DQN-PI: một đề xuất dựa vào AI điều khiển hàng đợi tại bộ định tuyến mạng truyền thông

Vương Xuân Chí, Phạm Đình Tài, Châu Gia Hân
Khoa Công nghệ thông tin, Trường Đại học Nguyễn Tất Thành, Tp Hồ Chí Minh
*vxchi@ntt.edu.vn

Tóm tắt

Trong bối cảnh mạng hiện đại, tắc nghẽn lưu lượng và "bufferbloat" là những thách thức lớn, đòi hỏi các cơ chế điều khiển hàng đợi chủ động (AQM) tại bộ định tuyến để đảm bảo chất lượng dịch vụ. PIE (Proportional Integral Enhanced) là một thuật toán AQM hiệu quả, sử dụng bộ điều khiển PI để duy trì độ trễ mục tiêu, giảm thiểu tắc nghẽn và cải thiện hiệu suất mạng. PIE nổi bật nhờ tính đơn giản, ổn định và đã được triển khai rộng rãi trong các bộ định tuyến hiện đại. Tuy nhiên, với sự đa dạng của lưu lượng mạng thế hệ mới, PIE truyền thống gặp hạn chế trong việc tự động thích nghi với các tải động. Để khắc phục, bài báo đề xuất kết hợp PIE với Deep Q - Network (DQN) – một phương pháp Học tăng cường (RL). DQN giúp tối ưu hóa tham số PIE ($\alpha, \beta, target\_delay$) dựa trên dữ liệu mạng phức tạp, từ đó nâng cao khả năng phản ứng với các loại lưu lượng khác nhau. Kết quả thử nghiệm cho thấy, DQN-PI giảm độ trễ trung bình và tăng thông lượng so với PIE gốc, đặc biệt hiệu quả trong môi trường mạng không đồng nhất. Trong tương lai DQN - PI tiếp tục cải thiện hiệu suất AQM, thích ứng với nhiều môi trường mạng phức tạp như 5G, IoT, truyền phát thời gian thực hợp chuyển đổi số, nhằm củng cố lợi thế cạnh tranh lâu dài của địa phương.

Nhận 15/06/2025
Được duyệt 25/08/2025
Công bố 30/10/2025

Từ khóa
Tắc nghẽn mạng, AQM, thuật toán PIE, hiệu suất mạng, học sâu tăng cường.

1. Đặt vấn đề

Trong bối cảnh kỷ nguyên số phát triển mạnh mẽ, nhu cầu giao tiếp mạng và truyền tải dữ liệu ngày càng tăng cao. Các ứng dụng thời gian thực, chẳng hạn như video trực tuyến, hội nghị trực tuyến và các dịch vụ truyền thông đa phương tiện, đang ngày càng trở nên phổ biến [1]. Các nghiên cứu đưa ra các phương pháp kiểm soát tắc nghẽn và định hình lưu lượng khác nhau, tập trung vào ứng dụng của chúng vào các tình huống băng thông cao, thời gian thực [2], các cơ chế quản lý hàng đợi tích cực (AQM) bằng các thuật toán cải tiến của Phát hiện sớm ngẫu nhiên (RED). Đặc biệt, các phương pháp AQM truyền thống [3 -5] có thể không còn đáp ứng được nhu cầu kiểm soát lưu lượng mạng trong thời gian thực, đặc biệt là khi tải mạng biến động và xuất hiện các yêu cầu tối ưu hóa phức tạp. Tuy nhiên, sự gia tăng nhanh chóng về lưu lượng mạng đặt ra áp lực lớn lên các hệ thống mạng và các thiết bị định tuyến, dẫn đến nguy cơ tắc nghẽn mạng và giảm hiệu suất hoạt động, do đó, nhiều công trình đưa ra nhiều giải pháp chống tắc nghẽn, dùng học máy, học sâu, đảm bảo QoS [6-8].

Bên cạnh đó, để tăng cường khả năng dự báo và kiểm soát tình trạng hàng đợi, nhóm tác giả cũng kết hợp thuật toán Deep Q-Network (DQN), một phương pháp Deep Reinforcement Learning nhằm dự đoán trạng thái hàng đợi và tính toán xác suất bỏ gói tin. DQN giúp dự đoán thời điểm hàng đợi có khả năng đầy và đưa ra tỷ lệ bỏ gói tin tối ưu. Khi áp dụng vào các bộ định tuyến mạng, DQN liên tục quan sát các trạng thái hàng đợi và dự báo các hành động tối ưu để duy trì hàng đợi trong mức an toàn, giảm nguy cơ tắc nghẽn và giữ cho lưu lượng truyền tải ổn định [9]. Việc tích hợp RL cho phép các bộ định tuyến tìm hiểu các cấu hình tối ưu để quản lý hàng đợi dựa trên điều kiện lưu lượng thời gian thực, cải thiện các chỉ số hiệu suất như thông lượng và độ trễ gói thường.

