% | 21.18% | 18.65% | 18.06% |
| 1800 | 24.67% | 23.83% | 21.56% | 18.61% | 18.17% |
| 1900 | 24.95% | 24.21% | 21.68% | 19.11% | 18.84% |
| 2000 | 25.20% | 24.25% | 22.10% | 19.60% | 19.40% |

Bảng 4.2 Tỉ lệ từ chối yêu cầu (mạng ANSNET mở rộng, băng thông  {10, 20, 30, 40, 50}, độ trễ  [40, 50] ms)

Ảnh 4.12 Đồ thị tỉ lệ từ chối yêu cầu (mạng NET1 mở rộng, băng thông  {10, 20, 30, 40, 50}, độ trễ  [15, 25] ms)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| NR | LDPRA | BDCRA | MDWCRA | OBDCRA | M-OBDCRA |
| 100 | 0.00% | 0.00% | 0.00% | 0.00% | 0.00% |
| 200 | 3.50% | 3.50% | 3.50% | 3.50% | 3.50% |
| 300 | 3.33% | 3.33% | 3.33% | 3.33% | 3.33% |
| 400 | 3.25% | 3.25% | 3.25% | 3.25% | 3.25% |
| 500 | 3.60% | 3.60% | 3.60% | 3.60% | 3.60% |
| 600 | 3.67% | 3.33% | 3.33% | 3.33% | 3.33% |
| 700 | 4.14% | 3.57% | 3.57% | 3.57% | 3.57% |
| 800 | 4.13% | 3.38% | 3.38% | 3.38% | 3.38% |
| 900 | 4.67% | 3.78% | 3.78% | 3.78% | 3.78% |
| 1000 | 4.90% | 4.00% | 4.00% | 4.00% | 4.00% |
| 1100 | 5.55% | 4.18% | 4.27% | 4.18% | 4.18% |
| 1200 | 6.17% | 4.25% | 4.42% | 4.25% | 4.25% |
| 1300 | 6.54% | 4.46% | 4.54% | 4.46% | 4.46% |
| 1400 | 6.93% | 4.86% | 4.86% | 4.86% | 4.86% |
| 1500 | 7.33% | 5.07% | 5.00% | 5.00% | 5.00% |
| 1600 | 7.75% | 5.56% | 5.56% | 5.31% | 5.31% |
| 1700 | 7.88% | 5.59% | 5.53% | 5.29% | 5.29% |
| 1800 | 8.06% | 5.78% | 5.72% | 5.33% | 5.33% |
| 1900 | 8.53% | 6.16% | 6.05% | 5.58% | 5.58% |
| 2000 | 9.00% | 6.55% | 6.25% | 5.85% | 5.85% |
| 2100 | 10.00% | 7.10% | 6.48% | 6.14% | 6.00% |
| 2200 | 10.09% | 7.27% | 6.77% | 6.36% | 6.27% |
| 2300 | 10.65% | 7.96% | 7.35% | 7.04% | 6.87% |
| 2400 | 10.92% | 8.46% | 7.79% | 7.58% | 7.50% |
| 2500 | 11.44% | 9.04% | 8.48% | 8.00% | 8.00% |
| 2600 | 12.04% | 9.88% | 9.31% | 8.69% | 8.54% |
| 2700 | 12.59% | 10.48% | 9.96% | 9.41% | 9.11% |
| 2800 | 12.54% | 10.71% | 9.93% | 9.39% | 9.18% |
| 2900 | 12.76% | 10.79% | 10.24% | 9.45% | 9.24% |
| 3000 | 13.17% | 11.20% | 10.57% | 9.77% | 9.67% |
| 3100 | 13.29% | 11.55% | 11.03% | 10.10% | 9.87% |
| 3200 | 13.56% | 11.84% | 11.50% | 10.44% | 10.19% |
| 3300 | 13.91% | 12.15% | 11.94% | 10.70% | 10.39% |
| 3400 | 13.88% | 12.18% | 12.03% | 10.74% | 10.50% |
| 3500 | 14.03% | 12.49% | 12.46% | 11.20% | 10.80% |
| 3600 | 14.47% | 13.03% | 12.89% | 11.69% | 11.25% |
| 3700 | 14.95% | 13.38% | 13.59% | 12.24% | 11.78% |
| 3800 | 15.39% | 13.97% | 14.16% | 12.84% | 12.37% |
| 3900 | 15.56% | 14.23% | 14.51% | 13.21% | 12.67% |
| 4000 | 15.63% | 14.40% | 14.58% | 13.30% | 12.78% |

