BỘ THÔNG TIN VÀ TRUYỀN THÔNG
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Lê Thị Thùy Dương

GIẢI PHÁP ĐIỀU KHIỂN TẮC NGHẼN TRONG MẠNG IoT VỚI GIAO THỨC CoAP

LUẬN ÁN TIẾN SĨ KỸ THUẬT

Hà Nội - 2023

BỘ THÔNG TIN VÀ TRUYỀN THÔNG
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Lê Thị Thùy Dương

GIẢI PHÁP ĐIỀU KHIỂN TẮC NGHẼN TRONG MẠNG IoT VỚI GIAO THỨC CoAP

CHUYÊN NGÀNH: KỸ THUẬT VIỄN THÔNG
MÃ SỐ: 9.52.02.08

LUẬN ÁN TIẾN SĨ KỸ THUẬT

NGƯỜI HƯỚNG DẪN KHOA HỌC

1. PGS. TSKH HOÀNG ĐĂNG HẢI

2. TS. PHẠM THIẾU NGA

Hà Nội – 2023

LỜI CAM ĐOAN

Tôi xin cam đoan luận án "Giải pháp điều khiển tắc nghẽn trong mạng IoT với giao thức CoAP" là công trình nghiên cứu của tôi, dưới sự hướng dẫn của PGS.TSKH Hoàng Đăng Hải và TS. Phạm Thiếu Nga. Các kết quả được trình bày trong luận án là hoàn toàn trung thực và không xung đột với bất kỳ tác giả nào khác. Các số liệu trong luận án được sử dụng là trung thực, một phần đã được công bố trên các tạp chí khoa học chuyên ngành với sự đồng ý và cho phép của đồng tác giả.

.

Người cam đoan

Lê Thị Thùy Dương

LỜI CẢM ƠN

Để hoàn thành luận án tôi xin bày tỏ lòng biết ơn sâu sắc đến PGS.TSKH. Hoàng Đăng Hải – Học viện Công nghệ Bưu chính viễn thông và TS Phạm Thiếu Nga – Đại học xây dựng Hà Nội đã tận tình hướng dẫn, tạo mọi điều kiện và giúp đỡ tôi thực hiện và hoàn thành luận án này.

Tôi xin trân trọng cảm ơn các thầy cô giáo của Học viện Công nghệ Bưu chính viễn thông đã có những nhận xét khoa học chân thành và sâu sắc trong các buổi báo cáo định hướng và tiến độ nghiên cứu cũng như báo cáo chuyên đề và tiểu luận tổng quan.

Tôi cũng xin trân trọng cảm ơn các thầy cô trong khoa Viễn thông và khoa Đào tạo sau đại học, Học viện Công nghệ Bưu chính viễn thông đã giảng dạy và giúp đỡ nhiệt tình trong suốt quá trình học tập tại Học viện Bưu chính viễn thông.

Tôi biết ơn những người thân trong gia đình đã luôn bên tôi, những đồng nghiệp, bạn bè đã động viên để tôi có thể hoàn thành bản luận án.

Nghiên cứu sinh

Lê Thị Thùy Dương

MỤC LỤC

DANH MỤC CÁC TỪ VIẾT TẮT vi
DANH MỤC CÁC KÝ HIỆU viii
DANH MỤC HÌNH VẼ x
DANH MỤC CÁC BẢNG xi
MỞ ĐẦU 1
CHƯƠNG 1.TỔNG QUAN VỀ MẠNG IoT VÀ VẤN ĐỀ ĐIỀU KHIỂN TẮC NGHẼN .8
1.1. Tổng quan về mạng IoT 8
1.1.1. Khái niệm về IoT 8
1.1.2. Các ứng dụng IoT 9
1.1.3. Mô hình kiến trúc mạng IoT 10
1.1.4. Tóm lược về các giao thức tầng ứng dụng của IoT 11
1.2. Tắc nghẽn và nguyên nhân tắc nghẽn 13
1.2.1. Khái niệm tắc nghẽn mạng 13
1.2.2. Nguyên nhân tắc nghẽn mạng 14
1.2.3. Tắc nghẽn mạng IoT 14
1.3. Điều khiển tắc nghẽn 15
1.3.1. Điều khiển vòng hở và điều khiển vòng kín 15
1.3.2. Điều khiển dựa cửa sổ và điều khiển dựa tốc độ 16
1.3.3. Điều khiển tắc nghẽn mạng IoT 17
1.4. Điều khiển mờ và khả năng áp dụng cho điều khiển tắc nghẽn 18
1.4.1. Logic mờ 18
1.4.2. Điều khiển mờ 20
1.4.3. Khả năng áp dụng điều khiển mờ cho điều khiển tắc nghẽn 22
1.5. Giao thức CoAP 24
1.5.1. Hoạt động của CoAP 24
1.5.2. Cơ chế điều khiển tắc nghẽn của CoAP 25
1.6. Các nghiên cứu liên quan cải tiến CoAP và những tồn tại 26
1.6.1. Các nghiên cứu liên quan cải tiến CoAP 26
1.6.2. Những tồn tại của CoAP và của các nghiên cứu liên quan 31
1.7. Các tham số đánh giá hiệu năng giao thức CoAP 34
1.8. Kết luận chương 1 35
CHƯƠNG 2. MÔ HÌNH TRUYỀN CHUỖI GÓI VÀ GIAO THỨC RCoAP ĐIỀU KHIỂN TẮC NGHẼN DỰA VÀO TỐC ĐỘ 37
2.1. Mô hình phân tích cho truyền chuỗi gói tin cậy với CoAP 37
2.1.1. Sơ đồ luồng tin kết nối đầu cuối của CoAP trong mạng IoT 37
2.1.2. Mô hình điều khiển tắc nghẽn cho CoAP 38
2.1.3. Tính toán tốc độ phát chuỗi gói tin của CoAP 40
2.2. Đề xuất giao thức RCoAP 45
2.2.1. Cơ chế hoạt động và điều khiển tắc nghẽn của RCoAP 45
2.2.2. Các trạng thái của giao thức RCoAP 47
2.2.3. Các thuật toán cơ bản của giao thức RCoAP 48
2.3. Tính toán hiệu năng giao thức RCoAP 53

2.3.1. Độ trễ gói tin....53
2.3.2. Tính toán mất gói tin khi truyền....54
2.3.3. Thông lượng....54
2.3.4. Các tham số hiệu năng khác....54
2.4. Kết quả mô phỏng cho RCoAP....55
2.4.1. Thiết lập môi trường mô phỏng....55
2.4.2. Kịch bản 2.1....56
2.4.3. Kịch bản 2.2....58
2.4.4. Kịch bản 2.3....61
2.5. Tổng hợp các thay đổi cải tiến của RCoAP so với CoAP....66
2.6. Kết luận chương 2....67
CHƯƠNG 3. GIAO THỨC FCoAP ĐIỀU KHIỂN TẮC NGHẼN DỰA VÀO HỆ ĐIỀU KHIỂN MỜ....69
3.1. Giải pháp điều khiển tắc nghẽn sử dụng hệ điều khiển mờ....69
3.1.1. Sơ đồ giải pháp và các tham số điều khiển....69
3.1.2. Phát hiện sớm tắc nghẽn....70
3.1.3. Tính toán băng thông cổ chai và tải lưu lượng chuỗi gói....71
3.1.4. Lựa chọn đầu vào cho hệ điều khiển mờ....72
3.1.5. Lựa chọn đầu ra cho hệ điều khiển mờ....75
3.2. Thiết kế hệ thống điều khiển mờ....76
3.2.1. Mờ hóa....76
3.2.2. Cơ sở luật mờ....79
3.2.3. Mô tơ suy diễn mờ....80
3.2.4.Giải mờ, điều chỉnh tốc độ phát, cập nhật RTO....81
3.3. Giao thức FCoAP cho điều khiển tắc nghẽn mạng....81
3.3.1. Cơ chế điều khiển của FCoAP....81
3.3.2. Các trạng thái của giao thức FCoAP....84
3.3.3. Các thuật toán của giao thức FCoAP....85
3.4. Tính toán hiệu năng giao thức FCoAP....88
3.4.1. Độ trễ đầu cuối của FCoAP....88
3.4.2. Thông lượng của FCoAP....90
3.5.3. Các thông số hiệu năng khác....90
3.5. Kết quả mô phỏng đánh giá FCoAP....90
3.5.1. Thiết lập môi trường mô phỏng cho FCoAP....90
3.5.2. Kịch bản 3.1: Kiểm tra hoạt động của hệ điều khiển mờ....92
3.5.3. Kịch bản 3.2: So sánh hiệu năng FCoAP và CoAP....94
3.5.4. Kịch bản 3.3: Hiệu năng FCoAP và CoAP khi có lưu lượng UDP thay đổi....96
3.5.5. Kịch bản 3.4: Hiệu năng FCoAP và CoAP khi có lưu lượng CoAP hỗn hợp....98
3.5.6. Kịch bản 3.5: FcoAP và CoAP khi có lưu lượng TCP/UDP hỗn hợp....100
3.5.7. Kịch bản 3.6: So sánh FCoAP với CoAP, CoCoA, CoCoA+....101
3.6. So sánh FCoAP với RCoAP....103
3.7. Kết luận chương 3....109
KẾT LUẬN....110
A. Đóng góp mới của luận án....111
B. Hướng phát triển tiếp....111

TÀI LIỆU THAM KHẢO 113
PHỤ LỤC 121

DANH MỤC CÁC TỪ VIẾT TẮT

Từ viết tắt	Tiếng Anh	Tiếng Việt
6LoWPAN	Ipv6 protocol over low power wireless PAN	Giao thức IPv6 với mạng vùng hẹp không dây công suất thấp
ABF	Adaptive-boundary Backoff Factor	Hệ số lùi thích nghi
ACK	Acknowledgement	Gói tin báo nhận
AIAD	Additive Increase / Additive Decrease	Tăng cộng/Giảm cộng
AIMD	Additive Increase Multiplicative Decrease	Tăng cộng giảm nhân
AMQP	Advanced Message Queuing Protocol	Giao thức xếp hàng tin nhắn nâng cao
BBR	Bottleneck Bandwidth and Round-trip propagation time	Tích băng thông cổ chai và thời gian truyền quay vòng
BDP-CoAP	Bandwidth-Delay Product CoAP	Bản cải tiến CoAP có sử dụng tích băng thông – độ trễ
BEB	Binary Exponential Backoff	Lùi theo hàm mũ nhị phân
BUNCON	Basic Uncconfirmable	Luồng CoAP ở chế độ không tin cậy
CoAP	Constrained Application Protocol	Giao thức ứng dụng có ràng buộc
CoAP_R	Rate-Based Congestion Control Mechanism in CoAP	Bản cải tiến CoAP có cơ chế điều khiển tắc nghẽn dựa vào tốc độ
CoCoA	Simple Congestion Control Advanced	Bản cải tiến CoAP có cơ chế điều khiển tắc nghẽn cải tiến đơn giản
CoG	Center-of-Gravity	Trung bình trọng tâm
CON	Confirmable	Gói tin truyền tin cậy
CWDN	Congestion Window	Cửa sổ tắc nghẽn
DNS	Domain Name System	Hệ thống phân giải tên miền
ETSI	European Telecommunications Standards Institute	Viện tiêu chuẩn viễn thông Châu Âu
FASOR	Fast-Slow RTO	Bản cải tiến CoAP sử dụng tính RTO nhanh và chậm
FCS	Fuzzy Control System	Hệ điều khiển mờ
Fuzzy-RED	Fuzzy Random Early Drops	Loại bỏ gói ngẫu nhiên sớm sử dụng điều khiển mờ
GW	Gateway	Trạm cửa ngõ
H2E	Human to Environment	Truyền tin người – với môi trường
H2M	Human to Machine	Truyền tin người - với máy
HTTP	Hypertext Transfer Protocol	Giao thức truyền tải siêu văn bản
ICN	Information Concentric Networks	Mạng tập trung thông tin
IEEE	Insitute of Electrical and Electronics Engineers	Viện kỹ sư điện và điện tử
IETF	Internet Engineering Task Force	Tổ chức chuyên trách về kỹ thuật Internet

IoT	Internet of Things	Internet vạn vật
IP	Ỉnternet Protocol	Giao thức Internet
IPI	Inter-packet interval	Khoảng thời gian giữa 2 gói tin liên tiếp
ITU	International Telecommunication Union	Tổ chức viễn thông thế giới
ITU-T	International Telecommunication Union – Telecomunication Standarization Sector	Tổ chức viễn thông quốc tế - Lĩnh vực tiêu chuẩn viễn thông
M2H	Machine to Human	Truyền tin máy - người
M2M	Machine to Machine	Truyền tin máy với máy
MAC	Medium Access Control	Điều khiển truy nhập môi trường
MQTT	Message Queue Telemetry Transport	Giao thức truyền tải từ xa hàng đợi bản tin
NON	Non-confirmable	Gói tin truyền không tin cậy
PBF	Probability of Backoff Factor	Hệ số xác suất của cơ chế lùi
RAP	Rate Adaptation Protocol	Giao thức thích nghi tốc dộ
RCAP	Rate Control Adaptive Protocol	Giao thức điều khiển tốc độ thích nghi
RCoAP	Rate-based CoAP	Bản cải tiến CoAP điều khiển tắc nghẽn dựa vào tốc độ
RED	Random Early Drop	Loại bỏ gói ngẫu nhiên sớm
RES	Reset	Bản tin báo hủy kết nối thiết lập mới
REST	Representational State Transfer	Chuyển trạng thái đại diện
RFC	Request for Comments	Tiêu chuẩn Internet của tổ chức IETF
ROTT	relative one-way trip time	Thời gian đi một chiều tương đối
RTO	Retransmission Timeout	Định thời phát lại
RTT	Round Trip Time	Thời gian quay vòng
TCP	Transmission Control Protocol	Giao thức điều khiển truyền tải
TFRC	TCP Friendly Rate Control	Giao thức điều khiển tốc độ thân thiện với TCP
UDP	User Datagram Protocol	Giao thức dữ liệu người dùng
XML	Extensible Markup Language	Ngôn ngữ đánh dấu mở rộng
XMPP	Extensible Messaging and Presence Protocol	Giao thức hiện diện và nhắn tin mở rộng

DANH MỤC CÁC KÝ HIỆU

Ký hiệu	Ý nghĩa	Đơn vị đo
$\theta(k)$	Thông lượng đo được ở thời điểm k	bps
$\theta(S)$	Hàm thông lượng, là hàm số của S	bps
ΔRTT_a	Độ biến thiên tuyệt đối của RTT	ms
ΔRTT_r	Độ biến thiên tương đối của RTT	Ms
$\theta_t(k)$	Thông lượng tức thời của luồng tin trong một khoảng thời gian T(k-1,k)	bps
θ_{TB}	Thông lượng trung bình	bps
ΔT_{min}	Hiệu số thời gian đến nhỏ nhất giữa 2 lần nhận gói tin CON liên tiếp	ms
$\mu_M(x)$	Độ thuộc	-
$B(S)$	Số gói tin được xử lý tại bên nhận khi không có mất gói	gói
$BDP(k)$	Số gói tin inflight	gói
$BG(k)$	Tỷ số của thông lượng trên băng thông cổ chai lớn nhất tại thời điểm k	-
$BW(k)$	Băng thông cổ chai ở thời điểm k	bps
$BW_{max}(k)$	Băng thông cổ chai tối đa của kết nối tại k	bps
$C(S)$	Số gói tin được xử lý tại bên nhận khi có mất gói	gói
$C\_Degree(k)$	Cấp độ tắc nghẽn tại thời điểm k	-
D	Độ trễ gói tin một chiều	ms
$D(S)$	Hàm độ trễ, là hàm số của S	ms
D_0	Độ trễ gói tin một chiều của gói tin đầu tiên trong luồng tin	ms
d_i	Độ trễ của gói tin i	ms
D_{TB}	Độ trễ gói tin trung bình của luồng tin	ms
N	Số gói tin phát đi trong một phiên kết nối	gói
$n(k)$	Lượng gói tin tích lũy trong chu kỳ k	gói
$nACK$	Số gói tin ACK bên gửi nhận được trong khoảng thời gian khởi tạo	gói
N_k	Số gói tin được phát đi thành công (nhận được ACK) trong chu kỳ k của một phiên kết nối	gói
P	Xác suất mất gói trong một lần phát	-
P_i	Xác suất mất gói sau *i* lần phát	-
R	Tốc độ phát	bps
$R(k)$	Tốc độ phát gói tin của bên gửi trong chu kỳ k	bps
R_{max}	Tốc độ tối đa cho phép phụ thuộc vào ứng dụng và điều kiện băng thông mạng	bps
$R_r(k)$	Tốc độ xử lý của bên nhận trong chu kỳ k	bps
$RT(k)$	Biến thiên của RTT	ms
$RT0_o$	Giá trị RTO khởi tạo	ms
$RTO_{backoff}$	Giá trị RTO được dùng cho chu kỳ lùi	ms
RTO_{init}	Giá trị RTO được dùng cho lần phát gói kế tiếp	ms
$RTO_{overall}$	Giá trị RTO tổng thể được sử dụng để tính RTO_{init}	ms
RTT	Thời gian quay vòng	ms
$RTT(k)$	Thời gian quay vòng của chu kỳ k	ms
$RTT_m(k)$	RTT đo thực tế tại thời điểm k	ms

$RTT_{max}(k)$	Giá trị RTT lớn nhất tính đến thời điểm k	ms
RTT_{min}	Giá trị RTT nhỏ nhất tới thời điểm hiện tại	ms
$RTT_{min}(k)$	Giá trị RTT nhỏ nhất tính đến thời điểm k	ms
$RTT_S(k)$	Giá trị RTT trung bình theo phương pháp EWMA	ms
$RTTVAR_x$	Giá trị ước trung bình ước tính của RTT theo 2 chế độ mạnh, yếu	ms
S	Số gói tin inflight	gói
$T(k)$	Thời gian của một chu kỳ k	ms
$T(k-1,k)$	Khoảng thời gian giữa 2 gói ACK liên tiếp	ms
t_0	Thời điểm bắt đầu kết nối	ms
$U(S)$	Hàm hiệu suất, là hàm số của S	-
α	Hệ số ước tính RTT	-
β	Hệ số ước tính RTT	-
γ	Hệ số ước tính RTO	-
∂	Hệ số điều khiển của hàm hiệu suất	-

DANH MỤC HÌNH VẼ

Hình 1.1 Kiến trúc mạng ICN 11
Hình 1.2 Kiến trúc mạng ICN cụ thể cho các ứng dụng IoT 11
Hình 1.3 Sơ đồ tóm lược các giao thức chính của IoT 11
Hình 1.4 Mối quan hệ giữa các đại lượng a) theo thời gian, b) theo tải lưu lượng 13
Hình 1.5 Ví dụ về trường hợp nghẽn cổ chai 14
Hình 1.6 Cơ chế tăng cộng – giảm nhân của TCP 16
Hình 1.7 Cách biểu diễn hàm thuộc a) hình thang, b) hình tam giác 19
Hình 1.8 Mô hình hệ thống điều khiển mờ 20
Hình 1.9 Tiêu đề cố định của gói tin CoAP 24
Hình 1.10 Cấu trúc một gói tin CoAP 24
Hình 1.11 Ví dụ về chế độ truyền tin cậy (a) và không tin cậy (b) 25
Hình 1.12 a) Trao đổi gói tin CON và ACK, b) Các lần phát lại 26
Hình 2.1 Sơ đồ luồng tin kết nối đầu cuối của CoAP trong mạng IoT 37
Hình 2.2 Mối quan hệ giữa các đại lượng điều khiển (dựa theo [64,27]) 38
Hình 2.3 Các chu kỳ phát gói tin theo chuỗi của CoAP 40
Hình 2.4 Sơ đồ triển khai RCoAP tại bên gửi và bên nhận 45
Hình 2.5 Bốn trạng thái của giao thức RCoAP 47
Hình 2.6 Mạng hình sao 55
Hình 2.7 Mạng hình xương cá 55
Hình 2.8 Xác định tốc độ ban đầu 57
Hình 2.9 Tăng/giảm tốc độ và lùi 57
Hình 2.10 Mất gói do lỗi kênh vô tuyến 58
Hình 2.11 Mất gói do tắc nghẽn 58
Hình 2.12 Độ trễ của các giao thức 59
Hình 2.13 Thông lượng của các giao thức 59
Hình 2.14 Độ trễ khi có TCP 61
Hình 2.15 Thông lượng khi có TCP 61
Hình 2.17 Độ trễ 3 luồng RCoAP (trong số 10 luồng) 61
Hình 2.18 Thông lượng 3 luồng RCoAP (trong số 10 luồng) 62
Hình 2.19 Độ trễ trung bình của 10 luồng RCoAP và 10 luồng CoAP 62
Hình 2.20 Thông lượng trung bình của 10 luồng RCoAP và 10 luồng CoAP 63
Hình 2.21 Độ trễ RcoAP và CoAP 64
Hình 2.22 Thông lượng RCoAP và CoAP 64
Hình 2.23 Độ trễ trung bình của 10 luồng RCoAP và 10 luồng CoAP 65
Hình 2.24 Thông lượng trung bình của 10 luồng RCoAP và 10 luồng CoAP 65
Hình 3.1 Sơ đồ giải pháp điều khiển mờ 70
Hình 3.2 a) Biến thiên RTT b) Biến thiên băng thông cổ chai 70
Hình 3.3 Biến thiên tải lưu lượng và BDP của CoAP 72
Hình 3.4 a) Biến thiên RTT khi tắc nghẽn và b) Biến thiên tương ứng của RT(k) 73
Hình 3.5 a) Biến thiên thông lượng b) Biến thiên BG_gradient 75
Hình 3.6 Mô hình hệ thống điều khiển mờ 76
Hình 3.7 a) Các hàm thuộc cho RT_gradient, b) Các hàm thuộc cho BG_gradient 77
Hình 3.8 a) Độ thuộc RT_gradient=0.7, b) Độ thuộc BG_gradient=0.65 78
Hình 3.9 Các hàm thuộc cho đầu ra C_degree 79
Hình 3.10: Tập mờ kết quả đầu ra. 81

Hình 3.11 Cơ chế điều khiển FCoAP (Fuzzy CoAP) 82
Hình 3.12 Ba trạng thái của giao thức FCoAP 85
Hình 3.13 Biến thiên RT_gradient, BG_gradient và C_degree 92
Hình 3.14 Biến thiên C_degree của ba luồng FCoAP 92
Hình 3.15 Độ trễ của FCoAP 93
Hình 3.16 Thông lượng của FCoAP 93
Hình 3.17 Độ trễ của FCoAP và CoAP 95
Hình 3.18 Thông lượng FCoAP và CoAP 95
Hình 3.19 Độ trễ của FCoAP và CoAP khi có lưu lượng UDP thay đổi 96
Hình 3.20 Số lần phát lại của FCoAP và CoAP khi có lưu lượng UDP thay đổi 97
Hình 3.21 Phát lại đúp của FCoAP và CoAP khi có lưu lượng UDP thay đổi 97
Hình 3.22 Thông lượng của FCoAP và CoAP khi có lưu lượng UDP thay đổi 97
Hình 3.23 Độ trễ của FCoAP và CoAP khi lưu lượng CoAP hỗn hợp 99
Hình 3.24 Thông lượng của FCoAP và CoAP khi lưu lượng CoAP hỗn hợp 99
Hình 3.25 Độ trễ của FCoAP và CoAP khi có lưu lượng nền TCP/UDP 100
Hình 3.26 Thông lượng của FCoAP và CoAP khi có lưu lượng nền TCP/UDP 101
Hình 3.27 So sánh độ trễ của FCoAP, CoAP, CoCoA và CoCoA+ 102
Hình 3.28 So sánh thông lượng của FCoAP, CoAP, CoCoA và CoCoA+ 102
Hình 3.29 Độ trễ của FCoAP và RCoAP trong chặng đơn 105
Hình 3.30 Thông lượng của FCoAP và RCoAP trong chặng đơn 106
Hình 3.31 Độ trễ của FCoAP và RCoAP trong chặng dài 107
Hình 3.32 Thông lượng của FCoAP và RCoAP trong chặng dài 108

DANH MỤC CÁC BẢNG

Bảng 2.1: Các ký hiệu cho mô hình phân tích 39
Bảng 2.2: Các ký hiệu tính toán hiệu năng RCoAP 53
Bảng 2.3: Các tham số chính để thiết lập mô phỏng RCoAP 56
Bảng 2.4: So sánh hiệu năng của RCoAP và các giao thức CoAP khác 59
Bảng 2.6: So sánh RCoAP và CoAP khi độ trễ liên kết D = 70ms 63
Bảng 2.7: So sánh RCoAP và CoAP khi độ trễ liên kết D =120ms 65
Bảng 2.8: Các cải tiến của RCoAP so với CoAP, CoCoA và CoCoA+ 67
Bảng 3.1: Tập luật mờ 79
Bảng 3.2: Các tham số chính để thiết lập mô phỏng RCoAP 91
Bảng 3.3: So sánh hiệu năng của các luồng tin FCoAP 94
Bảng 3.4: So sánh hiệu năng của FCoAP và CoAP 94
Bảng 3.5: Hiệu năng của FCoAP và COAP khi có lưu lượng UDP thay đổi 98
Bảng 3.6: So sánh hiệu năng của FCoAP và các giao thức CoAP khác 103
Bảng 3.7: So sánh FCoAP và RCoAP trong đơn chặng 106
Bảng 3.8: So sánh FCoAP và RCoAP trong chặng dài 108

MỞ ĐẦU

1. Lý do chọn đề tài và trọng tâm nghiên cứu

Trong vài năm gần đây, Internet vạn vật (IoT-Internet of Things) đã trở thành phổ biến trong nhiều lĩnh vực ứng dụng. Những tiến bộ công nghệ trong các lĩnh vực điện tử, viễn thông và công nghệ thông tin, điển hình là công nghệ cảm biến và mạng vô tuyến đã tạo động lực cho sự ra đời của kỷ nguyên kết nối Internet vạn vật. Theo các tổ chức tiêu chuẩn quốc tế [54, 56], IoT là mạng kết nối các thực thể (thiết bị) có khả năng thu thập, xử lý và trao đổi dữ liệu thông qua Internet. IoT kết nối đa dạng thiết bị mọi lúc mọi nơi, từ các thiết bị đơn giản gắn các cảm biến cho tới các thiết bị thông minh như điện thoại thông minh, thiết bị theo dõi sức khỏe. Thông qua kết nối các thiết bị thông minh với Internet, một nền tảng mạng mới được hình thành cho phép phát triển hàng loạt ứng dụng mới như ngôi nhà thông minh, đô thị thông minh, chăm sóc sức khỏe, giám sát môi trường, v.v.

Ứng dụng rộng rãi của IoT trong các lĩnh vực dẫn tới sự gia tăng thiết bị kết nối, nhu cầu phát triển tiêu chuẩn và giao thức cho IoT. Để kết nối với Internet, các thiết bị IoT hiện đang sử dụng các giao thức truyền thống như IP (Internet protocol), TCP (Transmission Control Protocol), UDP (User Datagram Protocol). Đặc biệt, IoT cần các giao thức tầng ứng dụng để trao đổi dữ liệu giữa các đầu cuối. Những giao thức tầng ứng dụng rất quan trọng đối với mạng IoT nhằm phục vụ cho các ứng dụng đa dạng, là một cơ sở quan trọng để triển khai IoT rộng rãi trong mọi lĩnh vực. Các tổ chức tiêu chuẩn quốc tế như ITU, IETF đã nỗ lực phát triển và chuẩn hóa các giao thức tầng ứng dụng mới cho IoT. Các giao thức điển hình tầng ứng dụng gồm: giao thức vận chuyển hàng đợi bản tin từ xa MQTT (Message Queue Telemetry Transport) [92], giao thức xếp hàng bản tin nâng cao AMQP (Advanced Message Queuing Protocol) [30], giao thức hiện diện và nhắn tin mở rộng XMPP (Extensible Messaging and Presence Protocol) [53] và giao thức ứng dụng có ràng buộc CoAP (Constrained Application Protocol) [100]. Các giao thức này có một đặc điểm chung là hạng nhẹ để phù hợp với môi trường mạng IoT, trong đó MQTT,

AMQP và XMPP hoạt động trên tầng TCP, còn CoAP hoạt động trên tầng UDP. TCP có nhiều hạn chế trong mạng IoT như: cần thời gian thiết lập và duy trì kết nối, chi phí tiêu đề gói tin lớn, độ trễ lớn do cơ chế bắt tay 3 bước, độ phức tạp cao, kém hiệu quả cho kích thước cửa sổ nhỏ [101, 36, 103, 45]. CoAP hoạt động dựa trên UDP nên có lợi thế là tiêu đề gói nhỏ, cơ chế hoạt động đơn giản, tốc độ nhanh, độ trễ thấp, không mất thời gian khởi tạo và duy trì kết nối, phù hợp cho các ứng dụng thời gian thực. Do đó, CoAP dựa trên UDP đã được đánh giá là thích hợp hơn cho nhiều ứng dụng IoT và trở thành nền tảng cho các thiết bị IoT có hạn chế tài nguyên như đã chỉ ra trong các tiêu chuẩn quốc tế [101, 103, 57, 99, 40]. Tuy nhiên, do thiết kế đơn giản nên CoAP còn nhiều hạn chế và cần được phát triển tiếp như đã nêu trong RFC 7252 [100], tiêu chuẩn của ITU [58] và một số RFC khác [102, 103]. **Nghiên cứu cải tiến CoAP đang là vấn đề rất được quan tâm và là chủ đề nghiên cứu của luận án này**.

Nhiều ứng dụng IoT ngày nay không chỉ trao đổi thưa thớt các gói tin, mà thường phải truyền các luồng dữ liệu lớn theo thời gian thực, ví dụ các ứng dụng trong y tế, chăm sóc bệnh nhân, theo dõi giám sát thảm họa, giám sát video an ninh [36, 57, 103]. Tắc nghẽn xảy ra khi tải lưu lượng vượt quá băng thông kết nối hoặc năng lực xử lý. Tắc nghẽn là vấn đề thường xuyên xảy ra trong mạng IoT như đã nêu trong các tiêu chuẩn quốc tế như ITU [56 - 59], các RFC [99, 102, 103] và tiêu chuẩn ETSI TR 103.375 [40]. Mạng IoT có tài nguyên hạn chế, lượng dữ liệu cần truyền rất lớn từ nhiều thiết bị IoT. Môi trường mạng IoT có nhiều biến động, có nhiều khả năng lỗi và mất gói. Biến động bất thường của băng thông liên kết dẫn đến bùng nổ dữ liệu (burstiness), xuất hiện các chuỗi gói làm gia tăng nguy cơ tắc nghẽn [102, 99, 103, 62]. Điều khiển tắc nghẽn có vai trò quan trọng trong việc giảm thiểu tắc nghẽn, giảm mất gói, duy trì độ trễ gói tin nhỏ, bảo đảm thông lượng và hiệu năng mạng, đáp ứng yêu cầu các ứng dụng, đặc biệt cho các ứng dụng nhạy cảm với trễ và mất gói.

Hạn chế cơ bản nhất của CoAP là cơ chế điều khiển tắc nghẽn như đã chỉ ra trong RFC 7252 [100] và các tiêu chuẩn khác như [102, 103, 58]. Các vấn đề tồn tại

cụ thể của CoAP trong cơ chế điều khiển tắc nghẽn gồm: sử dụng các tham số cố định, chỉ điều khiển tốc độ phát lại khi đã xảy ra mất gói (nghĩa là khi đã tắc nghẽn), không hỗ trợ chuỗi gói, không phát hiện sớm tắc nghẽn. Nhiều nghiên cứu mới đây đã chỉ ra sự cần thiết phải cải tiến cơ chế điều khiển của CoAP [107, 36, 101, 5, 37]. Các cải tiến CoAP đã có tới nay chủ yếu gồm: thay đổi cách tính toán thời gian quay vòng RTT (Round Trip-time) và định thời phát lại RTO (Retransmission Timeout) thay vì dùng tham số cố định [9, 15, 21, 23, 38, 5], cải tiến cơ chế lùi [18, 23, 87, 107], cải tiến thuật toán điều khiển [9, 11, 24, 37, 67, 69]. Tuy nhiên, CoAP và các bản cải tiến CoAP vẫn còn một số hạn chế sau:

(1) Hạn chế trong tính toán tham số.

(2) Chưa hỗ trợ chuỗi gói

(3) Hạn chế về điều khiển tốc độ để giảm tắc nghẽn

(4) Chưa phân biệt nguyên nhân mất gói

(5) Chưa phát hiện sớm tắc nghẽn

(6) Hạn chế trong tính toán băng thông cổ chai.

Ngoài ra, do hạn chế tài nguyên của các thiết bị IoT, cơ chế điều khiển cần gọn nhẹ, hiệu quả. Chi tiết cụ thể về các hạn chế này sẽ được trình bày trong Chương 1. Nghiên cứu giải pháp khắc phục các hạn chế trên là các vấn đề thách thức và là trọng tâm nghiên cứu của luận án này.

2. Mục tiêu nghiên cứu

Mục tiêu của luận án là nghiên cứu giải pháp khắc phục các hạn chế nêu trên và đề xuất cơ chế điều khiển tắc nghẽn hiệu quả cho giao thức CoAP để trao đổi tin cậy giữa các thiết bị đầu cuối trong mạng IoT. Các mục tiêu cụ thể gồm:

- Nghiên cứu xây dựng mô hình phân tích cho CoAP cho truyền tin theo chuỗi gói có tin cậy nhằm khắc phục hạn chế (1) và (2).
- Nghiên cứu đề xuất giải pháp điều khiển tăng/giảm tốc độ phát cho CoAP dựa vào phát hiện mất gói, trạng thái mạng và đường truyền nhằm điều khiển tắc nghẽn, nâng cao hiệu quả truyền tin nhằm giải quyết hạn chế (3) và (4).

- Nghiên cứu đề xuất giải pháp phát hiện sớm nguy cơ tắc nghẽn dựa vào biến động mạng, điều khiển tốc độ linh hoạt theo biến thiên động của trạng thái tắc nghẽn và môi trường truyền tin nhằm giải quyết hạn chế (5) và (6).

3. Đối tượng nghiên cứu

Đối tượng nghiên cứu là cơ chế điều khiển tắc nghẽn cho giao thức lớp ứng dụng CoAP, mô hình truyền tin theo chuỗi gói và tốc độ phát của CoAP.

4. Phạm vi nghiên cứu

Phạm vi nghiên cứu của luận án tập trung vào giao thức CoAP cài đặt trên tầng ứng dụng ở thiết bị IoT bên gửi và bên nhận tin với cơ chế điều khiển tắc nghẽn theo vòng kín, nghĩa là chỉ dựa trên thông tin trao đổi giữa các đầu cuối trong mạng IoT.

Mạng IoT đa dạng về thiết bị, công nghệ mạng và ứng dụng. Thiết bị IoT khác nhau về chủng loại, tính năng theo ứng dụng. Các thiết bị IoT được kết nối với nhau với đa dạng công nghệ mạng như WiFi IEEE 802.11, ZigBee, IEEE 802.15.4, Z-Wave, 6LoWPAN) [62]. Các ứng dụng của mạng IoT cũng rất đa dạng trong các lĩnh vực khác nhau [99]. Do CoAP ở tầng ứng dụng, phạm vi nghiên cứu của luận án không đề cập đến những vấn đề ở các tầng mạng khác cũng như nhiễu, lỗi vô tuyến ở tầng vật lý.

Ngoài ra, các kiến trúc mạng IoT cũng rất đa dạng. Tuy nhiên, kiến trúc mạng tập trung thông tin ICN (Information Concentric Networks) đã được chuẩn hóa trong [58, 60, 61, 99, 102] và là xu thế chủ yếu cho các ứng dụng IoT hiện nay [101], trong đó các thiết bị IoT thu thập dữ liệu từ môi trường để truyền về thiết bị đầu cuối phía Internet. Phạm vi nghiên cứu của luận án tập trung vào kiến trúc mạng ICN nói trên.

Lĩnh vực điều khiển tắc nghẽn mạng IoT khá rộng và đa dạng. Luận án chỉ tập trung vào cơ chế điều khiển tắc nghẽn của CoAP. Trong phạm vi nghiên cứu, luận án không đi sâu phân tích các vấn đề khác như: Tính toán độ phức tạp, tính toán tối ưu các tham số, chi phí năng lượng tiêu thụ, tác động của lỗi kênh vô tuyến, v.v. Đây là những hướng nghiên cứu tiếp trong tương lai.

5. Phương pháp nghiên cứu

Luận án sử dụng các phương pháp nghiên cứu sau:

- Khảo sát tài liệu kỹ thuật về CoAP và các bản đã chuẩn hóa, các nghiên cứu cải tiến liên quan để phân tích, đánh giá các tồn tại, hạn chế.
- Nghiên cứu lý luận, phân tích các phương pháp điều khiển tắc nghẽn để lựa chọn giải pháp.
- Mô hình hóa, phân tích mô hình truyền tin của CoAP để xây dựng cơ chế điều khiển phù hợp với thiết bị có hạn chế tài nguyên và môi trường IoT.
- Mô phỏng, kiểm chứng, so sánh, đánh giá kết quả nghiên cứu đạt được bằng cách sử dụng bộ công cụ mô phỏng NS3.

6. Định hướng nghiên cứu và các kết quả đóng góp của luận án

Luận án định hướng nghiên cứu vào giải pháp điều khiển tắc nghẽn mạng IoT với giao thức tầng ứng dụng CoAP. Các câu hỏi nghiên cứu đặt ra gồm:

- Những hạn chế, tồn tại trong cơ chế điều khiển tắc nghẽn của CoAP?
- Những đề xuất cải tiến CoAP hiện còn những nhược điểm gì?
- Làm thế nào để phát hiện sớm nguy cơ tắc nghẽn?
- Điều khiển tắc nghẽn khi nào, mức độ nào để hạn chế tắc nghẽn, duy trì hiệu năng cao trong điều kiện mạng và băng thông kênh truyền luôn biến động?