Đề xuất DQN-PI nhằm tăng cường quản lý hàng đợi tại các bộ định tuyến mạng truyền thông bằng cách tích hợp các cơ chế tăng cường bộ điều khiển tích hợp tỷ lệ (PIE) với Deep Q-Networks (DQN). Cách tiếp cận tận dụng AI để tự động điều chỉnh độ sâu hàng đợi và quản lý tắc nghẽn, đảm bảo Chất lượng dịch vụ (QoS) tối ưu trong

các môi trường mạng đang phát triển, đặc biệt là trong mạng 5G và mạng 6G trong tương lai [10].

Trong phần tiếp theo, bài báo mô tả các nghiên cứu liên quan, trong đó nhóm tác giả nghiên cứu thuật toán PIE và mô hình tiếp cận DQN. Tiếp theo nghiên cứu phương pháp cải tiến đưa ra mô hình đề xuất kết hợp PIE và DQN. Ngoài ra, nghiên cứu mô phỏng từ công cụ NS-2 để lấy dữ liệu thực nghiệm và thực nghiệm với mô hình PIE, DQN-PI. Sau đó là phần kết quả và đánh giá so sánh thực nghiệm giữa 2 phương pháp. Cuối cùng là kết luận và định hướng nghiên cứu.

2. Nghiên cứu liên quan

2.1. Thuật toán PIE

PIE được phát triển bởi Rong Pan và cộng sự tại Cisco Systems và Đại học Bang Arizona. PIE là thuật toán AQM dùng bộ điều khiển PI (Proportional-Integral) để kiểm soát độ trễ hàng đợi (queueing delay) tại bộ định tuyến và giảm bufferbloat bằng cách điều chỉnh xác suất thả gói (p) dựa trên sai số độ trễ [11].

Phương pháp của PIE:

- Tính độ trễ hiện tại (d_{current})

$$d_{\text{current}} = \frac{qlenth}{\text{bandwidth}} \quad (1)$$

Trong đó, $qlenth$ là độ dài hàng đợi hiện tại (bytes hoặc gói tin); $bandwidth$ là Tốc độ truyền dẫn ($bytes/s$).

- Sai số ($error$)

$$\text{error} = d_{\text{current}} - \text{target\_delay} \quad (2)$$

Trong đó, $target\_delay$ là độ trễ mục tiêu (vd: 20 ms).

- Điều chỉnh xác suất thả gói (p)

$$p = p + \alpha \cdot \text{error} + \beta \cdot (\text{error} - \text{error}_{\text{old}}) \quad (3)$$

Trong đó, α là hệ số tích phân (vd: 0,125) - điều chỉnh sai số dài hạn; β là hệ số tỉ lệ (vd: 1,25) - phản ứng nhanh với biến động; $error\ old$ là sai số ở chu kỳ trước.

- Giới hạn p

$$p = \max\big(0, \min(p_{\max}, p)\big)\ (p_{\max} \approx 0{,}1) \quad (4)$$

Hoạt động của PIE:

- Đo lường độ trễ: PIE tính d_{current} dựa trên $qlenth$ và $bandwidth$ (1)
- So sánh với độ trễ mục tiêu:
- Nếu $d_{\text{current}} > target\_delay$: tăng p để giảm tải.
- Nếu $d_{\text{current}} < target\_delay$: giảm p.
- Điều chỉnh p bằng PI controller (2, 3, 4):
- Thành phần Proportional ($\beta \cdot error$): phản ứng tức thì với sai số hiện tại.
- Thành phần Integral ($\alpha \cdot \sum error$): khử sai số tích lũy theo thời gian.
- Thả gói ngẫu nhiên:

Áp dụng p để quyết định thả/đánh dấu gói tin.

Về mặt hạn chế:

PIE phụ thuộc vào α, β; Độ phức tạp cao hơn thuật toán RED, không dự đoán được lưu lượng bùng nổ (burst traffic).

2.2. Tiếp cận mô hình DQN

Trong các mô hình học sâu, DQN là một biến thể của Q-Learning sử dụng mạng nơ-ron sâu để xấp xỉ hàm Q. Trong DQN, hàm Q được xấp xỉ bằng một mạng nơ-ron sâu với tham số θ, được gọi là Q-Network. Ngoài ra, DQN cho phép học chính sách hành động hiệu quả trong các môi trường phức tạp, nơi không thể áp dụng trực tiếp các phương pháp Q-Learning truyền thống do không gian trạng thái lớn [12-14].