Bảng 4.3 Tỉ lệ từ chối yêu cầu (mạng NET1 mở rộng, băng thông  {10, 20, 30, 40, 50}, độ trễ  [15, 25] ms)

Kết quả thực nghiệm cho thước đo *thời gian tính toán trung bình* được trình bài ở Ảnh 4.13 đến Ảnh 4.15. Trục *x* biểu diễn số yêu cầu, trục *y* biểu diễn thời gian tính toán trung bình của giải thuật (tính bằng mili giây). Từ Ảnh 4.13 đến Ảnh 4.15 chúng tôi thấy rằng, thời gian tính toán trung bình của thuật toán LDPRA thấp nhất nhờ sự đơn giản với độ phức tạp là O(*nlogn + m)*, thời gian tính toán trung bình dao động trong khoảng từ 0.11 đến 0.26 mili giây đối với sơ đồ MIRA, từ 0.25 đến 0.39 mili giây đối với sơ đồ ANSNET mở rộng và từ 0.42 đến 0.65 mili giây đối với sơ đồ NET1. Tiếp theo là thuật toán OBDCRA có độ phức tạp bằng hai lần giải thuật Dijkstra O(*nlogn + m*) vì thế thời gian tính toán trung bình cao hơn so với LDPRA, dao động trong khoảng từ 0.33 đến 0.46 mili giây đối với sơ đồ mạng MIRA, từ 0.56 đến 0.62 mili giây đối với sơ đồ ANSNET mở rộng và từ 1.21 đến 1.58 mili giây đối với sơ đồ mạng NET1. Tiếp theo là thuật toán BDCRA có thời gian tính toán trung bình dao động trong khoảng từ 0.25 đến 0.52 mili giây đối với sơ đồ MIRA, từ 0.41 đến 0.63 mili giây đối với sơ đồ ANSNET mở rộng và từ 1.51 đến 2.10 mili giây đối với sơ đồ mạng NET1. Kế tiếp là thuật toán M-OBDCRA có thời gian tính toán trung bình giao động trọng khoảng từ 1.02 đến 1.53 mili giây đối với sơ đồ mạng MIRA, từ 1.25 đến 1.65 mili giây đối với sơ đồ mạng ANSNET mở rộng và từ 6.90 đến 7.14 mili giây đối với sơ đồ mạng NET1. Còn lại thuật toán MDWCRA có thời gian tính toán chậm nhất do thuật toán này có độ phức tạp tương đối cao, thời gian tính toán trung bình giao động trong khoảng từ 10.65 đến 11.45 mili giây đối với sơ đồ mạng MIRA, từ 48.76 đến 52.93 mili giây đối với sơ đồ mạng ANSNET mở rộng và từ 454.73 đến 514.44 mili giây đối với sơ đồ mạng NET1.

Ảnh 4.13 Đồ thị thời gian tính toán trung bình (mạng MIRA, băng thông  {10, 20, 30, 40}, độ trễ  [35, 45] ms)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| NR | LDPRA | BDCRA | MDWCRA | OBDCRA | M-OBDCRA |
| **100** | 0.26 | 0.52 | 10.65 | 0.46 | 1.53 |
| **200** | 0.19 | 0.36 | 10.26 | 0.35 | 1.37 |
| **300** | 0.16 | 0.32 | 10.74 | 0.33 | 1.31 |
| **400** | 0.15 | 0.30 | 11.11 | 0.33 | 1.29 |
| **500** | 0.14 | 0.28 | 11.30 | 0.33 | 1.23 |
| **600** | 0.14 | 0.27 | 11.38 | 0.33 | 1.21 |
| **700** | 0.13 | 0.27 | 11.41 | 0.33 | 1.19 |
| **800** | 0.13 | 0.26 | 11.42 | 0.33 | 1.15 |
| **900** | 0.13 | 0.26 | 11.45 | 0.33 | 1.13 |
| **1000** | 0.13 | 0.26 | 11.47 | 0.33 | 1.12 |
| **1100** | 0.13 | 0.25 | 11.49 | 0.33 | 1.10 |
| **1200** | 0.12 | 0.25 | 11.50 | 0.33 | 1.09 |
| **1300** | 0.12 | 0.25 | 11.54 | 0.33 | 1.10 |
| **1400** | 0.12 | 0.25 | 11.57 | 0.33 | 1.09 |
| **1500** | 0.12 | 0.25 | 11.58 | 0.33 | 1.08 |
| **1600** | 0.12 | 0.25 | 11.61 | 0.33 | 1.07 |
| **1700** | 0.12 | 0.25 | 11.60 | 0.33 | 1.06 |
| **1800** | 0.12 | 0.25 | 11.59 | 0.33 | 1.04 |
| **1900** | 0.12 | 0.25 | 11.53 | 0.33 | 1.03 |
| **2000** | 0.11 | 0.25 | 11.45 | 0.33 | 1.02 |