Luận án có các kết quả đóng góp chính như sau:

1) Đề xuất một mô hình phân tích truyền tin theo chuỗi gói cho CoAP và giao thức mới RCoAP dựa trên tốc độ để điều khiển tắc nghẽn trong mạng IoT với cơ chế điều khiển tăng giảm tốc độ phát phù hợp với tình trạng tắc nghẽn nhằm đạt được hiệu năng cao về độ trễ, thông lượng, tỷ lệ mất gói, tỷ lệ phát lại và tỷ lệ phát lại đúp so với các cơ chế CoAP hiện có. Mô hình phân tích truyền tin theo chuỗi gói được công bố trong [J1, J2], giao thức RCoAP được công bố trong [J3, J4].
2) Đề xuất giao thức mới FCoAP điều khiển tắc nghẽn sử dụng hệ điều khiển mờ theo biến thiên động của tình trạng tắc nghẽn và các tham số mạng nhằm

đạt được hiệu năng cao về độ trễ, thông lượng, tỷ lệ mất gói, tỷ lệ phát lại và tỷ lệ phát lại đúp so với các cơ chế CoAP hiện có trong điều kiện mạng biến thiên động, kể cả khi có tắc nghẽn nghiêm trọng. Đóng góp này được công bố trong [J5, J6].

7. Ý nghĩa khoa học và thực tiễn

Ý nghĩa khoa học:

- Xây dựng một mô hình phân tích truyền tin theo chuỗi gói cho CoAP. Mô hình có thể áp dụng để nghiên cứu tính toán định lượng tốc độ phát.
- Hầu hết các nghiên cứu khác về CoAP chỉ tập trung vào cải thiện các tham số thời gian quay vòng, định thời phát lại và cơ chế phát lại. Luận án đưa ra một cách tiếp cận mới để tính toán định lượng tốc độ phát, điều khiển tốc độ phát thay vì chỉ điều khiển tốc độ phát lại trong các nghiên cứu trước đó.
- Do các tham số liên quan đến tắc nghẽn biến thiên động và khó xác định chính xác, luận án đề xuất sử dụng logic mờ vào điều khiển tắc nghẽn cho CoAP. Đây là một cách tiếp cận mới tới nay. Hệ điều khiển mờ đã được chứng minh có nhiều ưu điểm như: đơn giản dựa trên các tham số không chính xác, không cần mô hình toán học chính xác, cho phép điều khiển nhanh [70, 72, 39].

Ý nghĩa thực tiễn:

Các cơ chế điều khiển đề xuất trong luận án có thể sử dụng cho các ứng dụng IoT trong thực tế:

- Các ứng dụng IoT truyền lưu lượng lớn theo luồng tin, chuỗi gói tin có độ trễ nhỏ, thông lượng cao theo thời gian thực, ví dụ như ứng dụng truyền luồng tin video từ các camera giám sát [93, 103], hay luồng gói tin liên tục theo các khối [99, 103, 69, 25].
- Các mạng cảm biến phục vụ mục đích giám sát, ví dụ ứng dụng truyền dữ liệu thu thập từ các cảm biến theo các chuỗi gói liên tục về trung tâm như đã nêu trong [10, 11, 5, 59, 57].

- Trao đổi thông tin máy – máy M2M (machine to machine) phục vụ cho các ứng dụng theo dõi, giám sát, cảnh báo về thảm họa như đã nêu trong [9, 115, 57, 59], các ứng dụng điều khiển từ xa trong y tế, các ứng dụng theo dõi và dự báo thời tiết như đã nêu trong [98, 52, 116, 59].

8. Bố cục của luận án

Ngoài phần mở đầu, phần kết luận và phần phụ lục, luận án gồm ba chương với bố cục như sau.

Chương 1 trình bày tổng quan về cơ sở lý thuyết mạng IoT và các khía cạnh liên quan đến tắc nghẽn và điều khiển tắc nghẽn. Các nội dung cụ thể gồm: khái niệm mạng IoT, các ứng dụng điển hình, mô hình kiến trúc mạng tập trung thông tin, một số giao thức lớp ứng dụng của IoT, vấn đề tắc nghẽn và điều khiển tắc nghẽn mạng IoT, giao thức CoAP và các nghiên cứu cải tiến CoAP, điều khiển mờ và khả năng áp dụng vào điều khiển tắc nghẽn mạng IoT, vấn đề nghiên cứu cần giải quyết trong luận án.

Chương 2 đề xuất một giao thức điều khiển tắc nghẽn dựa vào tốc độ đặt tên là RCoAP (Rate-based CoAP). Nội dung chính bao gồm: xây dựng mô hình phân tích cho CoAP để tính toán tốc độ phát chuỗi gói tin, giao thức RCoAP với cơ chế điều khiển RCoAP, các trạng thái hoạt động và các thuật toán điều khiển, tính toán hiệu năng RCoAP, mô phỏng và đánh giá hiệu năng RCoAP.

Chương 3 đề xuất một giao thức điều khiển tắc nghẽn dựa vào hệ điều khiển mờ đặt tên là FCoAP (Fuzzy CoAP). Nội dung chính bao gồm: phân tích sự biến thiên của các đại lượng tác động đến điều khiển tắc nghẽn, phân tích lựa chọn đầu vào và đầu ra cho hệ điều khiển mờ, thiết kế hệ điều khiển mờ, giao thức FCoAP với cơ chế điều khiển, các trạng thái hoạt động và các thuật toán điều khiển, tính toán hiệu năng FCoAP, mô phỏng và đánh giá hiệu năng FCoAP.

CHƯƠNG 1. TỔNG QUAN VỀ MẠNG IoT VÀ VẤN ĐỀ ĐIỀU KHIỂN TẮC NGHẼN

Nội dung Chương 1 trình bày tổng quan về mạng IoT và vấn đề điều khiển tắc nghẽn. Các nội dung được tập trung trình bày gồm: khái niệm về IoT, các ứng dụng IoT, mô hình kiến trúc mạng tập trung thông tin, một số giao thức lớp ứng dụng điển hình, vấn đề tắc nghẽn và điều khiển tắc nghẽn, những hạn chế của CoAP và các nghiên cứu CoAP liên quan, điều khiển mờ và khả năng áp dụng vào điều khiển tắc nghẽn, các vấn đề nghiên cứu cần giải quyết trong luận án.

1.1. Tổng quan về mạng IoT

1.1.1. Khái niệm về IoT

Sự ra đời của Internet đã là một cuộc cách mạng làm thay đổi thế giới, có tác động đáng kể đến mọi lĩnh vực của đời sống. IoT là sự phát triển kế tiếp với ý tưởng kết nối mọi vật với nhau và với Internet để tận dụng các thành quả của Internet.

Theo tài liệu [56], Tổ chức viễn thông thế giới ITU đã đưa ra định nghĩa ngắn gọn như sau: "*IoT là một cơ sở hạ tầng toàn cầu cho xã hội thông tin, cho phép các dịch vụ tiên tiến bằng cách liên kết các vật thể (vật lý hoặc ảo) dựa trên các công nghệ thông tin và truyền thông hiện có và tương lai. Thông qua việc khai thác khả năng định danh, thu thập dữ liệu, xử lý và giao tiếp, IoT tận dụng mọi thứ để cung cấp dịch vụ cho tất cả các loại ứng dụng*". Tổ chức IEEE [54] đưa ra định nghĩa: "*IoT là một mạng kết nối các vật thể được định danh duy nhất vào Internet. Các vật thể này có khả năng cảm biến/tương tác và lập trình được*". Mặc dù các định nghĩa khác nhau, song về bản chất IoT được hiểu là *mạng kết nối các vật thể (sau đây gọi chung là thiết bị) có khả năng thu thập, xử lý và trao đổi dữ liệu thông qua Internet.*

Mạng IoT hội tụ các xu hướng phát triển trong nhiều lĩnh vực khác nhau. Xu hướng hội tụ công nghệ tạo ra các đặc trưng sau đây của mạng IoT.

- Tính phổ quát (ubiquitous): Mọi vật thể đều có khả năng kết nối mạng mọi lúc, mọi nơi với chi phí thấp và cung cấp các dịch vụ liên quan đến vật thể.
- Sử dụng nền tảng IP: Giao thức IP cung cấp một nền tảng chung cho các

thiết bị kết nối, trao đổi, chia sẻ thông tin một cách dễ dàng với chi phí thấp.

- Tính kết nối liên thông: Mọi vật thể có thể kết nối với nhau qua mạng lưới và cơ sở hạ tầng thông tin.
- Tính đa dạng, không đồng nhất: Thiết bị đa dạng và không đồng nhất vì nó có phần cứng/phần mềm, liên kết mạng khác nhau.
- Tính đa dạng trong giao tiếp: Các vật thể giao tiếp máy-máy bên cạnh các tương tác truyền thống khác (người-máy, máy-người, người-môi trường).
- Tính đa dạng về công nghệ: Hạ tầng mạng đa dạng như mạng cảm biến, mạng cục bộ, mạng di động thế hệ mới, điện toán đám mây, v.v.
- Quy mô lớn: Số lượng rất lớn thiết bị giao tiếp với nhau, lớn hơn nhiều so với số máy tính kết nối Internet hiện nay. Lượng thông tin truyền rất lớn.

1.1.2. Các ứng dụng IoT

Với tính thông minh, hiệu quả thiết thực, IoT đã và đang được tích hợp trên khắp mọi thứ, mọi nơi con người sống. Từ chiếc vòng đeo tay, đồ gia dụng trong nhà, mảnh vườn ươm hạt giống, cho đến những sinh vật sống đều sử dụng giải pháp IoT [6, 101, 57]. Một số ví dụ điển hình cho các ứng dụng IoT trong thực tiễn:

- Đô thị thông minh: Dữ liệu thu thập phục vụ quản trị dịch vụ, hệ thống giao thông, bãi đỗ xe, đèn giao thông, chiếu sáng đô thị; hỗ trợ điều khiển.
- Ngôi nhà thông minh: Tự động hóa ngôi nhà qua thu thập thông tin về điều kiện hoạt động, phân tích dữ liệu; giám sát các hệ thống trong tòa nhà; điều khiển các thiết bị chiếu sáng, sưởi ấm, thông gió, v.v.
- Y tế thông minh: Theo dõi sức khỏe từ xa và thông báo khẩn cấp; giám sát cấy ghép đặc biệt; theo dõi sức khỏe người già, bệnh nhân, v.v.
- Công nghiệp, nông nghiệp thông minh: IoT kết nối nhà máy với quy trình sản xuất, điều khiển lưới các hệ thống công nghiệp, nông nghiệp; giám sát quy trình phát triển cây cối, truy xuất nguồn gốc sản phẩm, v.v.
- Giám sát thảm họa: Giám sát nguy cơ thảm họa và môi trường độc hại; giám sát núi lửa, cháy rừng, lũ lụt, động đất, v.v.

Ứng dụng rộng rãi của IoT dẫn đến số thiết bị kết nối và lượng dữ liệu cần chuyển tiếp ngày càng lớn. Điều này thúc đẩy phát triển các giao thức tầng ứng dụng để bảo đảm truyền tải dữ liệu hiệu quả, tránh mất mát, giảm độ trễ truyền tin [57, 102]. Nhu cầu của các ứng dụng rất đa dạng. Ví dụ, trong ứng dụng dự báo thời tiết, các bản tin thu được từ các giá trị cảm biến và truyền về trung tâm. Độ trễ thấp quan trọng hơn mất gói tin, có thể bỏ qua một số gói tin do chúng được cập nhật với giá trị cảm biến tiếp theo. Ngược lại, trong ứng dụng y tế, mất gói tin quan trọng hơn là trễ gói vì mỗi giá trị cảm biến biểu thị một trạng thái khá nhạy cảm đối với bệnh nhân. Ứng dụng giám sát thảm họa thì nhiều lúc đòi hỏi cả trễ thấp và tỷ lệ mất gói thấp.

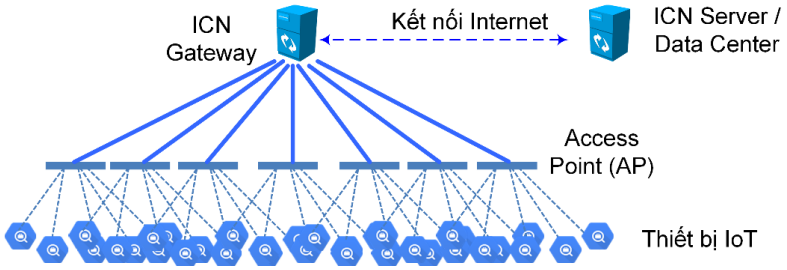
1.1.3. Mô hình kiến trúc mạng IoT

Theo tổ chức IEEE [54], kiến trúc cơ bản của mạng IoT gồm 3 lớp: 1) lớp cảm biến, 2) lớp liên kết mạng và truyền dữ liệu, 3) lớp ứng dụng. Lớp cảm biến là lớp vật lý hay lớp thu thập dữ liệu, tại đó các cảm biến thu thập thông tin về môi trường. Lớp liên kết mạng và truyền dữ liệu kết nối với các thiết bị khác, Internet và thiết bị đầu cuối, có vai trò xử lý và chuyển tiếp dữ liệu. Lớp ứng dụng cung cấp dịch vụ của ứng dụng tới người dùng, thực thi ứng dụng do mạng IoT cung cấp.

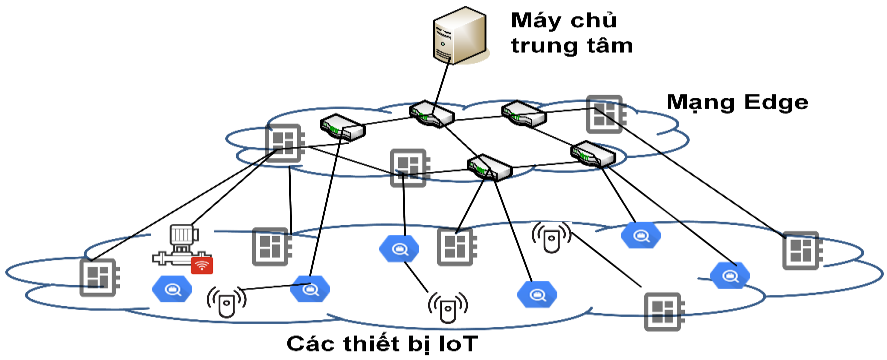
Kiến trúc ba lớp được mô tả trong RFC 7927 [102] và [58] theo một mô hình mạng tập trung thông tin ICN (Information Centric Networking). ICN được đề xuất làm hạ tầng mạng Internet mới cho phân phối thông tin [58, 61, 99, 102]. CoAP sử dụng kiến trúc ICN như đã chỉ ra trong tiêu chuẩn RFC 8763 (tháng 4/2020). Kiến trúc ICN lấy dữ liệu làm trung tâm [58, 57], trong đó các thiết bị IoT thu thập dữ liệu và chuyển qua các điểm truy nhập AP (Access point) hoặc Gateway để truyền về máy chủ ở trung tâm qua Internet [102, 57, 99]. Ứng dụng kiến trúc mạng ICN cho IoT đã được nghiên cứu trong một vài năm gần đây [102, 62, 99, 61]. Hình 1.1 mô tả kiến trúc mạng ICN điển hình cho các ứng dụng IoT thu thập dữ liệu phục vụ giám sát, xử lý dữ liệu tập trung tại trung tâm.

Trên hình 1.2, các điểm truy nhập AP và Gateway tạo thành mạng biên. Thiết bị IoT thu thập dữ liệu từ môi trường và truyền về máy chủ ở trung tâm qua các điểm

truy nhập / Gateway và kết nối Internet. Đây là mô hình kiến trúc ICN điển hình theo các tài liệu chuẩn [102, 99, 58] được sử dụng trong luận án.



Hình 1.1 Kiến trúc mạng ICN



Hình 1.2 Kiến trúc mạng ICN cụ thể cho các ứng dụng IoT

1.1.4. Tóm lược về các giao thức tầng ứng dụng của IoT

Hình 1.3 tóm lược các giao thức chính của mạng IoT. Các giao thức tầng ứng dụng và tầng vận chuyển đóng vai trò quan trọng [58]. Chi tiết về các giao thức ở nhóm hạ tầng mạng như định tuyến, IPv4/IPv6, 6LoWPAN, WiFi (802.11), ZigBee (802.15), LTE-A, Z-Wave có trong các tài liệu tiêu chuẩn [56, 57].

Giao thức tầng ứng dụng		MQTT	AMQP	XMPP	CoAP
Giao thức hạ tầng mạng	Giao thức vận chuyển	TCP, UDP			
	Giao thức tầng mạng	Định tuyến, IPv4/IPv6, 6LoWPAN			
	Giao thức liên kết	WiFi (802.11), ZigBee (802.15),…			
	Giao thức tầng vật lý	LTE-A, ZigBee, Z-Wave,…			

Hình 1.3 Sơ đồ tóm lược các giao thức chính của IoT

Phạm vi luận án chỉ tập trung vào giao thức tầng ứng dụng và giải pháp điều khiển tắc nghẽn từ đầu cuối tới đầu cuối. Vì vậy, luận án sẽ trình bày cụ thể hơn về các giao thức tầng ứng dụng. Các giao thức tầng ứng dụng cơ bản nhất đã chuẩn hóa

gồm: MQTT, AMQP, XMPP, và CoAP.

- **Giao thức MQTT**

MQTT [92] là một giao thức lớp ứng dụng hạng nhẹ, dùng để thu thập dữ liệu từ thiết bị IoT và gửi về máy chủ. MQTT tiêu thụ nguồn thấp, có thể truyền dữ liệu hiệu quả tới một số máy đích. MQTT dùng kiến trúc đẩy – kéo, nghĩa là quảng bá dữ liệu - đặt hàng dữ liệu. MQTT hoạt động trên tầng TCP nên hỗ trợ truyền tin cậy. Tuy nhiên, độ trễ gói tin có thể lớn do phải phát lại nhiều lần khi mạng nhiều lỗi. Mặt khác, TCP phải dùng gói tin bổ sung để thiết lập và duy trì kết nối, dẫn tới độ trễ tăng, tiêu thụ nguồn lớn hơn, mức độ rủi ro có lỗi khi truyền cao.

- **Giao thức AMQP**

AMQP [30] là một giao thức trao đổi bản tin hỗ trợ truyền tin cậy. AMQP chủ yếu hoạt động trong môi trường định hướng bản tin, bổ trợ cho giao thức truyền tải siêu văn bản HTTP (Hypertext Transfer Protocol). AMQP áp dụng cho nhiều kiểu kiến trúc mạng như: quảng bá, lưu trữ, chuyển tiếp, định tuyến bản tin. AMQP cũng dựa trên TCP, nên cũng có trễ gói lớn và hoạt động bị ảnh hưởng nhiều trong môi trường mạng có lỗi, mất gói cao.

- **Giao thức XMPP**

XMPP [53] là một giao thức truyền bản tin tức thời, được chuyển đổi từ truyền luồng tin dựa vào ngôn ngữ đánh dấu mở rộng XML (Extensible Markup Language) để dùng cho IoT. Nó cho phép trao đổi theo thời gian thực dữ liệu có cấu trúc giữa các vật thể. Thiết bị IoT kết nối với máy chủ và truyền các bản tin dựa vào luồng tin kiểu XML. XMPP sử dụng định dạng dữ liệu XML nên có tiêu phí mào đầu lớn. XMPP hoạt động dựa vào cả TCP và XML, nên nó có nhược điểm của TCP và không hiệu quả trong môi trường mạng IoT.

- **Giao thức CoAP**

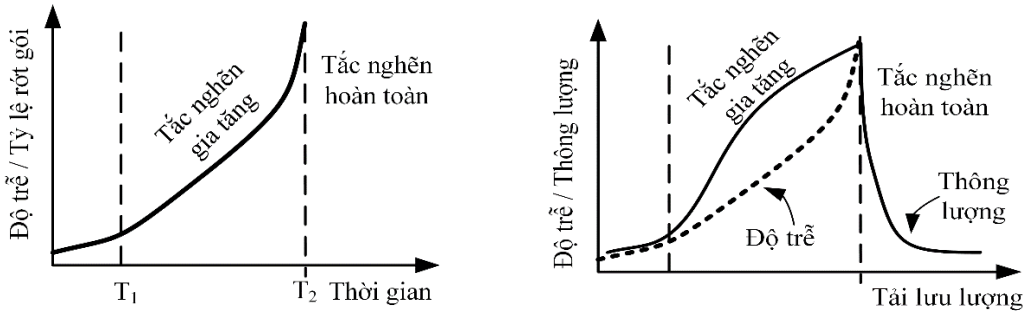
CoAP là giao thức lớp ứng dụng được tổ chức IETF chuẩn hóa trong RFC7252 [100] cho mạng IoT. CoAP trở thành chuẩn Internet năm 2014. Khác với các giao thức khác, CoAP hoạt động dựa trên UDP, có cấu trúc và tiêu đề gói tin đơn giản. CoAP là một giao thức hạng nhẹ được xây dựng dựa theo mô hình chuyển trạng

thái đại diện REST (Representational State Transfer) để sử dụng cho thiết bị có năng lực xử lý và tài nguyên hạn chế trong mạng IoT [100, 99, 117]. CoAP sử dụng cách thức gửi yêu cầu và phản hồi giữa các nút đầu cuối. Theo tài liệu RFC 7252 [100], CoAP có thể dễ dàng ánh xạ sang HTTP để biểu thị thông tin lên trang Web. CoAP cung cấp hai chế độ hoạt động: tin cậy và không tin cậy. Trong truyền tin cậy, bên gửi sẽ phát lại gói tin khi phát hiện gói tin đã gửi trước đó không đến đích, nghĩa là gói tin bị mất. Khi phải phát lại, độ trễ gói tin có thể lớn. Ngược lại, chế độ truyền không tin cậy không đòi hỏi phát lại gói tin khi có mất mát. Tỷ lệ mất gói tin có thể cao khi môi trường mạng có nhiều lỗi.

1.2. Tắc nghẽn và nguyên nhân tắc nghẽn

1.2.1. Khái niệm tắc nghẽn mạng

Tắc nghẽn là một hiện tượng phổ biến trên mạng với các dấu hiệu: độ trễ và tỷ lệ rớt gói tin tăng nhanh, thông lượng sụt giảm nhanh. Mạng Internet thiết kế theo cách lưu trữ và chuyển tiếp, nghĩa là các gói tin đến được lưu vào bộ đệm nút mạng, chờ xử lý để đưa ra khỏi nút mạng theo một tuyến đường đã chọn để đến đích. Nếu lượng gói tin đến càng lớn, thời gian nghẽn mạng càng kéo dài, số gói tin bị loại bỏ do hết chỗ lưu càng nhiều, dẫn đến nguy cơ mạng tê liệt hoàn toàn.



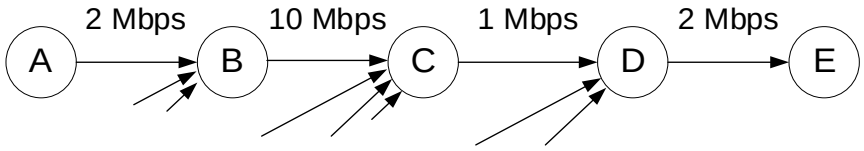
Hình 1.4 Mối quan hệ giữa các đại lượng a) theo thời gian, b) theo tải lưu lượng

Hình 1.4a biểu thị tình huống tắc nghẽn với độ trễ và tỷ lệ rớt gói theo thời gian. Hình 1.4b mô tả mối quan hệ giữa độ trễ, thông lượng tương ứng với tải lưu lượng (số gói tin gửi vào mạng). Tắc nghẽn xuất hiện trong khoảng $[T_1, T_2]$ với độ trễ, tỷ lệ rớt gói tăng nhanh với tải lưu lượng. Tại thời điểm T_2 mạng gần như tắc nghẽn hoàn toàn, độ trễ và tỷ lệ mất gói trở lên rất lớn. Khi tắc nghẽn hoàn toàn, tất cả các gói tin đều bị rớt.

1.2.2. Nguyên nhân tắc nghẽn mạng

Tắc nghẽn xảy ra do nhiều nguyên nhân, song chủ yếu là: 1) tốc độ phát của bên gửi vượt quá băng thông của liên kết mạng trong tuyến từ đầu cuối tới đầu cuối; 2) bộ đệm lưu trữ tạm thời gói tin để chuyển tiếp ở các nút trung gian bị quá tải dẫn đến tràn bộ đệm; 3) các nút mạng (kể cả nút trung gian và nút đích) không kịp xử lý các gói tin đến.

Hình 1.5 mô tả một ví dụ đơn giản với 4 đoạn liên kết. Đoạn A-B và D-E là các liên kết WiFi có băng thông 2 Mbps, đoạn B-C có băng thông 10 Mbps, đoạn C-D có băng thông 1 Mbps (do đoạn C-D có thể bị chia sẻ cho nhiều kết nối khác). Đoạn C-D được gọi là băng thông cổ chai (Bottleneck bandwidth). Bên gửi chỉ có thể phát với tốc độ 1 Mbps, bằng băng thông cổ chai. Nếu bên gửi phát quá 1 Mbps, tắc nghẽn sẽ xảy ra. Băng thông cổ chai cũng có thể bất kỳ đoạn nào tùy vào biến thiên tải lưu lượng. Vị trí băng thông cổ chai biến động, việc xác định băng thông cổ chai không đơn giản do số kết nối liên tục thay đổi ngẫu nhiên theo thời gian.



Hình 1.5 Ví dụ về trường hợp nghẽn cổ chai

1.2.3. Tắc nghẽn mạng IoT

Ngoài các nguyên nhân tắc nghẽn đã nêu ở 1.2.2, mạng IoT có thêm các nguy cơ gây tắc nghẽn khác do: 1) mất gói do lỗi kênh vô tuyến; 2) hạn chế tài nguyên của thiết bị IoT; 3) tính đa dạng của kết nối mạng; 4) yêu cầu băng thông của đa dạng ứng dụng; 5) khả năng xuất hiện chuỗi gói cao.

- ***Mất gói do lỗi kênh vô tuyến***

Khác với mạng Internet truyền thống, mạng IoT gồm nhiều chặng vô tuyến và phụ thuộc nhiều vào môi trường vô tuyến. Mất gói trong IoT có thể do nhiễu, lỗi kênh vô tuyến hoặc suy giảm tín hiệu vô tuyến. TCP [105] nhận biết tắc nghẽn kể cả khi có mất gói do lỗi kênh vô tuyến và giảm cửa sổ dẫn đến hiệu suất truyền thấp và tiêu phí tài nguyên vô tuyến. Nhiều nghiên cứu đã chỉ ra TCP có hạn chế trong

IoT, cần có cải tiến [45, 105, 103, 32, 112].

- ***Hạn chế tài nguyên của các thiết bị IoT***

Thiết bị IoT thường hạn chế về tài nguyên (bộ đệm, năng lực xử lý, băng thông, nguồn pin), dẫn đến nguy cơ tràn bộ đệm hoặc vượt tốc độ xử lý. Hạn chế về tài nguyên nên khó áp dụng các cơ chế phức tạp để xác định tắc nghẽn [65]. Mặt khác, cơ chế điều khiển cần đơn giản, gọn nhẹ, tiêu tốn ít tài nguyên và năng lượng, đạt hiệu quả cao để triển khai trên thiết bị IoT [7, 15, 86, 99].

- ***Tính đa dạng của kết nối mạng***

Kết nối mạng của các thiết bị IoT đa dạng với nhiều phân đoạn mạng. Số lượng kết nối gia tăng đột biến dẫn đến lượng dữ liệu truyền tải có thể rất lớn. Các tham số này biến thiên động khiến cho khó xác định tắc nghẽn [57, 112, 9, 85, 7].

- ***Yêu cầu băng thông đa dạng của các ứng dụng IoT***

Yêu cầu về băng thông của các ứng dụng IoT khá đa dạng. Kích cỡ gói tin trong mạng IoT nhỏ. Do vậy, việc xác định tắc nghẽn và cửa sổ tắc nghẽn khó khăn [11, 9, 75, 96].

- ***Khả năng xuất hiện chuỗi gói cao***

Lỗi hay nhiễu kênh vô tuyến dẫn đến gián đoạn kết nối trong những khoảng thời gian ngắn (ví dụ do suy giảm tín hiệu hoặc do vật cản chắn tạm thời) gây ra mất gói. Sau khi kênh truyền được khôi phục lại, khả năng xuất hiện các chuỗi gói cao do các gói tin chờ trong bộ đệm sẽ được phát lại liên tiếp. Chuỗi gói tin cũng xuất hiện khi có sự biến thiên của kết nối mạng, nghĩa là khi lượng gói tin dồn ứ ở nút trước đổ dồn về nút kế tiếp. Trong mạng IoT, sự biến thiên của băng thông và khả năng xuất hiện chuỗi gói tin cao là điều khó tránh khỏi [99, 57, 103].

1.3. Điều khiển tắc nghẽn

1.3.1. Điều khiển vòng hở và điều khiển vòng kín

Theo [68, 48, 45], các cơ chế điều khiển tắc nghẽn cho mạng Internet truyền thống chia thành hai nhóm: 1) Điều khiển vòng hở (có hỗ trợ của mạng) và 2) Điều khiển vòng kín (dựa vào trao đổi giữa hai đầu cuối, không có hỗ trợ của mạng). Điều khiển vòng hở không dựa vào thông tin phản hồi của bên nhận, mà chỉ dựa

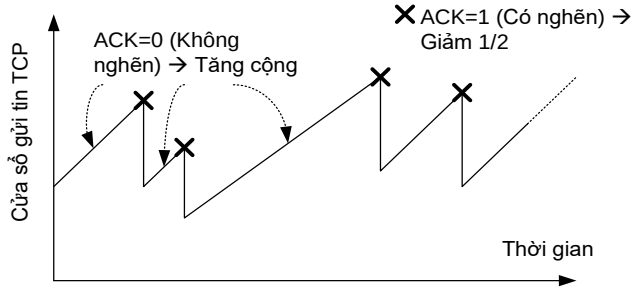
vào các nút trung gian. Nhóm này là dựa vào quản lý bộ đệm tích cực để tránh tràn bộ đệm [42], hoặc dựa vào tầng điều khiển truy cập [118, 110, 109] hay tầng liên kết [94, 39, 26, 16, 116]. Điều khiển vòng kín chỉ dựa vào trao đổi thông tin giữa bên gửi và bên nhận. Gói tin phản hồi ACK khẳng định đã nhận được gói tin gửi đi.

Có thể nhận thấy, cơ chế điều khiển tắc nghẽn của TCP, các giao thức tương tự TCP và của CoAP đều cũng thuộc nhóm điều khiển vòng kín.

1.3.2. Điều khiển dựa cửa sổ và điều khiển dựa tốc độ

- ***Điều khiển dựa cửa sổ***

Cơ chế điều khiển của TCP dựa vào cửa sổ. Cửa sổ *W* biểu thị số lượng gói tin tối đa được phát vào mạng. Các gói tin đang truyền trên mạng được gọi là gói *inflight*, là các gói tin đã phát đi song bên gửi vẫn chưa nhận được ACK cho chúng. Bên gửi cần phát sao cho tổng số gói tin *inflight* nhỏ hơn *W*. Cơ chế dựa cửa sổ của TCP là tăng cộng/giảm nhân (hình 1.6). *W* sẽ tăng cộng khi không có tắc nghẽn và giảm một nửa khi không có tắc nghẽn (không có mất gói xảy ra).



Hình 1.6 Cơ chế tăng cộng – giảm nhân của TCP

Các phiên bản của TCP đều dựa trên cơ chế cửa sổ, song có thêm các cải tiến cách tính RTT, RTO, cách xác định mất gói, v.v. [48, 3, 12]. Cơ chế dựa cửa sổ có hạn chế khi truyền chuỗi gói [84, 3, 7]. Cửa sổ *W* có thể tăng nhanh dẫn đến biến động trễ lớn, thông lượng giảm đáng kể [85, 68].

- ***Điều khiển dựa tốc độ***

Cơ chế dựa tốc độ phản ứng linh hoạt hơn với tắc nghẽn [51, 64, 68]. Cơ chế này đặt tốc độ phát ban đầu dựa vào tính toán băng thông cổ chai. Trong quá trình kết nối, tốc độ phát được điều chỉnh phù hợp với trạng thái mạng. Tốc độ phát có thể tăng nếu không thấy tắc nghẽn, giảm nhanh hoặc chậm tùy vào trạng thái mạng

và nguy cơ tắc nghẽn. Tốc độ phát được tính theo công thức dựa vào tỷ lệ mất gói, RTT, kích thước gói tin lớn nhất. Giao thức TCP dựa vào tích băng thông cổ chai và thời gian quay vòng BBR-TCP (Bottleneck Bandwidth and Round-trip time TCP) [27], các giao thức tương tự TCP [97, 44, 51] là ví dụ điển hình cho nhóm cơ chế dựa tốc độ. Khảo sát về cơ chế điều khiển dựa tốc độ cho mạng IoT được nêu trong [72, 85, 96]. Theo [27], các cơ chế dựa tốc độ có ưu điểm hơn cơ chế dựa cửa sổ.

1.3.3. Điều khiển tắc nghẽn mạng IoT

Các cơ chế điều khiển vòng kín trong mạng IoT được chia làm ba loại: 1) dựa vào mất gói, 2) dựa vào độ trễ, 3) dựa vào tốc độ [48, 68].

- ***Các cơ chế dựa vào mất gói***

Cơ chế dựa vào mất gói (loss-based) sử dụng mất gói làm chỉ báo tắc nghẽn. RTO được bên gửi đặt mỗi khi phát một gói tin mới. Khi quá thời gian RTO mà bên gửi vẫn chưa nhận được ACK thì bên gửi sẽ coi gói tin đã phát bị mất trên đường tới đích. Cần lưu ý là ACK cũng có thể mất trên đường quay về bên gửi. Mất gói cũng có thể phát hiện thông qua khoảng cách trong số thứ tự gói tin nhận được [49, 63, 42]. Cơ chế dựa vào mất gói được sử dụng phổ biến trong CoAP [100] và một số cải tiến của nó [14, 22, 77, 21]. Các cơ chế dựa vào mất gói chỉ kích hoạt khi xảy ra mất gói. Hạn chế lớn nhất của chúng là cần có cơ chế bổ sung để phát hiện sớm mất gói, tránh bị mất gói liên tiếp khi tắc nghẽn trở nên nghiêm trọng [42, 115].

- ***Các cơ chế dựa vào độ trễ***

Cơ chế dựa vào độ trễ (delay-based hoặc RTT-based) sử dụng thời gian trễ gói, RTT để dự đoán tắc nghẽn mạng. Độ trễ hoặc RTT càng lớn, tắc nghẽn càng nghiêm trọng. RTT được định nghĩa là thời gian từ lúc phát một gói tin cho đến khi bên gửi nhận được Độ trễ là thời gian từ lúc gửi gói tin đến thời điểm gói tin đó đến đích. Dựa vào độ trễ hoặc RTT, bên gửi tính toán nguy cơ tắc nghẽn mạng để hiệu chỉnh cửa sổ phát (số gói tin phát đi trong một khoảng thời gian) hoặc tải lưu lượng sau mỗi RTT. Cơ chế dựa độ trễ được dùng trong một số cải tiến của TCP như [64, 46, 47, 87, 81] và trong RTT-CoAP [9].

- ***Các cơ chế dựa vào tốc độ***

Cơ chế dựa vào tốc độ (rate-based) thực hiện đo lường RTT, tỷ lệ mất gói, thông lượng để dự đoán tắc nghẽn và hiệu chỉnh tốc độ phát để tránh tắc nghẽn. Cơ chế này tương đối giống cơ chế dựa vào độ trễ vì đều sử dụng RTT. Điểm khác biệt giữa chúng là đối tượng điều khiển. Cơ chế dựa độ trễ chỉ kiểm soát kích thước cửa sổ (window-based) [64], còn cơ chế dựa tốc độ thì điều khiển tốc độ phát [97, 44, 51, 90, 27]. Trong mạng IoT, cơ chế dựa tốc độ đã được đề xuất trong [10, 11, 69].

Cơ chế dựa tốc độ có ưu điểm về khả năng duy trì độ trễ thấp, thông lượng cao [10, 27, 93]. Khác với cơ chế dựa vào mất gói, cơ chế dựa tốc độ có khả năng phát hiện sớm tắc nghẽn và điều chỉnh tốc độ phát để tránh hoặc giảm bớt tắc nghẽn.

1.4. Điều khiển mờ và khả năng áp dụng cho điều khiển tắc nghẽn

1.4.1. Logic mờ

Zadeh [123] đưa ra lý thuyết mờ từ năm 1965 và được Mamdani [83] áp dụng vào hệ thống điều khiển thực tiễn. Kể từ đó, lý thuyết mờ đã được ứng dụng vào nhiều lĩnh vực. Theo lý thuyết này, logic mờ là một dạng logic nhiều giá trị khác với logic Bool chỉ dùng "0" và "1". Logic mờ xử lý nhiều trạng thái hơn, không chỉ những trạng thái rõ như "0/1", đúng/sai mà còn các trạng thái trung gian và sự chuyển dịch của chúng.

- ***Tập mờ***

Tập mờ là một khái niệm cơ bản của logic mờ. Tập mờ được định nghĩa là một lớp các đối tượng mà không có ranh giới rõ ràng giữa các đối tượng thuộc lớp đó hay lớp khác [17]. Zadeh [123] định nghĩa tập hợp A của tất cả các cặp (x, $\mu_A(x)$)

$$A = \{ x, \mu_A(x) \} \text{ với } x \in X \quad (1.1)$$

Trong đó, A gọi là tập mờ trên tập nền X; $\mu_A(x)$: $X \rightarrow \mathfrak{R}$ là một hàm thực được gọi là hàm thuộc; $\mu_A(x)$ của mỗi $x \in X$ được gọi là độ thuộc, $\mu_A(x) \in [0,1]$. Ví dụ, với tập nền $X = \{D_1, D_2, D_3\}$. Giả sử *"nhỏ"* là một tập mờ của biến độ trễ trên tập nền X và *"nhỏ" = {(D_1,0.95), (D_2,0.1)}*. Ta có:

$\mu_{nhỏ}(D_1) = 0.95 = 95\%$ nghĩa là D_1 có *95%* thuộc tập *"nhỏ"*.