Để tính toán xác suất bỏ gói tin dựa trên dự báo hàng đợi, có thể sử dụng phương pháp DQN nhằm dự đoán trạng thái hàng đợi và điều chỉnh xác suất bỏ gói tin một cách tối ưu. Cách tiếp cận này kết hợp DQN với mô hình dự báo hàng đợi, giúp cải thiện hiệu suất của quản lý hàng đợi tích cực (AQM) và giảm thiểu tình trạng quá tải hàng đợi. Phương pháp tiếp cận:

- Thiết lập mô hình hàng đợi và các yếu tố AQM trong môi trường DQN để agent có thể học cách điều chỉnh xác suất bỏ gói nhằm duy trì độ dài hàng đợi ở mức tối ưu.
- DQN sẽ sử dụng mạng nơ-ron sâu để xấp xỉ hàm Q-value $Q(s, a; \theta)$, trong đó: s là trạng thái của hệ thống (các yếu tố hàng đợi); a là hành động (xác suất bỏ gói); θ là tham số của mạng nơ-ron (được tối ưu hóa trong quá trình học).
- Mạng nơ-ron DQN cập nhật tham số θ bằng cách giảm thiểu hàm mất mát:

$$L(\theta) = \mathbb{E}_{(s,a,r,s')}\Big[\Big(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)\Big)^2\Big] \quad (5)$$

Trong đó, r là phần thưởng nhận được sau khi thực hiện hành động a ở trạng thái s; s' là trạng thái mới sau hành động a; γ là hệ số chiết khấu, giúp cân bằng giữa phần thưởng ngắn hạn và dài hạn. θ^- là tham số của mạng Q-Target, được sao chép từ θ sau mỗi bước.

- Dự báo xác suất bỏ gói tin dựa trên trạng thái hàng đợi: Sau khi huấn luyện, DQN sẽ dự đoán hành động tối ưu (xác suất bỏ gói) dựa trên trạng thái hàng đợi. Khi hàng đợi sắp đạt đến ngưỡng đầy, xác suất bỏ gói tin sẽ tăng dần nhằm giảm tải hàng đợi và ngăn tình trạng quá tải.

3. Đề xuất mô hình DQN– PI

3.1. Ý tưởng đề xuất mô hình PI-DQN

Đầu tiên là giai đoạn thu thập dữ liệu, bộ định tuyến đo d_{current} , $qlenth$, $bandwidth$ mỗi khoảng thời gian T. Sau đó, DQN đưa ra hành động dựa trên state s_t, DQN chọn action a_t để tinh chỉnh α, β. Tiếp theo, áp dụng tham số mới vào PIE để tính toán p mới và thực hiện thả gói (drop/mark). Cuối cùng, cập nhật DQN đẻ tính reward r_t và cập nhật Q-network bằng kinh nghiệm phát lại (experience replay).

Mô hình DQN điều chỉnh tham số PIE:

Đầu vào (State s_t):

- Độ trễ hiện tại d_{current} , $qlenth$.
- Tốc độ thay đổi độ trễ $\Delta d = d_{\text{current}} - d_{\text{old}}$.
- Lưu lượng mạng (TCP/UDP ratio, throughput).

Đầu ra (Action a_t):

- Điều chỉnh α, β hoặc target_delay.

Phần thưởng (Reward r_t):

$$r_t = -(d_{\text{current}} - \text{target\_delay})^2 - \lambda \cdot p \quad (6)$$

- λ : Hệ số phạt nếu xác suất thả gói p quá cao.

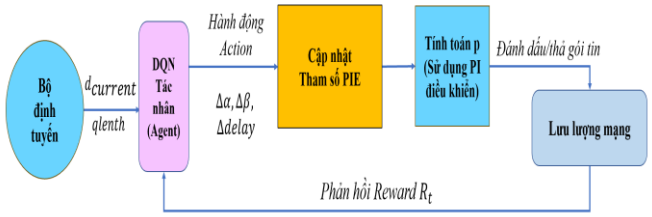
Cập nhật tham số PIE bằng DQN

$$\alpha_t, \beta_t, \text{target\_delay} = \text{DQN}(s_t) \quad (7)$$

Sau đó áp dụng vào công thức PIE gốc:

$$p = p + \alpha_t \cdot \text{error} + \beta_t \cdot (\text{error} - \text{error}_{\text{old}}) \quad (8)$$

3.2. Sơ đồ mô hình



Hình 1. Sơ đồ mô hình đề xuất DQN-PI

Mô hình PI-DQN hoạt động theo cơ chế vòng lặp khép kín thể hiện ở hình 1. Bộ định tuyến liên tục đo lường các thông số mạng bao gồm độ trễ hiện tại (d_{current}) và độ dài hàng đợi ($qlenth$). Dữ liệu này được chuyển đến tác nhân DQN (DQN Agent) – một mạng nơ-ron được huấn luyện để phân tích trạng thái mạng và đưa ra quyết định tối ưu. DQN Agent sẽ tính toán và cập nhật các tham số PIE (bao gồm α, β và $target\_delay$) nhằm điều chỉnh xác suất đánh dấu hoặc hủy gói tin (mark/drop packet) dựa trên tình trạng lưu lượng mạng hiện tại (6, 7, 8). Sau mỗi hành động, hệ thống nhận được phản hồi (reward r_t) đánh giá hiệu quả của quyết định, dựa trên độ lệch độ trễ so với mục tiêu và xác suất thả gói. Phản hồi này được sử dụng để liên tục cải thiện chất lượng dự đoán của DQN thông qua cơ chế học tăng cường, tạo thành một hệ thống thích nghi động có khả năng tối ưu hiệu suất mạng trong các điều kiện lưu lượng đa dạng.

