**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG**

**NGUYỄN THANH TÙNG – 23521744**

**ĐỖ ĐỨC MINH TRIẾT - 23521650**

**BÁO CÁO ĐỒ ÁN CUỐI KỲ**

**EVALUATING NETWORK CONGESTION CONTROL WITH IPERF3 AND LINUX TRAFFIC CONTROL (TC/NETEM)**

**GIẢNG VIÊN HƯỚNG DẪN**

**LÊ TRUNG QUÂN**

**NGUYỄN VĂN BẢO**

**TP. HỒ CHÍ MINH, 2025**

**Mục lục**

[I. CƠ SỞ LÝ THUYẾT 6](#_Toc212234211)

[1.1. ARRIVAL PROCESSES (QUÁ TRÌNH ĐẾN) 6](#_Toc212234212)

[**1.2. MÔ HÌNH HÀNG ĐỢI CƠ BẢN** 7](#_Toc212234213)

[1.3. RED (RANDOM EARLY DETECTION) 8](#_Toc212234214)

[1. Nguyên lý hoạt động: 8](#_Toc212234215)

[2. Ưu điểm: 8](#_Toc212234216)

[1.4. CUBIC (TCP CONGESTION CONTROL) 8](#_Toc212234217)

[1. Ý tưởng chính: 8](#_Toc212234218)

[2. Ưu điểm: 9](#_Toc212234219)

[1.5. Các hàng đợi (qdisc) phổ biến trong Linux 9](#_Toc212234220)

[1.6. iperf - Công Cụ Đo Băng Thông và Hiệu Năng 10](#_Toc212234221)

[1.7. tc (Traffic Control) và netem - Công Cụ Điều Khiển và Mô Phỏng Mạng 11](#_Toc212234222)

[1.8. tcpdump - Công Cụ Phân Tích Gói Tin Mạng 11](#_Toc212234223)

[1.9. ss - Công Cụ Thống Kê Socket 12](#_Toc212234224)

[1.10. ifstat - Công Cụ Thống Kê Lưu Lượng Giao Diện Mạng 12](#_Toc212234225)

[II. MÔ HÌNH TRIỂN KHAI 13](#_Toc212234226)

[2.1. Mô hình tổng quan 13](#_Toc212234227)

[2.2. Sơ đồ hệ thống 13](#_Toc212234228)

[2.3. Kịch bản thử nghiệm 13](#_Toc212234229)

Bảng

[Bảng 1. Hệ thống hàng đợi 8](#_Toc212317240)

[Bảng 2. hai đại lượng cơ bản 11](#_Toc212317241)

[Bảng 3. 4 pha chính BBR hoạt động 12](#_Toc212317242)

[Bảng 4. So sánh BBR với Cubic 12](#_Toc212317243)

[Bảng 6. Ví dụ về cấu hình trong Linux của RED 13](#_Toc212317244)

[Bảng 7. So sánh qdisc 13](#_Toc212317245)

**Hình ảnh**

[Hình 1. Sơ đồ hệ thống 13](#_Toc212292879)

# I. CƠ SỞ LÝ THUYẾT

## 1.1. ARRIVAL PROCESSES (QUÁ TRÌNH ĐẾN)

Trong lý thuyết hàng đợi, **arrival process** mô tả cách các gói tin (hoặc khách hàng) đến hệ thống theo thời gian.

Các đặc trưng chính của arrival process gồm:

* **Tốc độ đến trung bình (λ - arrival rate)**: số gói đến trung bình trong một đơn vị thời gian.
* **Phân phối thời gian giữa các lần đến (inter-arrival time)**: thời gian giữa hai gói tin liên tiếp.

**Các mô hình arrival phổ biến:**

* **Deterministic Arrival (D)**: gói đến theo chu kỳ cố định.
* **Poisson Process (M)**: thời gian giữa các gói đến tuân theo phân phối mũ — đây là mô hình phổ biến nhất do tính chất “không nhớ” (*memoryless*).
* **General Arrival (G)**: thời gian đến có phân phối bất kỳ.

Trong mô hình hàng đợi, ký hiệu M/D/1, M/M/1, v.v… biểu diễn:

* Chữ đầu: loại arrival process (M – Markov/Poisson, D – deterministic).
* Chữ thứ hai: loại service time distribution.
* Số cuối: số lượng server (thường là 1).