$\mu_{nhỏ}(D_2) = 0.1 = 10\%$ nghĩa là D_2 chỉ có *10%* thuộc tập *"nhỏ"*, tức là phần

lớn D_2 không nhỏ.

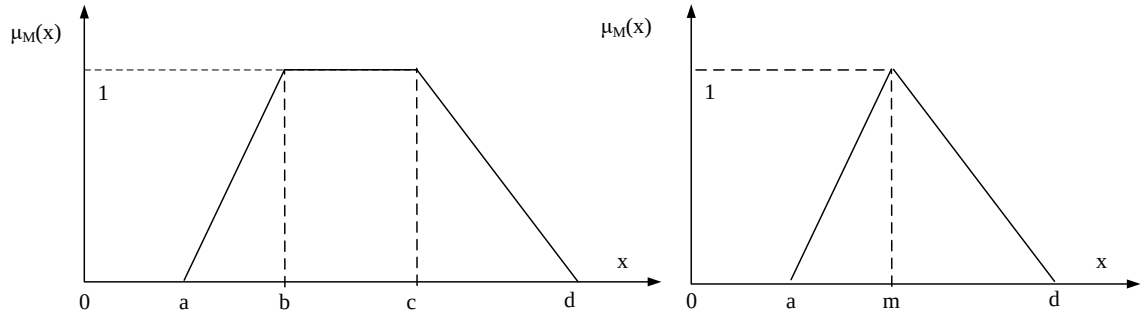
$\mu_{nhỏ}(D_3) = 0 = 0\%$ nghĩa là D_3 hoàn toàn không nhỏ.

- ***Các dạng hàm thuộc thường gặp***

Có nhiều cách biểu diễn hàm thuộc: có thể là hàm liên tục, rời rạc trong đó có thể tuyến tính, tuyến tính từng đoạn hoặc phi tuyến. Cách đơn giản là sử dụng các dạng tam giác, hoặc hình thang (hình 1.7).

Hình thang: $\mu_M(x) = max\left\{0; min\left\{1; \frac{x-a}{b-a}; \frac{d-x}{d-c}\right\}\right\}$ (1.2)

Hình tam giác: $\mu_M(x) = max\left\{0; min\left\{1; \frac{x-a}{m-a}; \frac{d-x}{d-m}\right\}\right\}$ (1.3)



Hình 1.7 Cách biểu diễn hàm thuộc a) hình thang, b) hình tam giác

Với hình thang, các điểm *a, b, c, d* biểu thị tọa độ trục *x* của một hàm thuộc $\mu_M(x)$ trong tập *M = (a, b, c, d)* như hình 1.7*a*, với *a* là biên trái, *d* là biên phải khi độ thuộc có giá trị *0*, còn *[b, c]* là một khoảng không đổi, tại đó độ thuộc có giá trị bằng *1*. Với hình tam giác, các điểm *a, m, d* biểu thị tọa độ trục *x* của một hàm thuộc $\mu_M(x)$ trong tập *M = (a, m, d)* như hình 1.7*b* với *a* là biên trái, *d* là biên phải khi độ thuộc có giá trị *0*, còn *m* là tọa độ của đỉnh tam giác khi độ thuộc có giá trị *1*.

- ***Toán tử mờ***

Có nhiều toán tử logic mờ, song trong phạm vi luận án, phần này chỉ trình bày hai toán tử quan trọng nhất được sử dụng là AND và OR. Toán tử AND là toán tử giao hay toán tử *min* biểu thị miền giao nhau của hai tập mờ *A* và *B*. Công thức phổ biến nhất của toán tử AND là:

$$A\ AND\ B = \{(x, \mu_{A\cap B}(x) | \mu_{A\cap B}(x) = min(\mu_A(x), \mu_B(x))\}\ \forall x\ \in X \quad (1.4)$$

Toán tử OR là toán tử hợp hay toán tử *max* biểu thị miền hợp nhau của hai tập

mờ *A* và *B*. Công thức phổ biến nhất của toán tử OR là:

$$A\ OR\ B = \{(x, \mu_{A\cup B}(x) | \mu_{A\cup B}(x) = max(\mu_A(x), \mu_B(x))\}\ \forall x\ \in X \quad (1.5)$$

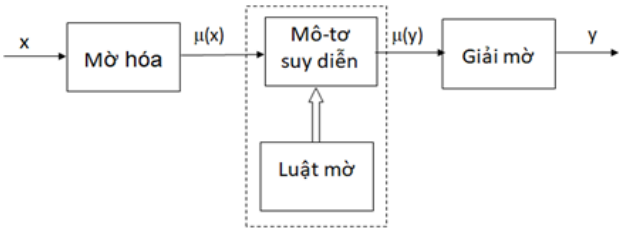
- ***Các biến ngôn ngữ***

Theo [126], các biến ngôn ngữ là các biến mà giá trị của chúng là từ ngữ hoặc câu theo kiểu ngôn ngữ tự nhiên. Một biến ngôn ngữ được đặc trưng bởi 3 phần chính *(V, X, T_V)* [1, 2], trong đó *V* là một biến ngôn ngữ trên tập tham chiếu *X*. Tập $T_V = \{A_1, A_2, ...\}$ gồm hữu hạn hoặc vô hạn các tập con mờ được chuẩn hóa của *X* dùng để đặc trưng cho *V*. A_i được ký hiệu đồng thời cho cả hạng thức ngôn ngữ (nhỏ, trung bình, lớn,…) và tập con mờ kết hợp với nó. Ví dụ, một biến ngôn ngữ *V* có tên là *"Độ trễ"* được xác định trên tập *X* là tập các giá trị thực của độ trễ. $T_{Độ\ trễ} = \{rất\ lớn, lớn, trung\ bình, nhỏ, rất\ nhỏ\}$, trong đó *"rất lớn", "lớn",* …là các tập mờ con A_i trên nền *X* xác định các giới hạn của những giá trị biến *V* lấy trong *X*.

1.4.2. Điều khiển mờ

- ***Cấu trúc chung***

Điều khiển mờ là một trong những lĩnh vực ứng dụng chính của logic mờ. Nó giúp con người có thể điều khiển hệ thống nhanh và đơn giản trên cơ sở các giá trị không chính xác của quá trình mà không cần biết chính xác mô hình toán học của quá trình. Nó có thể điều khiển cả những quá trình mà các thông số của nó rất khó hoặc phần nào không xác định được. Điều khiển mờ đặc biệt mạnh trong các hệ thống phi tuyến, hệ thống điều khiển mà các thông tin đầu vào hoặc đầu ra không đủ hoặc không chính xác. Thông tin chi tiết về các hệ điều khiển mờ có thể xem trong các tài liệu như: [123, 17] về các tập mờ, [124, 83] về các bộ điều khiển mờ, [31, 126] về độ thuộc và các hàm thuộc, [83, 126, 91] về quan hệ mờ, các biến ngôn ngữ, luật mờ, tiến trình mờ hóa, động cơ suy diễn và tiến trình giải mờ.



Hình 1.8 Mô hình hệ thống điều khiển mờ

Một hệ điều khiển mờ gồm 4 thành phần chính như trên hình 1.8 [83, 126]. Khối mờ hóa biến đổi các giá trị rõ đầu vào thành một hoặc nhiều tập mờ với hàm thuộc đã chọn ứng với các biến ngôn ngữ đã định nghĩa. Khối luật mờ bao gồm các tập luật *"nếu…thì"* dựa trên phép suy diễn từ các quan hệ mờ thích hợp với từng biến và các giá trị biến ngôn ngữ. Mô tơ suy diễn biến đổi các giá trị mờ hóa đầu vào thành các giá trị của biến ngôn ngữ đầu ra theo các luật mờ. Khối giải mờ biến đổi các giá trị mờ đầu ra thành các giá trị rõ để điều khiển đối tượng.

- ***Khối mờ hóa***

Khối này ánh xạ tập các giá trị thực $x \in R$ thành tập các giá trị mờ với hàm thuộc $\mu(x)$. Những giá trị rõ sẽ được sắp xếp vào tập mờ với những độ thuộc μ_i tương ứng. Các bước thực hiện gồm: 1) Định nghĩa các biến ngôn ngữ ứng với các tập mờ của đầu vào và đầu ra, 2) Xác định hàm thuộc của các tập mờ đã được định nghĩa, 3) Đọc độ thuộc từ những giá trị rõ đã biết.

- ***Khối luật mờ***

Luật mờ cơ sở là luật chứa một tập các luật ***"Nếu... Thì".*** Các loại hệ mờ gồm: 1) Một đầu vào x_0 và một đầu ra y (SISO - Single Input, Single Output); 2) Nhiều đầu vào x_i $(i=1..m)$ và một đầu ra y (MISO - Multiple Inputs, Single Output); 3) Nhiều đầu vào x_i $(i=1..m)$ và nhiều đầu ra y_k, $(k=1..n)$ (MIMO - Multiple Inputs, Multiple Outputs)). Với ***R*** là luật, ***A*** và ***B*** là các tập mờ, hệ mờ MISO có khối luật mờ (gồm r luật) dạng sau:

R_1: *Nếu* x_1 *là* A_{11} *và ... và* x_m *là* A_{1m} *thì y là* B_1

R_2: *Nếu* x_1 *là* A_{21} *và ... và* x_m *là* A_{2m} *thì y là* B_2

...

R_r: *Nếu* x_1 *là* A_{r1} *và ... và* x_m *là* A_{rm} *thì y là* B_r

Hệ mờ MIMO với m đầu vào, n đầu ra: Có thể tách thành n hệ, mỗi hệ có m đầu vào và 1 đầu ra. Tùy theo số đầu vào và số đầu ra, một hệ điều khiển mờ có thể là SISO, MISO hoặc MISO. Số lượng đầu vào và đầu ra là tùy vào mức độ phức tạp của hệ thống và yêu cầu của ứng dụng. Tuy nhiên để không phức tạp khi tính toán đầu ra hệ điều khiển mờ thường được lựa chọn phổ biến là MISO.

- ***Khối mô tơ suy diễn***

Mô-tơ suy diễn có nhiệm vụ dựa vào các tập mờ đầu vào và tập các luật mờ trong khối luật mờ tạo thành tập mờ đầu ra. Ký hiệu ⇒ nghĩa là ánh xạ tập mờ đầu vào thành tập mờ đầu ra theo các luật mờ sẵn có. Gọi *A* là tập mờ đầu vào; *B* là tập mờ đầu ra. Các suy luận $A \Rightarrow B$ được thực hiện qua các hàm thuộc $\mu_A(x)$, $\mu_B(y)$ và $\mu_R(x,y)$, *R* là quan hệ mờ giữa A và B. Có ba phương pháp suy diễn mờ phổ biến là max – min, max – prod và singlenton, trong đó phương pháp max – min được dùng phổ biến do đơn giản và tốc độ tính toán nhanh. Cách tính độ thuộc của kết quả theo phương pháp max – min như sau:

$$\mu_B(y) = max\{min[\mu_A(x)\mu_R(x,y)]\} \quad (1.6)$$

- ***Khối giải mờ***

Giải mờ là tiến trình tạo ra các giá trị rõ là đại lượng đầu ra của hệ điều khiển mờ từ các kết quả mờ do mô tơ suy diễn cung cấp. Có nhiều phương pháp để giải mờ như: phương pháp cực đại, phương pháp trọng tâm, phương pháp Singletons. Tuy nhiên phương pháp được dùng phổ biến nhất là phương pháp trọng tâm (CoG-Center-of-Gravity). Các giá trị rõ ở đầu ra được lấy theo điểm trọng tâm của hình bao bởi hàm thuộc của đầu ra và trục hoành theo công thức sau.

$$y_S = \frac{\int_S y\,\mu_B(y)dy}{\int_S \mu_B(y)dy} \quad (1.7)$$

Trong đó, *S* là miền xác định của tập mờ đầu ra *y*, y là các giá trị rõ của đầu ra. Để tránh phức tạp khi phải tính tích phân, phương pháp CoG có thể tính xấp xỉ bằng phương pháp tính trung bình tâm theo công thức sau với y_i là tọa độ của trọng tâm tập mờ thứ *i* trong tập hợp thành của đầu ra:

$$y_S = \frac{\sum_{i=1}^{N} y_i \cdot \mu(y_i)}{\sum_{i=1}^{N} \mu(y_i)} \quad (1.8)$$

1.4.3. Khả năng áp dụng điều khiển mờ cho điều khiển tắc nghẽn

Điều khiển mờ đã được áp dụng vào điều khiển tắc nghẽn trong nhiều nghiên cứu trước đây [70, 33, 65, 113, 29, 125] và một số nghiên cứu mới đây [98, 4, 35, 5]. Tắc nghẽn là một khái niệm không thực sự rõ ràng, có sự chồng lấp. Trạng thái

tắc nghẽn mạng có thể rất thấp, thấp, trung bình, cao, rất cao, v.v. Các tham số xác định tắc nghẽn đa dạng và có độ biến thiên cao, rất khó xác định rõ ràng. Khó có thể đưa ra một mô hình tính toán chính xác cho điều khiển tắc nghẽn trong điều kiện mạng động, biến thiên liên tục của các tham số liên quan như độ trễ, số lượng kết nối, lưu lượng, v.v. Tính không chắc chắn của các tham số như băng thông sẵn dùng, băng thông cổ chai, v.v. cũng tạo thêm khó khăn cho điều khiển. Bởi vậy, điều khiển mờ đã được nhiều nghiên cứu lựa chọn để điều khiển tắc nghẽn.

Keshav [70] đã chỉ ra thách thức trong ước tính tham số khi mạng biến động. Một bộ điều khiển mờ đã được đề xuất để ước tính các tham số chuỗi thời gian nhằm hiệu chỉnh tốc độ phát. Các tác giả [33] đề xuất cơ chế Fuzzy-RED dựa vào biến thiên của kích thước bộ đệm để dự đoán tắc nghẽn do tràn bộ đệm trong mạng. Bài báo [39] đề xuất sử dụng bộ điều khiển mờ để dự đoán tắc nghẽn cho TCP. Cơ chế điều khiển mờ trong [65] sử dụng mức độ chiếm dụng bộ đệm, tốc độ phát gói, số nút mạng tham gia để xác định mức độ tắc nghẽn mạng cảm biến không dây. Các tác giả [98] đề xuất kết hợp cơ chế quản lý bộ đệm tích cực với một bộ điều khiển mờ để hiệu chỉnh tốc độ phát cho từng nút trong mạng cảm biến không dây, sử dụng sự khác biệt về kích thước bộ đệm và biến thiên trễ tại mỗi nút mạng. Một hệ điều khiển mờ đề xuất trong [113] sử dụng số nút mạng tương tranh, tỷ lệ chiếm dụng bộ đệm của các nút ở chặng kế tiếp, tải lưu lượng của các luồng tin video qua mạng cảm biến không dây. Cơ chế điều khiển mờ trong [29] đề xuất định tuyến sử dụng kích thước bộ đệm và cửa sổ tắc nghẽn với cơ chế quản lý bộ đệm tích cực.

Một số nghiên cứu đề xuất ứng dụng điều khiển mờ để lựa chọn tuyến kết nối [125, 4, 52]. Trong [125], một bộ điều khiển mờ để định tuyến sử dụng cơ chế quản lý bộ đệm tích cực cho mạng cảm biến không dây. Mô hình điều khiển tách biệt các nút mạng bị lỗi khi định tuyến để giảm thiểu tắc nghẽn. Trong [4], một cơ chế RED sử dụng mức ưu tiên mờ được đề xuất để dự đoán mức tắc nghẽn tại từng nút mạng dựa vào xác suất rớt gói để tìm ra tuyến ít tắc nghẽn nhất. Cơ chế điều khiển mờ trong [52] sử dụng độ chiếm dụng bộ đệm tại từng nút, mức ưu tiên, tốc độ gói tin đến để dự đoán mức độ tắc nghẽn nhằm điều chỉnh tốc độ phát. Trong [35], một bộ

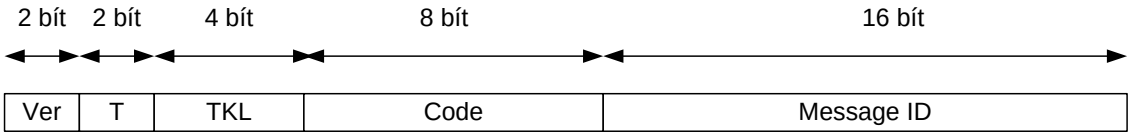
điều khiển mờ cho điều chỉnh tốc độ đã được đề xuất. Cơ chế này sử dụng mô hình hàng đợi để tính toán bộ đệm tại nút mạng nhằm dự đoán tắc nghẽn. Các tác giả sử dụng kích thước bộ đệm, tỷ lệ RTT/RTO để dự đoán trạng thái mạng và điều chỉnh tốc độ phát. Trong [5], một cơ chế điều khiển tắc nghẽn cho CoAP sử dụng logic mờ được đề xuất. Hệ điều khiển mờ sử dụng 3 đầu vào là: Tỷ lệ RTT, chu kỳ RTT và độ biến thiên RTT để dự đoán đầu ra RTO nhằm mục đích tính RTO cho cơ chế lùi, giảm thiểu số lần phát lại của CoAP. Cơ chế này có hạn chế do dựa vào mất gói và chưa điều chỉnh tốc độ phát.

Kết quả khảo sát đến nay cho thấy, điều khiển mờ đã được áp dụng cho điều khiển tắc nghẽn trong mạng Internet, song chủ yếu áp dụng trong quản lý bộ đệm tích cực tại các nút mạng. Cơ chế điều khiển mờ cho CoAP trong [5] chỉ phục vụ cho việc tính RTO nhằm mục đích phát lại. Vẫn chưa có nghiên cứu nào đề cập đến điều khiển mờ cho điều chỉnh tốc độ phát CoAP để tránh tắc nghẽn.

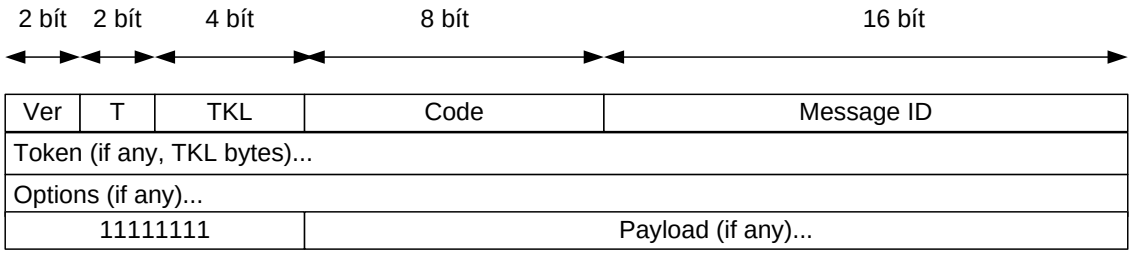
1.5. Giao thức CoAP

1.5.1. Hoạt động của CoAP

CoAP [100] là giao thức hạng nhẹ vì có phần tiêu đề gói tin đơn giản với kích thước cố định là 4 bytes. Giao thức sử dụng mô hình đẩy–kéo với 4 phương thức trao đổi cơ bản là: GET, POST, PUT và DELETE.



Hình 1.9 Tiêu đề cố định của gói tin CoAP



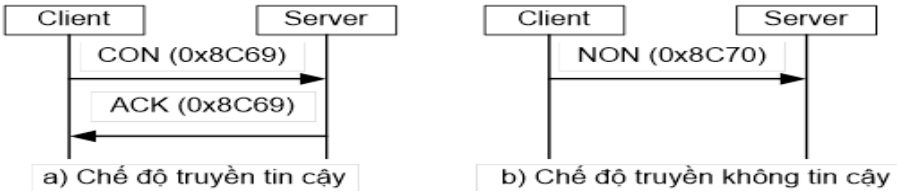
Hình 1.10 Cấu trúc một gói tin CoAP

Hình 1.9 biểu thị tiêu đề cố định của gói tin CoAP, hình 1.10 biểu thị một gói

tin CoAP đầy đủ các trường tin. Ý nghĩa các ký hiệu như sau: Ver (2 bit) là phiên bản CoAP; T (2 bit) biểu thị kiểu gói tin (00 = CON, 01=NON, 02=ACK, 03=RES); TKL (4 bit) là độ dài của trường thẻ (Token); Option là phần tùy chọn. CoAP có 4 kiểu gói tin gồm: CON (Confirmable) là gói tin có tin cậy; NON (Non-confirmable) là gói tin không tin cậy; ACK là gói tin phản hồi; RES (Reset) là gói tin báo hủy kết nối. CoAP có cấu trúc gọn nhẹ về quy tắc và cú pháp. Phần tiêu đề nhỏ nên có thể đạt hiệu quả truyền cao, tiêu tốn tài nguyên thấp. Do dựa trên UDP nên CoAP không cần thiết lập và duy trì kết nối, quá trình xử lý đơn giản, nhanh, phù hợp cho các ứng dụng IoT.

CoAP hỗ trợ hai chế độ truyền: Tin cậy và không tin cậy. Ở chế độ tin cậy, gói tin CON được phát đi và chuyển tới bên nhận. Bên gửi chờ phản hồi ACK từ bên nhận (hình 1.11a). ID 0x8C69 là ví dụ một mã định danh gói tin. Nếu bên gửi không nhận được phản hồi ACK sau RTO đặt trước, bên gửi sẽ thực hiện phát lại gói tin. Số lần phát lại tối đa có thể đặt trước, ví dụ bằng 4 [100]. Ngược lại, ở chế độ truyền không tin cậy, bên gửi không đòi hỏi phản hồi ACK từ bên nhận (hình 1.11b).

Hình 1.11 mô tả phương thức hoạt động của CoAP. Bên gửi phát đi một yêu cầu (Request) tới bên nhận bằng cách sử dụng phương thức GET hay POST. Bên nhận sẽ trả về phản hồi (Response) theo phương thức PUT.



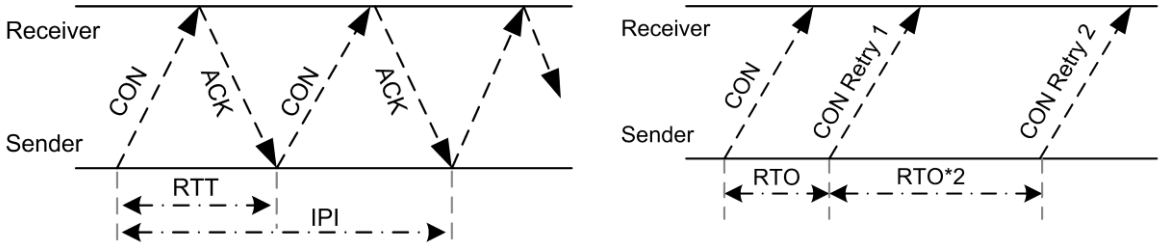
Hình 1.11 Ví dụ về chế độ truyền tin cậy (a) và không tin cậy (b)

1.5.2. Cơ chế điều khiển tắc nghẽn của CoAP

Do thiết kế hạng nhẹ, cơ chế điều khiển tắc nghẽn của CoAP khá đơn giản theo kiểu dừng và đi tiếp (Stop-and-Go). Dừng là bên gửi chờ phản hồi ACK sau khi đã phát một gói tin. Đi tiếp nghĩa là bên gửi chỉ phát tiếp gói tin sau khi đã nhận được ACK cho gói tin trước. Cơ chế truyền tin cậy này biểu thị trên hình 1.12.

Ở hình 1.12a, sau khi phát một gói tin, bên gửi sẽ chờ ACK. RTT là thời gian

quay vòng từ lúc gửi đến khi nhận ACK tương ứng. Sau khi nhận được ACK, bên gửi phát gói tin CON tiếp theo. IPI (Inter-packet interval) là thời gian giữa hai gói CON phát đi thành công. Hình 1.12b mô tả các lần phát lại của CoAP. Sau khi phát một gói CON, bên gửi đặt RTO để chờ gói ACK. Nếu thời gian chờ quá RTO mà vẫn chưa nhận được ACK, bên gửi sẽ phát lại gói CON với RTO mới gấp đôi RTO cũ. Quá trình lặp lại nếu bên gửi vẫn chưa nhận được ACK. Số lần phát lại tối đa có thể đặt trước [100] hoặc tùy ứng dụng. Cơ chế này được gọi là lùi theo hàm mũ nhị phân BEB (Binary Exponential Backoff).



Hình 1.12 a) Trao đổi gói tin CON và ACK, b) Các lần phát lại

Cơ chế BEB được giải thích như sau: Khoảng thời gian từ khi phát gói CON tới khi phát lại lần thứ nhất là RTO. Tốc độ phát lại là 1/RTO (1 gói tin trong một RTO). Ở lần phát lại thứ 2, RTO tăng gấp đôi, nghĩa là tốc độ phát lại sẽ giảm ½ sau mỗi lần phát lại. Đây là cách CoAP hạn chế tắc nghẽn, là giảm tốc độ phát lại.

1.6. Các nghiên cứu liên quan cải tiến CoAP và những tồn tại

1.6.1. Các nghiên cứu liên quan cải tiến CoAP

Các nghiên cứu liên quan cải tiến CoAP chia theo ba tính năng chính của CoAP là: 1) cách tính RTO, 2) cơ chế lùi, và 3) thuật toán điều khiển.

- ***Các nghiên cứu liên quan cải tiến cách tính RTO của CoAP***

Tham số RTO cố định của CoAP không phù hợp với điều kiện biến động của mạng. Do vậy, hầu hết các cơ chế cải tiến đều tập trung vào cách tính RTO động.

Trong [34], các tác giả đề xuất thay thế RTO cố định bằng việc nhân RTO với một hệ số ngẫu nhiên. Giá trị hệ số này nhỏ nên bên gửi có thể phát nhanh hơn mà không cần chờ lâu trong thời gian có tắc nghẽn. Tuy nhiên, điều này lại làm phát sinh mất nhiều gói. Các tác giả trong [77] đề xuất đưa một biến đếm số lần phát lại vào trường tiêu đề của CoAP để cập nhật RTO. Do RTO biến thiên theo RTT, một

cơ chế kết hợp giữa RTO nhanh và RTO chậm được đề xuất trong [66, 67] với cơ chế RTO nhanh - chậm FASOR (Fast-Slow RTO). RTO nhanh được tính tương tự RTO của TCP [104] khi các gói ACK trùng khớp với các gói CON. RTO chậm được đo từ thời điểm phát một gói đến khi nhận được ACK của nó, bất kể bao nhiêu lần phát lại. RTO nhanh dùng để dự đoán lỗi kênh, còn RTO chậm để dự đoán tắc nghẽn.

Độ chính xác của tính toán RTO phụ thuộc vào ước lượng đúng RTT. Trong [76], một bộ đếm được đưa vào trường tùy chọn của tiêu đề gói tin nhằm xác định đúng số hiệu gói có ACK để từ đó tính RTT chính xác hơn và cập nhật RTO. Các tác giả trong [95] đề xuất cách tính độ dốc RTT theo thời gian. Giá trị RTT của từng gói được dùng để dự báo tắc nghẽn. Cách này khó khả thi vì RTT biến động ngẫu nhiên cho từng gói tin. Tầm quan trọng của độ biến thiên RTT đã được nhìn nhận trong hầu hết các nghiên cứu. Đã có nhiều phương pháp đưa ra ước lượng phù hợp hơn cho độ biến thiên RTT [21, 71, 115], điển hình như trong một số nghiên cứu mới đây [67, 37, 117, 107, 38].

Giao thức CoCoA (Simple Congestion Control Advanced) [22, 23] là một cơ chế điển hình đưa ra một số cải tiến để khắc phục nhược điểm của CoAP. CoCoA đo liên tục RTT để tính RTO nhằm hạn chế tần suất phát lại, sử dụng RTT để tính RTO dựa theo thuật toán tính RTO của TCP [104]. Hai bộ ước lượng RTO được đề xuất là ước lượng *mạnh* và ước lượng *yếu*. Các bộ ước lượng này được cập nhật tương ứng với giá trị đo RTT mạnh hay yếu. RTT mạnh đo được khi bên gửi nhận được ACK ngay cho lần phát đầu tiên. RTT yếu đo được trong trường hợp nhận ACK sau ít nhất một lần phát lại. Các công thức sau được sử dụng để tính *RTO*:

$$RTTVAR_x = (1-\beta)RTTVAR_x + \beta\left|RTT_x - RTT_{x,new}\right| \quad (1.9)$$

$$RTT_x = (1-\alpha)RTT_x + \alpha RTT_{x,new} \quad (1.10)$$

$$RTO_x = RTT_x + K_x \times RTTVAR_x \quad (1.11)$$

$$RTO_{overall} = \gamma_x \times RTO_x + (1-\gamma_x) \times RTO_{overall} \quad (1.12)$$

Trong đó *RTTVAR* là giá trị trung bình ước tính của *RTT*, *x* là ký hiệu cho *mạnh* hoặc *yếu*, *new* biểu thị giá trị *RTT* đo được ở chu kỳ hiện thời, *overall* biểu thị ước

tính tổng thể, $\alpha = 0.25$ và $\beta = 0.125$ là các hệ số được chọn theo TCP [104], $K_x=4$ và $\gamma_x=0.5$ nếu *x* là *mạnh*, $K_x=1$ và $\gamma_x=0.5$ nếu *x* là *yếu* [22, 23]. $RTO_{overall}$ được sử dụng để khởi tạo RTO (RTO_{init}) cho lần phát gói kế tiếp. Thông thường RTO_{init} sẽ được chọn ngẫu nhiên trong khoảng từ $RTO_{overall}$ đến $1.5 \times RTO_{overall}$.

Hạn chế của CoCoA là kém hiệu quả cho chuỗi gói tin, việc tính toán bộ ước lượng yếu chưa rõ. Giá trị RTO mới có thể bị ước lượng quá lớn dẫn đến thời gian chờ phát lâu. Nếu giá trị RTO_{init} thấp hoặc vượt quá giá trị RTO định sẵn, có thể khiến cho các lần phát lại diễn ra liên tiếp chỉ trong một khoảng thời gian ngắn, dẫn đến tăng thêm tắc nghẽn mạng. Do vậy, cơ chế CoCoA-S trong [13] cải tiến bằng cách chỉ sử dụng bộ ước lượng *mạnh*. Tuy nhiên, CoCoA-S có thông lượng thấp hơn CoCoA. CoCoA+ [15] là cơ chế cải tiến nổi bật của CoCoA [22, 23] bằng cách thay đổi cách tính $RTO_{overall}$ như sau:

$$\begin{aligned} RTO_{overall} &= 0.25 * RTO_{weak} + 0.75 * RTO_{overall} \\ RTO_{overall} &= 0.5 * RTO_{strong} + 0.5 * RTO_{overall} \end{aligned} \quad (1.13)$$

Trong đó, RTO_{strong} là giá trị ước lượng *mạnh* và RTO_{weak} là giá trị ước lượng *yếu* cho RTO. Ngoài ra, CoCoA+ có thêm cơ chế tuổi thọ (aging) để đặt lại giá trị RTO nếu như RTT không được cập nhật mới sau một khoảng thời gian. Do thay đổi trọng số trong cách tính $RTO_{overall}$, ảnh hưởng của bộ ước lượng yếu ít hơn nhiều so với bộ ước lượng mạnh. Tuy nhiên, việc điều chỉnh này chỉ hiệu quả khi tải mạng ổn định. Tính toán các bộ ước lượng chỉ thực hiện cho lần phát đầu hoặc lần phát lại đầu. CoCoA+ cũng không đưa ra RTO phù hợp cho truyền chuỗi gói. Ước tính RTT không đúng có thể dẫn đến phát lại trùng lặp các gói tin. Các nghiên cứu trong [21, 117, 77, 11] đã chỉ ra CoCoA+ hoạt động thậm chí kém hơn đáng kể so với bản CoAP gốc [100] trong nhiều điều kiện mạng, đặc biệt khi có lưu lượng chuỗi gói.

Các tác giả trong [21] chỉ ra cách tính RTO chưa đúng trong các cơ chế hiện nay dẫn đến phát lại gói tin trùng lặp và đưa ra cách tính chính xác hơn cho RTO trong một cơ chế mới có tên là pCoCoA. Cơ chế này cũng sử dụng một bộ đếm số lần phát trong trường tiêu đề của CoAP tương tự như [77] để so khớp gói ACK với gói tin CON tương ứng, kể cả cho gói tin phát lại. pCoCoA đưa ra các cách khác

nhau để cập nhật RTT và giới hạn giá trị nhỏ nhất của RTO để giảm thiểu việc phát lại trùng lặp. Việc tính RTO của pCoCoA vẫn dựa theo cách tính của TCP song có sử dụng thêm trọng số cho biến RTT trong phép tính. Trong [37], các tác giả đề xuất cơ chế AdCoCoA với cơ chế ước tính cụ thể biến thiên của RTT nhằm cải thiện độ chính xác của RTO so với CoCoA+. Độ sai lệch tối đa của RTO được tính để tránh ảnh hưởng của biến thiên RTT. RTO được tính qua một hệ số biến đổi động phụ thuộc vào hiệu số giữa giá trị RTT đo được và RTT trước đó, qua đó tính được giá trị RTO đủ lớn để giảm thiểu phát lại. Vì tính toán RTO phụ thuộc vào việc ước tính hệ số biến đổi động, cơ chế này có hạn chế khi RTT biến thiên nhanh. Kết quả là RTO có thể rất lớn dẫn đến các khoảng chờ đợi dài. Nếu RTO lớn, bên gửi không thể nhận tiếp các ACK. Điều này lại dẫn đến tính toán sai RTT.

- ***Các nghiên cứu liên quan cải tiến cơ chế lùi của CoAP***

Cơ chế lùi của CoAP dựa trên điều chỉnh RTO cho phát lại nhằm giảm tắc nghẽn. RTO được đặt giá trị cố định mỗi khi gửi gói tin và được nhân đôi sau mỗi lần phát lại gói tin. Cải tiến cách tính RTO có thể dẫn đến RTO rất nhỏ hoặc rất lớn.

Để tránh giá trị khởi tạo RTO_{init} quá lớn dẫn đến thời gian chờ lâu hoặc quá nhỏ dẫn đến phát lại quá nhanh, CoCoA [13] sử dụng một hệ số lùi biến đổi VBF (Variable Backoff Factor) như sau:

$$VBF = \begin{cases} 3 & RTO_{init} < 1s \\ 2 & 1s \leq RTO_{init} \leq 3s \\ 1.3 & RTO_{init} > 3s \end{cases} \quad (1.14)$$

RTO được tính trong chu kỳ lùi như sau:

$$RTO_{backoff} = RTO_{init} \times VBF \quad (1.15)$$

Khi RTO_{init} lớn hơn 3s, hệ số VBF nhỏ sẽ không làm tăng thời gian chờ giữa các lần phát lại. Tuy nhiên, khi RTO_{init} nhỏ, hệ số VBF khiến phát lại quá nhanh trong khoảng thời gian ngắn gây thêm tắc nghẽn. Do đó, CoCoA+ [15] thay đổi cách tính VBF như sau:

$$VBF = \begin{cases} 2.5 & RTO_{init} < 1s \\ 2 & 1 \leq RTO_{init} < 3s \\ 1.5 & RTO_{init} \geq 3s \end{cases} \quad (1.16)$$

So với các giá trị [3, 2, 1.3] trong CoCoA [13], VBF trong CoCoA+ [15] thay đổi thành [2.5, 2, 1.5]. Ngoài ra, CoCoA+ bổ sung thêm biến tuổi thọ để làm mới lại RTO_{init} thành $RTO_{init} \times 16$ nếu như RTO_{init} liên tục nhỏ hơn *1* s sau *16* lần phát lại. Các tác giả [18] đề xuất CoCoA-4-State thay đổi VBF theo 4 trạng thái như sau:

$$VBF = \begin{cases} 1.1 & VBF_1\ trạng\ thái\ 1\ (Không\ phát\ lại) \\ 1.3 & VBF_2\ trạng\ thái\ 2\ (1\ lần\ phát\ lại) \\ 1.7 & VBF_3\ trạng\ thái\ 3\ (2\ lần\ phát\ lại) \\ 2.5 & VBF_4\ trạng\ thái\ 4\ (\geq 3\ lần\ phát\ lại) \end{cases} \quad (1.17)$$

VBF được gán trọng số theo từng trạng thái. Trạng thái CoCoA-4-State giảm đi 1 mỗi khi một gói phát thành công và tăng lên 1 khi một gói phải phát lại. Cách tính RTO cho cơ chế lùi cũng có tính mạnh yếu tương tự như CoCoA và CoCoA+.

Các giá trị tính VBF ở trên là theo thực nghiệm, có thể tốt trong một số trường hợp song lại kém hiệu quả ở các trường hợp khác. Đây cũng là điểm yếu của cách tính RTO trong cải tiến cơ chế lùi. pCoCoA [21] đưa ra cách tính chính xác hơn cho RTO, song cách tính vẫn chủ yếu tương tự như CoCoA+ với hệ số VBF.

Cơ chế lùi của FASOR [66, 67] đưa ra ba trạng thái chuyển đổi cho RTO: Fast/Fast-Slow-Fast/Slow-Fast. RTO nhanh (Fast) được tính bằng các mẫu RTT rõ (ít biến đổi), còn RTO chậm (Slow) được tính bằng các mẫu RTT không rõ (có biến đổi). Cơ chế này có hạn chế khi RTO chậm có khoảng dừng chờ của bên gửi. Các tác giả [95] đề xuất cơ chế CoCoA++ với việc thay thế hệ số VBF bằng hệ số xác suất PBF (Probability of Backoff Factor) như sau:

$$PBF = \begin{cases} 1.42 & P\_backoff > RandNo\ và\ g > 0 \\ 0.7 & Trường\ hợp\ khác \end{cases} \quad (1.18)$$

Trong công thức trên, *P_backoff* là giá trị xác suất của độ lệch giữa các giá trị trễ hoặc RTT, *RandNo* là một biến ngẫu nhiên giữa 0 và 1, g là độ lệch của các giá trị trễ. Cơ chế lùi chỉ thực hiện với RTO trong trường hợp hiệu số độ trễ (hay RTT) có giá trị dương, biểu thị bởi $g > 0$. CoCoA++ lùi với hệ số *PBF = 1,42* khi tắc nghẽn. Nếu không tắc nghẽn, giá trị $RTO = 0{,}7 \times RTO_{init}$. Như vậy, nếu tốc độ phát gói cao, nguy cơ phát lại sẽ rất cao dẫn đến tắc nghẽn thêm.