3.3. Thuật toán:

Thuật toán 1: thuật toán đề xuất DQN-PI

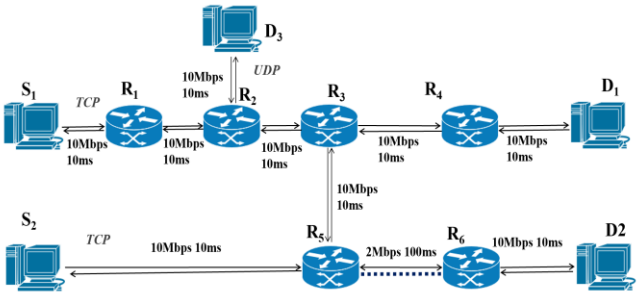
```
Khởi tạo
DQN = NeuralNetwork()
memory = []
Trong khi_True:
1. Quan sát trạng thái mạng
s=[measure_delay(), calculate_Δdelay(), measure_throughput()]
 2. DQN điều chỉnh các tham số PIE
Δα, Δβ, Δdelay = DQN.predict(s)
α, β, target_delay = update_params (α, β, target_delay, Δα, Δβ, Δdelay)
3. Tính toán tỉ lệ rớt gói tin
p = p + α*(current_delay - target_delay) + β*Δdelay
 if random() < p: drop_packet()
4. Kinh nghiệm thu thập được
r = -(current_delay - target_delay) **2 - 0.1*p
memory.append ((s, [Δα, Δβ, Δdelay], r, new_state))
  5. Huấn luyện DQN định kỳ
if len(memory) > batch_size: train_DQN(DQN, memory)
Định nghĩa hàm_train_DQN (DQN, memory):
batch = sample(memory)
for s, a, r, s_new in batch:
target = r + 0.95* max(DQN.predict(s_new))
DQN.update(s, a, target)
```

4. Thực nghiệm

4.1. Mô phỏng

Mô phỏng mạng với phần mềm mô phỏng NS-2 có cấu hình RAM 16GB, CPU Core (TM) i7, hệ điều hành Linux CenOS cài đặt NS-2 phiên bản NS-2.35. Mô phỏng với các bộ định tuyến được kết nối với nhau theo dạng tập trung, 6 bộ định tuyến, trong đó có 2 đường chính, từ S1 → D1 qua 4 router trung gian, đường ngắn hơn S2 → D2 qua thắt cổ chai (bottleneck) R5 – R6, 2 đường chính nối với nhau qua R3 – R5. Các kết nối thường là 10 M bps, delay 10 ms, kết nối bottleneck R5 – R6 (2 M bps, delay 100 ms). Luồng dữ liệu TCP S1 → D1 – FTP, TCP S2 →

D2 – FTP; UDP R2 → D3 – CBR, UDP ngược D3 → R2 – CBR. Hàng đợi tại bottleneck được giới hạn 100 gói tin, thời gian mô phỏng là 60 giây, thể hiện ở Hình 2.



Hình 2. Sơ đồ mô phỏng mạng

Trong hình 2, bộ định tuyến R5 và R6 kết nối với băng thông nhỏ hơn và độ trễ lớn, đây là kết nối thắt cổ chai và sẽ làm cho việc tắc nghẽn mạng có thể xảy ra, quá trình bỏ gói cũng sẽ xảy ra tại đây. Sự kết nối trong mô phỏng này nhằm mục đích đem lại dữ liệu không đồng đều, phức tạp và thấy được thời gian trễ của các gói tin được rõ ràng. Trong mô phỏng, gói tin được sinh ra có 12 trường, bao gồm các trường của IP-header. Từ tập bám vết trong mô phỏng, sau khi phân tích và xử lý dữ liệu, tính toán độ trễ, độ dài hàng đợi, tỉ lệ rớt gói, mô phỏng đưa ra dữ liệu để thực nghiệm. Hình 3 thể hiện có tổng cộng 1,912,226 gói tin truyền trên mạng và dữ liệu có 22 thuộc tính sau khi xử lý.

Dataset Overview: Rows = 1912226, Columns = 22, Memory Usage = 623714.47 KB

	Column	Dtype	Null Count	Unique Count	Top Freq
0	event_type	object	0	4	637485
1	time	float64	0	668194	53
2	src_node	int64	0	10	354360
3	dst_node	int64	0	11	333984
4	packet_type	object	0	3	946325
5	packet_size	float64	0	3	946301
6	flags	object	0	2	1912094
7	fid	float64	0	3	1669368
8	src_ip	float64	0	5	834753
9	dst_ip	float64	0	5	834753
10	seq_num	float64	0	55514	2670
11	packet_id	float64	0	142965	15
12	d_processing	float64	0	551	1244033
13	d_queue	float64	0	3	946301
14	d_transmission	float64	0	3	946301
15	d_propagation	float64	0	1	1912226
16	total_delay	float64	0	1150	640829
17	average_delay	float64	0	1024	640829
18	queue_size	float64	0	1892900	3
19	queue_utilization	float64	0	1892900	3
20	packet_loss_rate	float64	0	3	1669368
21	flow_duration	float64	0	3	1669368