**1.2. MÔ HÌNH HÀNG ĐỢI CƠ BẢN**

**Hệ thống hàng đợi được mô tả bởi các thành phần:**

|  |  |
| --- | --- |
| **Thành phần** | **Ý nghĩa** |
| **Arrival Process** | Cách các gói tin đến hàng đợi |
| **Service Process** | Cách hệ thống xử lý (truyền) từng gói |
| **Queue Discipline** | Quy tắc chọn gói tiếp theo để phục vụ (ví dụ: FIFO, Priority, Round Robin) |
| **Number of Servers** | Số tiến trình phục vụ đồng thời |
| **System Capacity** | Số lượng gói tối đa hệ thống có thể chứa |
| **Population Source** | Tổng số nguồn có thể sinh gói (hữu hạn hoặc vô hạn) |

Bảng 1. Hệ thống hàng đợi

**1. Một số mô hình hàng đợi chuẩn:**

* **M/M/1**: Arrival Poisson, service time exponential, 1 server.

với .

* **M/M/1/K**: Có giới hạn hàng đợi K (mất gói khi hàng đầy).
* **M/D/1**: Service time cố định, giúp giảm độ dao động trễ.

Các mô hình này giúp mô phỏng và đánh giá **hiệu năng của bộ định tuyến hoặc link mạng**, như độ trễ trung bình, xác suất mất gói và thông lượng.

## 1.3. RED (RANDOM EARLY DETECTION)

**RED** là một cơ chế quản lý hàng đợi chủ động (**AQM – Active Queue Management**) nhằm tránh hiện tượng **tắc nghẽn toàn cục (global synchronization)** trong mạng TCP.

### 1. Nguyên lý hoạt động:

* RED không chờ hàng đợi đầy mới loại bỏ gói, mà **bắt đầu loại ngẫu nhiên sớm hơn**, khi độ dài trung bình của hàng vượt ngưỡng.
* Tính **độ dài trung bình (avg\_len)** bằng bộ lọc trung bình trượt theo hàm mũ:
* Khi vượt qua **min\_th** (ngưỡng dưới), gói sẽ bị **drop** hoặc **mark (ECN)** với xác suất tăng dần cho đến **max\_th** (ngưỡng trên).

### 2. Ưu điểm:

* Giảm hiện tượng đồng bộ hóa mất gói của TCP.
* Duy trì độ trễ ổn định.
* Tăng thông lượng và hiệu quả đường truyền.

**3. Nhược điểm:**

* Khó cấu hình các tham số (min\_th, max\_th, w\_q).
* Không tối ưu trong các mạng có tải thay đổi nhanh.

## 1.4. CUBIC (TCP CONGESTION CONTROL)

**Cubic** là thuật toán điều khiển tắc nghẽn TCP mặc định trong hầu hết các nhân Linux hiện nay (thay thế cho Reno và BIC).

### 1. Ý tưởng chính:

* Thay vì tăng tuyến tính như **Reno**, Cubic tăng theo **hàm bậc ba (cubic function)** của thời gian kể từ lần mất gói gần nhất.
  + : cửa sổ nghẽn tại thời điểm t
  + : cửa sổ tại thời điểm mất gói gần nhất
  + : hằng số tăng
  + : khoảng thời gian để cwnd đạt lại

### 2. Ưu điểm:

* Phục hồi nhanh sau mất gói.
* Tận dụng tốt băng thông trong mạng tốc độ cao – độ trễ lớn (high-BDP).
* Giữ ổn định khi tải cao.

## 1.5. TCP BBR (Bottleneck Bandwidth and Round-trip Propagation Time)

### 1. Giới thiệu chung

BBR (Bottleneck Bandwidth and RTT) là một thuật toán TCP congestion control do Google phát triển và công bố năm 2016, hiện được tích hợp trong Linux kernel từ phiên bản 4.9 trở lên.  
Không giống các thuật toán truyền thống như Reno hay Cubic, vốn dựa vào tín hiệu mất gói (packet loss) để xác định tắc nghẽn, BBR dựa trên mô hình băng thông – độ trễ thực tế của đường truyền để duy trì tốc độ truyền tối ưu.

Mục tiêu của BBR là tối đa hóa thông lượng mà vẫn giữ độ trễ thấp, giúp tránh hiện tượng *bufferbloat* (độ trễ tăng cao do hàng đợi bị đầy).

### 2. Nguyên lý hoạt động

BBR xây dựng một **mô hình ảo của đường truyền** dựa trên hai đại lượng cơ bản:

|  |  |
| --- | --- |
| **Tham số** | **Ý nghĩa** |
| **BtlBw (Bottleneck Bandwidth)** | Băng thông tối đa mà liên kết có thể truyền (tính bằng bytes/s). |
| **RTprop (Round-trip propagation time)** | Thời gian truyền tối thiểu của một gói đi và về mà không bị hàng đợi ảnh hưởng. |

Bảng 2. hai đại lượng cơ bản

=> Hai giá trị này được ước lượng liên tục dựa trên kết quả đo thực tế từ các gói ACK nhận được.