Các tác giả [5] đề xuất ước tính RTO để lùi với một trọng số tương đối được

tính dựa vào hiệu số giữa các RTT đo được theo các khoảng thời gian. Cơ chế lùi thực hiện với một hệ số lùi giới hạn thích nghi ABF (Adaptive-boundary Backoff Factor). Giá trị ABF được tính bằng một hệ logic mờ. RTO ở mỗi chu kỳ lùi được tính bằng tích số giữa RTO ở chu kỳ trước đó nhân với một hệ số RTO_{gain}. Hệ số này biểu thị tốc độ tăng của RTO sau mỗi chu kỳ lùi.

- ***Các nghiên cứu liên quan cải tiến thuật toán điều khiển***

RTT-CoAP [9] là một cơ chế điều khiển dựa vào độ trễ. Biến thiên RTT được giám sát để dự đoán trạng thái mạng có tắc nghẽn hay không, từ đó xác định tốc độ phát để tránh tắc nghẽn. RTT-CoAP định nghĩa 4 vùng lưu lượng theo số gói phát đi: 1) tắc nghẽn thấp, 2) hoạt động bình thường, 3) hoạt động trung bình, và 4) biến thiên lớn. Ứng với các vùng, tốc độ được điều chỉnh tăng cộng/giảm cộng AIAD (Additive Increase/Additive Decrease) theo mức: tăng nhanh/chậm hoặc giảm nhanh/chậm tùy vào trạng thái tắc nghẽn trong các vùng.

CoAP-R [10] là một cơ chế dựa vào tốc độ, sử dụng định tuyến theo hình nhánh cây, dựa vào thông tin ở mỗi nút của cây để phát hiện nghẽn cổ chai. Tắc nghẽn được phát hiện dựa vào tải lưu lượng kênh truyền hiện tại và trước đó, đồng thời dùng tỷ số của tốc độ phát trên băng thông cấp phát cho từng nút. Cơ chế điều chỉnh tốc độ là tăng nhân/giảm nhân. Cơ chế này hỗ trợ truyền chuỗi gói, có xét băng thông cổ chai.

BDP-CoAP [11] là một cơ chế dựa vào tốc độ, sử dụng cách tính băng thông cổ chai như BBR-TCP [27]. Dựa vào đó, BDP-CoAP điều chỉnh tốc độ phát và giữ cho số gói tin *inflight* không vượt quá băng thông cổ chai. Hệ số điều chỉnh tốc độ là *0,6* hoặc *0,2* nếu tần số phát lại tương ứng nhỏ hơn hoặc lớn hơn 20%.

CoAP-SC [69] điều chỉnh tốc độ phát dựa vào điều khiển luồng, xử lý lỗi truyền cho các dịch vụ truyền tin theo luồng. Số thứ tự gói được đưa vào tiêu đề gói tin. CoAP-SC phát hiện tắc nghẽn và điều chỉnh tốc độ luồng dựa vào kích thước bộ đệm và sự khác nhau giữa số thứ tự của gói gửi và gói nhận.

1.6.2. Những tồn tại của CoAP và của các nghiên cứu liên quan

(1) Hạn chế trong tính toán tham số điều khiển

CoAP có hạn chế là chỉ sử dụng RTO đặt cố định (từ 2-3 giây [100]). Nếu như gói ACK quay về chậm hơn so với RTO, bên gửi sẽ phát lại ngay khi hết định thời RTO. Điều này dẫn đến khả năng phát đúp (phát trùng lặp) gói tin [9, 15]. Mặt khác, RTO nhân đôi mỗi khi phát lại dẫn đến giá trị rất lớn. CoAP sẽ kém hiệu quả khi phải chờ ACK lâu, dẫn đến lãng phí băng thông kênh truyền [21, 23]. Sử dụng các tham số cố định, CoAP bỏ qua biến động RTT và trạng thái biến thiên của tắc nghẽn và mạng [38]. Do vậy, hầu hết các cơ chế cải tiến CoAP đều tập trung vào thay đổi cách tính RTO động thay vì dùng giá trị cố định.

Các nghiên cứu liên quan chủ yếu chỉ xét biến thiên của RTT, chưa xem xét biến thiên của các tham số khác như tải, thông lượng, tỷ lệ mất gói. Các tham số này phản ánh tình trạng tắc nghẽn, cần được nghiên cứu xem xét.

(2) Hạn chế về hỗ trợ chuỗi gói

Cơ chế dừng và đi tiếp của CoAP không hỗ trợ chuỗi gói tin. Điều này thể hiện qua tham số NSTART đặt cố định bằng 1 [100]. NSTART là tham số truyền vượt mức, biểu thị số gói tin được phép phát đi khi chưa có ACK quay về, nghĩa là các gói *inflight* (đang trên đường, chưa tới đích). NSTART=1 thể hiện chỉ 1 gói CON được phép trên đường. RFC 7252 [100] đã nêu hạn chế này của CoAP và chỉ ra giá trị NSTART >1 sẽ được nghiên cứu tiếp trong tương lai. Chuỗi gói tin có khả năng xuất hiện cao trong mạng IoT và cần được xử lý phù hợp [11, 24, 99, 103, 57]. Mặt khác, chuỗi gói tin có tác động lớn đến tính toán RTO [77].

Mới có rất ít nghiên cứu đề cập đến xử lý chuỗi gói tin. Cơ chế trong [24] đưa ra một lựa chọn cho truyền dữ liệu theo từng khối. Một cơ chế CoAP chuyển theo khối [25] đã được đề xuất chuẩn hóa bởi IETF. Tuy nhiên, các cơ chế này chỉ hoạt động cho chế độ truyền không tin cậy. Chúng không thực sự hỗ trợ truyền chuỗi gói mà chỉ chủ yếu phân chia các khối dữ liệu lớn cần truyền thành các khối nhỏ hơn. Cơ chế đề xuất trong [69] bổ sung số hiệu gói vào phần tiêu đề nhằm phát hiện mất gói và tạo ra các báo hiệu ACK cho nhiều gói tin. Cơ chế này điều khiển truyền dữ liệu theo luồng, song không đề cập đến chuỗi gói tin. Cơ chế điều khiển dựa vào tốc độ có khả năng hỗ trợ truyền chuỗi gói tin [10, 79, 93].

(3) Hạn chế về điều khiển tốc độ phát

Điều chỉnh tốc độ phát (tăng hoặc giảm) là rất cần thiết để tránh hoặc giảm tắc nghẽn. Tuy nhiên, CoAP chỉ giảm tốc độ phát một nửa mỗi khi phải phát lại với RTO cố định. Đa số các nghiên cứu liên quan cải tiến CoAP chỉ thay đổi RTO [117, 5, 36, 79], nghĩa là thay đổi tốc độ phát lại khi tắc nghẽn đã xảy ra.

Mới có một số ít nghiên cứu đề xuất điều chỉnh tốc độ phát [9, 10, 11, 69]. Cơ chế RTT-CoAP [9] đề xuất điều chỉnh tốc độ phát bằng tăng cộng/ giảm cộng tốc độ theo 4 trạng thái tắc nghẽn. Tuy nhiên, chưa nêu rõ cách xác định 4 trạng thái cũng như cách chọn hệ số tăng/giảm. Cơ chế CoAP-R [10] đề xuất điều chỉnh tốc độ phát bằng tăng nhân/ giảm nhân, song đòi hỏi có thông tin hỗ trợ từ các nút mạng. BDP-CoAP [11] tính toán băng thông cổ chai và số gói *inflight* để điều chỉnh tốc độ phát. Tuy nhiên, BDP-CoAP đưa ra hệ số cứng cho điều chỉnh tốc độ phát là *0,6* hoặc *0,2* tùy thuộc vào tần suất phát lại, dẫn đến không linh hoạt với biến động mạng và trạng thái tắc nghẽn. CoAP-SC [69] điều chỉnh tốc độ luồng tin video dựa vào số thứ tự gói tin và kích thước bộ đệm của bên gửi/bên nhận. Tốc độ phát luồng được điều chỉnh để số gói trong bộ đệm nhỏ hơn 1/3 kích thước bộ đệm. CoAP-SC chưa đưa ra cách tính tốc độ, đòi hỏi chèn kích thước bộ đệm vào tiêu đề gói tin gửi đi dẫn đến chi phí tiêu đề lớn.

(4) Hạn chế về phân biệt nguyên nhân mất gói

Trong mạng IoT, mất gói có thể do lỗi kênh vô tuyến ngoài lý do tắc nghẽn. Cần phân biệt nguyên nhân mất gói để có điều khiển phù hợp. Nếu mất gói tạm thời do lỗi kênh mà giảm tốc độ phát ngay sẽ dẫn tới suy giảm đáng kể hiệu năng. Đã có nhiều phương pháp phân biệt nguyên nhân mất gói với TCP trên Internet [19, 20, 28, 106, 119]. Có rất ít tài liệu đề cập đến phương pháp phân biệt nguyên nhân mất gói trong IoT. Tài liệu [32] phân tích các phương pháp phân biệt nguyên nhân mất gói dùng trong TCP cho mạng IoT. Cơ chế phát lại và lùi sẽ kém hiệu quả trong trường hợp lỗi kênh vô tuyến hoặc biến động mạng dẫn đến hiệu năng truyền tin của CoAP trở nên rất thấp với độ trễ gói tin lớn [37, 48, 66, 68, 75].

Cho tới thời điểm này, mới chỉ có tài liệu [67] đề cập cách sử dụng biến thiên

RTT và độ trễ của các gói tin liên tiếp để phân biệt mất gói do tắc nghẽn hay do lỗi kênh vô tuyến cho CoAP. Luận án sử dụng cách tương tự [67].

(5) Hạn chế về phát hiện sớm tắc nghẽn

CoAP và đa số các nghiên cứu cải tiến CoAP đều chỉ dựa vào mất gói để phát hiện tắc nghẽn, nghĩa là cơ chế điều khiển chỉ hoạt động khi đã có tắc nghẽn xảy ra. Còn rất ít nghiên cứu đưa ra tính năng phát hiện sớm tắc nghẽn cho CoAP. Cơ chế CoCo-RED [115] cho khả năng phát hiện sớm dựa vào quản lý bộ đệm, song cần hỗ trợ của các nút mạng trung gian. RTT-CoAP [9], CoAP-R [10] và BDP-CoAP [11] dựa vào RTT hoặc độ trễ để dự đoán tắc nghẽn. Tuy nhiên, các tham số RTT, độ trễ, tải lưu lượng, băng thông cổ chai liên tục biến thiên. Như đã nêu ở mục 1.4, các trạng thái tắc nghẽn không thực sự rõ ràng, có sự chồng lấp. Các tính toán RTT, RTO tới nay đều chỉ dựa vào thực nghiệm vì rất khó đưa ra một công thức tính chính xác cho các tham số [15, 7, 9, 85, 117].

(6) Hạn chế về tính toán băng thông cổ chai

Băng thông cổ chai có tác động lớn đến tắc nghẽn nên đã được nêu trong nhiều nghiên cứu về TCP [26, 27, 74, 64]. Tuy nhiên, còn rất ít nghiên cứu về CoAP đề cập đến băng thông cổ chai như CoAP-R [10] và BDP-CoAP [11]. CoAP-R tính băng thông cổ chai dựa vào thông tin từ các nút trên mạng, còn BDP-CoAP thì dựa vào công thức tính của BBR-TCP [27].

1.7. Các tham số đánh giá hiệu năng giao thức CoAP

Hiệu năng giao thức CoAP được đánh giá bằng các tham số cơ bản như sau:

(1) Số gói phát thành công và tỷ lệ phát thành công: Số gói tin phát thành công được lưu để tính tỷ lệ gói tin phát thành công. Tỷ lệ thành công là tỷ số giữa số gói tin nhận được ACK trên tổng số gói tin đã gửi.

(2) Số gói phát lại và tỷ lệ phát lại gói: Số gói tin phát lại được lưu để tính tỷ lệ gói tin phát lại. Tỷ lệ phát lại là tỷ số giữa số gói tin phát lại trên tổng số gói tin đã gửi. Một gói tin có thể phải phát lại 1, 2, 3 hoặc 4 lần (không phát lại sau 4 lần).

(3) Số gói phát lại đúp và tỷ lệ phát lại đúp: Số gói tin phát lại bị đúp được lưu

để tính tỷ lệ gói tin phát lại bị đúp (phát trùng lặp). Tỷ lệ phát lại đúp là tỷ số giữa số gói tin phát lại đúp trên tổng số gói tin đã gửi.

(4) Số gói đến đích thành công và tỷ lệ đến đích thành công: Là tổng số gói tin đã gửi trừ đi số gói bị mất. Tỷ lệ đến đích thành công là tỷ số giữa số gói đến đích thành công trên tổng số gói tin đã gửi.

(5) Số gói bị mất và tỷ lệ mất gói: Số gói tin bị mất và số gói tin đã gửi được lưu để tính tỷ lệ mất gói. Tỷ lệ mất gói là tỷ số giữa số gói tin bị mất trên tổng số gói tin đã gửi.

(6) Độ trễ gói tin (đơn vị là ms hoặc s): Độ trễ của gói tin được tính bằng hiệu số thời gian từ khi phát đi tại bên gửi đến khi nhận được tại bên thu.

(7) Thông lượng (đơn vị là gói/s hoặc bit/s): Thông lượng luồng tin là số gói tin nhận được trong một đơn vị thời gian, được tính bằng tốc độ nhận gói tin tại bên nhận. Thông lượng tức thời được tính trong một chu kỳ RTT.

Các giá trị trung bình được tính cho các tham số bằng cách tính trung bình cộng của các tham số theo tổng số gói đã gửi. Khoảng tin cậy được tính cho các giá trị trung bình theo công thức sau:

$$CI_X = X_{TB} \pm z\frac{\sigma_X}{\sqrt{N_X}} \qquad (1.19)$$

Trong đó, *CI* là khoảng tin cậy, X_{TB} là giá trị trung bình của tham số X cần tính khoảng tin cậy, z là mức tin cậy được chọn là 99%, σ là phương sai, N_X là số mẫu.

Các tham số nêu trên được sử dụng khi mô phỏng để đánh giá hiệu năng giao thức.

1.8. Kết luận chương 1

Trong chương 1, luận án đã trình bày tổng quan về mạng IoT và vấn đề tắc nghẽn mạng. Luận án đã khái quát về các ứng dụng IoT, phân tích và nêu mô hình kiến trúc mạng ICN đang được sử dụng phổ biến, nêu tầm quan trọng và tóm lược các giao thức tầng ứng dụng điển hình của IoT. Tiếp đó, luận án đã trình bày cơ sở lý thuyết về tắc nghẽn, điều khiển tắc nghẽn, điều khiển mờ và khả năng áp dụng cho điều khiển tắc nghẽn. Luận án đã phân tích ưu nhược điểm của các giao thức tầng ứng dụng và chỉ ra CoAP là một giao thức còn có những hạn chế, cần nghiên

cứu phát triển tiếp. Đã có nhiều nghiên cứu cải tiến CoAP, song vẫn còn 6 hạn chế cơ bản như đã nêu ở mục 1.6. Qua phân tích các điểm tồn tại của CoAP và các nghiên cứu cải tiến CoAP liên quan, có thể đưa ra định hướng nghiên cứu như sau.

Từ các hạn chế (1) và (2) đã nêu ở mục 1.6, cần phân tích mô hình truyền tin theo chuỗi gói tin cậy cho CoAP, tính toán các tham số theo tình trạng tắc nghẽn và biến thiên động của mạng. Mô hình truyền tin theo chuỗi gói sẽ giúp xác định cơ chế điều chỉnh tốc độ phát để khắc phục hạn chế (3). Để khắc phục hạn chế (4), có thể sử dụng phương pháp phân biệt nguyên nhân mất gói đã nêu trong [66, 67] kết hợp với cơ chế điều khiển tắc nghẽn CoAP.

Trong chương 2, luận án đề xuất giao thức RCoAP để khắc phục các hạn chế (1), (2), (3) và (4) đã nêu ở mục 1.6.

Để khắc phục hạn chế (5), chương 3 sẽ trình bày giao thức FCoAP dựa trên hệ điều khiển mờ nhằm giải quyết tính biến thiên động, không rõ ràng và có sự chồng lấp của các tham số. Mặt khác, chương 3 cũng đưa cách tính băng thông cổ chai vào hệ điều khiển mờ để khắc phục hạn chế (6) đã nêu ở mục 1.6. Giao thức FCoAP kết hợp các phương pháp để giải quyết các vấn đề hạn chế (1), (2), (3), (5) và (6).

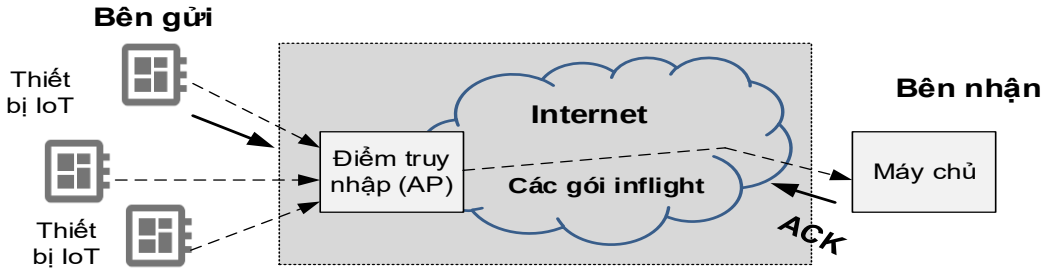
CHƯƠNG 2. MÔ HÌNH TRUYỀN CHUỖI GÓI VÀ GIAO THỨC RCoAP ĐIỀU KHIỂN TẮC NGHẼN DỰA VÀO TỐC ĐỘ

Nội dung chương này trình bày đóng góp thứ nhất của luận án: mô hình phân tích truyền tin theo chuỗi gói tin cậy và giao thức điều khiển tắc nghẽn dựa vào tốc độ RCoAP (Rate-based CoAP). Hạn chế về hỗ trợ chuỗi gói, tính toán các tham số có biến thiên của CoAP và các nghiên cứu liên quan đã chỉ ra sự cần thiết phải đưa ra một mô hình phân tích truyền tin theo chuỗi gói tin cậy cho CoAP. Mô hình này sẽ giúp tính toán các tham số điều khiển tắc nghẽn tốt hơn là sử dụng tham số cố định trong CoAP. Mô hình phân tích được đưa vào một giao thức mới có tên là RCoAP nhằm điều chỉnh tăng/giảm tốc độ phát và điều khiển tắc nghẽn tương ứng với trạng thái tắc nghẽn. Ngoài ra, RCoAP được bổ sung cơ chế phân biệt nguyên nhân mất gói để tăng hiệu quả truyền tin. Phần cuối chương thực hiện tính toán hiệu năng RCoAP và trình bày các kết quả mô phỏng cho RCoAP, so sánh hiệu năng RCoAP với các giao thức CoAP khác.

2.1. Mô hình phân tích cho truyền chuỗi gói tin cậy với CoAP

2.1.1. Sơ đồ luồng tin kết nối đầu cuối của CoAP trong mạng IoT

Hình 2.1 mô tả sơ đồ một luồng tin có kết nối giữa hai đầu cuối sử dụng giao thức CoAP trong mô hình kiến trúc ICN của mạng IoT. Các thiết bị đầu cuối IoT kết nối tới các điểm truy nhập và Gateway để truyền dữ liệu về máy chủ trung tâm qua Internet.



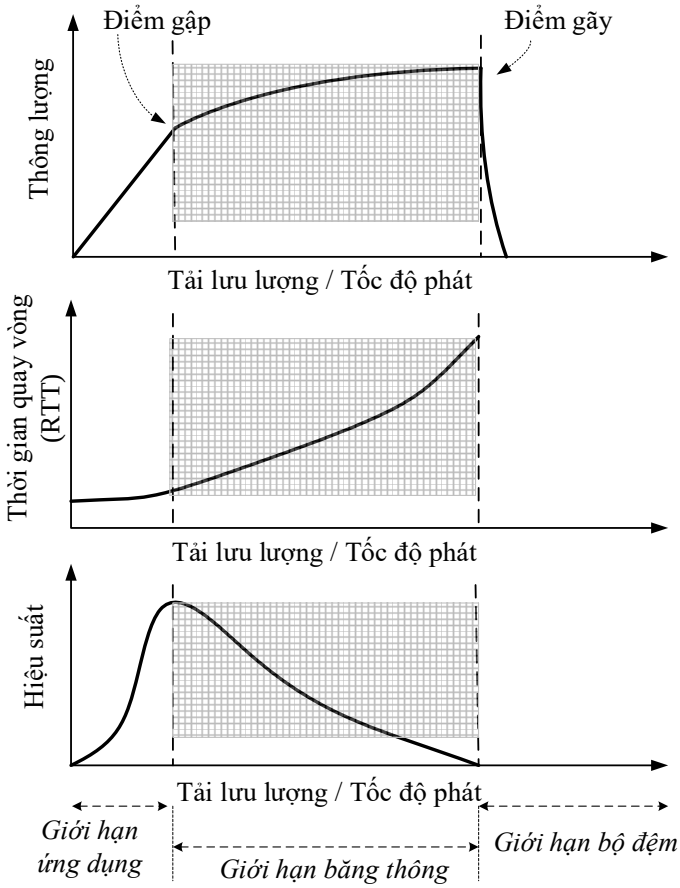
Hình 2.1 Sơ đồ luồng tin kết nối đầu cuối của CoAP trong mạng IoT

Các thiết bị IoT có thể phát các gói tin theo chuỗi. Luồng tin giữa bên gửi và bên nhận gồm các gói tin *inflight* – tức là các gói đang di chuyển trên mạng, chưa

tới đích. Khi nhận gói tin, máy chủ phản hồi với gói tin ACK. Nếu gói tin bị mất (không tới đích), bên gửi sẽ không nhận được ACK từ bên nhận. Do các giao thức tầng ứng dụng như CoAP chỉ quan tâm đến kết nối giữa hai đầu cuối, mạng có thể coi như một hộp đen (hình chữ nhật đậm trên hình 2.1). Mô hình hộp đen phù hợp với mô tả các mạng hỗn hợp giúp cho việc phân tích dễ dàng hơn [64].

2.1.2. Mô hình điều khiển tắc nghẽn cho CoAP

Mô hình điều khiển tắc nghẽn cho CoAP được đặc trưng bởi các đại lượng chủ yếu gồm: thời gian quay vòng RTT, thông lượng, tải lưu lượng và tốc độ phát.



Hình 2.2 Mối quan hệ giữa các đại lượng điều khiển (dựa theo [64,27])

Hình 2.2 biểu thị mối quan hệ hình thức giữa các đại lượng đặc trưng nêu trên. Trục hoành biểu thị tải lưu lượng hoặc tốc độ phát gói. Trục tung biểu thị thông lượng, trễ RTT, hiệu suất mạng. Điểm gập là nơi bắt đầu có nguy cơ tắc nghẽn. Điểm gẫy là nơi tắc nghẽn toàn cục xảy ra [64]. Tốc độ phát tăng đồng nghĩa với tải lưu lượng tăng. Theo [64, 27], trạng thái mạng được mô tả với ba vùng: 1) giới hạn

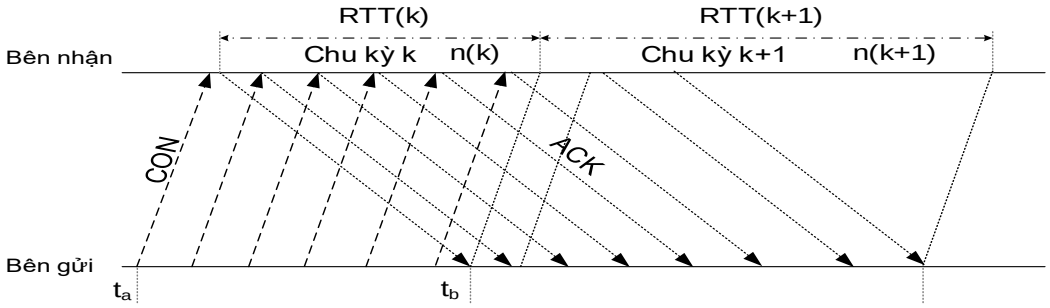
ứng dụng, 2) giới hạn băng thông và 3) giới hạn bộ đệm. Vùng 1 đặc trưng với tốc độ phát nhỏ, thông lượng tăng tuyến tính với tải và tốc độ phát, RTT tương đối ổn định. Hiệu suất gia tăng theo tải/tốc độ. Ở vùng 2, thông lượng bắt đầu tăng chậm. RTT tăng nhanh đáng kể do dồn ứ gói tin. Hiệu suất suy giảm khi tăng tải. Ở điểm gẫy vùng 3, các gói tin bị loại bỏ do tắc nghẽn. Thông lượng giảm nhanh về 0. RTT tăng rất lớn. Mạng tắc nghẽn toàn cục. Cơ chế điều khiển dựa vào mất gói hoạt động ở rìa trái điểm gẫy, còn cơ chế dựa vào độ trễ hay tốc độ hoạt động ở vùng giới hạn băng thông. Điểm gập là điểm tối ưu về lý thuyết, song không thể xác định do biến động của mạng [27]. Cơ chế điều khiển tắc nghẽn đề xuất trong luận án hoạt động trong vùng giới hạn băng thông (kẻ sọc).

Bảng 2.1: Các ký hiệu cho mô hình phân tích

Tên ký hiệu	Ý nghĩa
k-1, k, k+1	Ký hiệu chu kỳ trước *k-1*, chu kỳ thứ *k*, chu kỳ tiếp theo *k+1*
R(k)	Tốc độ phát gói tin của một bên gửi trong chu kỳ *k*
α	Hệ số điều khiển
ρ	Hệ số sử dụng
$R_r(k)$	Tốc độ xử lý của bên nhận trong chu kỳ *k*
n(k), n(k+1)	Lượng gói tin tích lũy trong chu kỳ *k*, chu kỳ *k+1*
T(k)	Thời gian của một chu kỳ *k*
RTT(k)	Thời gian quay vòng của chu kỳ *k*
Q(k)	Lượng gói tin bên gửi phát đi trong chu kỳ *k*
S	Số gói tin inflight
B(S)	Số gói tin được xử lý tại bên nhận khi không có mất gói
C(S)	Số gói tin được xử lý tại bên nhận khi có mất gói
U(S)	Hàm hiệu suất, là hàm số của S
$\theta(S)$	Hàm thông lượng, là hàm số của S
D(S)	Hàm trễ quay vòng, là hàm số của S

2.1.3. Tính toán tốc độ phát chuỗi gói tin của CoAP

Hình 2.3 mô tả các chu kỳ phát tin của bên gửi. Luồng tin CoAP gồm chuỗi các gói CON và ACK mô tả bằng một mô hình thời gian rời rạc. Đây là mô hình thường sử dụng cho phân tích hiệu năng Internet với TCP [70, 74]. Mô hình này áp dụng được cho hệ thống có quyết định điều khiển tạo ra ở các khoảng thời gian rời rạc. Mô hình CoAP trong luận án khác với mô hình TCP trong [70, 74, 64] ở các đặc điểm: CoAP dựa vào tốc độ và các gói tin *inflight* khác với TCP dựa vào cửa sổ; CoAP phát hiện mất gói với 1 gói ACK trong khi TCP với 3 gói tin ACK.



Hình 2.3 Các chu kỳ phát gói tin theo chuỗi của CoAP

Mỗi chu kỳ *k* kéo dài một khoảng *T(k)*, bằng RTT. Trong chu kỳ *k*, bên gửi phát đi liên tục các gói với tốc độ *R(k)*, tốc độ xử lý (tốc độ nhận gói) của bên nhận là $R_r(k)$. Số lượng gói do bên gửi phát đi trong chu kỳ *k* được tính như sau:

$$Q(k) = R(k) \times T(k) \tag{2.1}$$

Trong số gói tin phát đi, có $R_r(k) \times T(k)$ gói được bên nhận xử lý (máy chủ nhận được và phản hồi với ACK). Lượng gói tin tích lũy trong chu kỳ kế tiếp k+1 sẽ là:

$$n(k+1) = n(k) + R(k) \times T(k) - R_r(k) \times T(k) \tag{2.2}$$

Từ đó ta có: $$R(k) = R_r(k) + \frac{\Delta n}{T(k)} \tag{2.3}$$

Với $$\Delta n = n(k+1) - n(k) \tag{2.4}$$

Theo [70], một hệ số sử dụng ρ được định nghĩa như sau:

$$\rho = \frac{R}{R_r} \tag{2.5}$$

Hệ thống ổn định nếu như $\rho \leq 1$, nghĩa là $R \leq R_r$. Điều đó nghĩa là tốc độ phát của bên gửi phải nhỏ hơn hoặc bằng tốc độ xử lý của bên nhận để hệ thống ổn định. Nói một cách khái quát, điều kiện đã nêu là để tránh tắc nghẽn. Trường hợp $R > R_r$,

tắc nghẽn sẽ xảy ra. Xét tốc độ phát tối đa *R(k-1)* ở chu kỳ *k-1* trước đó với điều kiện tránh tắc nghẽn, ta có $R(k-1) = R_r(k)$. Thay vào (2.3), ta được:

$$R(k) = R(k-1) + \frac{\Delta n}{T(k)} \quad (2.6)$$

Hiệu số *Δn* có thể âm hoặc dương. *Δn* âm nếu số gói *inflight* ở chu kỳ *k+1* nhỏ hơn ở chu kỳ *k* và ngược lại. Tỷ số *Δn/T(k)* biểu thị lượng tăng/giảm tốc độ phát sau mỗi chu kỳ *k*. Dễ dàng nhận thấy, trường hợp *Δn* âm thì điều kiện $R<R_r$ đạt được và tắc nghẽn không xảy ra. Do vậy, trường hợp *Δn* dương được quan tâm hơn, nghĩa là tắc nghẽn có thể xảy ra khi tăng tốc độ để phát chuỗi gói.

Mặt khác, cần nhắc lại tính chất cơ bản của CoAP là chỉ phát tiếp 1 gói tin sau khi nhận được ACK, nghĩa là sau khi hết chu kỳ của gói tin trước đó. Để duy trì tính chất này, số gói tin phát thêm sau mỗi chu kỳ *T(k)* cần không vượt quá 1, nghĩa là giá trị tăng thêm của *Δ*n chỉ có thể là *1*. Ta viết lại được (2.6) như sau.

$$R(k) = R(k-1) + \frac{1}{T(k)} \quad (2.7)$$

Công thức (2.7) biểu thị khả năng điều chỉnh tăng tốc độ phát để có thể truyền chuỗi gói tin trong điều kiện tránh tắc nghẽn. Tuy nhiên, theo hình 2.2, tốc độ phát tăng dẫn đến nguy cơ tắc nghẽn gia tăng. Khi có tắc nghẽn, tốc độ phát cần phải giảm. Cơ chế giảm nhân được trình bày trong phần tiếp sau đây.

Bên gửi coi mạng là một hộp đen và tương tác với bên nhận chỉ thông qua phát gói tin và nhận phản hồi ACK. Trong [74], Kleinrock đã chỉ ra khả năng mô hình hóa một kết nối đầu cuối tới đầu cuối dưới dạng một đường ống vật lý. Đường kính của ống biểu thị băng thông cổ chai, nghĩa là tốc độ phát lớn nhất được phép. Chiều dài ống mô tả trễ truyền tin. Giả sử mô tả đường ống dưới dạng một hình chữ nhật trên mặt phẳng với độ rộng *Y* là băng thông cổ chai, chiều dài *X* là trễ quay vòng. Tích số $X \times Y$ sẽ là số gói tin tối đa mà bên gửi có thể phát đi (số gói tin *inflight*).

Để mô tả các mối quan hệ giữa thông lượng, trễ quay vòng và số gói tin *inflight* như trên hình 2.2, ta định nghĩa một hàm hiệu suất từ ý tưởng của Jain [64] như sau:

$$U(S) = \frac{\theta(S)^{\partial}}{D(S)} \quad (2.8)$$

Trong đó, *S* là số gói tin *inflight*, *U(S)* là hàm hiệu suất theo *S*, $\theta(S)$ là hàm

thông lượng theo *S*, *D(S)* là hàm trễ quay vòng theo *S*, ∂ là hệ số điều khiển, $\partial > 0$. Trễ quay vòng là trễ RTT, là thời gian phát gói đến khi bên gửi nhận ACK cho gói. Mục đích sử dụng hàm mũ là để mô tả đặc tính phi tuyến của các tham số. Hàm hiệu suất biểu thị mối quan hệ giữa thông lượng và trễ RTT theo biến *S* (xem hình 2.2). Thông lượng được định nghĩa là tỷ số của số gói được bên nhận xử lý trong một đơn vị thời gian. Tỷ số này chính là tốc độ xử lý gói tin (tốc độ nhận gói) của bên nhận.

Mặc dù công thức (2.8) gần tương tự với công thức hàm hiệu suất trong [64], cần lưu ý là công thức (2.8) khác với công thức hàm hiệu suất trong [64] như sau:

- Mô hình CoAP dựa vào tốc độ phát, còn TCP [64] dựa vào cửa sổ tắc nghẽn.
- Công thức (2.8) tính các hàm hiệu suất, thông lượng và trễ RTT theo *S* (số gói *inflight*), còn mô hình [64] sử dụng các hàm số của cửa sổ tắc nghẽn.
- Mục tiêu của công thức (2.8) trong bài này là để tính tốc độ phát, còn mục tiêu của mô hình trong [64] là tính toán kích thước cửa sổ tắc nghẽn.
- Mô hình TCP có bắt tay ba bước, dùng 3 gói tin ACK làm chỉ báo mất gói. CoAP không có bắt tay ba bước, chỉ sử dụng một ACK làm chỉ báo mất gói.

Hàm *U(S)* tăng theo tải và tốc độ phát đến giá trị cực đại và giảm dần khi xuất hiện tắc nghẽn (xem hình 2.2). Giá trị cực đại của *U(S)* không phải toàn cục. Các điều kiện giả thiết là: chưa xét tác động của nhiều luồng tin; số gói *S* không bị giới hạn; các hàm *U(S)*, *D(S)* và $\theta(S)$ tính đạo hàm được; hàm *U(S)* có tính lồi. Từ công thức (2.8), thực hiện lấy logarit và lấy đạo hàm hai vế ta có:

$$logU(S) = \partial\ log(\theta(S) - log\ D(S) \tag{2.9}$$

$$\frac{dU(S)}{U(S)} = \partial \frac{d\theta(S)}{\theta(S)} - \frac{dD(S)}{D(S)} \tag{2.10}$$

Hàm *U(S)* đạt giá trị cực đại khi đạo hàm của nó bằng không, do đó ta có:

$$\partial \frac{d\theta(S)}{\theta(S)} = \frac{dD(S)}{D(S)} \tag{2.11}$$

Tỷ số $d\theta(S)/\theta(S)$ biểu thị sự biến thiên thông lượng, hay tốc độ bên nhận, còn tỷ số *dD(S)/D(S)* biểu thị sự biến thiên trễ RTT theo số gói tin *inflight S*. Giá trị ∂ biểu thị tỷ lệ biến thiên của hai đại lượng nêu trên như sau:

- Nếu ∂ <1: Biến thiên của trễ RTT sẽ tăng nhanh hơn biến thiên của thông

lượng. Điều khiển sẽ có xu hướng nghiêng về ưu tiên độ trễ nhỏ.

- Nếu ∂ >1: Biến thiên của trễ RTT sẽ tăng chậm hơn biến thiên của thông lượng. Điều khiển sẽ có xu hướng nghiêng về ưu tiên thông lượng cao.
- Nếu ∂ =1: Biến thiên trễ RTT tương ứng với biến thiên thông lượng. Điều khiển sẽ có xu hướng duy trì sự cân bằng giữa độ trễ gói tin và thông lượng.

Sau đây, ta tính *B(S)* và *C(S)* cho các trường hợp không mất gói (không tắc nghẽn) và có mất gói (có tắc nghẽn) nhằm tính toán tốc độ phát cần điều chỉnh giảm để tránh tắc nghẽn. Số gói bên nhận xử lý (số gói đến đích) ở cuối một chu kỳ *k* phụ thuộc vào *S* tích lũy trong chu kỳ *k*, nên sẽ là hàm số của *S*. Gọi *B(S)* là số gói bên nhận xử lý trong trường hợp không mất gói ở cuối một chu kỳ *k*, *C(S)* là số gói bên nhận xử lý trong trường hợp có mất gói ở cuối một chu kỳ *k*.