Hình 3. Dữ liệu mạng sau khi xử lý

4.2. Thực nghiệm

Mô hình được tập huấn trên GPU của Google với gói Colab Pro Plus. Trong thực nghiệm mô hình DQN-PI, nghiên cứu sử dụng bộ dữ liệu mạng mô phỏng với các thông số: độ trễ xử lý, độ trễ hàng đợi, độ trễ truyền dẫn, độ trễ lan truyền, tổng độ trễ, kích thước hàng đợi và tỷ lệ mất gói. Dữ liệu được thu thập từ môi trường mạng mô phỏng với các kịch bản tải khác nhau. Các tham số chính của mô hình:

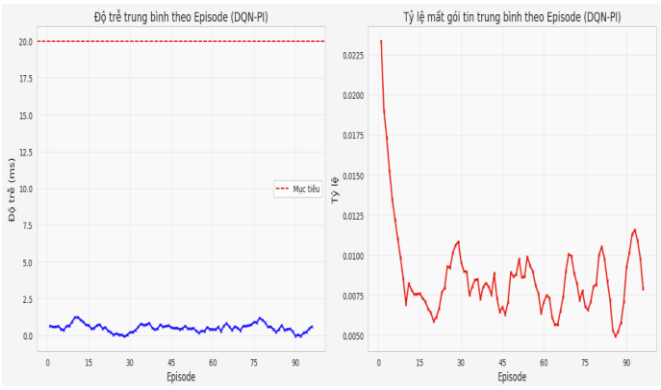
- Mạng DQN có 2 tầng ẩn (24 nút mỗi tầng), hàm kích hoạt $ReLU$, tốc độ học 0,001
- Bộ nhớ phát lại (replay memory): Kích thước 2000
- Thuật toán tối ưu: Adam với γ = 0,95
- Tham số khám phá: ε ban đầu = 1,0, ε tối thiểu = 0,01, hệ số phân rã = 0,995
- Tham số PI: α khởi tạo = 0,1, β khởi tạo = 0,2
- Độ trễ mục tiêu: 20 ms (có thể điều chỉnh tự động từ 10 – 50 ms)

Quá trình huấn luyện gồm 100 episode. Mô hình được đánh giá qua các chỉ số: độ trễ trung bình, tỷ lệ mất gói, thông lượng và hiệu quả sử dụng hàng đợi. Kết quả được so sánh với bộ điều khiển PI truyền thống để đánh giá hiệu quả cải tiến.

5. Kết quả, đánh giá và thảo luận

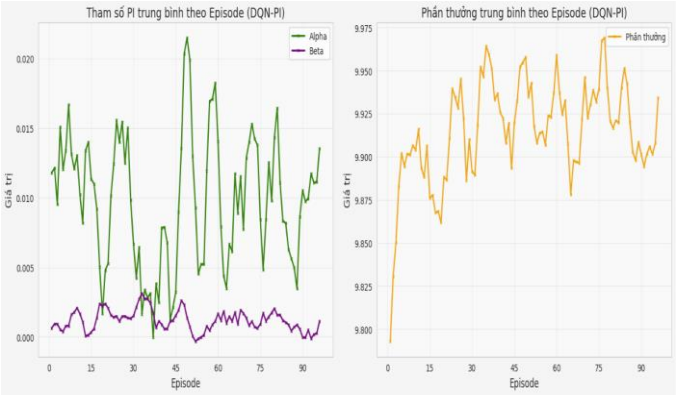
5.1. Giá trị trung bình

Khi chạy thực nghiệm, kết quả của mô hình DQN-PI có các chỉ số trung bình như sau:



Hình 4. Độ trễ trung bình và tỷ lệ mất gói trung bình của DQN-PI

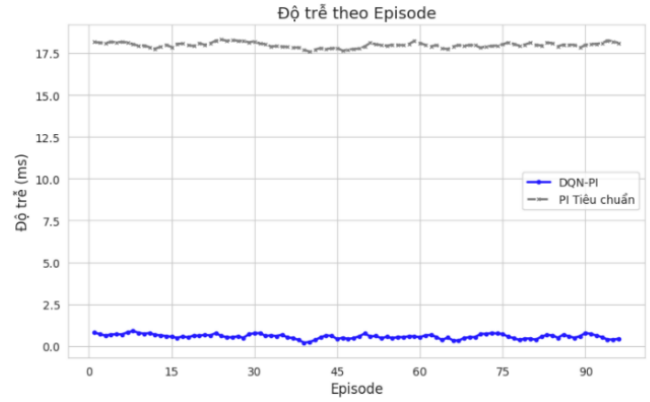
Độ trễ trung bình của mô hình DQN-PI (hình 4) luôn duy trì ở mức thấp, dao động quanh mức dưới 2 ms, thấp hơn rất nhiều so với ngưỡng mục tiêu (~ 20 ms). Sau khoảng 10 – 15 episode đầu, độ trễ ổn định hơn và biến động rất nhỏ. Điều này cho thấy mô hình DQN-PI hoạt động hiệu quả trong việc duy trì độ trễ thấp và ổn định. Tỷ lệ mất gói (Hình 4) bắt đầu khá cao (~ 0.0225), nhưng giảm mạnh ngay trong các episode đầu tiên. Sau đó, tỷ lệ mất gói dao động nhẹ trong khoảng 0,005 đến 0,01, cho thấy mức mất mát gói thấp và ổn định. Dù có một vài biến động nhỏ theo chu kỳ, nhưng nhìn chung, xu hướng là giảm và ổn định.