Bảng 4.4 Số liệu thời gian tính toán trung bình (ms) (mạng MIRA, băng thông  {10, 20, 30, 40}, độ trễ  [35, 45] ms)

Ảnh 4.14 Đồ thị thời gian tính toán trung bình (mạng ANSNET mở rộng, băng thông  {10, 20, 30, 40, 50}, độ trễ  [40, 50] ms)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **NR** | LDPRA | BDCRA | MDWCRA | OBDCRA | M-OBDCRA |
| **100** | 0.39 | 0.64 | 52.93 | 0.62 | 1.65 |
| **200** | 0.32 | 0.48 | 53.06 | 0.55 | 1.50 |
| **300** | 0.29 | 0.44 | 52.51 | 0.52 | 1.39 |
| **400** | 0.29 | 0.44 | 52.48 | 0.51 | 1.38 |
| **500** | 0.28 | 0.42 | 52.37 | 0.51 | 1.34 |
| **600** | 0.27 | 0.42 | 52.06 | 0.52 | 1.32 |
| **700** | 0.27 | 0.42 | 52.17 | 0.53 | 1.30 |
| **800** | 0.27 | 0.42 | 51.95 | 0.54 | 1.29 |
| **900** | 0.27 | 0.42 | 51.52 | 0.55 | 1.30 |
| **1000** | 0.26 | 0.42 | 51.47 | 0.55 | 1.30 |
| **1100** | 0.26 | 0.42 | 51.40 | 0.55 | 1.30 |
| **1200** | 0.26 | 0.42 | 51.23 | 0.56 | 1.29 |
| **1300** | 0.26 | 0.41 | 51.03 | 0.56 | 1.28 |
| **1400** | 0.26 | 0.41 | 50.88 | 0.56 | 1.27 |
| **1500** | 0.26 | 0.41 | 50.80 | 0.56 | 1.27 |
| **1600** | 0.26 | 0.41 | 50.69 | 0.56 | 1.27 |
| **1700** | 0.26 | 0.41 | 50.15 | 0.56 | 1.26 |
| **1800** | 0.26 | 0.41 | 49.64 | 0.56 | 1.26 |
| **1900** | 0.25 | 0.41 | 49.18 | 0.56 | 1.26 |
| **2000** | 0.25 | 0.41 | 48.76 | 0.56 | 1.25 |

Bảng 4.5 Số liệu thời gian tính toán trung bình (ms) (mạng ANSNET mở rộng, băng thông  {10, 20, 30, 40, 50}, độ trễ  [40, 50] ms)