- *Trường hợp không mất gói:*

  Thông lượng đạt được sẽ là:

  $$\theta(S) = \frac{B(S)}{D(S)} \quad (2.12)$$

  Lấy đạo hàm của θ theo *B*, ta có:

  $$\frac{d\theta(S)}{dB(S)} = \frac{1}{D(S)} \quad (2.13)$$

  Thay (2.12) và (2.13) vào (2.11), tại điểm cực đại của *U(S)* ta có:

  $$B(S) = \partial\, D(S) \frac{dB(S)}{dD(S)} \quad (2.14)$$

Do không mất gói, *B(S)* sẽ bằng số gói *inflight S* tích lũy trong chu kỳ *k*, tức là bằng lượng gói do bên gửi phát đi trong chu kỳ *k* theo công thức (2.1). Khoảng thời gian *T(k)* là thời gian tích lũy *S* gói *inflight* cũng là trễ RTT của *S* gói từ lúc phát đến lúc được bên nhận xử lý, nghĩa là *D(S)=T(k)* và *B(S)=R(k)×D(S)*. Ta có:

$$\frac{dB(S)}{dD(S)} = R(k) \quad (2.15)$$

Thay (2.15) vào (2.14), ta được:

$$B(S) = \partial\, D(S) R(k) \quad (2.16)$$

- *Trường hợp có mất gói:*

  Xét trường hợp có mất gói, thông lượng đạt được sẽ là:

$$\theta(S) = \frac{C(S)}{D(S)} \quad (2.17)$$

Thay (2.17) vào (2.8), lấy logarit và đạo hàm hai vế ta được:

$$logU(S) = \partial\ log(C(S) - (1+\partial)\ log D(S) \quad (2.18)$$

$$\frac{dU(S)}{U(S)} = \partial \frac{dC(S)}{C(S)} - (1+\partial)\frac{dD(S)}{D(S)} \quad (2.19)$$

Hàm *U(S)* đạt giá trị cực đại khi đạo hàm của nó bằng không, do đó có:

$$\frac{dU(S)}{U(S)} = \partial \frac{dC(S)}{C(S)} - (1+\partial)\frac{dD(S)}{D(S)} = 0 \quad (2.20)$$

$$C(S) = \frac{\partial}{(1+\partial)}\ D(S)\frac{dC(S)}{dD(S)} \quad (2.21)$$

So với (2.17), tỷ lệ *dC(S)/dD(S)* trong (2.21) chính là thông lượng ở chu kỳ *k*. Theo định nghĩa thông lượng đã nêu ở phần trên, tỷ số này cũng là tốc độ nhận gói tin R_r của bên nhận ở chu kỳ *k*. Do đó, công thức (2.21) trở thành:

$$C(S) = \frac{\partial}{(1+\partial)}\ D(S)R_r(k) \quad (2.22)$$

Ở phần trên, ta đã lập luận điều kiện tránh tắc nghẽn là $R \leq R_r$. Khi $R=R_r$, tắc nghẽn có thể xảy ra. Để ý là CoAP phát hiện có tắc nghẽn khi mất 1 gói tin, khi đó cơ chế điều khiển tắc nghẽn cần kích hoạt và giảm tốc độ phát. Không mất tính tổng quát, có thể giả thiết mất gói trong chu kỳ *k* là do *R* đạt ngưỡng R_r trong chu kỳ *k*. Với mục đích so sánh (2.22) và (2.16) khi $R_r = R$ ta viết lại (2.22) cho trường hợp xảy ra mất 1 gói ở cuối chu kỳ *k* như sau:

$$C(S) = \frac{\partial}{(1+\partial)}\ D(S)R(k) \quad (2.23)$$

So sánh (2.16) và (2.23), ta có nhận xét sau:

- Số gói tin *C(S)* do bên nhận xử lý thành công trong trường hợp mất 1 gói ở cuối một chu kỳ *k* giảm *1/(1+ ∂)* so với *B(S)* trong trường hợp không mất gói trong điều kiện hiệu suất *U(S)* đạt cực đại.

Như đã lập luận ở phần trên cho các giá trị của *∂*, nếu *∂ =1* thì cơ chế điều khiển sẽ duy trì được cân bằng giữa trễ RTT và thông lượng. Nếu chọn *∂* =1, *C(S)* sẽ giảm một nửa so với *B(S)*. Điều đó nghĩa là khi tắc nghẽn (có mất gói), bên nhận chỉ có thể xử lý một nửa lượng gói tin *inflight* so với điều kiện không mất gói.

Xét chu kỳ *k-1* không mất gói với lượng gói tin *inflight B(S)* và tốc độ phát *R(k-1)*, chu kỳ *k* khi mất 1 gói với *C(S)* và tốc độ phát *R(k)*. Từ nhận xét trên và từ (2.16), (2.23) có thể rút ra trong điều kiện duy trì cùng *D(S)* như sau:

$$D(S)R(k-1) = \frac{1}{2} D(S)R(k) \quad (2.24)$$

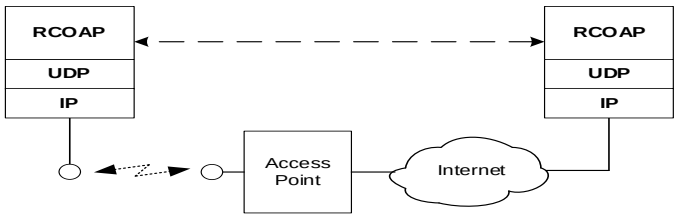
$$R(k) = 0{,}5 \times R(k-1) \quad (2.25)$$

Nghĩa là, khi xảy ra mất gói, tốc độ phát ở chu kỳ tiếp theo phải được điều chỉnh giảm đi ít nhất một nửa so với chu kỳ trước đó để giảm tắc nghẽn. Các công thức (2.7) và (2.25) được dùng để điều chỉnh tốc độ CoAP hạn chế tắc nghẽn.

2.2. Đề xuất giao thức RCoAP

2.2.1. Cơ chế hoạt động và điều khiển tắc nghẽn của RCoAP

Giao thức RCoAP hoạt động theo cơ chế điều khiển dựa vào tốc độ, cài đặt ở trên tầng UDP tại bên gửi và bên nhận như hình 2.4.



Hình 2.4 Sơ đồ triển khai RCoAP tại bên gửi và bên nhận

Các nghiên cứu trước đây [27, 11] đã chỉ ra chu kỳ ngắn nhất để điều chỉnh tốc độ là một RTT, tức là *T(k) = RTT*. Công thức (2.7) và (2.25) sẽ đơn giản là:

Khi tăng tốc độ: $R = R + 1 / RTT$ (2.26)

Khi giảm tốc độ: $R = 0{,}5 \times R$ (2.27)

RCoAP sử dụng tiêu đề gói tin CoAP [100], song cơ chế điều khiển tốc độ phát theo công thức (2.26) và (2.27) nêu trên, cụ thể như sau:

- RCoAP khởi tạo và phát các gói tin theo tốc độ tăng dần theo công thức (2.26) khi không có tắc nghẽn.
- RCoAP giảm tốc độ phát đi một nửa theo công thức (2.27) khi có mất gói, tức là khi có tắc nghẽn xảy ra.

Cơ chế hoạt động của RCoAP đơn giản, không quá phức tạp so với CoAP song khắc phục được các hạn chế (1), (2) và (3) của CoAP như đã nêu ở chương 1. Để

tăng hiệu quả truyền tin, luận án đề xuất bổ sung cho RCoAP hai cơ chế: Xác định tốc độ phát ban đầu và phân biệt nguyên nhân mất gói.

Nếu tốc độ phát ban đầu nhỏ, RCoAP sẽ mất một khoảng thời gian để đạt được tốc độ tối đa. Do vậy, nếu xác định tốc độ ban đầu phù hợp băng thông cổ chai và trạng thái mạng, RCoAP sẽ hiệu quả hơn. Đặt R_{max} là tốc độ tối đa cho phép của ứng dụng. Việc lựa chọn R_{max} là tùy thuộc vào ứng dụng và điều kiện mạng. Luận án đề xuất các xác định tốc độ phát ban đầu như sau:

- Khi bắt đầu kết nối, RCoAP gửi một chuỗi gói tin thử liên tiếp tương tự như [11, 27] để xác định tốc độ phát ban đầu. Gói tin thử là gói RCoAP rỗng (chỉ có tiêu đề). Thời gian phát thử kéo dài $2\times RTT$. Bên gửi đếm số gói ACK nhận được với biến *nACK*.
- Sau khoảng thời gian $2\times RTT$, bên gửi căn cứ vào R_{max} và *nACK* để xác định tốc độ phát ban đầu như công thức sau:

$$R = min\left(R_{max}, \frac{max(1,\ nACK)}{2\times RTT}\right) \quad (2.28)$$

Hàm *max* là để tránh *R=0*, hàm *min* là để hạn chế tốc độ tối đa. Số gói *nACK* nhận được phụ thuộc vào băng thông cổ chai và trạng thái tắc nghẽn.

Để phân biệt nguyên nhân mất gói (khắc phục hạn chế (4) của CoAP đã nêu ở chương 1), luận án sử dụng phương pháp sau:

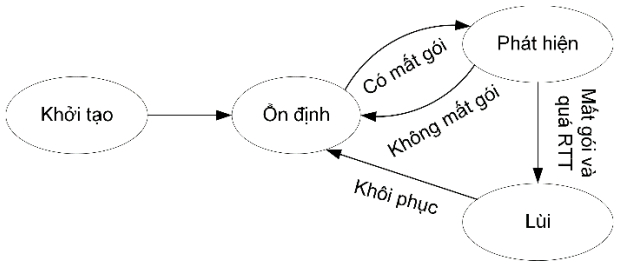
- Khi phát hiện mất gói, bên gửi RCoAP gửi tiếp một số gói tin thử để kiểm tra nguyên nhân mất gói. Nếu sau RTT bên gửi nhận được ACK, mất gói được coi là do lỗi kênh vô tuyến. Nếu quá RTT mà vẫn chưa nhận được ACK, mất gói được coi là do tắc nghẽn. Lý do là: 1) lỗi kênh vô tuyến thường chỉ gây mất gói đơn (do gián đoạn kênh trong thời gian ngắn), 2) tắc nghẽn thường kéo theo mất nhiều gói kế tiếp [67, 32, 119].
- Nếu mất gói do lỗi vô tuyến, RCoAP duy trì tốc độ phát. Nếu mất gói do tắc nghẽn, RCoAP chia đôi tốc độ phát theo (2.27) để tránh tắc nghẽn tiếp.

Mất gói được phát hiện bằng hai cách: 1) quá RTO mà bên gửi vẫn chưa nhận được ACK cho gói đã phát, 2) phát hiện khoảng trống trong số thứ tự nhận được. Số thứ tự được chèn vào trường tùy chọn (dùng 4 bytes) của tiêu đề RCoAP để kiểm

tra khoảng trống và xác định mất gói (Phụ lục 2).

2.2.2. Các trạng thái của giao thức RCoAP

Theo cơ chế hoạt động đã nêu, giao thức RCoAP được thiết kế với 4 trạng thái: 1) Khởi tạo, 2) Ổn định, 3) Phát hiện và 4) Lùi như mô tả trên hình 2.5.



Hình 2.5 Bốn trạng thái của giao thức RCoAP

- ***Trạng thái khởi tạo***

Khi bắt đầu kết nối, RCoAP ở trạng thái khởi tạo trong khoảng *2× RTT*. Mục đích của trạng thái này là để xác định tốc độ phát ban đầu theo công thức (2.28). Sau khoảng *2× RTT*, RCoAP sẽ chuyển sang trạng thái hoạt động ổn định.

- ***Trạng thái ổn định***

Đây là trạng thái hoạt động chính để truyền tin của RCoAP. Trạng thái này được duy trì nếu không có tắc nghẽn xảy ra. Bên gửi bắt đầu phát với tốc độ khởi tạo theo (2.28) và tăng dần tốc độ nếu không thấy mất gói. RTT được cập nhật tương tự như cách tính RTT cho TCP [104] hay CoCoA+ [15] cụ thể như sau:

$$RTT(k) = (1 - \gamma) \times RTT(k-1) + \gamma \times RTT(k) \quad (2.29)$$

Trong đó *RTT(k)* là giá trị RTT đo được ở chu kỳ mới, *RTT(k-1)* là giá trị RTT ở chu kỳ trước đó, $\gamma = 0.25$ là hệ số theo [104, 15]. Cách tính này làm giảm thiểu tác động biến thiên tức thời của RTT. Sau mỗi RTT, bên gửi điều chỉnh tốc độ phát *R* nếu không thấy mất gói như sau:

$$R = min\left(R_{max},\ R + \frac{1}{RTT}\right) \quad (2.30)$$

R sẽ được tăng thêm *1* gói tin sau mỗi RTT theo (2.30). Khi phát một gói mới, bên gửi đặt một giá trị RTO cho mỗi gói tin. RTO khởi tạo tương tự như CoAP (*2*s theo [100]). Nếu không nhận được ACK cho gói tin đã phát đi, bên gửi sẽ dừng trạng thái ổn định để chuyển sang trạng thái phát hiện.

- ***Trạng thái phát hiện***

Khi phát hiện mất gói, RCoAP lập tức giảm tốc độ một nửa theo công thức (2.27). Tiếp đó, RCoAP xác định mất gói do lỗi kênh vô tuyến hay do tắc nghẽn theo cách đã nêu ở mục 2.2.1 với việc phát một số gói tin thử. Nếu có ACK trong khoảng một RTT, RCoAP xác định mất gói do lỗi vô tuyến. Khi đó, RCoAP khôi phục lại tốc độ phát để duy trì hiệu năng và trở về trạng thái ổn định. Nếu sau RTT vẫn chưa nhận được ACK, RCoAP xác định mất gói do tắc nghẽn và chuyển sang trạng thái lùi.

- ***Trạng thái lùi***

Trong trạng thái này, RCoAP giảm tốc độ phát theo công thức (2.27) để tránh tắc nghẽn thêm. Tiếp đó, RCoAP thực hiện phát lại các gói tin đã mất. Tiến trình phát lại sử dụng cơ chế lùi tương tự như CoCoA+ [15], cụ thể như sau. RCoAP đo và cập nhật RTT theo công thức (2.29) để tính độ biến thiên *RTVAR* và *RTO*:

$$RTVAR(k) = (1-\beta) \times RTVAR(k-1) + \beta \times |RTT(k-1) - RTT(k)| \quad (2.31)$$

với $RTT(k-1) = (1-\alpha)RTT(k-1) + \alpha RTT(k)$

$$RTO(k) = 0.5 \times (RTT(k) + 4 \times RTVAR(k)) + 0.5 \times RTO(k-1) \quad (2.32)$$

$$RTO_{backoff} = RTO(k) \times VBF \quad (2.33)$$

Với $\alpha = 0.25$ và $\beta = 0.125$ theo [15].

Giá trị VBF được tính theo công thức (1.8) như trong CoCoA+ [15].

Hệ số VBF biểu thị chiến lược lùi biến động trong trường hợp có tắc nghẽn. Số lần phát lại mỗi gói tối đa là *4* theo [100]. Nếu quá *4* lần phát lại, gói tin tương ứng được coi là mất. Trong thời gian lùi, nếu bên gửi nhận được một ACK, tắc nghẽn được coi là hết. Khi đó, RCoAP dừng trạng thái lùi và chuyển về trạng thái ổn định với tốc độ phát hiện có. Trường hợp có tắc nghẽn kéo dài, RCoAP sẽ tự động lặp lại quá trình chuyển từ trạng thái ổn định sang phát hiện và lùi như mô tả ở trên.

2.2.3. Các thuật toán cơ bản của giao thức RCoAP

Phần sau trình bày các thuật toán cơ bản của RCoAP theo 4 trạng thái đã nêu. RCoAP hoạt động theo sự kiện. Một số hàm sự kiện không trình bày chi tiết ở đây

gồm: nhận gói, gửi gói, định thời RTO và hàm phát lại gói. Biến ACK=*False* mỗi khi gửi một gói mới, ACK chỉ là *True* khi bên gửi nhận một gói ACK.

- ***Thuật toán khởi tạo***

Thuật toán khởi tạo thực hiện một lần khi bắt đầu kết nối và kéo dài $2\times RTT$ nhằm xác định tốc độ phát ban đầu. Đầu vào là R_{max} và RTO_{init}. RTO_{init} là giá trị định thời phát lại của CoAP. R_{max} là tốc độ tối đa cho phép ứng dụng (tùy chọn). Đầu ra là tốc độ R. Biến *nACK* là số gói ACK đếm được. *T* là thời điểm phát tiếp một gói tin; t_0 là thời điểm bắt đầu. *Tm* là thời gian tối đa chờ ACK (tương ứng 4 lần phát lại). Biến ACK được khởi tạo = *False* (dòng 5). Lưu ý là ACK=*False* mỗi khi phát gói mới với hàm SendNextPacket, chỉ là *True* khi có sự kiện bên gửi nhận được ACK.

Thuật toán 2.1. Khởi tạo

Đầu vào: R_{max} , RTO_{init}
Đầu ra: *R*
1: Function InitState()
2: *nACK* = 0
3: $T = t0 + 1 / R_{max}$
4: $Tm = t0 + 4\times RTO_{init}$
5: *ACK = False*
6: WHILE (*ACK == False*) DO
7: IF ($t \geq T$) THEN
8: packet ← SendNextPacket()
9: $T = t + 1 / R_{max}$
10: ENDIF
11: IF (*t > Tm*) THEN Restart()
12: ENDIF
13: ENDDO
14: *RTT* ← CalculateRTT()
15: WHILE ($t \leq 2\times RTT$) DO
16: IF (*ACK == True*) THEN
17: *nACK = nACK* + 1
18: *ACK = False*
19: ENDIF
20: ENDDO
21: $R = min (R_{max}, max (1, nACK / (2\times RTT)))$
22: **END Function.**

Vòng WHILE thứ nhất gửi chuỗi gói thử (dòng 6-13). RCoAP phát từng gói tin với hàm SendNextPacket() tại các thời điểm $t \geq T$ với *t* là thời gian tức thời. Nếu

quá *Tm* mà vẫn chưa có ACK, RCoAP có lỗi kết nối và cần khởi tạo lại. Khi có ACK, RCoAP tính RTT và chuyển sang vòng WHILE thứ 2. Vòng WHILE này thực hiện trong *2×RTT* (dòng 15-20). RCoAP đếm *nACK* mỗi khi nhận được gói ACK. Sau *2×RTT*, RCoAP tính tốc độ phát ban đầu (dòng 21) và kết thúc khởi tạo, chuyển sang trạng thái ổn định.

- ***Thuật toán ở trạng thái ổn định***

Trạng thái ổn định duy trì khi không có mất gói xảy ra. Biến *T* xác định thời điểm phát tiếp một gói tin theo tốc độ *R*, *t* là thời thời gian tức thời, *Tn* là thời điểm điều chỉnh tốc độ phát, biến *ACK* chỉ là *True* mỗi khi có sự kiện nhận một gói ACK, biến *pLoss* biểu thị có gói tin bị mất.

Thuật toán 2.2. Ổn định

Đầu vào: *R*
Đầu ra: *pLoss*
1: **Function SteadyState()**
2: *T = t + 1 / R*
3: *Tn = t + RTT*
4: *ACK = False*
5 *pLoss = False*
6: WHILE (*pLoss == False*) DO
7: IF ($t \geq T$) THEN
8: packet ← SendNextPacket()
9: *T = t + 1 / R*
10: ENDIF
11: IF (*ACK == True*) THEN *RTT* ← CalculateRTT()
12: ENDIF
13: IF (*t > Tn*) THEN
14: *R* = min (*R*+ *1 / RTT*, R_{max})
15: *Tn = t + RTT*
16: ENDIF
17: *pLoss* ← CheckLoss()
18: ENDDO
19: **DetectState**()
21: **END Function.**

Vòng WHILE (dòng 6-18) duy trì khi không có mất gói. RCoAP phát từng gói tin với hàm SendNextPacket() tại các thời điểm T ≥ *t + 1/R* (dòng 7-9) với *t* là thời gian tức thời. Biến *ACK=True* mỗi khi có sự kiện nhận được gói ACK, *RTT* được cập nhật mới (dòng 11). Do không có mất gói, tốc độ phát R được cập nhật theo

công thức (2.30) sau mỗi khoảng *RTT* (dòng 13-16). Biến *pLoss* được kiểm tra với hàm CheckLoss(). Hàm này kiểm tra khoảng trống trong số thứ tự gói nhận được. Biến *pLoss* cũng có thể bằng *True* khi hàm sự kiện định thời *RTO* được kích hoạt báo hiệu mất gói. Khi *pLoss = True*, thuật toán rời vòng WHILE và chuyển sang trạng thái phát hiện bằng cách gọi hàm DetectState() khi xác định có mất gói.

- ***Thuật toán phát hiện***

Thuật toán bắt đầu khi RCoAP phát hiện có mất gói với *pLoss=True*. Các biến *T, R, t, Tn, ACK, pLoss* tương tự như ở thuật toán trước. RCoAP lưu lại tốc độ hiện có vào biến R_{old} và lập tức giảm tốc độ *R* đi một nửa (dòng 5) theo công thức (2.27).

Thuật toán 2.3. Phát hiện

Đầu vào: *R* , *pLoss*
Đầu ra: *R* , *pLoss*

1: **Function DetectState()**
2: $T = t + 1 / R$
3: $Tn = t + RTT$
4: *ACK = False* ; *pLoss = False*
5: *Rold = R* ; $R = R_{old} / 2$
6: WHILE ($t \leq Tn$) DO
7: IF ($t \geq T$) THEN
8: packet ← SendNextPacket()
9: $T = t + 1 / R$
10: ENDIF
11: IF (*ACK == True*) THEN
12: $R = R_{old}$
13: *ACK = False*
14: **return SteadyState**()
15: ENDIF
16: ENDDO
17: IF (*ACK == False*) THEN
18: *pLoss = True*
19: **BackoffState**()
20: ENDIF
21: **END Function.**

Trong khoảng một *RTT* (vòng WHILE dòng 6-21), bên gửi sẽ phát tiếp một số gói tin thử với hàm SendNextPacket() để xác định nguyên nhân mất gói (dòng 6-10). Nếu có ACK trong khoảng một *RTT* (dòng 11), mất gói là do lỗi vô tuyến. Khi đó RCoAP khôi phục lại tốc độ cũ (dòng 12) và quay về trạng thái ổn định (dòng 14). *ACK=True* chỉ khi bên gửi nhận được gói ACK trong sự kiện hàm nhận. Nếu

ACK=False và vẫn chưa hết một *RTT (biến Tn)*, RCoAP tiếp vòng WHILE. Nếu đã hết một *RTT* mà vẫn chưa nhận được ACK, RCoAP xác định mất gói do tắc nghẽn và chuyển sang trạng thái lùi (dòng 17-19).

- ***Thuật toán lùi***

Các biến *T, R, t, Tn, ACK, pLoss* tương tự như ở các thuật toán trước. Trạng thái lùi duy trì khi không nhận được bất kỳ gói tin ACK nào (vòng WHILE ở dòng 6-20). Do đã xác định tắc nghẽn ở thuật toán phát hiện, RCoAP giảm tiếp tốc độ phát theo công thức (2.27) để tránh tắc nghẽn thêm (dòng 5). Tiếp đó, RCoAP thực hiện phát lại các gói tin đã mất (dòng 7-10) với hàm SendNextRTX().

Thuật toán 2.4. Lùi

```
Đầu vào: R , pLoss
Đầu ra: R
1: Function BackoffState()
2:      T = t + 1 / R
3:      Tn = t + RTT
4:      ACK = False
5:      R = R / 2
6:      WHILE ( ACK == False ) DO
7:          IF ( t ≥ T ) THEN
8:              packet ← SendNextRTX()
9:              T = t + 1 / R
10:         ENDIF
11:         IF ( ACK == True) THEN
12:             ACK = False
13:             return SteadyState()
14:         ENDIF
15:         IF ( t ≥ Tn ) THEN
16:             R = R / 2
17:             T = t + 1 / R
18:             Tn = t + RTT
19:         ENDIF
20:     ENDDO
21: END Function.
```

Nếu có bất kỳ gói ACK nào nhận được (dòng 11), RCoAP sẽ quay về trạng thái ổn định (dòng 13) với tốc độ phát *R* hiện tại. Nếu có tắc nghẽn kéo dài, RCoAP sẽ tự động lặp lại quá trình chuyển từ trạng thái ổn định sang phát hiện và lùi như mô tả ở trên. Sau mỗi RTT (dòng 15), RCoAP sẽ tiếp tục lùi với việc chia đôi tốc độ phát (dòng 16), cập nhật lại *T* và *Tn* (dòng 17-18). Trường hợp không nhận được

ACK trong thời gian quá dài, triển khai RCoAP thực tế có thể kiểm tra thời gian để kết thúc kết nối.

2.3. Tính toán hiệu năng giao thức RCoAP

Theo các tham số đánh giá hiệu năng đã nêu ở chương 1, phần này tính toán các đại lượng: độ trễ, tỷ lệ mất gói và thông lượng.

Bảng 2.2: Các ký hiệu tính toán hiệu năng RCoAP

Tên ký hiệu	Ý nghĩa
D	Độ trễ gói tin một chiều
D_0	Độ trễ gói tin một chiều của gói tin đầu tiên của luồng tin
d_i	Độ trễ gói tin thứ i
D_{TB}	Độ trễ gói tin trung bình của luồng tin
F	Hệ số ngẫu nhiên
N_k	Tổng số gói phát đi thành công
M	Số lần phát lại thứ m
P	Xác suất mất gói trong một lần phát
P_i	Xác suất mất gói sau *i* lần phát
$RT0_o$	Giá trị RTO khởi tạo
θ_{TB}	Thông lượng trung bình của luồng tin

2.3.1. Độ trễ gói tin

Độ trễ là thời gian để truyền một gói tin từ bên gửi tới bên nhận thành công. Độ trễ d_i của gói tin thứ *i* được tính theo công thức:

$$d_i = D_0 + (2^i - 1) \times RTO_0 \times \frac{(f+1)}{2} \quad (2.34)$$

Trong đó, *i* là số lần gói tin phải phát lại với $1 \le i \le m$, *m* là số lần phát lại tối đa, RTO_0 là giá trị *RTO* khởi tạo, *f* là một hệ số ngẫu nhiên để biểu thị bên gửi sẽ chờ một khoảng giữa *[RTO, RTO×f]* để phát lại sau mỗi *RTO*. 2^i vì mỗi lần phát lại, giá trị *RTO* sẽ nhân đôi. $2^i - 1$ vì trong số *i* lần phát lại, có ít nhất một lần phát lại thành công để tính được độ trễ. Xác suất phát gói thành công có trễ d_i sau *i* lần phát lại là:

$$Pr(D = d_i) = p^i(1 - p) \quad (2.35)$$

Với p là xác suất mất gói trong một lần phát (*1-p)* là xác suất phát thành công một gói). Với $TO = RTO_0(f+1)/2$, độ trễ trung bình của các gói tin sẽ là:

$$D_{TB} = \sum_{i=0}^{m} \Pr(D = d_i)d_i = \sum_{i=0}^{m} p^i(1-p)[\, D_0 + (2^i - 1)TO] \quad (2.36)$$

$$D_{TB} = D_0 \sum_{i=0}^{m} p^i(1-p) + TO \sum_{i=0}^{m} (2p)^i(1-2p)\frac{(1-p)}{(1-2p)} - TO \sum_{i=0}^{m} p^i(1-p) \quad (2.37)$$

Vì: $\sum_{i=0}^{m} p^i(1-p) = 1 - p^{m+1}$ (2.38)

Nếu coi p rất nhỏ so với m, có thể tính xấp xỉ D_{TB} như sau:

$$D_{TB} = D_0(1 - p^{m+1}) + TO[1 + p^{m+1} - (2p)^{m+1}] \quad (2.39)$$

Nếu giới hạn số lần phát lại m thì có thể giới hạn độ trễ D_{TB} của luồng tin.

2.3.2. Tính toán mất gói tin khi truyền

Xác suất phát gói thành công sau i lần phát lại là: $p^i(1-p)$, do vậy xác suất mất một gói tin sau i lần phát lại là:

$$P_i = 1 - p^i(1-p) \quad (2.40)$$

Tỷ lệ mất gói là số gói tin bị mất khi truyền so với tổng số gói tin đã phát đi.

2.3.3. Thông lượng

RCoAP phát đi chuỗi gói *inflight* tương tự TCP nên có thể sử dụng công thức tính thông lượng trung bình cho TCP [43]. Tuy nhiên, TCP phát hiện mất gói bằng dấu hiệu mất ba ACK liên tiếp, còn RCoAP phát hiện mất gói chỉ bằng 1 ACK bị mất. Vì vậy, công thức tính thông lượng trung bình cho RCoAP sẽ như sau:

$$\theta_{TB}(p, RTT) = \frac{1}{RTT\sqrt{\frac{2p}{3}} + RTO_0\left(3\sqrt{\frac{3p}{8}} \times p(1+32p^2)\right)} \quad (2.41)$$

Có thể tính xấp xỉ:

$$\theta_{TB}(p, RTT) = \frac{1.42}{RTT\sqrt{p}\,(1 + 2.5 \times RTO_0 \times p(1+32p^2)} \quad (2.42)$$

Nếu bỏ qua các gói tin phát lại với RTO_0, ta có công thức xấp xỉ sau:

$$\theta_{TB}(p, RTT) = \frac{1.42}{RTT\sqrt{p}} \quad (2.43)$$

2.3.4. Các tham số hiệu năng khác

Các tham số hiệu năng khác như: số gói tin phát đi thành công (có phản hồi ACK), tỷ lệ gói tin phát thành công, số gói tin phải phát lại, tỷ lệ gói tin phát lại, số gói tin phát lại đúp (phát lại gói trùng với gói tin đã nhận được), tỷ lệ gói tin phát lại

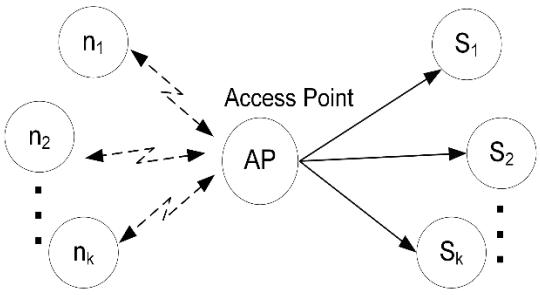
đúp được tính bằng giá trị trung bình của các tham số đo được khi mô phỏng và tính độ tin cậy theo công thức đã nêu ở chương 1.

2.4. Kết quả mô phỏng cho RCoAP

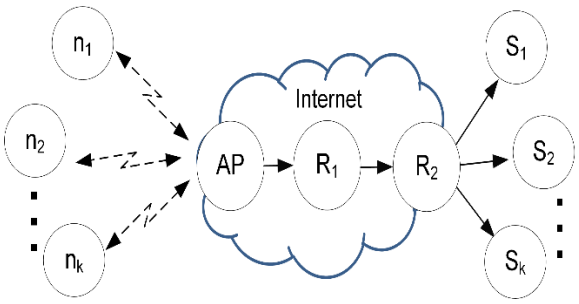
2.4.1. Thiết lập môi trường mô phỏng

Bộ công cụ sử dụng cho mô phỏng là NS3.36 [88]. NS-3 là một bộ công cụ mô phỏng mạng theo sự kiện rời rạc, hỗ trợ xây dựng và phát triển các giao thức truyền tin nhanh, hiệu quả và sát với thực tế. NS-3 rất linh hoạt, tạo được nhiều topo mạng và kịch bản đa dạng tương tự như trên mạng thực. Phần mềm CoAP dựa trên các hàm cơ bản của Maesor [82] và Take [50]. Phần mềm CoCoA và CoCoA+ được tạo từ CoAP với việc bổ sung và sửa đổi hàm tính RTO. Phần mềm RCoAP được xây dựng dựa trên CoAP với việc bổ sung các hàm cho truyền chuỗi gói, tính tốc độ phát và điều chỉnh tốc độ (Phụ lục 3).

Mô hình mạng ICN cho ứng dụng CoAP được đưa ra các tiêu chuẩn [99, 57, 60, 62], trong đó các mô đun IoT truyền tin qua IoT Gateway, qua mạng Internet về máy chủ ở trung tâm như mô tả ở hình 1.3. Các nghiên cứu liên quan thường sử dụng mạng hình sao để mô phỏng ứng dụng một chặng (cho ứng dụng đơn giản [99]) và mạng hình xương cá cho ứng dụng nhiều chặng kết nối với nhiều bên nhận. Luận án cũng sử dụng 2 mô hình này như trên hình 2.6 và 2.7 với các mô đun IoT bên gửi và các máy chủ bên nhận, AP là điểm truy nhập (hay Gateway), R_1 và R_2 biểu thị các nút mạng Internet chuyển tiếp dữ liệu.



Hình 2.6 Mạng hình sao



Hình 2.7 Mạng hình xương cá

Theo tiêu chuẩn ITU, các mô đun IoT sử dụng phổ biến công nghệ ZigBee (802.15.4), WiFi (802.11n/g) hoặc Ethernet [57, 99]. CoAP trong các mô đun IoT thường hoạt động ở tốc độ thấp, ví dụ tốc độ tối đa đạt 250 Kbps với 802.15.4, một

đặc trưng cho mạng 6LoWPAN [100]. Các nghiên cứu liên quan như CoCoA [14], CoCoA+ [15], BDP-CoAP [11], RTT-CoAP [9] sử dụng cấu hình mô phỏng gồm: ZigBee (802.15.4) hoặc WiFi (802.11) cho tầng vật lý, tốc độ phát của mỗi luồng tin từ 1 Kbps - 250 Kbps, thời gian mô phỏng phổ biến từ 10s -300s. Các tình trạng tắc nghẽn được mô phỏng bằng cách thay đổi băng thông cổ chai và độ trễ của tuyến kết nối giữa hai đầu cuối [68, 66, 10]. Để so sánh với các nghiên cứu liên quan, luận án sử dụng cấu hình mạng mô phỏng tương tự [13, 15, 22, 93, 68] để so sánh RCoAP với các giao thức CoAP đã chuẩn hóa gồm: CoAP, CoCoA và CoCoA+. Các tham số cấu hình ở bảng 2.3 sau.

Bảng 2.3: Các tham số chính để thiết lập mô phỏng RCoAP

Tham số		**Giá trị**
Thời gian mô phỏng		50s – 300s
Số lần chạy mỗi kịch bản		30 lần
Số mô đun IoT bên gửi		3 – 30
Tầng vật lý và MAC		WifiPhy 2,4 GHz; IEEE 802.11b/n [88, 93, 68]
Liên kết AP		250 Kbps, 64 ms (tương tự [14, 15, 22, 23])
Kết nối Internet		10/100 Mbps (Ethernet)
Kịch bản mô phỏng		Mô phỏng băng thông cổ chai với BW và trễ liên kết D theo cách trong [66, 21, 10, 68, 15, 115]:
+ Kịch bản 2.1:	Thử nghiệm chức năng	BW = 500 Kbps; D = 500 ms
+ Kịch bản 2.2:	So sánh giao thức	BW = 1 Mbps, 500 Kbps; D = 64 ms, 500 ms
+ Kịch bản 2.3:	Lưu lượng biến động	Kịch bản 2.3a: BW = 1 Mbps; D = 70 ms
		Kịch bản 2.3b: BW = 1 Mbps; D = 120 ms
Mô hình mạng		Kịch bản 2.1: Mạng hình sao (hình 2.6) Kịch bản 2.2, 2.3: Mạng hình xương cá (hình 2.7)
Thời điểm bắt đầu phát		Ngẫu nhiên trong khoảng 0-200 ms
Tính khoảng tin cậy		Theo công thức (1.11) với mức tin cậy 99%

2.4.2. Kịch bản 2.1

Kịch bản 2.1 sử dụng mạng hình sao (hình 2.6) nhằm thử nghiệm chức năng xác

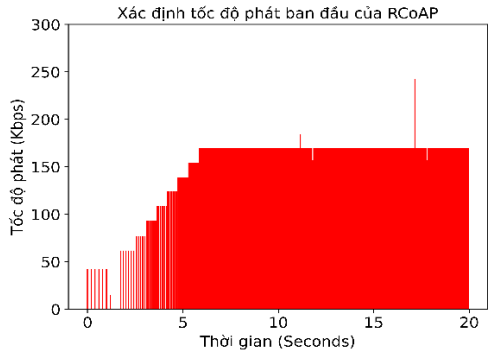
định tốc độ phát ban đầu, điều khiển tăng/giảm tốc độ phát và nhận biết nguyên nhân mất gói của RCoAP.

- ***Xác định tốc độ phát ban đầu***

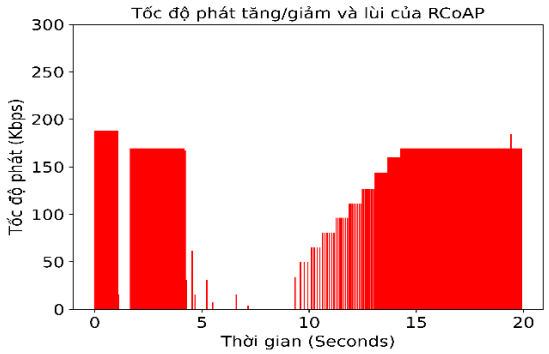
Hình 2.8 biểu thị kết quả xác định tốc độ phát ban đầu sau 2× RTT (xấp xỉ 1,3s). Sau khi khởi tạo, RCoAP bắt đầu phát các gói tin thử cho đến khi nhận được ACK đầu tiên. Khi không có tắc nghẽn, RCoAP tăng dần tốc độ phát và đạt tốc độ tối đa 160 Kbps ở thời điểm 7s.

- ***Tăng giảm tốc độ và lùi***

Mất gói được mô phỏng bằng cách loại bỏ một số gói, tương tự như cách ở các nghiên cứu liên quan. Hình 2.9 biểu thị điều khiển của RCoAP. Khi phát hiện mất gói do tắc nghẽn (mất gói liên tiếp trong khoảng 4,5s-6,5s), RCoAP giảm tốc độ phát đi một nửa ở 4,5s và tiếp tục giảm dần (chia đôi) về 0. Từ 9s, do không còn mất gói (hết tắc nghẽn), RCoAP tăng dần tốc độ phát để đạt tốc độ tối đa 160 Kbps ở thời điểm 14s.



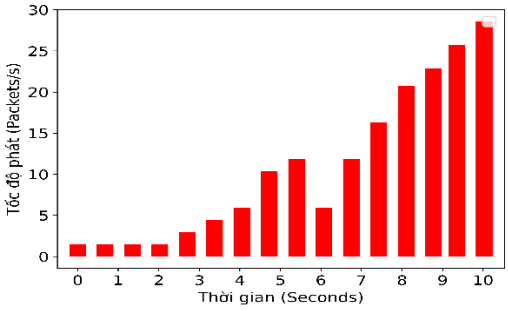
Hình 2.8 Xác định tốc độ ban đầu



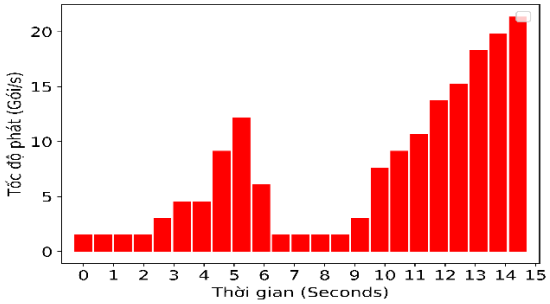
Hình 2.9 Tăng/giảm tốc độ và lùi

- ***Phân biệt nguyên nhân mất gói***

Hình 2.10 biểu thị RCoAP phát hiện mất gói do lỗi vô tuyến ở thời điểm 5,5s do chỉ mất một gói. Tốc độ phát được khôi phục lại giá trị trước thời điểm mất gói.