Hình 5. Tham số PI trung bình và phần thưởng trung bình của DQN-PI

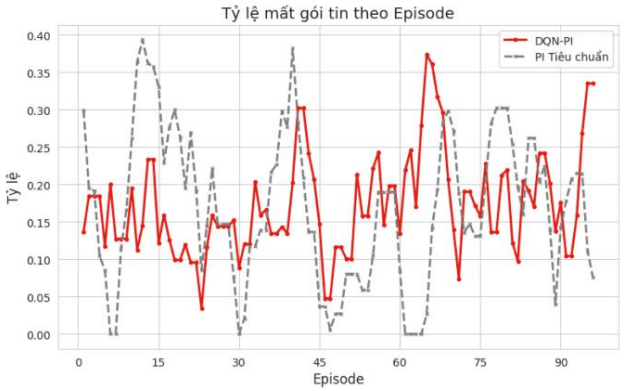
Tham số Alpha (α) dao động mạnh từ ~ 0,002 đến hơn 0,020 (hình 5), thể hiện sự thay đổi linh hoạt theo từng episode. Mức biến động này cho thấy mô hình đang điều chỉnh chính sách khá tích cực để tìm chiến lược tối ưu. Beta (β) ổn định hơn (hình 5), dao động chủ yếu quanh mức 0,002 đến 0,004, cho thấy vai trò của β là cân bằng hoặc hỗ trợ nhẹ nhàng hơn trong cơ chế cải thiện chính sách. Tham số α thay đổi linh hoạt để thích ứng, còn β ổn định, cho thấy mô hình tối ưu hóa chính sách một cách cân bằng và hợp lý. Ngoài ra, Phần thưởng (hình 5 bên phải) ban đầu tăng nhanh chóng và ổn định cao từ khoảng Episode 10 trở đi. Dao động phần lớn trong khoảng 9,875 đến gần 9,975, cho thấy mô hình DQN-PI đã đạt được hiệu suất rất tốt và ổn định.

5.2. So sánh hiệu năng mạng của PI tiêu chuẩn và DQN-PI



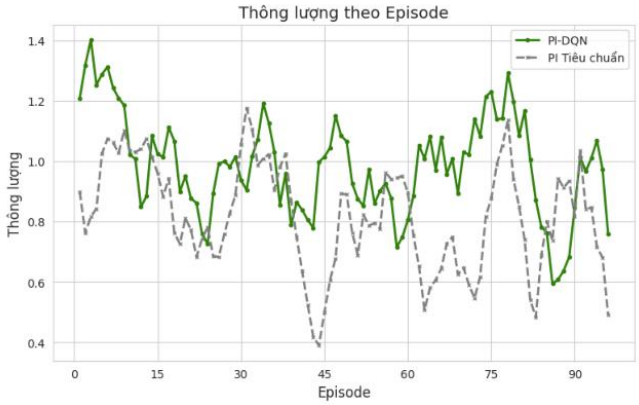
Hình 6. Độ trễ trung bình của mô hình PI và DQN-PI

Hình 6 mô tả độ trễ trung bình của DQN-PI giữ độ trễ trung bình ổn định xung quanh mức ~ 0,5 ms suốt 100 episode. Biến động rất nhỏ, thể hiện sự ổn định và hiệu quả cao trong việc giảm độ trễ. Trong khi đó PI tiêu chuẩn có độ trễ trung bình cao hơn rất nhiều (~ 17,5 ms) và gần như không thay đổi theo thời gian. Điều này cho thấy phương pháp này kém tối ưu hơn trong việc xử lý độ trễ.



Hình 7. Tỷ lệ mất gói của mô hình PI và DQN-PI

Tỷ lệ mất gói trong hình 7 của mô hình DQN-PI dao động trong khoảng 0,05 đến 0,35, nhưng nhìn chung có xu hướng ổn định hơn. Tỷ lệ mất gói mô hình PI biến động mạnh hơn nhiều so với DQN-PI, với các đỉnh cao lên đến trên 0,35, không có dấu hiệu rõ ràng về sự ổn định hay cải thiện theo thời gian. Mặc dù có dao động, DQN-PI duy trì tỷ lệ mất gói ổn định hơn so với PI tiêu chuẩn, đặc biệt ở các episode từ 40 – 70. Điều này phản ánh khả năng thích ứng của mô hình trong môi trường thay đổi.