Ảnh 4.15 Đồ thị thời gian tính toán trung bình (mạng NET1 mở rộng, băng thông  {10, 20, 30, 40, 50}, độ trễ  [15, 25] ms)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| NR | LDPRA | BDCRA | MDWCRA | OBDCRA | M-OBDCRA |
| 100 | 0.65 | 2.10 | 454.73 | 1.58 | 7.14 |
| 200 | 0.51 | 1.97 | 493.94 | 1.42 | 6.99 |
| 300 | 0.46 | 1.96 | 504.36 | 1.35 | 7.36 |
| 400 | 0.45 | 1.96 | 514.44 | 1.30 | 7.67 |
| 500 | 0.44 | 1.98 | 507.46 | 1.29 | 7.89 |
| 600 | 0.44 | 1.97 | 491.66 | 1.29 | 7.92 |
| 700 | 0.43 | 1.93 | 478.95 | 1.27 | 7.80 |
| 800 | 0.43 | 1.92 | 469.63 | 1.26 | 7.76 |
| 900 | 0.43 | 1.88 | 461.49 | 1.25 | 7.61 |
| 1000 | 0.43 | 1.87 | 454.82 | 1.26 | 7.60 |
| 1100 | 0.43 | 1.87 | 449.50 | 1.26 | 7.60 |
| 1200 | 0.43 | 1.87 | 445.59 | 1.26 | 7.71 |
| 1300 | 0.43 | 1.87 | 442.28 | 1.26 | 7.79 |
| 1400 | 0.42 | 1.87 | 440.91 | 1.25 | 7.80 |
| 1500 | 0.42 | 1.88 | 443.73 | 1.24 | 7.85 |
| 1600 | 0.42 | 1.87 | 458.17 | 1.22 | 7.87 |
| 1700 | 0.42 | 1.88 | 471.71 | 1.21 | 7.99 |
| 1800 | 0.42 | 1.87 | 480.62 | 1.20 | 7.99 |
| 1900 | 0.42 | 1.85 | 485.17 | 1.19 | 7.98 |
| 2000 | 0.42 | 1.84 | 491.47 | 1.19 | 7.97 |
| 2100 | 0.42 | 1.81 | 493.55 | 1.19 | 7.91 |
| 2200 | 0.42 | 1.79 | 494.50 | 1.19 | 7.86 |
| 2300 | 0.42 | 1.78 | 494.34 | 1.19 | 7.83 |
| 2400 | 0.42 | 1.76 | 493.87 | 1.19 | 7.78 |
| 2500 | 0.42 | 1.74 | 490.23 | 1.20 | 7.67 |
| 2600 | 0.42 | 1.71 | 487.42 | 1.20 | 7.59 |
| 2700 | 0.42 | 1.69 | 485.29 | 1.20 | 7.50 |
| 2800 | 0.42 | 1.68 | 481.89 | 1.20 | 7.47 |
| 2900 | 0.42 | 1.66 | 478.75 | 1.20 | 7.42 |
| 3000 | 0.42 | 1.65 | 475.86 | 1.20 | 7.38 |
| 3100 | 0.42 | 1.63 | 473.61 | 1.20 | 7.31 |
| 3200 | 0.42 | 1.62 | 471.70 | 1.20 | 7.24 |
| 3300 | 0.42 | 1.60 | 469.27 | 1.20 | 7.16 |
| 3400 | 0.42 | 1.59 | 467.01 | 1.20 | 7.17 |
| 3500 | 0.42 | 1.58 | 464.79 | 1.21 | 7.14 |
| 3600 | 0.42 | 1.56 | 463.05 | 1.21 | 7.09 |
| 3700 | 0.42 | 1.54 | 461.05 | 1.21 | 7.01 |
| 3800 | 0.42 | 1.53 | 459.26 | 1.21 | 6.93 |
| 3900 | 0.42 | 1.52 | 457.71 | 1.21 | 6.91 |
| 4000 | 0.42 | 1.51 | 456.27 | 1.21 | 6.90 |

Bảng 4.6 Số liệu thời gian tính toán trung bình (ms) (mạng NET1 mở rộng, băng thông  {10, 20, 30, 40, 50}, độ trễ  [15, 25] ms)

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết quả đạt được

Chúng tôi đã đề xuất được một thuật toán định tuyến mới (OBDCRA) thỏa mãn cả băng thông và ràng buộc độ trễ. Thuật toán này thật sự hiệu quả trong việc định tuyến cải thiện việc sử dụng tài nguyên mạng và thời gian tính toán thấp có thể được chấp nhất bởi các hệ thống mạng lớn. Nhận thấy OBDCRA tồn tại một số khuyết điểm, vì vậy chúng tôi đã tiếp tục cải tiến OBDCRA thành M-OBDCRA và kết quả thực nghiệm đã cho thấy M-OBDCRA có kết quả luôn tốt hơn OBDCRA và các thuật toán định tuyến liên quan.