Hình 2.10 Mất gói do lỗi kênh vô tuyến



Hình 2.11 Mất gói do tắc nghẽn

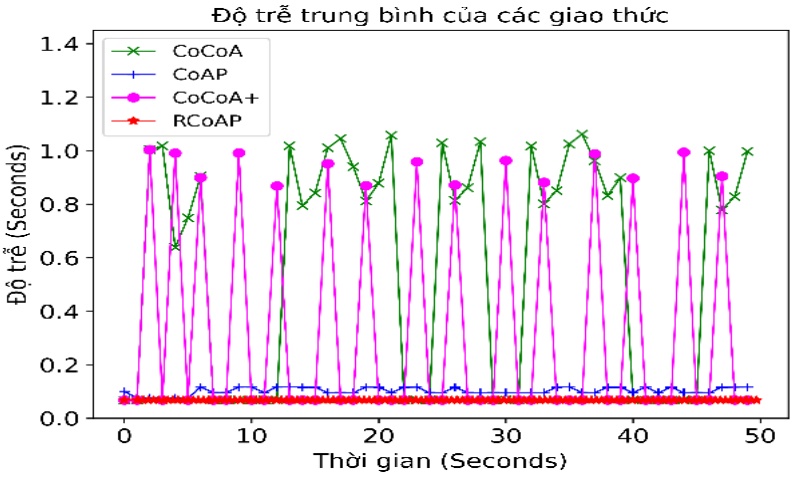
Hình 2.11 biểu thị RCoAP phát hiện mất gói do tắc nghẽn khi có mất gói liên tiếp trong khoảng 4,5s - 6s. Tiếp đó, RCoAP giảm tốc độ phát một nửa sau mỗi RTT và phát tiếp các gói tin thử để kiểm tra trong khoảng 6s – 8.5s. Khi khắc phục được tắc nghẽn, RCoAP bắt đầu tăng dần tốc độ phát từ thời điểm 9s.

2.4.3. Kịch bản 2.2

Kịch bản 2.2 sử dụng mô hình mạng 2.7 để so sánh hiệu năng của RCoAP với các giao thức CoAP, CoCoA và CoCoA+ dùng các tham số mô phỏng ở bảng 2.3. Tốc độ phát của các luồng tin thay đổi từ 2 Kbps tới 2,5 Kbps. Các đại lượng: độ trễ gói, thông lượng, tỷ lệ mất gói, số gói phát thành công, số gói phát lại được đo lường để so sánh hiệu năng của các giao thức.

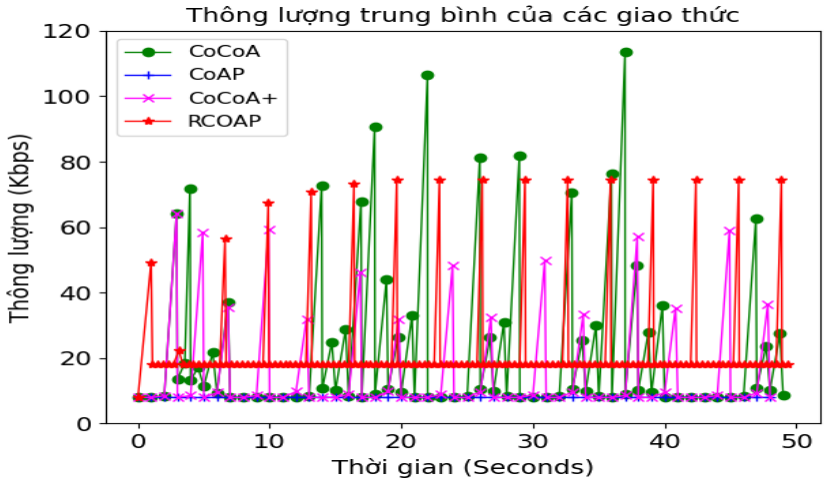
- ***So sánh hiệu năng RCoAP với CoAP, CoCoA và CoCoA+***

Hình 2.12 biểu thị độ trễ trung bình của các giao thức. Độ trễ của CoCoA và CoCoA+ cao và biến động nhanh do có nhiều gói tin phải phát lại (xem Bảng 2.4). Độ trễ trung bình trong khoảng 400 ms -700 ms với đỉnh lên tới 1,0s. Độ trễ của CoAP và RCoAP thấp hơn và biến thiên nhỏ. Đô trễ của CoAP nhỏ và biến đổi ít hơn CoAP, CoCoA và CoCoA+. Nguyên nhân là RCoAP không phải phát lại gói tin nào trong thí nghiệm này (xem Bảng 2.4).



Hình 2.12 Độ trễ của các giao thức

Hình 2.13 so sánh thông lượng của RCoAP, CoAP, CoCoA và CoCoA+. Thông lượng các giao thức biến thiên với một số đỉnh nhọn do sự thay đổi trong RTO và số lần phát lại. Các số liệu đo lường được liệt kê trong Bảng 2.4. Kết quả mô phỏng cho thấy thông lượng trung bình của CoAP, CoCoA và CoCoA+ xấp xỉ 16,96 Kbps (gồm cả phát lại), trong khi thông lượng trung bình của RCoAP cao hơn do phát chuỗi gói, đạt xấp xỉ 23,24 Kbps trong cùng điều kiện mô phỏng này.



Hình 2.13 Thông lượng của các giao thức

Trong kịch bản 2.2, băng thông cổ chai chưa đủ tạo ra tắc nghẽn. Chưa có mất gói xảy ra trong cả 4 giao thức, mặc dù CoAP, CoCoA và CoCoA+ phải phát lại tới 43 gói như trong bảng 2.4.

Bảng 2.4: So sánh hiệu năng của RCoAP và các giao thức CoAP khác

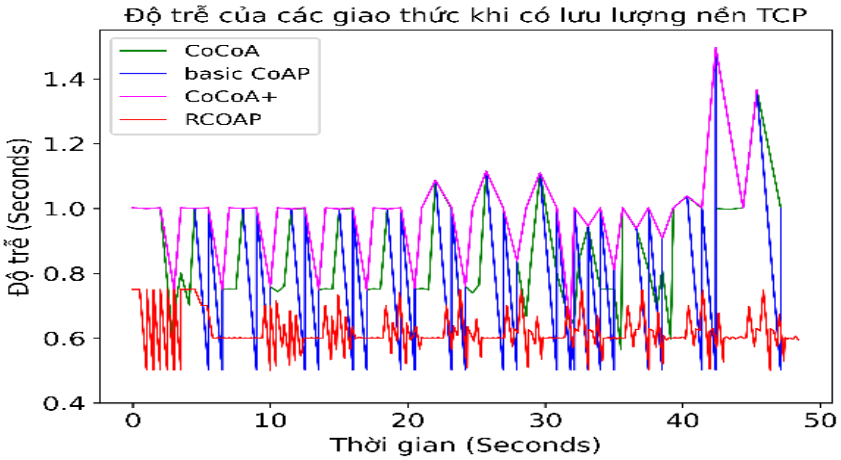
Giao thức	Độ trễ trung bình (ms)	Thông lượng trung bình (Kbps)	Tỷ lệ mất gói (%)	Số gói phát thành công	Số gói phát lại
CoAP	102,4	16,96	0	50	43
CoCoA	585,7	16,96	0	50	43
CoCoA+	316,7	16,96	0	50	43
RCoAP	78,8	23,24	0	137	0

- ***So sánh hiệu năng các giao thức CoAP khi có lưu lượng TCP biến động***

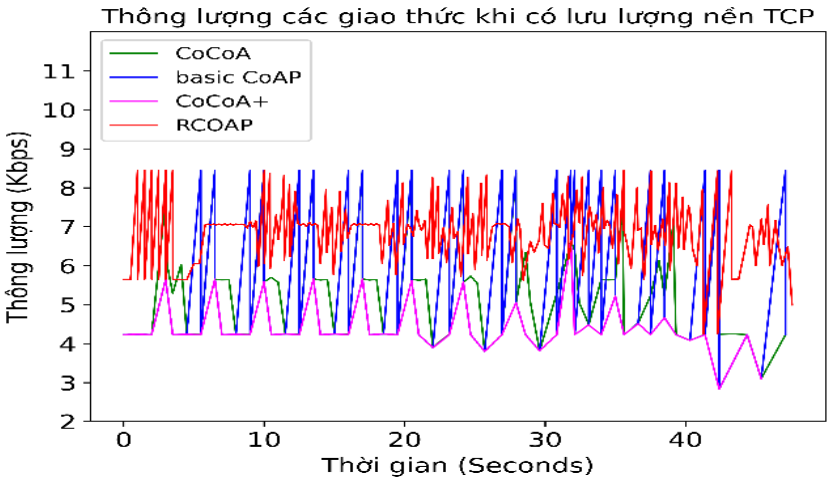
Để tạo môi trường lưu lượng động cho mô phỏng, một luồng TCP được tạo thêm bằng cách sử dụng socket TCP chuẩn của NS-3 [88]. Lưu lượng TCP biến thiên cao, cạnh tranh với các luồng CoAP, CoCoA, CoCoA+ và RCoAP, tạo sự biến động về tải lưu lượng trong mạng.

Hình 2.14 biểu thị độ trễ của các giao thức khi có lưu lượng nền TCP. Độ trễ trung bình của RCoAP là 613,25 ms, CoCoA là 800,81 ms, CoAP là 804,57 và của CoCoA+ là 941,24 ms. Độ trễ gói biến thiên theo thay đổi tải lưu lượng. Khi tải cao, độ trễ gói của CoAP, CoCoA và CoCoA+ tăng nhanh chóng và biến thiên lớn. Ngược lại, độ trễ của RCoAP có biến thiên song không vượt quá 750 ms.

Hình 2.15 so sánh thông lượng của các giao thức. Thông lượng trung bình của CoCoA+ là 4,6 Kbps, CoCoA là 5,6 Kbps, CoAP là 5,8 Kbps, và của RCoAP là 7,0 Kbps. Số gói tin phát đi thành công (kể cả phát lại) của CoAP là 80, của CoCoA là 81, của CoCoA+ là 54 và của RCoAP là 216. Thông lượng trung bình của 4 giao thức thấp hơn thí nghiệm trước do có lưu lượng TCP tương tranh băng thông.



Hình 2.14 Độ trễ khi có TCP



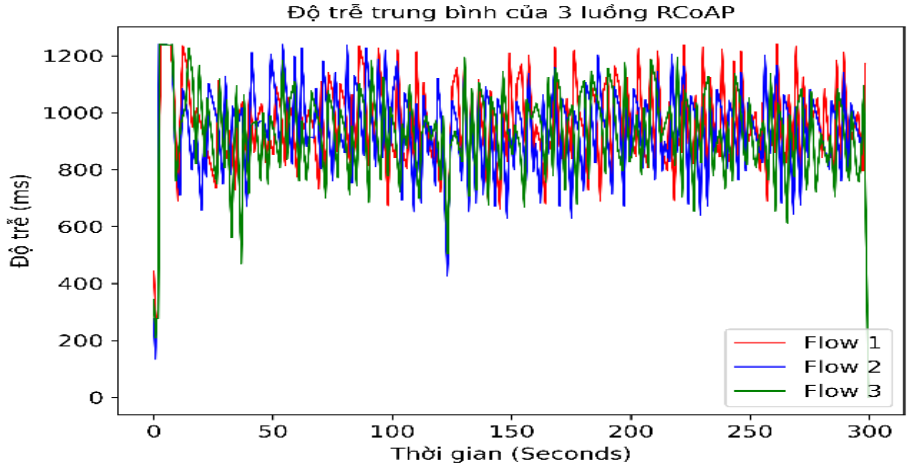
Hình 2.15 Thông lượng khi có TCP

2.4.4. Kịch bản 2.3

Kịch bản 2.3 sử dụng mô hình mạng 2.7 để đánh giá hiệu năng RCoAP trong các điều kiện độ trễ liên kết khác nhau thể hiện tình trạng tắc nghẽn với các tham số mô phỏng ở bảng 2.3.

a. Kịch bản 2.3a (độ trễ liên kết D = 70ms)

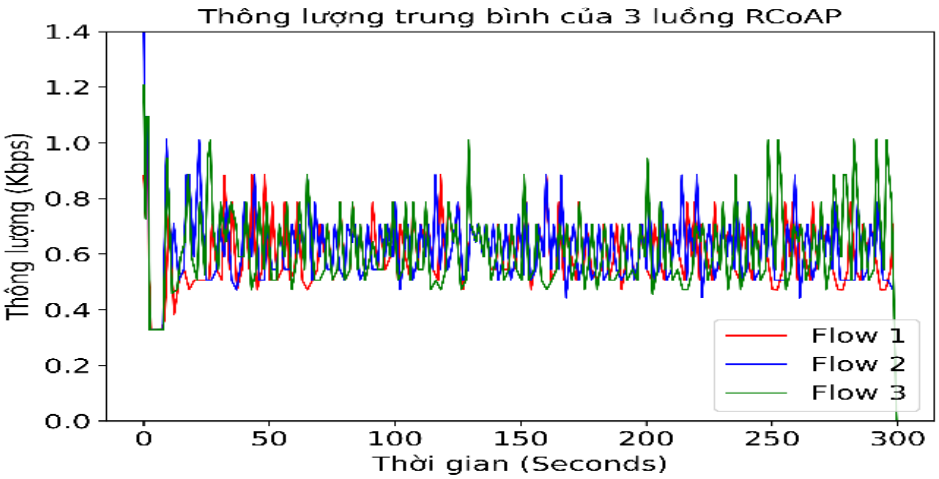
- ***Hiệu năng RCoAP***



Hình 2.16 Độ trễ 3 luồng RCoAP (trong số 10 luồng)

Trong kịch bản mô phỏng này cho phép lưu thoát nhiều gói hơn, nguy cơ xảy ra tắc nghẽn thấp trong điều kiện 10 luồng tin cùng phát đồng thời. Hình 2.17 hiển thị độ trễ trung bình của *3* luồng tin RCoAP (từ 10 luồng). Các luồng RCoAP duy trì được độ trễ ổn định giống nhau với độ trễ trung bình xấp xỉ 920 ms cho mỗi

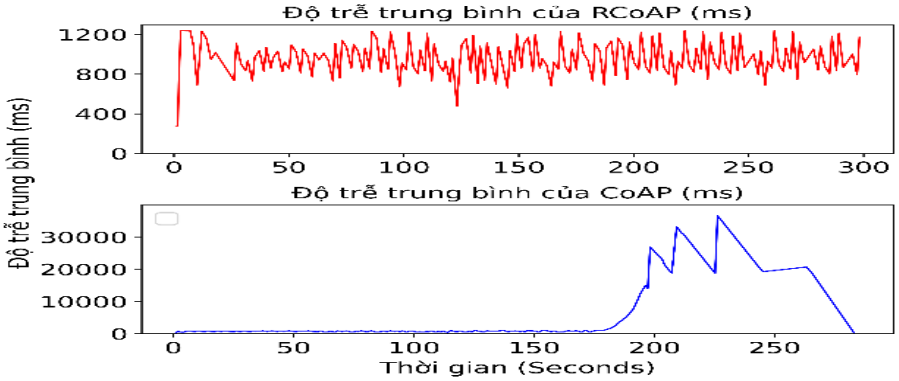
luồng tin. Hình 2.18 biểu thị thông lượng trung bình của *3* luồng tin RCoAP. Thông lượng trung bình của các luồng đạt xấp xỉ nhau với giá trị trung bình 0,7 Kbps.



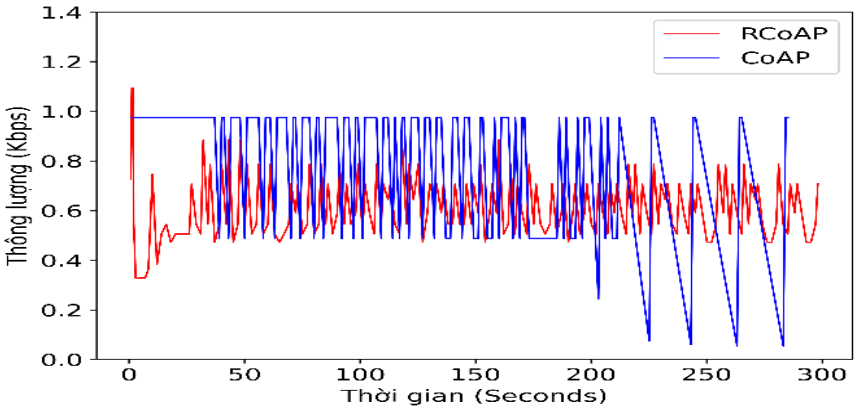
Hình 2.17 Thông lượng 3 luồng RCoAP (trong số 10 luồng)

- ***So sánh RCoAP và CoAP***

Hình 2.19 so sánh độ trễ trung bình của 10 luồng RCoAP và 10 luồng CoAP trong kịch bản 2.3a. Các luồng RCoAP duy trì được độ trễ trong khoảng 790 ms đến 1200 ms. Trong khi đó, độ trễ các luồng CoAP duy trì ở 900 ms trong khoảng từ 0s–160s, sau đó tăng nhanh và đạt tới 30s trong khoảng 170s-250s. Tắc nghẽn đã xảy ra trong khoảng này. CoAP phải phát lại nhiều gói đã mất với RTO tăng gấp đôi mỗi khi phát lại (xem số gói phát lại ở bảng 2.6).



Hình 2.18 Độ trễ trung bình của 10 luồng RCoAP và 10 luồng CoAP



Hình 2.19 Thông lượng trung bình của 10 luồng RCoAP và 10 luồng CoAP

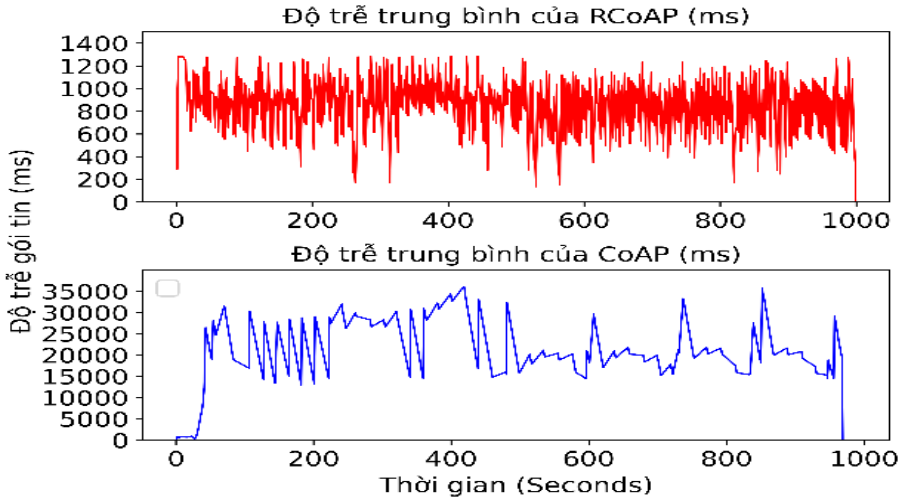
Hình 2.20 so sánh thông lượng trung bình của 10 luồng RCoAP và 10 luồng CoAP. Trong khoảng bắt đầu (0s-20s), RCoAP dò tìm băng thông cổ chai. Khi chưa xảy ra tắc nghẽn (0s-170s), thông lượng trung bình của RCoAP và CoAP xấp xỉ bằng nhau. Khi tắc nghẽn xảy ra (khoảng 170s-290s), thông lượng CoAP giảm nhanh chóng, còn RCoAP vẫn giữ được thông lượng trung bình ở 0,711 Kbps, cao hơn so với CoAP.

Bảng 2.6: So sánh RCoAP và CoAP khi độ trễ liên kết D = 70ms

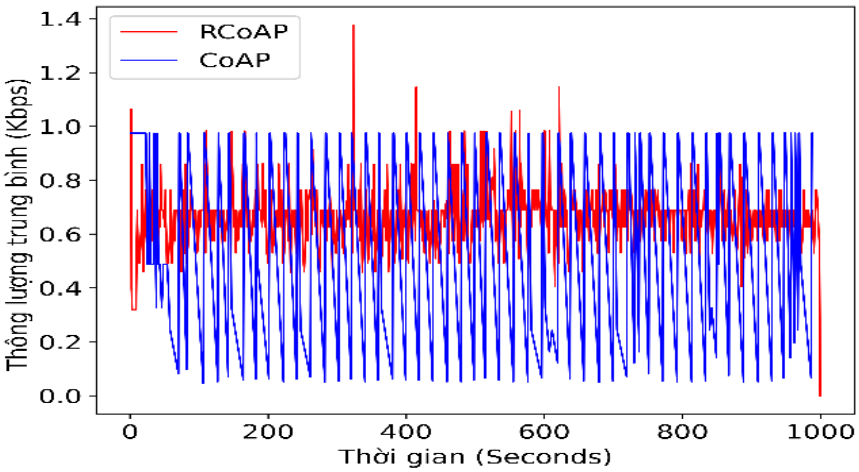
Giá trị trung bình của 10 luồng tin	**RCoAP**	**CoAP**
Tổng số gói gửi	216	162
Số gói phát thành công	216 (100%)	159 (98,21%)
Số gói phát lại	0 (0.00%)	33 (20,53%)
Số gói phát lại đúp	0 (0.00%)	29 (17,93%)
Số gói đến đích thành công	216 (100%)	130 (80,27%)
Số gói bị mất	0 (0.00%)	32 (19,73%)
Trễ trung bình	922,41ms	4175,40ms
Thông lượng trung bình	0,7111 Kbps	0,5353 Kbps

Để kiểm tra tính ổn định của cơ chế điều khiển trong thời gian hoạt động dài, thời gian mô phỏng được đặt là 1000s (≈ 17 phút). Hình 2.21 và 2.22 so sánh độ trễ trung bình và thông lượng của RCoAP và CoAP. Kết quả đo được cụ thể như sau: độ trễ trung bình của RCoAP là 844,76 ms với khoảng tin cậy (826,19; 863,33) nhỏ hơn nhiều so với 18702,23 ms với khoảng tin cậy (17320,67; 20083,79) của CoAP.

Thông lượng trung bình của RCoAP là 0,669 Kbps với khoảng tin cậy (0,657; 0,680) cao hơn so với 0,664 Kbps với khoảng tin cậy (0,591; 0,737) của CoAP. Các khoảng tin cậy được tính với mức 99%.



Hình 2.20 Độ trễ RcoAP và CoAP



Hình 2.21 Thông lượng RCoAP và CoAP

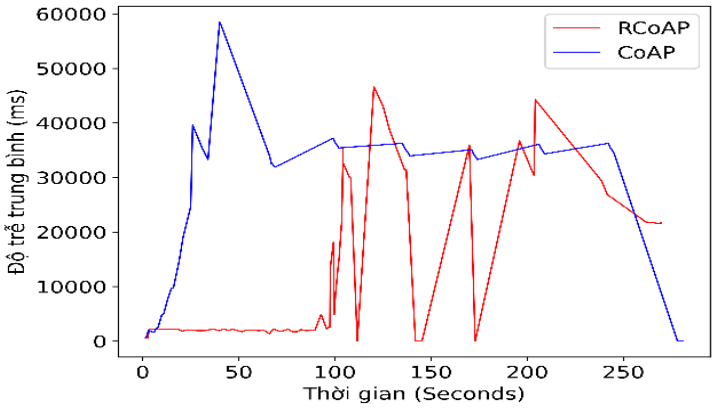
b. Kịch bản 2.3b (độ trễ liên kết D = 120ms)

- ***So sánh RCoAP và CoAP***

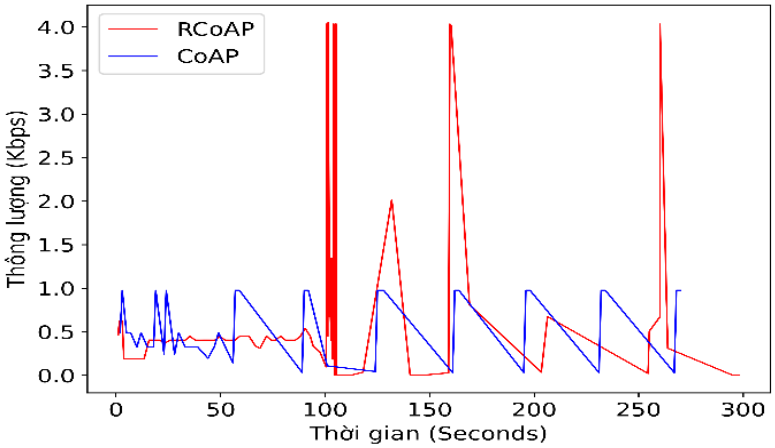
Khi thay đổi độ trễ D = 70ms lên 120ms tạo ra nguy cơ tắc nghẽn cao khi 10 luồng CoAP và 10 luồng RCoAP cùng phát đồng thời. Hình 2.23 hiển thị độ trễ trung bình của 10 luồng RCoAP và 10 luồng CoAP trong điều kiện này. Tắc nghẽn xảy ra ngay khi các luồng bắt đầu phát tin. Độ trễ tăng nhanh trong CoAP. Trong khi đó, RCoAP hạn chế được độ trễ trong khoảng 0s-100s. Sau đó độ trễ tăng do

xảy ra tắc nghẽn. Tuy nhiên, RCoAP duy trì được độ trễ trung bình nhỏ hơn CoAP.

Hình 2.24 so sánh thông lượng của RCoAP và CoAP. Do biến động của độ trễ liên kết, thông lượng của RCoAP và CoAP đều dao động song ở mức nhỏ (dưới *1* Kbps). RCoAP có xung cao do phát thêm gói thử khi có mất gói, thể hiện chuỗi gói *inflight* lớn.



Hình 2.22 Độ trễ trung bình của 10 luồng RCoAP và 10 luồng CoAP



Hình 2.23 Thông lượng trung bình của 10 luồng RCoAP và 10 luồng CoAP

Bảng 2.7 so sánh hiệu năng giữa RCoAP và CoAP khi độ trễ liên kết D = 120ms. Các tham số tương tự như trong bảng 2.6, song có giá trị nhỏ hơn do các luồng tin phát đi ít gói hơn trong điều kiện kịch bản mô phỏng này. Do thời gian chạy mô phỏng là 300s nên đến thời điểm kết thúc mô phỏng có một số gói tin inflight nên số gói bị mất (không nhận được ACK) của RCoAP nhỏ hơn số gói phát lại. Các kết quả chỉ ra các tham số hiệu năng của RCoAP tốt hơn CoAP.

Bảng 2.7: So sánh RCoAP và CoAP khi độ trễ liên kết D =120ms

Giá trị trung bình của 10 luồng tin	RCoAP	CoAP
Tổng số gói gửi	74	49
Số gói phát thành công	63 (85,33%)	44 (89,57%)
Số gói phát lại	35 (47,24%)	48 (97,55%)
Số gói phát lại đúp	27 (36,88%)	40 (82,62%)
Số gói đến đích thành công	36 (48,45%)	4 (6,95%)
Số gói bị mất	38 (51,55%)	45 (93,05%)
Trễ trung bình	10902,82ms	30665,06ms
Thông lượng trung bình	0,2459 Kbps	0,1519 Kbps

2.5. Tổng hợp các thay đổi cải tiến của RCoAP so với CoAP

Cải tiến cơ chế điều khiển tắc nghẽn cho CoAP là một yêu cầu bắt buộc như đã chỉ ra trong các RFC [100, 102, 101, 103], khuyến nghị của ITU [57, 58, 99] và trong các nghiên cứu liên quan trước đây như [36, 37, 9-11, 66].

So với CoAP, RCoAP có một số thay đổi cải tiến chính như sau:

(1) RCoAP cần thời gian khởi tạo kéo dài 2×RTT cho một phiên kết nối mới để xác định tốc độ phát ban đầu. Chi phí này nhỏ so với toàn bộ thời gian truyền tin. Ví dụ, nếu RTT = 100 ms, thời gian kết nối phiên truyền tin là 300s thì thời gian phát sinh do khởi tạo là 0,033%. Nếu phiên kết nối kéo dài thì chi phí cho thời gian khởi tạo càng nhỏ.

(2) So sánh CoAP gốc, RCoAP có những bổ sung sau:

- Tiêu đề gói tin RCoAP có nhiều hơn 4 bytes ở phần tùy chọn để gửi số thứ tự gói tin phục vụ cho kiểm tra nguyên nhân mất gói (Phụ lục 2),
- Hàm gửi (SendPacket) và hàm nhận (Receive) có thêm 6 dòng lệnh để xử lý tiêu đề gói tin,
- Hàm SendPacket có thêm 17 dòng lệnh để điều khiển tốc độ phát.
- Hàm Receiver có thêm 19 dòng lệnh để kiểm tra nguyên nhân mất gói.

(3) So sánh CoAP gốc, RCoAP có thêm 2 hàm mới là hàm Detected để kiểm tra mất gói và hàm Backoff để lùi khi có tắc nghẽn với tổng số 38 dòng lệnh.

Như vậy, RCoAP đã có thêm chi phí tính toán so với CoAP gốc, do đó thời

gian xử lý của RCoAP cao hơn so với CoAP. Tuy nhiên, cả RCoAP và CoAP đều chỉ sử dụng các vòng lặp đơn trong tính toán. Vì vậy, độ phức tạp tính toán của cả CoAP và RCoAP đều là O(n). Trong phạm vi nghiên cứu, luận án không đi sâu vào tính toán độ phức tạp, tối ưu các tham số, chi phí năng lượng tiêu thụ, tác động của lỗi kênh vô tuyến. Đây là những hướng nghiên cứu tiếp trong tương lai.

Bảng 2.8 thống kê các cải tiến chính của RCoAP so với các giao thức đã chuẩn hóa là CoAP, CoCoA và CoCoA+

Bảng 2.8: Các cải tiến của RCoAP so với CoAP, CoCoA và CoCoA+

Giao thức	Thuật toán lùi	Xác định tốc độ khởi tạo	Hỗ trợ chuỗi gói	Phân biệt mất gói	Điều khiển tốc độ
CoAP [100]	BEB	Không	Không	Không	Không
CoCoA [23]	BEB	Không	Không	Không	Không
CoCoA+ [15]	VBF	Không	Không	Không	Không
RCoAP	VBF	Có	Có	Có	Có

2.6. Kết luận chương 2

Chương 2 đã trình bày một đóng góp của luận án về xây dựng mô hình phân tích truyền tin theo chuỗi gói tin cậy và giao thức điều khiển tắc nghẽn dựa vào tốc độ RCoAP. Trong phần đầu chương, một mô hình tuyền tin theo chuỗi gói tin cậy cho CoAP được xây dựng nhằm giải quyết hạn chế (1) và (2) của CoAP đã nêu ở chương 1. Dựa vào mô hình này, luận án đã đề xuất giao thức mới RCoAP. Nội dung chương 2 đã trình bày chi tiết cơ chế hoạt động, trạng thái và các thuật toán điều khiển tắc nghẽn cho RCoAP. Chương 2 cũng thực hiện tính toán các tham số hiệu năng cho RCoAP và thực hiện mô phỏng để so sánh, đánh giá hiệu năng RCoAP với các giao thức CoAP khác. Kết quả cho thấy RCoAP có khả năng đạt hiệu năng tốt hơn các giao thức đã chuẩn hóa hiện có là CoAP, CoCoA, CoCoA+ về thông lượng, độ trễ, tỷ lệ mất gói, tỷ lệ phát thành công, tỷ lệ phát lại, tỷ lệ phát lại trùng lặp. Phần cuối chương đề cập vấn đề độ phức tạp của RCoAP so với CoAP. Yêu cầu cải tiến cơ chế điều khiển tắc nghẽn cho CoAP là bắt buộc, như đã chỉ ra trong một số tiêu chuẩn và các nghiên cứu liên quan. Độ phức tạp của RCoAP không cao hơn nhiều so với CoAP, nên RCoAP vẫn bảo đảm tính hạng nhẹ. Giao

thức RCoAP đã khắc phục được các hạn chế (3) và (4) của CoAP đã nêu ở cuối chương 1. Các kết quả đạt được trong chương 2 đã được công bố trong các công trình [**J1**] và [**J2**] cho mô hình phân tích truyền tin, [**J3**] và [**J4**] cho giao thức RCoAP.

Từ nghiên cứu ở chương 1 và 2 cho thấy: các đại lượng tác động đến tắc nghẽn cũng như trạng thái tắc nghẽn là khá mờ, không rõ ràng, có sự chồng lấp. Một hệ điều khiển mờ sẽ là một phương án khác để thực hiện cơ chế điều khiển tắc nghẽn cho CoAP. Chương 3 sẽ trình bày phương án đề xuất này.

CHƯƠNG 3. GIAO THỨC FCoAP ĐIỀU KHIỂN TẮC NGHẼN DỰA VÀO HỆ ĐIỀU KHIỂN MỜ

Chương này 3 trình bày đóng góp thứ hai của luận án: hệ điều khiển mờ và giao thức FCoAP (Fuzzy CoAP) dựa vào hệ điều khiển mờ để điều khiển tắc nghẽn. Giải pháp hệ điều khiển mờ được đưa ra nhằm khắc phục các hạn chế (4), (5) và (6) đã nêu ở chương 1. Các tham số điều khiển được lựa chọn nhằm tạo khả năng phát hiện sớm tắc nghẽn, tính toán băng thông cổ chai và lưu lượng chuỗi gói phục vụ điều khiển tắc nghẽn. Hệ điều khiển mờ được thiết kế với việc lựa chọn các đầu vào và đầu ra, giải pháp điều chỉnh tốc độ phát để điều khiển tắc nghẽn. Tiếp đó, luận án trình bày thiết kế giao thức FCoAP dựa vào hệ điều khiển mờ. Phần cuối chương thực hiện tính toán hiệu năng FCoAP và trình bày các kết quả mô phỏng đánh giá cho FCoAP so sánh với CoAP, các giao thức đã chuẩn hóa CoCoA, CoCoA+ và với giao thức RCoAP đã được đề xuất ở chương 2.

Do dùng hệ điều khiển mờ, FCoAP có điểm khác biệt so với RCoAP về khả năng phát hiện sớm tắc nghẽn, điều khiển tốc độ dựa vào tính toán băng thông cổ chai và biến thiên động của các tham số liên quan.

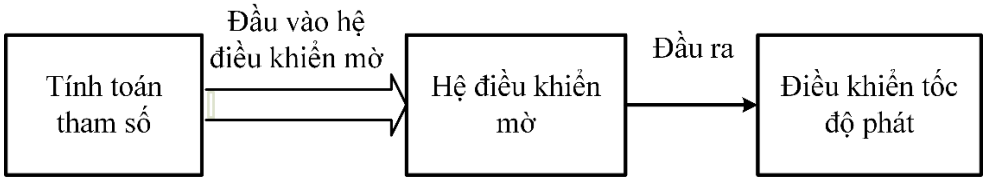
3.1. Giải pháp điều khiển tắc nghẽn sử dụng hệ điều khiển mờ

3.1.1. Sơ đồ giải pháp và các tham số điều khiển

Điều khiển tắc nghẽn phụ thuộc chủ yếu vào các tham số RTT, tải lưu lượng, thông lượng, tốc độ phát gói và mất gói trong điều kiện động của mạng. Sự biến thiên của các đại lượng này có tác động lớn đến điều khiển tắc nghẽn. Các đại lượng đều biến thiên nhanh, có tính mờ, khó xác định một cách rõ ràng, không chắc chắn, có sự chồng lấp giữa tác động của chúng và có tính phi tuyến. Rất khó có thể đưa ra một mô hình giải tích chính xác cho điều khiển tắc nghẽn. Một hệ điều khiển mờ sẽ phát huy được ưu điểm đối với hệ thống phi tuyến phức tạp này như đã phân tích ở chương 1. Giải pháp điều khiển tắc nghẽn sử dụng hệ điều khiển mờ đề xuất trong bài được mô tả trên hình 3.1.

Như đã phân tích ở mục 1.2 chương 1 và mục 2.1.2 chương 2, tình trạng tắc nghẽn có liên quan mật thiết đến RTT, băng thông cổ chai, tốc độ phát và tải lưu

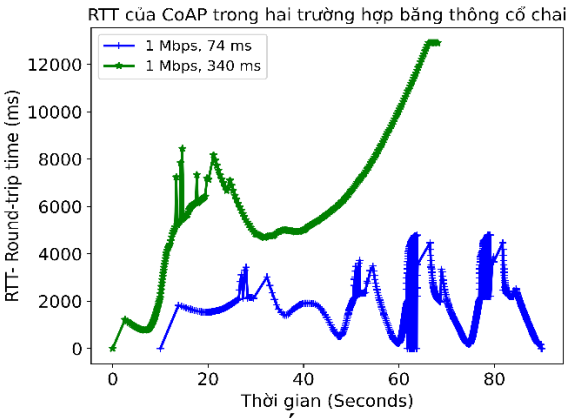
lượng. Các tham số mô tả trạng thái mạng như RTT, băng thông cổ chai, tải lưu lượng, thông lượng được tính toán để đưa vào hệ điều khiển mờ. Đầu ra hệ điều khiển mờ phục vụ cho điều khiển tốc độ phát nhằm hạn chế tắc nghẽn. Tốc độ phát được điều khiển dựa theo băng thông cổ chai và hiện trạng tải lưu lượng.