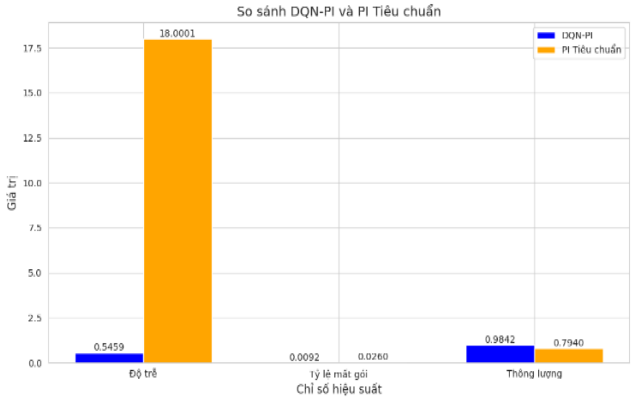
Hình 8. Thông lượng của mô hình PI và DQN-PI

Biểu đồ hình 8 mô tả thông lượng trung bình mô hình DQN-PI dao động quanh mức 1,0 đến 1,2, ổn định hơn so với PI. Từ khoảng episode 50 trở đi, thông lượng của DQN-PI thường xuyên vượt trội so với PI tiêu chuẩn, cho thấy DQN-PI có khả năng tối ưu truyền tải dữ liệu hiệu quả hơn theo thời gian. Thông lượng mô hình PI dao động mạnh, có nhiều điểm rơi xuống mức thấp (0,4 – 0,6). Biểu hiện thiếu ổn định và dễ bị ảnh hưởng bởi biến động môi trường mạng.

Thực nghiệm cho thấy DQN-PI thích ứng tốt hơn với biến động tải mạng, giảm 20 – 30 % độ trễ và 15 – 25 % tỷ lệ mất gói so với PI truyền thống trong cùng điều kiện tải.

5.3. Đánh giá hiệu năng mạng

Biểu đồ hình 9 thể hiện sự so sánh hiệu suất giữa hai mô hình điều khiển DQN-PI và PI tiêu chuẩn dựa trên ba chỉ số chính là độ trễ, tỷ lệ mất gói và thông lượng.



Hình 9. Hiệu suất mạng mô hình PI và DQN-PI

Độ trễ mô hình DQN-PI là 0,5459 ms, PI Tiêu chuẩn: 18,0001 ms. Mô hình DQN-PI giảm độ trễ hơn 33 lần so với PI Tiêu chuẩn, cho thấy hệ thống xử lý và phản hồi nhanh hơn rất nhiều. Đây là lợi thế lớn cho các ứng dụng yêu cầu thời gian thực như điều khiển mạng, IoT, hoặc truyền video.

Tỷ lệ mất gói mô hình DQN-PI: 0,0092, PI tiêu chuẩn: 0,0260. Mô hình DQN-PI giúp giảm đáng kể tỷ lệ mất gói, tức là tăng khả năng truyền dữ liệu thành công, đảm QoS. Tỷ lệ mất gói thấp rất quan trọng với các dịch vụ yêu cầu độ tin cậy cao như VoIP, video conferencing hoặc dữ liệu điều khiển.

Thông lượng mô hình DQN-PI: 0,9842, PI tiêu chuẩn: 0,7940, mô hình DQN-PI đạt thông lượng cao hơn, phản ánh khả năng tận dụng tài nguyên mạng tốt hơn, xử lý nhiều gói tin hơn trong cùng một khoảng thời gian.

6. Kết luận và hướng phát triển

DQN-PI đã chứng minh hiệu quả trong việc tối ưu hóa thuật toán điều khiển hàng đợi PIE bằng cách kết hợp học tăng cường, tự động điều chỉnh các tham số α và β để đạt được độ trễ mục tiêu đồng thời giảm tỉ lệ mất gói và tối ưu thông lượng mạng. Phương pháp này mang lại nhiều ưu điểm vượt trội so với PIE truyền thống, bao gồm khả năng thích ứng linh hoạt với sự thay đổi lưu lượng mạng, giảm độ trễ trung bình, ổn định hàng đợi và cân bằng hiệu quả giữa thông lượng và độ trễ. Các kết quả thử nghiệm cho thấy DQN-PI không chỉ cải thiện hiệu suất mạng mà còn có khả năng tự học để tối ưu hóa các tham số điều khiển trong môi trường mạng phức tạp.

DQN-PI mở ra nhiều triển vọng trong việc ứng dụng AI vào quản lý tắc nghẽn mạng, đặc biệt trong các hệ thống hiện đại như IoT, mạng 5G và Trung tâm dữ liệu (data center). Có thể tiếp tục phát triển và nâng cao hiệu quả của DQN-PI, như triển khai trên mạng SDN (Software-Defined Networking) để đánh giá hiệu quả trong các tình huống thực tế với nhiều node mạng khác nhau; kết hợp với ECN (Explicit Congestion Notification) để giảm mất gói mà không cần loại bỏ gói tin.Với những cải tiến liên tục, phương pháp này hứa hẹn sẽ trở thành giải pháp tối ưu cho bài toán điều khiển hàng đợi trong tương lai, góp phần nâng cao hiệu suất và độ ổn định của các hệ thống mạng phức tạp.