Về việc thực nghiệm, chúng tôi đã xây dựng được một môi trường giả lập thuật toán định tuyến đơn giản và hiệu quả để so sánh các thuật toán định tuyến khác nhau. Bên cạnh đó chúng tôi cũng đã xây dựng một bộ phát sinh yêu cầu định tuyến theo kịch bản bằng việc thiết lập các thông số để phục vụ cho các mục đích hay tiêu chí đánh giá khác nhau.

## Hạn chế

Thực nghiệm cho thấy OBDCRA và M-OBDCRA thực sự hiệu quả với thời gian tính toán thấp. Song đường đi tìm được chưa thực sự là tối ưu do chưa quan tâm đến việc hạn chế ảnh hưởng đối với các yêu cầu khác trong tương lai.

## Hướng phát triển

Trong tương lai chúng tôi sẽ xem xét các vấn đề như: dự đoán yêu cầu trong tương lai, thống kê dữ liệu yêu cầu trong quá khứ, áp dụng máy học vào thuật toán định tuyến … để tiếp tục cải thiện hiệu quả của OBDCRA và M-OBDCRA.

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | S. Chen, "Routing Support for Providing Guaranteed End-To-End Quality of Service," 1999. |
| [2] | W. Z and C. J, "Qos routing for supporting resource reservation," *IEEE Journal on Selected areas in Communications,* 1996. |
| [3] | R. Guerin, A. Orda and D. Williams, "QoS Routing Mechanisms and OSPF Extensions," *Global Telecommunications Conference, 1997. GLOBECOM '97., IEEE ,* vol. 3, pp. 1903-1908, 1997. |
| [4] | K. Kar, M. Kodialam and T. Lakshman, "Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications," *Selected Areas in Communications, IEEE Journal on ,* vol. 18, no. 12, pp. 2566 - 2579, 2000. |
| [5] | A. Kotti, R. Hamza and K. Bouleimen, "Bandwidth Constrained Routing Algorithm for MPLS Traffic Engineering," *Networking and Services, 2007. ICNS. Third International Conference on ,* pp. 20-20, 2007. |
| [6] | Z. Wang and J. Crowcroft, "Bandwidth-Delay Based Routing Algorithms," *Global Telecommunications Conference, 1995. GLOBECOM '95., IEEE ,* vol. 3, pp. 2129 - 2133, 1995. |
| [7] | A. Kotti, R. Hamza and K. Bouleimen, "Bandwidth Constrained Least Delay Least Cost Routing Algorithm," *EUROCON, 2007. The International Conference on "Computer as a Tool" ,* pp. 1005-1008, 2007. |
| [8] | Y. Yang, L. Zhang, J. K. Muppala and S. T. Chanson, "Bandwidth-Delay Constrained Routing Algorithms," *Computer Networks: The International Journal of Computer and Telecommunications Networking,* vol. 42, no. 4, pp. 503-520, 15 July 2003. |
| [9] | M. Kodialam and T. Lakshman, "Minimum interference routing with applications to MPLS traffic engineering," *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE ,* vol. 2, pp. 884 - 893, 2000. |
| [10] | B. Waxman, "Routing of Multipoint Connections," *Selected Areas in Communications, IEEE Journal on ,* vol. 6, no. 9, pp. 1617- 1622 , 1988 . |
| [11] | J. Zhou, "A new distributed routing algorithm for supporting delay-sensitive applications," *Communication Technology Proceedings, 1998. ICCT '98. 1998 International Conference on ,* vol. 2, 1998. |
| [12] | D. Reeves and H. Salama, "A Distributed Algorithm for Delay-Constrained," *Networking, IEEE/ACM Transactions on ,* vol. 8, no. 2, pp. 239- 250 , 2000. |
| [13] | A. A. Turky and A. Mitschele-Thiel, "MPLS Online Routing Optimization Using Prediction," *Lecture Notes in Computer Science,* vol. 5425 , pp. 45-52, 2008. |
| [14] | P.-T. C. Thai and H. T. Cong, "A Study of Bandwidth Guaranteed Routing Algorithms for Traffic Engineering," *Advanced Methods for Computational Collective Intelligence,* vol. 457, pp. 313-322, 2013. |