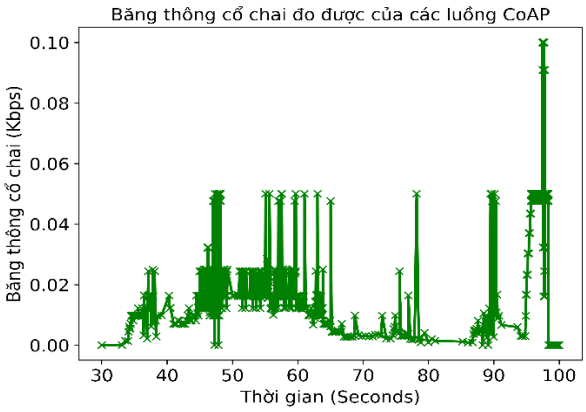


Hình 3.1 Sơ đồ giải pháp điều khiển mờ

3.1.2. Phát hiện sớm tắc nghẽn

RTT là tham số quan trọng thể hiện tắc nghẽn mạng. Khi có nhiều gói tin phát đi, tải lưu lượng gia tăng, RTT sẽ tăng nhanh phản ánh nguy cơ tắc nghẽn cao. RTT giảm thể hiện nguy cơ tắc nghẽn thấp. Để khảo sát biến thiên của RTT, một thí nghiệm mô phỏng được thực hiện cho CoAP [100] theo sơ đồ như hình 2.7 (chương 2) với băng thông cổ chai 1 Mbps. Hình 3.2a biểu thị biến thiên RTT của CoAP khi các luồng CoAP cùng chia sẻ băng thông cổ chai 1 Mbps với trễ liên kết lần lượt là 74 ms và 340 ms.





Hình 3.2 a) Biến thiên RTT *b) Biến thiên băng thông cổ chai*

Khi bắt đầu kết nối, giá trị RTT thường là nhỏ nhất. RTT biến thiên nhanh khi nhiều gói được phát đi, tương đương với tải lưu lượng tăng và nguy cơ tắc nghẽn gia tăng. RTT lớn nhất là khi xảy ra tắc nghẽn hoàn toàn (điểm gãy ở hình 2.2). Theo dõi độ lớn của RTT và chiều biến thiên tăng hay giảm của RTT có thể phát hiện sớm nguy cơ tắc nghẽn.

RTT được tính toán tại các thời điểm lấy mẫu rời rạc *k-1*, *k*. Chu kỳ lấy mẫu *T(k-1,k)* là khoảng thời gian giữa hai gói tin ACK liên tiếp. Gọi *RTT(k-1)* và *RTT(k)* là các giá trị *RTT* tại các thời điểm *k-1* và *k*, độ biến thiên tương đối của *RTT* tại thời điểm *k* sẽ là

$$\Delta RTT_r = RTT(k) - RTT(k-1) \quad (3.1)$$

Công thức (3.1) thể hiện độ lớn và chiều biến thiên (tăng hay giảm) của *RTT* tại mỗi thời điểm *k*.

3.1.3. Tính toán băng thông cổ chai và tải lưu lượng chuỗi gói

- ***Băng thông cổ chai***

Băng thông cổ chai *BW(k)* ở thời điểm *k* có đơn vị là gói/s (hoặc bit/s) được tính theo công thức tương tự như trong [27, 11] như sau:

$$BW(k) = max\ (\theta(k)) \quad với\ \forall k \in [T\text{-}RTT, T] \quad (3.2)$$

Trong đó $\theta(k)$ là thông lượng đo được ở thời điểm *k*, *T* là thời điểm nhận được ACK, *T-RTT* là thời điểm nhận được ACK trước đó. Băng thông cổ chai được định nghĩa là băng thông hẹp nhất khi truyền một luồng gói tin từ bên gửi tới bên nhận [27], có thể coi là tốc độ phát lớn nhất được phép ở trạng thái mạng hiện tại. Băng thông cổ chai càng nhỏ thì thông lượng cho phép càng nhỏ, nguy cơ tắc nghẽn càng cao. Ngược lại, nếu băng thông cổ chai lớn thì nguy cơ tắc nghẽn ít hơn.

Hình 3.2b biểu thị biến thiên băng thông cổ chai khi các luồng tin CoAP cùng chia sẻ liên kết trong điều kiện thí nghiệm nêu trên. Băng thông cổ chai càng nhỏ thì nguy cơ tắc nghẽn càng cao và ngược lại.

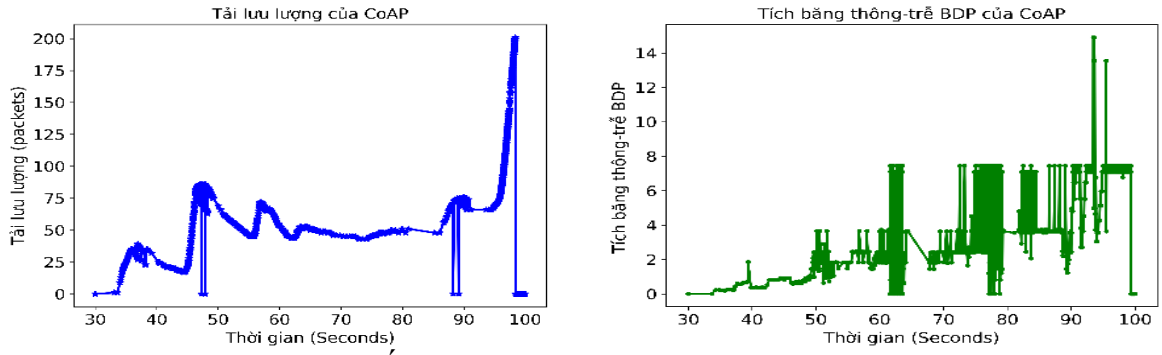
- ***Tải lưu lượng chuỗi gói***

Tải lưu lượng là tổng số gói tin đang gửi cho tới thời điểm hiện tại, bao gồm cả gói *inflight*, gói mới phát tiếp và gói vừa có ACK. Các gói *inflight* có tác động lớn đến nguy cơ tắc nghẽn. Khái niệm tích số băng thông-trễ (BDP-Bandwidth Delay Product) được đưa ra [11, 26] để kiểm soát số gói *inflight*. Gọi *BDP(k)* là số gói tin tối đa được phép phát đi tại một thời điểm *k*, ta có:

$$BDP(k) = BW_{max}(k) \times RTT_{min} \quad (3.3)$$

Với $BW_{max}(k)$ là băng thông cổ chai lớn nhất của kết nối tại *k*, RTT_{min} là giá trị

RTT nhỏ nhất tới thời điểm hiện tại. Hình 3.3 biểu thị biến thiên tải lưu lượng chuỗi gói và BDP khi các luồng CoAP cùng chia sẻ liên kết như ở thí nghiệm nêu trên. Tải càng lớn, BDP (nghĩa là số gói *inflight*) càng lớn thì nguy cơ tắc nghẽn càng cao và ngược lại. Băng thông cổ chai lớn thi nguy cơ tắc nghẽn ít hơn, số gói tin *inflight* có thể nhiều để truyền chuỗi gói.



Hình 3.3 Biến thiên tải lưu lượng và BDP của CoAP

3.1.4. Lựa chọn đầu vào cho hệ điều khiển mờ

Theo sơ đồ ở hình 3.1 và theo phân tích ở chương 1, một hệ điều khiển mờ FCS có thể gồm nhiều đầu vào và nhiều đầu ra, song hệ MISO là phù hợp cho bài toán điều khiển tắc nghẽn trong luận án. Việc lựa chọn đầu ra và đầu vào cho hệ FCS rất quan trọng, đặc biệt đối với yêu cầu đơn giản và gọn nhẹ cho các thiết bị IoT. Như đã chỉ ra trong [124, 108], số lượng đầu vào càng nhiều thì độ phức tạp càng cao, các đầu vào càng ít có sự liên quan đến nhau càng tốt. Vì vậy, luận án lựa chọn **hai đầu vào cho hệ FCS** cụ thể như sau.

- ***Đầu vào RT_gradient***

Như đã nêu ở mục 3.1.2, độ lớn và chiều biến thiên RTT được theo dõi để phát hiện sớm tắc nghẽn. Biến thiên RTT cũng đã được sử dụng trong [11, 10, 9] để dự đoán tắc nghẽn. Do vậy, có thể chọn RTT làm một đầu vào cho hệ FCS. Tuy nhiên, thay vì dùng độ biến thiên tương đối của RTT theo công thức (3.1), luận án đề xuất sử dụng độ biến thiên tuyệt đối của RTT như sau:

$$\Delta RTT_a = RTT(k) - RTT_{min}(k) \quad (3.4)$$

Trong đó, $RTT_{\min}(k)$ là giá trị nhỏ nhất của RTT tới thời điểm *k* hiện tại với:

$$RTT_{min}(k) = \min\big(RTT_{min}(k-1), RTT(k)\big) \quad (3.5)$$

Dùng độ biến thiên tuyệt đối ΔRTT_a sẽ tốt hơn vì không phải duy trì các giá trị đo RTT liên tiếp để xác định chiều hướng tăng hay giảm. ΔRTT_a càng lớn thì nguy cơ tắc nghẽn càng cao và ngược lại, ΔRTT_a càng nhỏ thì nguy cơ tắc nghẽn càng thấp. Để chuẩn hóa đầu vào hệ FCS, luận án định nghĩa một đại lượng gọi là độ dốc RTT (RTT-Gradient) viết tắt là RT(k) cho đầu vào thứ nhất như sau:

$$RT(k) = \frac{RTT_S(k) - RTT_{min}}{RTT_{max}(k) - RTT_{min}} \quad (3.6)$$

Giá trị của *RT(k)* nằm trong khoảng *[0, 1]*. *RT(k)* bằng *0* nếu $RTT_s(k) = RTT_{min}$ và bằng *1* nếu $RTT_s(k) = RTT_{max}(k)$.

$RTT_s(k)$ là giá trị RTT tính được ở thời điểm *k*, $RTT_{max}(k)$ là giá trị RTT lớn nhất ở thời điểm k hiện tại. So với (3.4), công thức (3.6) chỉ lấy một giá trị RTT_{min} thay vì lấy ở từng chu kỳ *k*. Lý do là giá trị nhỏ nhất RTT_{min} được xác định trong thời gian khởi tạo kết nối và duy trì trong suốt thời gian kết nối.

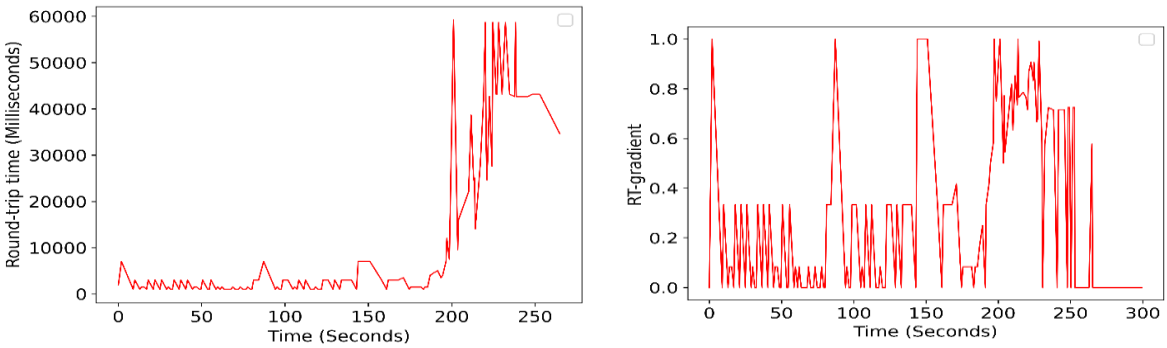
$RTT_s(k)$ tính theo công thức [104]:

$$RTT_s(k) = 0{,}75 \times RTT_s(k-1) + 0{,}25 \times RTT_m(k) \quad (3.7)$$

Trong đó, $RTT_m(k)$ là giá trị RTT thực tế đo được tại thời điểm *k*. Công thức (3.7) giúp làm mịn kết quả đo và đưa ra ước lượng RTT trung bình cho tổng số *k* gói tin đã phát đi. Tuy nhiên, làm mịn sẽ giảm sự biến thiên tức thời của RTT. Giá trị $RTT_{max}(k)$ có thể liên tục cập nhật theo các chu kỳ lấy mẫu bằng công thức:

$$RTT_{max}(k) = \max\left(RTT_{max}(k-1), RTT_m(k)\right) \quad (3.8)$$

Trong đó, *k* và *k-1* là các bước lấy mẫu rời rạc của chu trình điều khiển. Chu kỳ lấy mẫu *T(k)* là một khoảng giữa hai lần nhận gói tin ACK.



Hình 3.4 a) Biến thiên RTT khi tắc nghẽn và b) Biến thiên tương ứng của RT(k)

Để khảo sát biến thiên của *RTT* và *RT_gradient*, một thí nghiệm mô phỏng thực hiện theo sơ đồ hình 2.7 (chương 2) với băng thông cổ chai 250 Kbps, trễ liên kết 50 ms với 10 luồng CoAP đồng thời phát trong 300s. Hình 3.4a biểu thị biến thiên *RTT* khi tắc nghẽn xảy ra, hình 3.4b là biến thiên tương ứng của *RT(k)* giữa 0 và 1. *RT(k)* càng gần *1* biểu thị nguy cơ tắc nghẽn càng cao. Do vậy, *RT(k)* dùng cho phát hiện sớm tắc nghẽn.

- ***Đầu vào BG_gradient***

Để thể hiện băng thông cổ chai và tải lưu lượng chuỗi gói, luận án định nghĩa một đại lượng gọi là độ dốc *BG (BG_gradient)* viết tắt là *BG(k)* cho đầu vào thứ hai như sau:

$$BG(k) = \frac{\min(BW_{max}(k),\ \theta_t(k))}{BW_{max}(k)} \tag{3.9}$$

$$\theta_t(k) = \frac{De(k-1,\ k)}{T(k-1,\ k)} \tag{3.10}$$

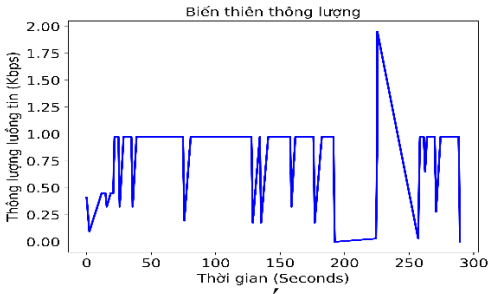
Giá trị *BG(k)* được chuẩn hóa trong khoảng *[0, 1]*. *BG(k)=0* nếu $\theta_t(k)=0$, *BG(k)=1* nếu $\theta_t(k) \geq BW_{max}(k)$. $BW_{max}(k)$ là băng thông cổ chai ở thời điểm *k*, $\theta_t(k)$ là thông lượng tức thời của luồng tin trong *T(k-1, k)*, *De(k-1, k)* là lượng dữ liệu đến bên nhận trong khoảng *T(k-1, k)*. Cần lưu ý, thông lượng tức thời $\theta_t(k)$ đo được có thể lớn hơn băng thông cổ chai ở thời điểm xảy ra tắc nghẽn. Do đó cần hàm *min*. Điều khiển tốc độ phát là nhằm giảm thông lượng tức thời so với băng thông cổ chai hiện có để giảm tắc nghẽn.

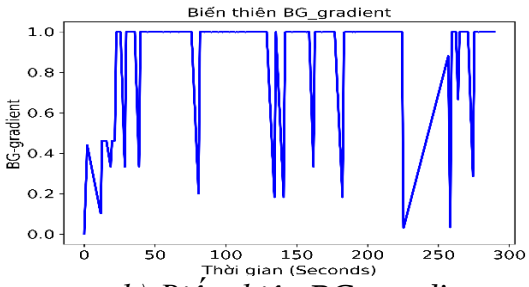
Giá trị $BW_{max}(k)$ có thể được xác định ở thời điểm khởi tạo kết nối, và được cập nhật liên tục trong quá trình kết nối. Để ước tính giá trị BW_{max}(k) ban đầu, bên gửi CoAP sẽ phát liên tiếp các gói tin thử tới máy chủ trong giai đoạn khởi tạo. Máy chủ sẽ tính hiệu số thời gian đến nhỏ nhất (ΔT_{min}) giữa hai lần nhận gói CON liên tiếp và gắn giá trị này vào gói ACK phản hồi về bên gửi. Bên gửi tính BW_{max}(k) dựa vào hiệu số ΔT_{min} như sau.

$$BW_{max}(k) = \max(BW(k),\ \frac{b}{\Delta T_{min}}) \tag{3.11}$$

Trong đó, *b* là lượng bit nhận thêm trong ΔT_{min} khi CoAP phát gói tin thử.

BW(k) là băng thông cổ chai tính được ở bước *k*. Hình 3.5a mô tả biến thiên của thông lượng và hình 3.5b là biến thiên tương ứng của BG_gradient trong thí nghiệm mô phỏng đã nêu ở mục trên. *BG(k)* càng gần tới *1*, nguy cơ tắc nghẽn càng cao. Nếu mạng tắc nghẽn ít, *BG(k)* sẽ gần với giá trị *0*.





Hình 3.5 a) Biến thiên thông lượng *b) Biến thiên BG_gradient*

3.1.5. Lựa chọn đầu ra cho hệ điều khiển mờ

Để thiết kế gọn nhẹ, luận án sử dụng một đầu ra cho hệ FCS. Luận án định nghĩa một hệ số ***C_Degree*** (Congestion Degree) làm đầu ra cho hệ FCS và dựa trên công thức tăng tốc độ phát của mô hình phân tích trong chương 2, nghiên cứu sinh đề xuất công thức điều khiển tính tốc độ phát như sau.

$$R(k) = R(k-1) + C\_degree(k) \times \frac{1}{RTT_S(k)} \quad (3.12)$$

Trong đó *R(k)* và *R(k-1)* là tốc độ phát tương ứng ở thời điểm *k* và *k-1* trước đó. *C_Degree(k)* biểu thị cấp độ tắc nghẽn tại *k* và có giá trị trong khoảng *[-1; 1]*. Lý do sử dụng C_degree trong khoảng *[-1; 1]* để biểu thị trạng thái tắc nghẽn mạng có thể giải thích như sau:

- Nếu *C_Degree* $\in$ *(0; 1]*, mạng được xác định là ít tắc nghẽn. C_degree càng lớn hơn *0* và càng gần *1*, trạng thái mạng sẽ được coi là càng ít tắc nghẽn. Nếu *C_degree=1*, mạng được coi là không có tắc nghẽn. Điểm hoạt động của FCS tương ứng khoảng điểm gập ở hình 2.2 ở chương 2. Tốc độ phát gói tin có thể được điều chỉnh tăng để tăng hiệu năng truyền tin.
- Nếu *C_Degree* $\in$ *[-1; 0)*, mạng được xác định là có tắc nghẽn. C_degree càng âm và càng gần *-1*, trạng thái mạng sẽ được coi là càng tắc nghẽn. Nếu *C_degree=-1*, mạng được coi là tắc nghẽn hoàn toàn. Giá trị *C_degree = -1*

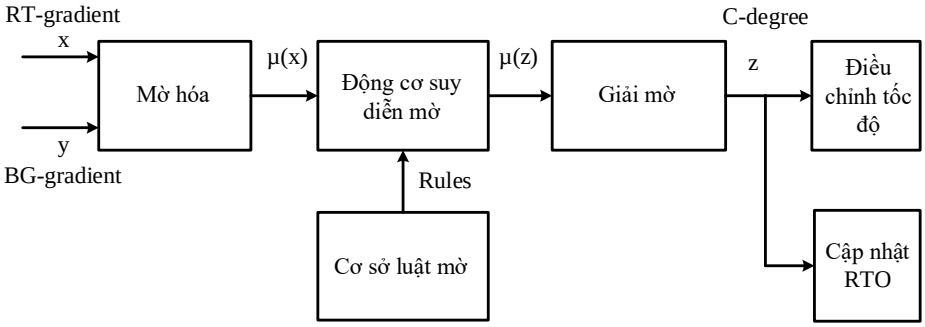
tương ứng với điểm gẫy trên hình 2.2 ở chương 2. Cần điều chỉnh giảm tốc độ phát theo giá trị *C_Degree* để giảm tắc nghẽn.

- Nếu *C_Degree = 0*, mạng ở trạng thái cân bằng không có sự thay đổi lưu lượng, không cần điều chỉnh tốc độ phát.

Như vậy có thể thấy, biến *C_degree* cung cấp cả hướng và biên độ để điều chỉnh tốc độ phát của luồng tin. Tải lưu lượng thấp tương ứng với nguy cơ tắc nghẽn thấp và *C_degree* dương. Ngược lại, nếu tải lưu lượng cao thì nguy cơ tắc nghẽn sẽ cao ứng với *C_degree* âm. Với *C_degree*, hệ FCS có thể phát hiện sớm tắc nghẽn, xác định được hướng tăng hay giảm và cả biên độ cần điều chỉnh tăng / giảm cho tốc độ phát để hạn chế tắc nghẽn.

3.2. Thiết kế hệ thống điều khiển mờ

Hệ thống điều khiển mờ cho CoAP được đề xuất như trên hình 3.6 với 2 đầu vào và 1 đầu ra. Kiến trúc hệ thống tương tự như kiến trúc chung của một hệ FCS đã trình bày ở mục 1.4.2 chương 1 với các thành phần mờ hóa, động cơ suy diễn, tập luật mờ và phần giải mờ.



Hình 3.6 Mô hình hệ thống điều khiển mờ

3.2.1. Mờ hóa

Nhiệm vụ của thành phần này là biểu diễn các đầu vào và đầu ra với các biến ngôn ngữ và diễn giải các hàm thuộc của chúng. Khối mờ hóa biến đổi dữ liệu đầu vào ở dạng rõ thành dữ liệu mờ. Mỗi đầu vào có thể có nhiều hạng thức và thường là số lẻ, số lượng thường từ 3 đến 7 như khuyến nghị trong [126]. Có thể nhận thấy, con số này là tùy thuộc vào tính phức tạp của các biến đầu vào và đầu ra. Không có quy tắc cụ thể cho việc chọn số lượng, Tuy nhiên, nếu số hạng thức nhiều, độ phức

tạp hệ thống sẽ tăng. Trong trường hợp CoAP, luận án đề xuất sử dụng 3 hạng thức cho mỗi đầu vào và 5 hạng thức cho đầu ra.

Đầu vào *RT_gradient* và *BG_gradient* được chuẩn hóa trong khoảng *[0, 1]*. Như định nghĩa trong [126], các hạng thức có giá trị không phải là con số, mà là từ ngữ. Vì vậy, luận án định nghĩa các hạng thức ("nhỏ", "trung bình", "lớn") cho các đầu vào như sau:

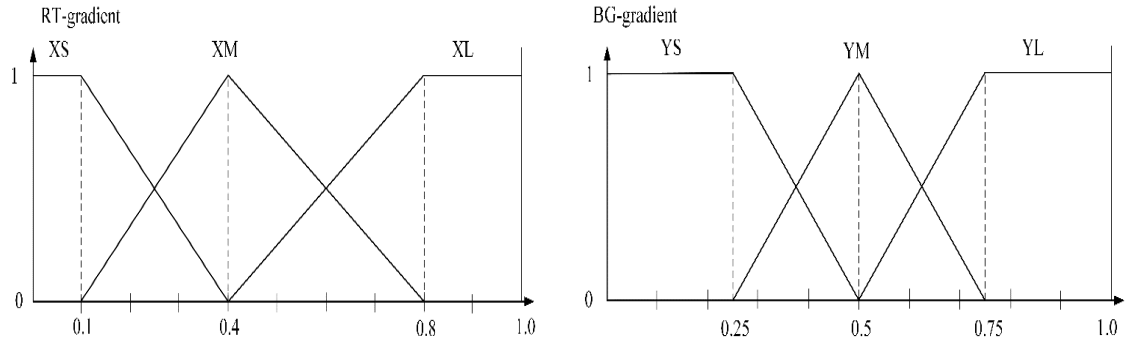
- *RT_gradient: small, medium, large*
- *BG_gradient: small, medium, large*

Đối với đầu ra *C_degree*, bài định nghĩa 5 hạng thức: "rất thấp", "thấp", "trung bình", "cao", "rất cao" như sau:

- *C_degree*: *very low, low, medium, high, very high*

Tương ứng với mỗi hạng thức là một tập mờ con. Các tập mờ con như đã trình bày ở mục 1.4.1. Tập mờ tam giác và hình thang là những tập mờ được sử dụng phổ biến trong điều khiển và luận án sử dụng những tập mờ này để biểu thị các hàm thuộc cho các biến ngôn ngữ ở đầu vào và đầu ra của hệ điều khiển mờ.

Như đã trình bày trong chương 1, để triển khai hệ điều khiển mờ thực tế với tập mờ tam giác cần xác định các điểm a, m, d và tập mờ hình thang cần xác định các điểm a, b, c, d. Đối với đầu vào các giá trị này có thể được đặt dựa vào việc xác định mức độ ảnh hưởng của các khoảng giá trị đầu vào đối với tắc nghẽn. Chúng có thể được căn chỉnh chính xác sau triển khai thử nghiệm. Luận án này đề xuất các hàm thuộc của các hạng thức cho các đầu vào hệ điều khiển mờ như trên hình 3.7a cho *RT_gradient* và hình 3.7b cho *BG_gradient*.



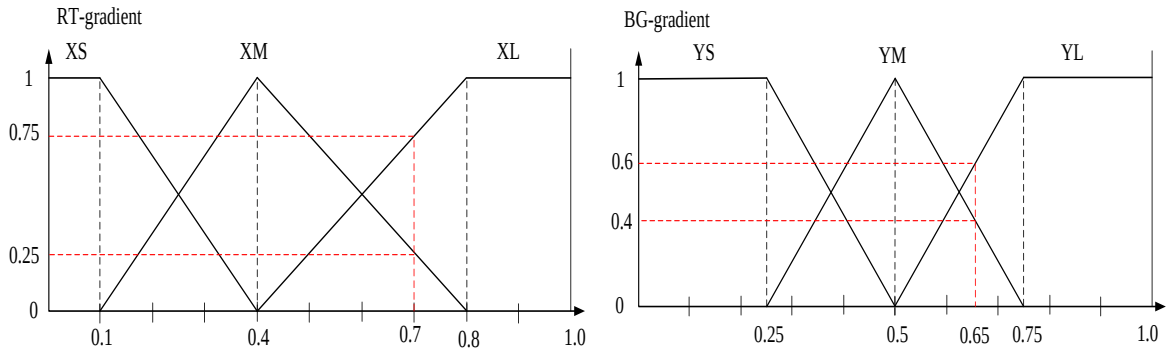
Hình 3.7 a) Các hàm thuộc cho RT_gradient, b) Các hàm thuộc cho BG_gradient

Ba tập mờ con cho các hạng thức của đầu vào *RT_gradient* là: “small (XS)”, “medium (XM)”, và “large (XL)”. Tương tự có ba tập mờ cho các hạng thức của đầu vào BG_gradient là: “small (YS)”, “medium (YM)”, và “large (YL)”. Ký hiệu X dùng cho đầu vào *RT_gradient*, ký hiệu Y dùng cho đầu vào *BG_gradient*. Có thể thấy các vùng chồng lấp giữa các tập mờ hạng thức. Kích thước các vùng chồng lấp phụ thuộc vào đặc trưng của đầu vào.

Có thể giải thích thêm về đồ thị của các biến ngôn ngữ này như sau. Với đầu vào *RT_gradient* đã được chuẩn hóa trong khoảng [0; 1], các giá trị rõ từ 0 đến 0,4 tương ứng với hạng thức “small”, từ 0,1 – 0,8 tương ứng với “medium”, từ 0,4 -1 tương ứng với “large”. Với các giá trị rõ trong khoảng [0; 0,1], chính xác tại 0,4 và trong khoảng [0,8; 1] các tập mờ tam giác và hình thang đạt độ thuộc bằng 1.

Tương tự với đầu vào BG_Gradient đã được chuẩn hóa trong khoảng [0; 1], các giá trị rõ từ 0 đến 0,5 tương ứng với hạng thức “small”, từ 0,25 – 0,75 tương ứng với “medium”, từ 0,5 -1 tương ứng với “large”. Với các giá trị rõ trong khoảng [0, 0,25], chính xác tại 0,5 và trong khoảng [0,75; 1] các tập mờ tam giác và hình thang đạt độ thuộc bằng 1.

Ví dụ: với các đầu vào đã được chuẩn hóa RT_gradient (x = 0,7) và BG – Gradient (y = 0,65), có thể đọc độ thuộc từ đồ thị các tập hạng thức trên hình 3.8a và hình 3.8b.



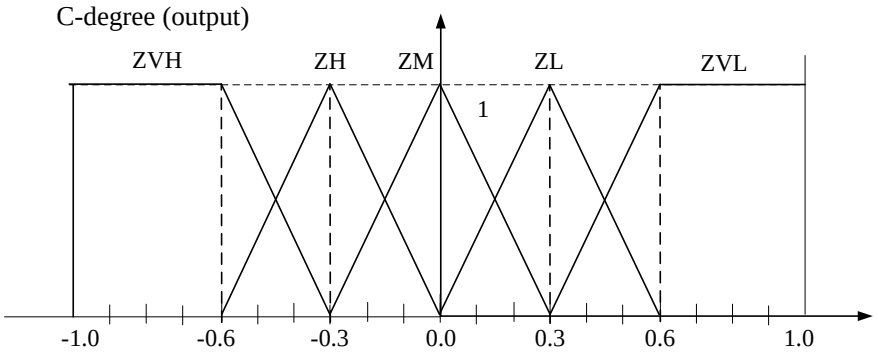
Hình 3.8 a) Độ thuộc RT_gradient=0,7, b) Độ thuộc BG_gradient=0,65

Kết quả thu được như sau: $\mu_{XL}(x) = 0{,}75$; $\mu_{XM}(x) = 0{,}25$

và $\mu_{YM}(y) = 0{,}4$; $\mu_{YL}(y) = 0{,}6$

Với đầu ra *C_degree*, biến ngôn ngữ này sẽ được chia thành 5 tập mờ là: very

low (ZVL), low (ZL), medium (ZM), high (ZH), very high (ZVH). Ký hiệu Z là dùng cho đầu ra. Các hàm thuộc của các tập mờ cho đầu ra biểu thị trên hình 3.9.



Hình 3.9 Các hàm thuộc cho đầu ra C_degree

Đầu ra *C- degree* đã được chuẩn hóa trong khoảng [-1;1], các giá trị rõ từ -1 đến -0,3 tương ứng với hạng thức "very high", từ -0,6 đến 0,0 tương ứng với "high", từ -0,3 đến 0,3 tương ứng với "medium", từ 0 đến 0,6 tương ứng "low", từ 0,3 đến 1 tương ứng với "very low". Với các giá trị rõ trong khoảng [-1,0; -0,6], chính xác tại -0,3, 0,0, 0,3 và trong khoảng [0,6; 1] các tập mờ tam giác và hình thang đạt độ thuộc bằng 1.

3.2.2. Cơ sở luật mờ

Cơ sở luật mờ chứa các luật mờ để xác định các biến ngôn ngữ đầu ra dựa vào các biến ngôn ngữ đầu vào. Tập luật mờ được đề xuất trong bảng 3.1.

Bảng 3.1: Tập luật mờ

Rules	**RT_gradient**	**BG_gradient**	**Congestion Degree**
1	*Small (XS)*	*Small (YS)*	*Very Low (ZVL)*
2	*Small (XS)*	*Medium (YM)*	*Very Low (ZVL)*
3	*Small (XS)*	*Large (YL)*	*Low (ZL)*
4	*Medium (XM)*	*Small (YS)*	*Low (ZL)*
5	*Medium (XM)*	*Medium (YM)*	*Medium (ZM)*
6	*Medium (XM)*	*Large (YL)*	*Medium (ZM)*
7	*Large (XL)*	*Small (YS)*	*High (ZH)*
8	*Large (XL)*	*Medium (YM)*	*High (ZH)*
9	*Large (XL)*	*Large (YL)*	*Very High (ZVH)*

Hệ mờ của luận án là một hệ MISO với hai đầu vào và một đầu ra. Do hai đầu vào, mỗi đầu vào có ba hạng thức nên bộ luật của hệ mờ có 9 luật. Trong tập luật mờ ở bảng 3.1, toán tử AND được sử dụng để kết hợp giữa các đầu vào. Ví dụ, một luật có thể được mô tả như sau:

Luật 3: IF *RT_gradient* là "small" AND BG_gradient là "large", THEN *C_degree* sẽ là "low".

3.2.3. Mô tơ suy diễn mờ

Mô tơ suy diễn mờ có nhiệm vụ xác định tập mờ đầu ra dựa vào các tập mờ đầu vào và cơ sở luật mờ. Tiến trình suy diễn gồm 2 bước:

(1) Tính toán độ thuộc của tập kết quả dựa vào cơ sở luật mờ trong Bảng 3.1.

(2) Sử dụng phương pháp suy diễn mở để tạo ra các tập mờ đầu ra. Có một số phương pháp suy diễn mờ như đã nêu ở mục 1.4 chương 1 như max-min, max-product, singletons. Trong luận án này, phương pháp suy diễn max-min được sử dụng vì tốc độ tính toán nhanh.

Ví dụ: Với các độ thuộc thu được ở 3.2.1, các bước sẽ được thực hiện như sau:
Bước 1: Sử dụng bảng 3.1 với các độ thuộc trên ta thấy các luật 5, 6, 8, 9 sẽ được sử dụng. Do có 2 điều kiện là *RT_gradient* và *BG_gradient* thì độ thuộc của tập hợp kết quả sẽ là sự kết hợp giữa các điều kiện này với nhau ở đây là sự kết hợp AND tương ứng với toán tử min giữa các độ thuộc. Cụ thể như sau:

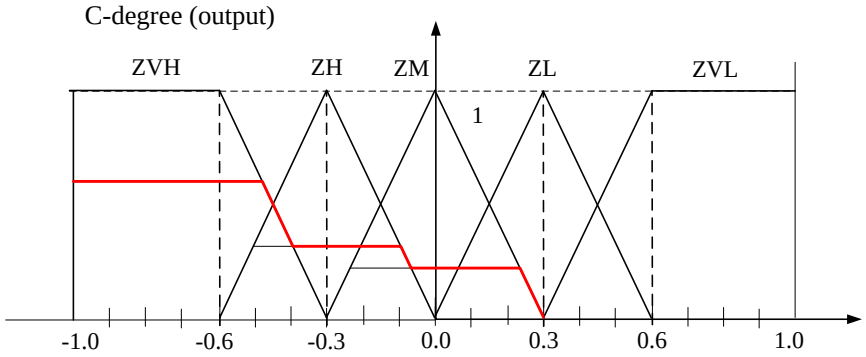
Luật 5: IF *RT_gradient* là "medium" AND *BG_gradient* là "medium", THEN *C_degree* sẽ là "medium". Với $\mu_{XM}(x) = 0{,}25$ và $\mu_{YM}(y) = 0{,}4$, chọn toán tử min cho phép AND thì $\mu_{ZM}(z) = \min(0{,}25;\ 0{,}4) = 0{,}25$.

Tương tự với các luật 6, 8, 9 ta có:

$\mu_{ZM}(z) = 0{,}25;\ \mu_{ZH}(z) = 0{,}4;\ \mu_{ZVH}(z) = 0{,}6.$

Do hai luật 5 và 6 đều cho ra kết quả là tập mờ ZM, trong trường hợp này 2 giá trị độ thuộc kết quả bằng nhau nên $\mu_{ZM}(z) = 0{,}25$ không đổi. Trong trường hợp các giá trị độ thuộc kết quả khác nhau ta cần chọn chiến lược kết hợp các điều kiện sao cho có thể phản ánh được đặc thù của quá trình tốt như có thể. Các chiến lược đó có thể là chọn cực đại hoặc trung bình cộng hoặc tổng hai tập mờ.

Bước 2: Suy diễn bằng phương pháp max – min ta có đồ thị kết quả như hình 3.10, phần dưới đường màu đỏ là tập mờ kết quả của đầu ra



Hình 3.10: Tập mờ kết quả đầu ra.

3.2.4.Giải mờ, điều chỉnh tốc độ phát, cập nhật RTO

Giải mờ là thực hiện chuyển đổi dữ liệu mờ đầu ra thành dữ liệu rõ. Như đã nêu ở mục 1.4 chương 1, phương pháp xấp xỉ trọng tâm CoG là phổ biến hơn. Luận án sử dụng phương pháp này để giải mờ. Các kết quả rõ ở đầu ra có thể tính được xấp xỉ bằng công thức trung bình tâm như sau.

$$C\_degree = Z_c = \frac{\sum_{i=1}^{N} z_i \times \mu(z_i)}{\sum_{i=1}^{N} \mu(z_i)} \quad (3.13)$$

Trong đó Z_c là đầu ra rõ (trung bình tâm của phần diện tích hợp bởi các tập mờ N), z_i là trọng tâm của mỗi tập mờ đầu ra, $\mu(z_i)$là độ thuộc tương ứng. Giá trị Z_c ở đầu ra là *C_degree*, có giá trị trong khoảng giữa [-1; 1], biểu thị trạng thái tắc nghẽn như đã chỉ ra ở mục 3.2.4 trên. Áp dụng công thức với ví dụ xuyên suốt các mục 3.2.1; 3.2.2; 3.2.3 ta có:

$$C\_degree = Z_c = \frac{0 \times 0{,}25 + (-0{,}3) \times 0{,}4 + (-0{,}8) \times 0{,}6}{0{,}25 + 0{,}4 + 0{,}6} = -0.48$$

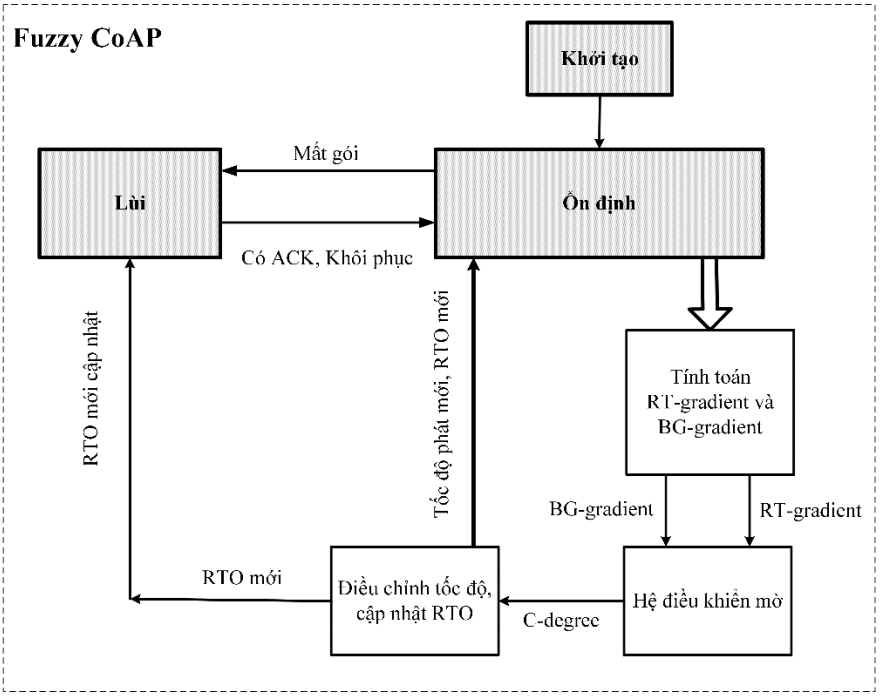
Giá trị Z_c được dùng để điều chỉnh tốc độ phát và cập nhật RTO. Trong trường hợp này đã tắc nghẽn ở mức trung bình cần phải điều chỉnh tốc độ phát.