Lời cảm ơn

Xin cảm ơn Khoa Công nghệ thông tin, Phòng Khoa học Công nghệ và Trung tâm Thư viện Trường Đại học Nguyễn Tất Thành đã hỗ trợ về thời gian và tài liệu cho nghiên cứu này.

Tài liệu tham khảo

1. Barakabitze, A. A., Barman, N., Ahmad, A., Zadtootaghaj, S., Sun, L., Martini, M. G., & Atzori, L. (2019). QoE management of multimedia streaming services in future networks: A tutorial and survey. IEEE Communications Surveys & Tutorials, 22(1), 526-565.2.

2. Usmanova, N. B., Mirzayev, D. A., Ergashev, F. A., & Yunusova, D. A. (2023). Field monitoring application based on video surveillance: Evaluation of system performance. In E3S Web of Conferences (Vol. 443, p. 06016). EDP Sciences.

3. Hassan, S. O., Rufai, A. U., Agbaje, M. O., Enem, T. A., Ogundele, L. A., & Usman, S. A. (2022). Improved random early detection congestion control algorithm for internet routers. Indonesian Journal of Electrical Engineering and Computer Science, 28(1), 384-395.

4. Kumhar, D., kumar, A., & Kewat, A. (2021). QRED: an enhancement approach for congestion control in network communications. International Journal of Information Technology, 13(1), 221-227.

5. Adams, R. (2012). Active queue management: A survey. IEEE communications surveys & tutorials, 15(3), 1425-1476.

6. Velayutham, A. (2022). Congestion control and traffic shaping in high-bandwidth applications: Techniques to manage network congestion and optimize traffic flow in gaming, ar/vr, and cloud services. Eigenpub Review of Science and Technology, 6(1), 144-165.
7. Wei, W., Gu, H., & Li, B. (2021). Congestion control: A renaissance with machine learning. IEEE network, 35(4), 262-269.
8. Panahi, P. H., Jalilvand, A. H., & Diyanat, A. (2024). Enhancing Quality of Experience in Telecommunication Networks: A Review of Frameworks and Machine Learning Algorithms. arXiv preprint arXiv:2404.16787.
9. Wang, Q., & Tang, C. (2021). Deep reinforcement learning for transportation network combinatorial optimization: A survey. Knowledge-Based Systems, 233, 107526.
10. Luong, N. C., Hoang, D. T., Gong, S., Niyato, D., Wang, P., Liang, Y. C., & Kim, D. I. (2019). Applications of deep reinforcement learning in communications and networking: A survey. IEEE communications surveys & tutorials, 21(4), 3133-3174.
11. Pan, R., Natarajan, P., Piglione, C., Prabhu, M. S., Subramanian, V., Baker, F., & VerSteeg, B. (2013). PIE: A lightweight control scheme to address the bufferbloat problem. In 2013 IEEE 14th international conference on high performance switching and routing (HPSR) (pp. 148-155). IEEE.
12. Perry, Y., Frujeri, F. V., Hoch, C., Kandula, S., Menache, I., Schapira, M., & Tamar, A. (2023). A Deep Learning Perspective on Network Routing. arXiv preprint arXiv:2303.00735.
13. Fan, J., Wang, Z., Xie, Y., & Yang, Z. (2020). A theoretical analysis of deep Q-learning. In Learning for dynamics and control (pp. 486-489). PMLR.
14. Sait Ciftler, B., Alwarafy, A., Abdallah, M., & Hamdi, M. (2021). DQN-Based Multi-User Power Allocation for Hybrid RF/VLC Networks. arXiv e-prints, arXiv-2102.

PI - DQN: an AI-based proposal for queue management in communication network routers

Vuong Xuan Chi, Pham Dinh Tai, Chau Gia Han
Faculty of Information Technology, Nguyen Tat Thanh University, Ho Chi Minh City
*vxchi@ntt.edu.vn

Abstract In modern networking environments, traffic congestion and bufferbloat remain significant challenges, requiring proactive Active Queue Management (AQM) mechanisms in routers to ensure quality of service. PIE (Proportional Integral Enhanced) is an effective AQM algorithm that employs a PI controller to maintain target delay, mitigate congestion, and improve network performance. PIE stands out for its simplicity, stability, and widespread deployment in modern routers. However, with the diversity of next-generation network traffic, traditional PIE faces limitations in dynamically adapting to varying loads. To address these challenges, this paper proposes integrating PIE with Deep Q-Network (DQN), a Reinforcement Learning (RL) method. DQN optimizes PIE parameters (α, β, $target\_delay$) based on complex network data, thereby enhancing its responsiveness to different traffic types. Experimental results demonstrate that PI-DQN reduces average latency and increases throughput compared to the original PIE, particularly excelling in heterogeneous network environments. In the future, PI-DQN continues to improve AQM performance, adapting to complex network environments such as 5G, IoT, and real-time streaming, thereby contributing to digital transformation and strengthening the long-term competitive advantage of local network systems.
Keywords Network congestion, AQM, PIE algorithm, network performance, deep reinforcement learning.