# PHỤ LỤC

1. **Bộ phát sinh sơ đồ mạng Waxman**

Hàm MATLAB để giả lập và hiển thị sơ đồ mạng ngẫu nhiên bằng phương pháp được đề xuất bởi Waxman (1998). Tài liệu chi tiết có thể tìm thấy tại <http://www.math.uu.se/research/telecom/software/>

Tác giả:

* Ingemar Kaj <ikaj@math.uu.se>
* Raimundas Gaigalas <jaunas@math.uu.se>

Mã nguồn:

|  |
| --- |
| function [adj\_matr, nd\_coord]=waxtop(lambda, alpha, beta, domain)  % WAXTOP Simulate and plot a random network topology by the method  % suggested by Waxman (1988):  % - nodes are a Poisson process in the plane with scaled  % Lebesgue mean measure  % - nodes u and v are connected with probability  % P(u,v)=alpha\*exp(-d(u, v)/(beta\*L)),  % where alpha>0, beta<=1, d(u,v) is Euclidean distance,  % L is the maximum distance between any two nodes  %  % [adj\_matr, nd\_coord] = waxtop(lambda,alpha, beta, domain)  %  % Inputs:  % lambda - intensity of the Poisson process  % alpha - maximal link probability  % beta - parameter to control length of the edges. Increased <beta>  % yields a larger ratio of long edges to short edges  % domain - bounds for the region. A 4-dimensional vector in  % the form [x\_min x\_max y\_min y\_max].  %  % Outputs:  % adj\_matr - adjacency matrix of the graph of the topology  % nd\_coord - coordinates of the nodes  % Authors: R.Gaigalas, I.Kaj  % Detailed documentation at  % http://www.math.uu.se/research/telecom/software  % v1.5 Created 07-Nov-01  % Modified 23-Nov-05 changed variable names  %  if (nargin<4) % default parameter values  lambda = 0.6; % intensity of the Poisson process  alpha = 0.99; % parameter for the link probability  beta = 0.13; % parameter for the link probability  domain = [0 3000 0 2400]; % bounds for the "geografical" domain  end  xmin = domain(1);  xmax = domain(2);  ymin = domain(3);  ymax = domain(4);  clear domain;    % number of points is Poisson distributed  % with intensity proportional to the area  area = (xmax-xmin)\*(ymax-ymin);  % npoints = poissrnd(lambda\*area)  npoints = 50;    % given the number of points, nodes are uniformly distributed  nd\_coord = rand(npoints, 2);  nd\_coord(:, 1) = nd\_coord(:, 1)\*(xmax-xmin)+xmin;  nd\_coord(:, 2) = nd\_coord(:, 2)\*(ymax-ymin)+ymin;    % create a matrix with all possible distances  x\_rep = repmat(nd\_coord(:, 1), 1, npoints);  y\_rep = repmat(nd\_coord(:, 2), 1, npoints);  dist\_matr = sparse(triu(((x\_rep-x\_rep').^2 + ...  (y\_rep-y\_rep').^2).^0.5, 1));    % create the matrix of probabilities  prob\_matr = alpha\*spfun('exp', ...  -dist\_matr./(beta\*max(max(dist\_matr))));  % generate the adjacency matrix  runi = sprand(dist\_matr);  adj\_matr = (runi>0) & (runi < prob\_matr);  % test for connectivity  % s\_matr = speye(size(adj\_matr));  % for i=1:npoints-1  % s\_matr = s\_matr+adj\_matr^i;  % end  % length(find(s\_matr==0))    % plot the network  figure(1);  clf;  hold on;  plot(nd\_coord(:, 1), nd\_coord(:, 2), '.');  gplot(adj\_matr, nd\_coord);  hold off;  display(adj\_matr);  display(nd\_coord); |

1. Single-source shortest path là vấn đề đường đi ngắn nhất từ tất cả các nút trong đồ thị có hướng đến một nút đích *v*, vấn đề này có thể được giải quyết bằng giải thuật của Dijkstra hoặc Bellman-Ford. [↑](#footnote-ref-1)
2. Destination-source shortest path là một biến thể của vấn đề single-source shotest path mà có thể được giải quyết như vấn đề single-source shortest path bằng cách đổi chiều tất cả các liên kết trong đồ thị. [↑](#footnote-ref-2)