- Công suất gửi tối ưu được BBR tính là: **Pacing Rate=BtlBw**

- Kích thước cửa sổ nghẽn (congestion window): **cwnd=BtlBw×Rtprop**

=> Điều này giúp BBR gửi lượng dữ liệu vừa đủ để “lấp đầy đường ống” (*pipe*), không gây tắc nghẽn hàng đợi như Cubic.

### 3. Các giai đoạn (phases) của BBR

BBR hoạt động theo chu kỳ, gồm 4 pha chính:

|  |  |  |
| --- | --- | --- |
| **Pha** | **Mục tiêu** | **Mô tả** |
| **Startup** | Tìm băng thông cực đại | Tăng tốc độ gửi theo cấp số nhân cho đến khi không còn thấy băng thông tăng. |
| **Drain** | Xả hàng đợi | Giảm tốc độ gửi để làm rỗng hàng đợi tạm thời. |
| **ProbeBW** | Duy trì và kiểm tra băng thông | Tăng/giảm nhẹ tốc độ gửi để xác minh liệu có thêm băng thông khả dụng không. |
| **ProbeRTT** | Đo lại RTT thực | Giảm tốc độ gửi trong thời gian ngắn để đo lại RTprop chính xác. |

Bảng 3. 4 pha chính BBR hoạt động

- Chu kỳ ProbeBW – ProbeRTT lặp liên tục để duy trì mô hình cập nhật của mạng.

### 4. So sánh BBR với Cubic

|  |  |  |
| --- | --- | --- |
| **Đặc tính** | **Cubic** | **BBR** |
| **Nguyên lý** | Dựa trên packet loss | Dựa trên băng thông và RTT |
| **Điều chỉnh cwnd** | Hàm bậc ba theo thời gian | Theo công thức cwnd = BtlBw × RTprop |
| **Độ trễ trung bình** | Có thể cao (bufferbloat) | Thấp và ổn định |
| **Tốc độ phục hồi sau mất gói** | Giảm cwnd mạnh | Ít bị ảnh hưởng bởi mất gói |
| **Hiệu suất ở mạng tốc độ cao** | Giảm hiệu quả do phụ thuộc vào RTT | Hiệu quả cao trong mạng có RTT lớn |
| **Phù hợp** | Liên kết truyền thống | Liên kết tốc độ cao, mạng 5G, WAN, Data Center |

Bảng 4. So sánh BBR với Cubic

=> Nhờ đó, **BBR thường đạt thông lượng cao hơn và độ trễ thấp hơn** trong hầu hết các môi trường mạng hiện đại.

### 5. Ưu điểm và hạn chế

**Ưu điểm:**

* Duy trì **throughput cao** mà **độ trễ thấp**.
* **Không phụ thuộc vào packet loss**, tránh giảm tốc độ không cần thiết.
* **Ổn định** trong môi trường có RTT và băng thông thay đổi.
* Giảm đáng kể hiện tượng **bufferbloat**.

**Hạn chế:**

* **Không công bằng với Cubic** trong một số trường hợp: BBR có thể chiếm nhiều băng thông hơn.
* Phiên bản đầu tiên (BBR v1) có thể gây *burst* trong liên kết nhỏ; Google đã khắc phục trong **BBR v2** (ra mắt 2021).
* Cần **ACK chính xác và ổn định**, nếu bị trễ hoặc mất ACK, hiệu quả giảm.

### 6. Ứng dụng và triển khai

Được **Google triển khai trên YouTube, Google Cloud, và gRPC**, giúp giảm độ trễ video streaming hàng chục phần trăm.

Có thể kích hoạt trên Linux:

|  |
| --- |
| sudo sysctl -w net.ipv4.tcp\_congestion\_control=bbr |

Bảng 5. Lệnh kích hoạt BBR trên Linux

Kiểm tra trạng thái:

|  |
| --- |
| sysctl net.ipv4.tcp\_congestion\_control |

Bảng 6. Lệnh kiểm tra trạng thái trên Linux

## 1.6. Các hàng đợi (qdisc) phổ biến trong Linux

**Qdisc (Queueing Discipline)** là cơ chế trong Linux kernel để quản lý cách gói tin được xếp hàng và gửi đi. Một số qdisc thường dùng:

**1. pfifo\_fast / pfifo**

* **pfifo**: đơn giản nhất, xếp hàng theo **FIFO (First In First Out)**, bỏ gói khi đầy.
* Thích hợp cho thử nghiệm cơ bản hoặc môi trường không cần kiểm soát trễ.

**2. RED (Random Early Detection)**

* Như mô tả ở trên, là qdisc AQM chủ động để giảm tắc nghẽn.
* Cấu hình trong Linux:

|  |
| --- |
| tc qdisc add dev eth0 root red limit 1000 min 200 max 600 avpkt 1000 burst 20 probability 0.02 |

Bảng 7. Ví dụ về cấu hình trong Linux của RED

**3. fq\_codel (Fair Queue Controlled Delay)**

* Là qdisc hiện đại, **kết hợp Codel (Controlled Delay)** và **Fair Queuing**.
* Tự động phát hiện và loại bỏ gói trễ quá lâu, giảm bufferbloat.
* Không cần tinh chỉnh tham số, hoạt động hiệu quả trong mạng thực tế.

|  |  |  |
| --- | --- | --- |
| **Qdisc** | **Đặc điểm chính** | **Loại** |
| **pfifo** | Đơn giản, FIFO | Passive |
| **RED** | Loại ngẫu nhiên sớm, chống tắc nghẽn | AQM |
| **fq\_codel** | Fair Queue + kiểm soát độ trễ tự động | AQM nâng cao |

Bảng 8. So sánh qdisc

## 1.7. iperf - Công Cụ Đo Băng Thông và Hiệu Năng

**Mục đích:** iperf3 là công cụ dòng lệnh được sử dụng rộng rãi để đo lường thông lượng tối đa (maximum throughput) trên mạng IP. Nó có thể kiểm tra hiệu năng của cả TCP và UDP.

**Nguyên lý hoạt động:**

* Thiết lập mô hình Client-Server. Máy chủ (server) lắng nghe kết nối, máy khách (client) khởi tạo và gửi dữ liệu.
* iperf3 tạo ra các luồng dữ liệu (streams) và gửi đi trong một khoảng thời gian xác định, sau đó báo cáo lại lượng dữ liệu đã truyền tải, từ đó tính toán ra băng thông đạt được.

**Các thông số quan trọng có thể đo được:**

* Bandwidth (Băng thông): Tốc độ truyền tải dữ liệu (Mbits/sec, MBytes/sec).
* Jitter (Độ biến động trễ): Sự thay đổi về độ trễ giữa các gói tin (chỉ cho UDP).
* Packet Loss (Tỷ lệ mất gói): Phần trăm gói tin bị mất trên đường truyền (chỉ cho UDP).

**Ứng dụng trong đồ án**: Dùng để kiểm chứng băng thông thực tế của liên kết mạng, so sánh hiệu năng giữa các cấu hình khác nhau (ví dụ: thay đổi kích thước cửa sổ TCP - window size), hoặc đánh giá hiệu quả của một giải pháp mạng mới.

## 1.8. tc (Traffic Control) và netem - Công Cụ Điều Khiển và Mô Phỏng Mạng

**Mục đích**: tc là công cụ mạnh mẽ trong Linux để điều khiển lưu lượng mạng. Kết hợp với module netem (Network Emulator), nó cho phép mô phỏng các điều kiện mạng không lý tưởng ngay trên một máy chủ.

**Nguyên lý hoạt động**:

* tc hoạt động bằng cách áp dụng các "qdisc" (queueing disciplines - kỷ luật xếp hàng) lên các giao diện mạng. Các qdisc này sẽ kiểm soát cách các gói tin được gửi đi, nhận về hoặc xử lý.
* netem là một loại qdisc đặc biệt, cung cấp khả năng thêm độ trễ, tỷ lệ mất gói, jitter, và nhiều hiện tượng mạng phức tạp khác.

**Các tình huống mạng có thể mô phỏng**:

* Độ trễ (Delay): netem delay 100ms 10ms
* Mất gói (Packet Loss): netem loss 5%
* Jitter: netem delay 100ms 20ms (độ trễ trung bình 100ms, dao động ±20ms).
* Gói tin trùng lặp (Duplication), Đảo thứ tự (Reordering), v.v.

**Ứng dụng trong đồ án**: Tạo ra một môi trường thử nghiệm có điều kiện mạng giống thực tế (ví dụ: mạng di động, đường truyền vệ tinh, hoặc mạng WAN có độ trễ cao) để đánh giá xem ứng dụng của bạn hoạt động như thế nào trong những điều kiện đó.

## 1.9. tcpdump - Công Cụ Phân Tích Gói Tin Mạng

**Mục đích**: tcpdump là công cụ phân tích giao thức mạng dựa trên dòng lệnh. Nó cho phép "bắt" (capture) và hiển thị các gói tin đang được truyền đi hoặc nhận về trên một giao diện mạng.

**Nguyên lý hoạt động**:

* tcpdump sử dụng thư viện libpcap để lấy các gói tin từ lớp driver của card mạng.
* Nó có thể lọc và hiển thị nội dung của các gói tin dựa trên nhiều tiêu chí (địa chỉ IP, cổng, giao thức, cờ TCP, v.v.).

**Thông tin chi tiết có thể thu thập**:

* Toàn bộ header của gói tin (Ethernet, IP, TCP/UDP, v.v.).
* Dữ liệu payload (nếu không mã hóa).
* Thời gian chính xác khi gói tin được gửi/ nhận.
* Các sự kiện trong quá trình bắt tay TCP (SYN, SYN-ACK, ACK) và kết thúc kết nối (FIN, RST).

**Ứng dụng trong đồ án**: Phân tích sâu các vấn đề về hiệu năng (ví dụ: xác định nguyên nhân độ trễ bằng cách phân tích các khoảng thời gian giữa các gói tin, phân tích hiện tượng tắc nghẽn thông qua cờ TCP Window, hoặc gỡ lỗi các kết nối không thành công).

## 1.10. ss - Công Cụ Thống Kê Socket

**Mục đích**: ss (socket statistics) là công cụ hiện đại thay thế cho netstat. Nó dùng để điều tra thông tin về socket, kết nối mạng, và các bảng routing.

**Nguyên lý hoạt động**:

* ss lấy thông tin trực tiếp từ nhân Linux thông qua giao diện netlink, làm cho nó nhanh hơn và hiệu quả hơn netstat trên các hệ thống có nhiều kết nối.

**Các thông tin quan trọng**:

* Danh sách tất cả các kết nối TCP, UDP, và UNIX socket.
* Trạng thái của kết nối (ESTABLISHED, TIME-WAIT, LISTEN, v.v.).
* Thông tin về bộ nhớ sử dụng (send/receive buffer), số gói tin đang chờ,...
* Thông tin chi tiết về cài đặt TCP (như rtt, cwnd - congestion window).

**Ứng dụng trong đồ án**: Giám sát trạng thái kết nối thời gian thực, phân tích vấn đề về kết nối (ví dụ: quá nhiều kết nối ở trạng thái TIME-WAIT), và thu thập các chỉ số hiệu năng cấp độ socket như kích thước cửa sổ tắc nghẽn.

## 1.11. ifstat - Công Cụ Thống Kê Lưu Lượng Giao Diện Mạng

**Mục đích**: ifstat là công cụ đơn giản để giám sát lưu lượng mạng trên các giao diện (interface) theo thời gian thực.

**Nguyên lý hoạt động:**

* ifstat định kỳ đọc các bộ đếm trong file /proc/net/dev (chứa thống kê về từng giao diện mạng) và tính toán tốc độ truyền tải dựa trên sự chênh lệch giữa các lần đọc.

**Các thông tin hiển thị**:

* Băng thông inbound (KB/s, MB/s) và outbound (KB/s, MB/s) của từng giao diện.
* Tổng số byte/packet đã truyền đi và nhận về.

**Ứng dụng trong đồ án**: Cung cấp một cái nhìn tổng quan, trực quan về việc sử dụng băng thông trên các giao diện mạng. Hữu ích để xác nhận xem liệu lưu lượng thử nghiệm từ iperf3 có thực sự được tạo ra hay không, hoặc để giám sát mức độ tải tổng thể của hệ thống.

# II. MÔ HÌNH TRIỂN KHAI

## 2.1. Mô hình tổng quan

- 3 máy ảo Ubuntu 22.04 LTS: Client, Bottleneck, Server.

## 2.2. Sơ đồ hệ thống

A screenshot of a computer

AI-generated content may be incorrect.

Hình 1. Sơ đồ hệ thống

## 2.3. Kịch bản thử nghiệm

- Trường hợp bandwidth bình thường và bandwidth bị giới hạn xuống 3 Mbps, 1 flow, CUBIC cho fq\_codel/pfifo/RED.

- Trường hợp bandwidth bình thường và bandwidth bị giới hạn xuống 3 Mbps, 5/10 flows với impairment (delay, jitter, loss) cho fq\_codel/pfifo/RED cùng với CUBIC/RED.