3.3. Giao thức FCoAP cho điều khiển tắc nghẽn mạng

3.3.1. Cơ chế điều khiển của FCoAP

FCoAP là viết tắt của Fuzzy CoAP. Giao thức FCoAP là một phiên bản cải tiến cho CoAP [100] sử dụng hệ điều khiển mờ để phát hiện sớm tắc nghẽn và điều

chỉnh tốc độ để giảm tắc nghẽn. Hình 3.11 là sơ đồ tổng quát của giao thức FCoAP đã đề xuất. Các khối màu đậm trên hình là các khối chức năng hoạt động chính. Các khối còn lại gồm các chức năng bổ trợ cho FCoAP.



Hình 3.11 Cơ chế điều khiển FCoAP (Fuzzy CoAP)

FCoAP hoạt động ở trên UDP. Khác với CoAP, FCoAP không phải cơ chế dựa vào mất gói, mà là cơ chế dựa vào tốc độ với các tính năng bổ sung mới. Điểm mới của FCoAP so với CoAP gồm: Hỗ trợ truyền chuỗi gói tin cậy, phát hiện sớm tắc nghẽn, tính toán biến thiên của các tham số và băng thông cổ chai, điều khiển tốc độ phát dựa vào đầu ra hệ FCS để giảm tắc nghẽn. Các tham số được tính theo thời gian thực gồm: RTT, băng thông cổ chai, thông lượng tức thời, số gói tin *inflight*. Các đại lượng này được bao hàm trong hai biến là RT_gradient và BG_gradient của hệ FCS. Giao thức FCoAP sử dụng tiêu đề chuẩn của CoAP [100] song có bổ sung 8 bytes trong tùy chọn tiêu đề gói tin, trong đó 4 bytes để ghi số thứ tự gói tin phục vụ kiểm tra mất gói, 4 bytes để ghi hiệu số thời gian nhận phục vụ cho tính thông lượng tức thời và băng thông cổ chai (Phụ lục 4).

Cơ chế hoạt động của FCoAP cụ thể như sau. Khi bắt đầu kết nối (trạng thái khởi tạo), bên gửi FCoAP thực hiện tính các giá trị $\theta_t(k)$ và $BW_{max}(k)$ trong khoảng

thời gian từ 6 đến 10 lần RTT. Khoảng thời gian này là cần thiết để kết nối được ổn định để xác định băng thông cổ chai. Để xác định băng thông cổ chai, các tác giả trong [27, 11] đã chỉ ra có thể dùng 8 hoặc 10 chu kỳ RTT cho giai đoạn khởi tạo. Thực nghiệm cho thấy, $BW_{max}(k)$ có thể xác định chính xác sau thời gian từ *6* tới *10* lần RTT ở cuối giai đoạn khởi tạo trong điều kiện mạng xác định (số nút mạng, tuyến kết nối). Giá trị tức thời của $BW_{max}(k)$ biến thiên khi thay đổi số luồng hoặc tải lưu lượng. Giá trị $\theta_t(k)$ được dùng để xác định băng thông cổ chai $BW_{max}(k)$.

Mặt khác, tốc độ phát ban đầu của FCoAP được xác định bằng cách sử dụng một biến *nACK* để đếm số gói ACK. Mỗi khi nhận được một gói ACK từ bên nhận, bên gửi tăng biến nACK như sau:

$$nACK = nACK + 1 \tag{3.14}$$

Kết thúc giai đoạn khởi tạo sau thời gian *T*, FCoAP tính tốc độ phát ban đầu như sau:

$$R = \frac{nACK}{T} \tag{3.15}$$

Trong giai đoạn hoạt động ổn định, bên gửi FCoAP liên tục phát các gói tin và tính toán RTT, $BW(k)$, $\theta_t(k)$ mỗi khi nó nhận được một gói tin ACK để tính RT_gradient và BG_gradient. Các giá trị này được đưa vào hệ FCS để tính ra cấp độ tắc nghẽn C_degree phục vụ cho việc tính toán lại tốc độ phát cần điều chỉnh. Việc cập nhật mới RT_gradient và BG_gradient được thực hiện định kỳ sau mỗi khoảng thời gian một RTT để tính C_degree. Tốc độ phát được điều chỉnh dựa vào C_degree như sau:

- Nếu 1 ≥ C_degree(k) > 0, tốc độ phát cần tăng. Lượng tăng tốc độ phát bằng giá trị tuyệt đối của C_degree. Giá trị này xác định tăng nhanh hay tăng chậm.

$$R(k) = R(k-1) + C\_degree(k) \times \frac{1}{RTT_S(k)} \tag{3.16}$$

- Nếu -1 ≤ C_degree(k) < 0, tốc độ phát cần giảm. Lượng giảm tốc độ phát bằng giá trị tuyệt đối của C_degree. Giá trị này xác định giảm nhanh hay giảm chậm.

$$R(k) = R(k-1) + C\_degree(k) \times \frac{1}{RTT_S(k)} \tag{3.17}$$

- Nếu C_degree = 0, tốc độ phát sẽ giữ nguyên, không thay đổi.

Ngoài ra, FCoAP cũng sử dụng C_degree để hiệu chỉnh lại giá trị RTO như sau:

$$RTT_s(k) = 0{,}75 \times RTT_s(k-1) + 0{,}25 \times RTT_m(k) \quad (3.18)$$

$$D\_RTT_s(k) = RTT_s(k) - RTT_s(k-1) \quad (3.19)$$

$$RTO(k) = RTT_s(k) + C\_degree(k) \times D\_RTT_s(k) \quad (3.20)$$

Trong đó $D\_RTT_s(k)$ là biến thiên RTT, $C\_degree(k)$ là đầu ra hệ điều khiển mờ và $RTO(k)$ là giá trị timeout cho phát lại tính ở bước k.

Trong khi hoạt động, nếu bên gửi không nhận được ACK trong khoảng thời gian RTO đã thiết lập, bên gửi sẽ thử phát lại gói tin đã mất và có thể gửi tiếp các gói mới. Có tối đa 4 lần thử phát lại đối với mỗi gói tin được phát lại. Giá trị RTO được cập nhật mới cho mỗi lần phát lại theo công thức (3.20) ở trên. FCoAP không tính toán lại băng thông cổ chai khi thực hiện phát lại các gói tin đã mất. Lý do là các tham số RTT và độ trễ của gói tin phát lại sẽ dẫn đến tính toán sai băng thông cổ chai như đã chỉ ra trong [11, 27, 104]. Nếu đã phát lại một gói tin 4 lần vẫn không thành công, gói tin sẽ được bên gửi đánh dấu là mất.

Trong giai đoạn ổn định, FCoAP liên tục kiểm tra mất gói thông qua: Kiểm tra khoảng trống số thứ tự gói tin nhận được, sự kiện của hàm định thời RTO. Nếu FCoAP phát hiện có dấu hiệu mất gói, nó chuyển sang trạng thái lùi. Khi đó có thể vẫn còn nhiều gói tin đang di chuyển trên mạng mà chưa tới đích (các gói *inflight*) bao gồm cả gói tin vừa mới phát và gói tin vừa được phát lại.

Trong giai đoạn lùi, FCoAP kiểm tra gói tin ACK sau mỗi RTT. Nếu có ACK, FCoAP cập nhật mới giá trị C_degree và quay về trạng thái ổn định. Nếu không nhận được ACK sau khoảng thời gian tối đa do ứng dụng cho phép [100], kết nối được coi là bị lỗi và cần khởi tạo lại từ đầu.

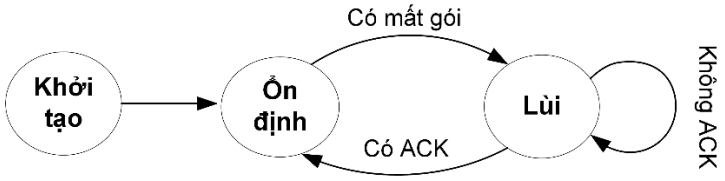
3.3.2. Các trạng thái của giao thức FCoAP

Giao thức FCoAP có 3 trạng thái: 1) Khởi tạo, 2) Ổn định, và 3) Lùi như mô tả trên hình 3.12.

- ***Trạng thái khởi tạo***

Khi bắt đầu kết nối, FCoAP ở trạng thái khởi tạo. FCoAP thực hiện tính băng thông cổ chai để tính ra tốc độ phát ban đầu. Để thực hiện điều này, bên gửi sẽ phát

các gói tin liên tiếp để kiểm tra băng thông cổ chai như đã nêu ở mục 3.3.1. Giai đoạn khởi tạo không sử dụng định thời RTO cho các gói tin thử, nghĩa là các gói tin gửi trong giai đoạn khởi tạo có thể bị mất và không được phát lại ngay. Bên gửi đếm số gói tin phản hồi ACK cho các gói tin đã phát để tính ra tốc độ phát ban đầu cho FCoAP theo công thức (3.15). Tiếp theo, bên gửi sẽ chuyển sang trạng thái hoạt động ổn định.



Hình 3.12 Ba trạng thái của giao thức FCoAP

- ***Trạng thái ổn định***

Đây là trạng thái hoạt động chính của FCoAP. Trong trạng thái này FCoAP sẽ liên tục phát các gói tin theo tốc độ được điều chỉnh cập nhật bởi hệ điều khiển mờ. Tốc độ phát gói được điều chỉnh tùy thuộc vào tình trạng mạng thể hiện qua cấp độ tắc nghẽn C_degree như đã nêu ở mục 3.3.1. Nếu không nhận được ACK cho gói tin đã phát, bên gửi FCoAP sẽ thực hiện phát lại gói tin đã mất theo giá trị RTO được cập nhật bởi C_degree. Nếu quá 4 lần phát lại không thành công, bên gửi FCoAP sẽ đánh dấu gói tin bị mất và chuyển sang trạng thái lùi.

- ***Trạng thái lùi***

Trong trạng thái này, bên gửi FCoAP liên tục kiểm tra xem có nhận được gói tin ACK không. Nếu có ACK, FCoAP sẽ phát tiếp một gói tin, cập nhật lại giá trị C_degree, tốc độ phát hiện tại và khôi phục lại trạng thái ổn định. Nếu vẫn không nhận được ACK sau một khoảng thời gian tối đa do ứng dụng cho phép [100], FCoAP sẽ hủy bỏ kết nối.

3.3.3. Các thuật toán của giao thức FCoAP

- ***Thuật toán khởi tạo***

Thuật toán 3.1 là thuật toán khởi tạo của FCoAP thực hiện một lần khi bắt đầu kết nối và kéo dài từ 6 đến 10 lần RTT như đã nêu ở mục 3.3.1 (FCoAP sử dụng 6 RTT trong thử nghiệm này).

Thuật toán 3.1. Khởi tạo

Đầu vào: *t0, maxCycle = 6*

Đầu ra: *R*, BW_{max}

1: Function Startup()

2: *nACK* = 0; *Cycle* = 0

3: *T = t0 + 1*

4: *ACK = False*

5: WHILE (*Cycle* < *maxCycle*) DO

6: IF ($t \geq T$) THEN

7: packet ← SendNextPacket()

8: *T = t + 1*

9: *Cycle = Cycle + 1*

10: ENDIF

11: IF (*ACK == True*) THEN

12: *nACK = nACK + 1*

13: Update (*RT(k), BG(k),* $BW_{max}(k)$)

14: *ACK = False*

15: ENDIF

16: ENDDO

17: IF ((*nACK* == 0) && (*Cycle == maxCycle*)) THEN

18: Restart()

19: ENDIF

20: *R = nACK* / (*t – t0*)

21: Update (*RT(k), BG(k),* $BW_{max}(k)$)

22: END Function.

Các giá trị khởi tạo khi bắt đầu kết nối gồm: *maxCycle = 6* là số chu kỳ RTT cho khởi tạo, *t0* là thời điểm bắt đầu, *t* là thời gian hiện tại. Vòng WHILE thực hiện trong thời gian 6 chu kỳ RTT (dòng 5 – 16). Trong mỗi chu kỳ RTT, bên gửi phát đi một gói tin liên tiếp (dòng 6 - 9). Hàm sự kiện nhận gói (không nêu trong thuật toán) đặt biến ACK = True khi có một gói ACK quay về. Mỗi khi nhận được ACK cho gói tin đã phát (dòng 11), bộ đếm *nACK* sẽ đếm số gói tin ACK nhận được (dòng 12). Hàm Update(..) tính toán cập nhật các tham số *RT(k), BG(k)* và $BW_{max}(k)$ và gán ACK = *False* (dòng 13-14). Khi kết thúc vòng WHILE, nếu *nACK = 0* và *Cycle = maxCycle* (nghĩa là không nhận được ACK nào sau *6* vòng RTT), bên gửi sẽ phải khởi tạo lại kết nối từ đầu (dòng 17-18). Nếu $nACK \neq 0$, FCoAP tính tốc độ phát ban đầu theo công thức (3.15) và gọi hàm Update(..) để cập nhật các tham số (dòng 20-21). Tiếp đó, FCoAP kết thúc trạng thái khởi tạo và chuyển sang trạng thái ổn định.

- ***Thuật toán ở trạng thái ổn định***

Thuật toán 3.2 mô tả trạng thái ổn định của FCoAP duy trì khi không có mất gói xảy ra (*pLoss = False*).

Thuật toán 3.2. Ổn định

Đầu vào: R, $BW_{max}(k)$

Đầu ra: *pLoss*

```
1: Function Steady ()
2:      last_adjust = t
3:      ACK = False
4       pLoss = False
5:      WHILE ( pLoss == False) DO
6:          BDP(k) = BWmax(k) × RTT_min
7           IF ( inflight ≤ BDP(k) ) THEN
8:              packet ← SendNextPacket()
9:              IF ( t ≥ last_adjust + RTT ) THEN
10:                 Update ( RT(k), BG(k), BW_max(k))
11:                 C_degree = FCS ( RT(k), BG(k), BW_max(k) )
12:                 R(k) = R(k-1) + C_degree × ( 1 / RTT )
13:                 last_adjust = t
14:             ENDIF
15:         ELSE
16:             IF ( ACK == True ) THEN
17:                 packet ← SendNextPacket()
18:             ENDIF
19:         ENDIF
20:         pLoss ← CheckLoss()
21:     ENDDO
22:     IF ( pLoss ) THEN   Backoff ()
23:     ENDIF
24: END Function.
```

Trong trạng thái này, *last_adjust* (dòng 2) là thời điểm để cập nhật tốc độ phát mới (dòng 12-13) dựa vào đầu ra C_degree của hàm điều khiển mờ ***FCS*** (dòng 11) nếu số gói *inflight* vẫn nhỏ hơn BDP (dòng 6-7). Nếu số gói *inflight* vượt quá BDP, tốc độ giữ nguyên để tránh quá tải. FCoAP sẽ chỉ gửi 1 gói mới mỗi khi nhận được ACK (dòng 16-17). Biến *ACK=True* mỗi khi có sự kiện nhận 1 gói ACK.

Trong quá trình truyền tin, một gói tin có thể không đến đích sau khoảng định thời RTO. Gói tin sẽ được phát lại (ở dòng 8 hoặc dòng 17), nghĩa là hàm SendNextPacket phát lại một gói đã mất (nếu có) hoặc phát tiếp một gói mới. Nếu quá 4 lần phát lại không thành công hoặc khi có khoảng trống thứ tự gói đến, hàm

CheckLoss(..) sẽ lập cờ báo *pLoss=True* (dòng 20) để báo mất gói. Khi đó, FCoAP sẽ chuyển sang trạng thái lùi (dòng 22).

- ***Thuật toán lùi***

Thuật toán 3.3. Lùi
Đầu vào: *pLoss, Tmax*
Đầu ra: *R(k)*
1: Function Backoff ()
2: *next_pacing = t*
3: *ACK = False*
4: WHILE (*ACK == False*) DO
5: IF ($t \geq next\_pacing + RTT$) THEN
6: packet ← SendNextPacket()
7: Update ($RT(k), BG(k), BW_{max}(k)$)
8: $C\_degree = \boldsymbol{FCS}$ ($RT(k), BG(k), BW_{max}(k)$)
9: RTO_Update (*C_degree, RTO*)
10: $R(k) = R(k\text{-}1) + \boldsymbol{C\_degree \times (1 / RTT)}$
11: *next_pacing = t*
12: ENDIF
13: IF ($t > Tmax$) THEN **Restart ()**
14: ENDIF
15: ENDDO
16: **return Steady ()**
17: END Function.

Thuật toán 3.3 mô tả trạng thái lùi của FCoAP khi có mất gói xảy ra. Trong vòng WHILE (dòng 4-15), FCoAP liên tục kiểm tra biến ACK. Nếu vẫn chưa có ACK, FCoAP phát lại gói tin đã mất (dòng 5-6) sau mỗi RTT, cập nhật lại các tham sô (dòng 7-11). Nếu quá thời gian tối đa *Tmax* do ứng dụng cho phép [100] (dòng 13), FCoAP sẽ hủy bỏ kết nối và khởi tạo lại từ đầu. Nếu nhận được ACK, FCoAP sẽ trở về trạng thái ổn định (dòng 16).

3.4. Tính toán hiệu năng giao thức FCoAP

Theo các tiêu chí đánh giá hiệu năng đã nêu ở chương 1, phần này tính toán các đại lượng: độ trễ, tỷ lệ mất gói và thông lượng cho FCoAP như sau. Các ký hiệu tính toán hiệu năng tương tự như ở bảng 2.2. Do RTO biến thiên ở FCoAP, cách tính độ trễ đầu cuối và thông lượng phụ thuộc vào RTO.

3.4.1. Độ trễ đầu cuối của FCoAP

Độ trễ của một gói tin bất kỳ sẽ phụ thuộc vào giá trị $TO = RTO(k)$ tại thời

điểm phát lại ở chu kỳ *k*. Ta có:

$$d_i = D_0 + TO \quad (3.21)$$

Xác suất phát gói thành công có trễ d_i sau *i* lần phát lại là:

$$\Pr(D = d_i) = p^i(1 - \mathrm{p}) \quad (3.22)$$

Với *p* là xác suất mất gói trong một lần phát (nghĩa là *1-p* là xác suất phát thành công một gói), *i* là số lần gói tin phải phát lại với $1 \le i \le m$, *m* là số lần phát lại tối đa. Độ trễ trung bình của các gói tin sẽ là:

$$D_{TB} = \sum_{i=0}^{m} \Pr(D = d_i) d_i = \sum_{i=0}^{m} p^i(1-p)[\, D_0 + TO] \quad (3.23)$$

$$D_{TB} = \; D_0(1 - p^{m+1}) + TO(1 - p^{m+1}) \quad (3.24)$$

$$\text{Vì: } \sum_{i=0}^{m} p^i(1-p) = 1 - p^{m+1} \quad (3.25)$$

Mặt khác, công thức cập nhật RTO của FCoAP tại một chu kỳ *k* là:

$$RTO(k) = RTT_s(k) + C\_degree(k) \times D\_RTT_s(k) \quad (3.26)$$

Trong đó $RTT_s(k)$ là giá trị RTT tại chu kỳ *k* hiện tại, $D\_RTT_s(k)$ là độ biến thiên RTT, *C_degree(k)* là đầu ra hệ điều khiển mờ. Giá trị lớn nhất của $D\_RTT_s(k)$ là $RTT_s(k)$ nên có:

$$TO = RTO(k) \le RTT_s(k)\, [1 + |C\_degree(k)|] \quad (3.27)$$

Do | *C_degree* | ≤ 1 nên:

$$TO \le 2 * RTT_s(k) \quad (3.28)$$

Mặt khác, ta có công thức điều chỉnh tốc độ phát với *R(k)* và *R(k-1)* là tốc độ tại chu kỳ *k* và *k-1* là:

$$R(k) = R(k-1) + C_{degree(k)} / \; RTT_s(k) \quad (3.29)$$

Do | *C_degree* | ≤ 1, đặt $\Delta R = R(k) - R(k\text{-}1)$ (3.30)

Ta có: $RTT_s(k) \le 1/\, \Delta R$ (3.31)

Từ (3.28) và (3.31) có:

$$TO \le \; 2 \; / \Delta R \quad (3.32)$$

Thay vào (3.24) ta có:

$$D_{TB} \le \; D_0(1 - p^{m+1}) + 2(1 - p^{m+1}) \; / \; \Delta R \quad (3.33)$$

Công thức (3.33) cho thấy, nếu giới hạn tốc độ phát, có thể giới hạn độ trễ *D* của luồng tin.

3.4.2. Thông lượng của FCoAP

Do FCoAP cũng phát đi chuỗi gói *inflight* tương tự TCP, ta có thể theo cách tính thông lượng trung bình cho TCP như trong công thức (2.41) đã nêu ở mục 2.3.3. Tuy nhiên, giá trị RTO không còn là cố định mà biến thiên theo thời gian. Do vậy, có thể dùng công thức tính xấp xỉ thông lượng cho FCoAP như sau.

$$\theta_{TB}(p, RTT) = \frac{1}{RTT\sqrt{\frac{2p}{3}}} = \frac{\sqrt{3/2}}{RTT\sqrt{p}} \quad (3.34)$$

3.5.3. Các thông số hiệu năng khác

Cách tính các đại lượng hiệu năng khác như tỷ lệ gói tin phát lại, tỷ lệ gói tin phát thành công, tỷ lệ gói tin phát lại bị đúp, tỷ lệ mất gói không phụ thuộc RTO, nên có thể dùng cách tính tương tự như đã nêu ở mục 2.3 cho RCoAP.

3.5. Kết quả mô phỏng đánh giá FCoAP

3.5.1. Thiết lập môi trường mô phỏng cho FCoAP

FCoAP được mô phỏng thử nghiệm với bộ công cụ mô phỏng NS3.36 [88]. So với RCoAP, phần mềm FCoAP có một số thay đổi với việc bổ sung hàm FCS với 3 hàm con: tính hàm thuộc và xác định tập mờ các đầu vào, tính hàm thuộc và xác định tập mờ đầu ra, giải mờ để xác định hệ số điều khiển tốc độ (Phụ lục 5).

Các tham số cấu hình cho các kịch bản mô phỏng ở bảng 3.2 với công nghệ khuyến nghị của ITU [57, 99] cho ứng dụng IoT. Cụ thể: Mô đun IoT sử dụng WiFi (802.11n/g), tốc độ phát tối đa mỗi luồng CoAP là 250 Kbps (đặc trưng cho mạng 6LoWPAN [100]). Thời gian mô phỏng phổ biến là 300s tương tự các nghiên cứu trước đây. Số lần chạy mỗi kịch bản là 30 lần. Số nút IoT bên gửi phát đồng thời từ 10 đến 30. Tầng vật lý và MAC sử dụng chuẩn của NS-3.36 [88]. Các kịch bản mô phỏng các tình trạng tắc nghẽn được thực hiện bằng cách thay đổi băng thông cổ chai và độ trễ liên kết tương tự [68, 66, 10]. Cấu hình mạng mô phỏng tương tự [13, 15, 22, 93, 68]. Mô phỏng sử dụng mạng như hình 2.6 và 2.7 ở chương 2 cho các kịch bản như thể hiện ở bảng 3.2.

FCoAP sử dụng tiêu đề gói tin chuẩn của CoAP [100] với 106 Bytes cho gói CON và 49 Bytes cho gói ACK. FCoAP sử dụng 8 Bytes trong tùy chọn tiêu đề: 4

Bytes để ghi thứ tự gói tin phục vụ kiểm tra mất gói, 4 Bytes để ghi thời gian nhận dùng cho tính thông lượng tức thời và băng thông cổ chai. Hiệu năng FCoAP được so sánh với CoAP do CoAP là đại diện cho các cơ chế dựa vào mất gói trong các nghiên cứu liên quan tới nay. Các điều kiện mô phỏng có thể thay đổi tùy biến. Điều quan trọng là tạo cùng một điều kiện mô phỏng để so sánh hiệu năng.

Bảng 3.2: Các tham số chính để thiết lập mô phỏng FCoAP

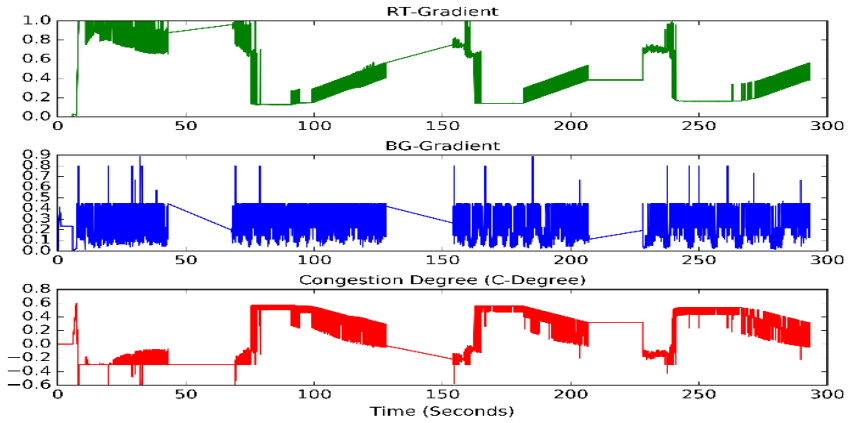
Tham số		**Giá trị**
Thời gian mô phỏng		100s – 300s
Số lần chạy mỗi kịch bản		30 lần
Số mô đun IoT bên gửi		3 – 30
Tầng vật lý và MAC		WifiPhy 2,4 GHz, IEEE 802.11b/n [88,93]
Liên kết AP		250 Kbps, 50 ms (tương tự [14, 15, 22, 23])
Kết nối Internet		10/100 Mbps (Ethernet)
Kịch bản mô phỏng		Mô phỏng băng thông cổ chai với BW và trễ liên kết D theo cách trong [66, 21, 10, 68, 15]
+ Kịch bản 3.1:	Thử nghiệm chức năng	BW = 45 Kbps, D = 300 ms
+ Kịch bản 3.2:	So sánh giao thức	BW = 60 Kbps, D = 300 ms
+ Kịch bản 3.3:	Lưu lượng UDP thay đổi	BW = 75 Kbps, D = 300 ms
+ Kịch bản 3.4:	Lưu lượng CoAP hỗn hợp	BW = 75 Kbps, D = 300 ms
+ Kịch bản 3.5:	Lưu lượng UDP/ TCP hỗn hợp	BW = 75 Kbps, D = 300 ms
+ Kịch bản 3.6:	So sánh với các giao thức CoAP khác	BW = 60 Kbps, D = 300 ms,
+ Kịch bản 3.7:	So sánh FCoAP với RCoAP chặng ngắn	BW = 60 Kbps, D = 300 ms,
+ Kịch bản 3.8:	So sánh FCoAP với RCoAP chặng dài	BW = 75 Kbps, D = 500 ms,
Mô hình mạng		Kịch bản 3.1, 3.7: hình sao (hình 2.6) Kịch bản 3.2, 3.3, 3.4, 3.5, 3.6, 3.8: hình xương cá (hình 2.7)
Thời điểm bắt đầu phát		Ngẫu nhiên trong khoảng 0-200 ms
Tính khoảng tin cậy		Công thức (1.11) với mức tin cậy 99%

Để có được các kết quả tin cậy, mỗi thí nghiệm được thực hiện 30 lần chạy thử và lấy giá trị trung bình với khoảng tin cậy tính theo công thức (1.11) cho mức tin cậy 99% với phân bố chuẩn. Thời điểm bắt đầu phát được gieo ngẫu nhiên trong khoảng 0-200 ms. Các dữ liệu được đo theo thời gian thực trong chương trình.

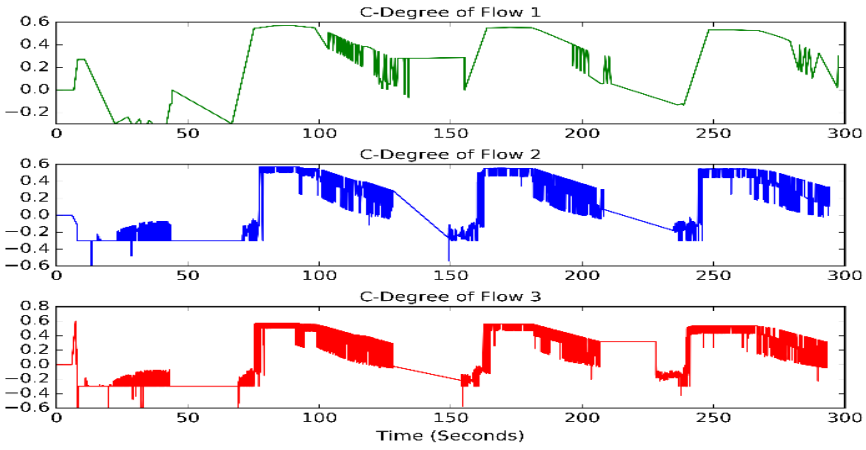
3.5.2. Kịch bản 3.1: Kiểm tra hoạt động của hệ điều khiển mờ

- ***Biến thiên của RT_gradient, BG_gradient và C_degree***

Kịch bản 3.1 sử dụng mạng hình sao như hình 2.6 (chương 2) với các tham số như ở bảng 3.2. Hình 3.13 biểu thị biến thiên của hai đầu vào và đầu ra của hệ FCS cho mỗi luồng FCoAP. Hình 3.14 là biến thiên C_degree của 3 luồng FCoAP.



Hình 3.13 Biến thiên RT_gradient, BG_gradient và C_degree



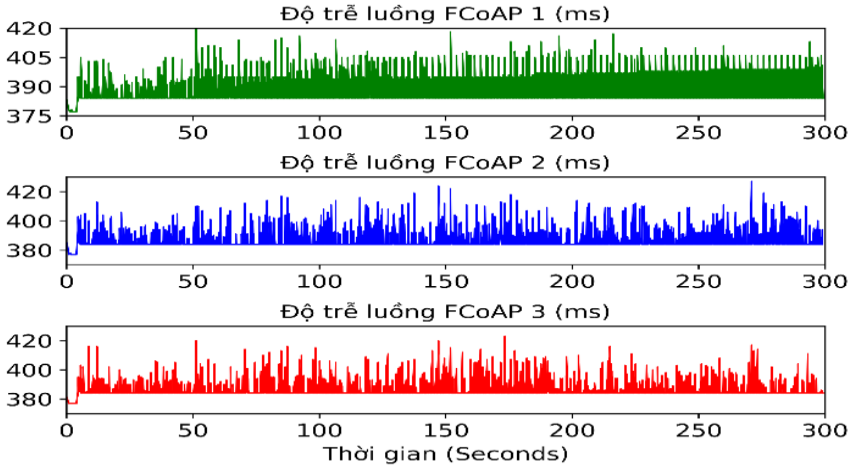
Hình 3.14 Biến thiên C_degree của ba luồng FCoAP

C_degree biến thiên trong khoảng [-1; 1] theo RT_gradient và BG_gradient, thể hiện tình trạng tắc nghẽn. Trong giai đoạn đầu, C_degree luôn có giá trị dương phản ánh nguy cơ tắc nghẽn thấp. Các khoảng giá trị âm của C_degree phản ánh nguy cơ

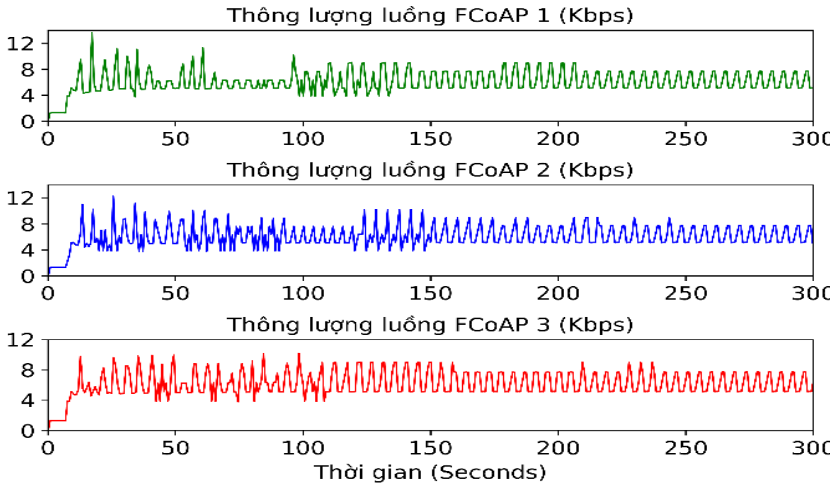
tắc nghẽn cao, khi RT_gradient tăng nhanh và BG_gradient có mật độ dày đặc hơn. Biến thiên C_degree của các luồng FCoAP ở hình 3.14 thể hiện tính bình đẳng của các luồng cùng tương tranh băng thông cổ chai trong thí nghiệm. Kết quả mô phỏng cho thấy biến thiên C_degree phù hợp với tình trạng tắc nghẽn và biến động của mạng.

- ***Hiệu năng FCoAP khi lưu lượng đồng nhất***

Trong kịch bản 3.1, lưu lượng đồng nhất được tạo ra từ các luồng FCoAP. Hình 3.15 biểu thị độ trễ, hình 3.16 biểu thị thông lượng của các luồng tin FCoAP.



Hình 3.15 Độ trễ của FCoAP



Hình 3.16 Thông lượng của FCoAP

Kết quả mô phỏng cho thấy độ trễ trung bình có giá trị nhỏ hơn *410* ms cho các luồng tin FCoAP. Chưa có mất gói xảy ra trong điều kiện mô phỏng, do vậy các

luồng FCoAP duy trì được độ trễ nhỏ và thông lượng trung bình đạt khoảng 8 Kbps. Độ biến thiên là do dao động của băng thông cổ chai khi các luồng tin cùng phát đồng thời, tương tranh băng thông chia sẻ. Trễ và thông lượng tăng khi tải tăng, nghĩa là số gói *inflight* tăng và ngược lại, giảm khi số gói *inflight* giảm. Bảng 3.3 so sánh hiệu năng cho 3 luồng FCoAP trong các thí nghiệm trên.

Bảng 3.3: So sánh hiệu năng của các luồng tin FCoAP

Chỉ số hiệu năng	**Luồng 1**	**Luồng 2**	**Luồng 3**
Tổng số gói gửi	2601	2641	2618
Số gói phát thành công	2601 (100%)	2641 (100%)	2618 (100%)
Số gói phát lại	1 (0,04%)	1 (0,04%)	1 (0,04%)
Số gói phát lại đúp	1 (0,04%)	1 (0,04%)	1 (0,04%)
Số gói bị mất	0 (0%)	0 (0%)	0 (0%)
Trễ trung bình	387,79 ms	386,41 ms	385,92 ms
Thông lượng trung bình	8,49 Kbps	8,62 Kbps	8,55 Kbps

Theo kết quả trong bảng: không có mất gói xảy ra, số lần phát lại thấp (chỉ 1 gói trong số hơn 2600 gói), do đó FCoAP duy trì được độ trễ nhỏ, thông lượng cao.

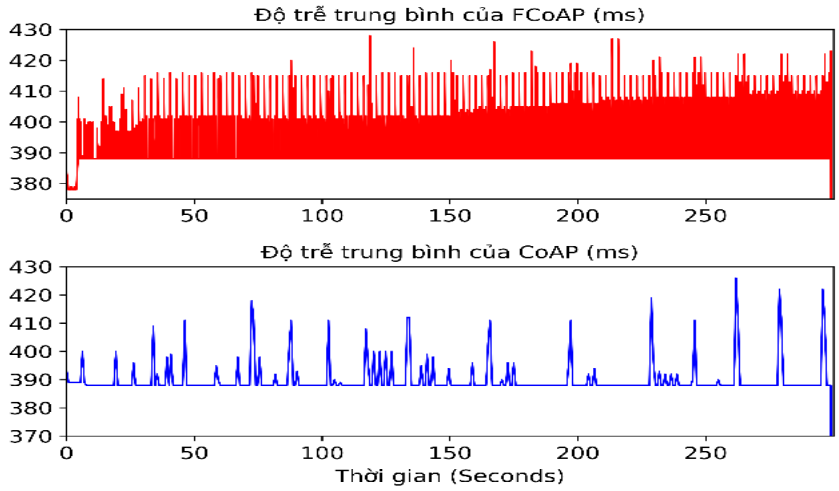
3.5.3. Kịch bản 3.2: So sánh hiệu năng FCoAP và CoAP

Kịch bản 3.2 sử dụng mạng như hình 2.7 (chương 2) với các tham số như ở bảng 3.2, có 10 luồng FCoAP và 10 luồng CoAP để so sánh hiệu năng. Bảng 3.4 tổng hợp kết quả so sánh hiệu năng giữa FCoAP và CoAP trong thí nghiệm này.

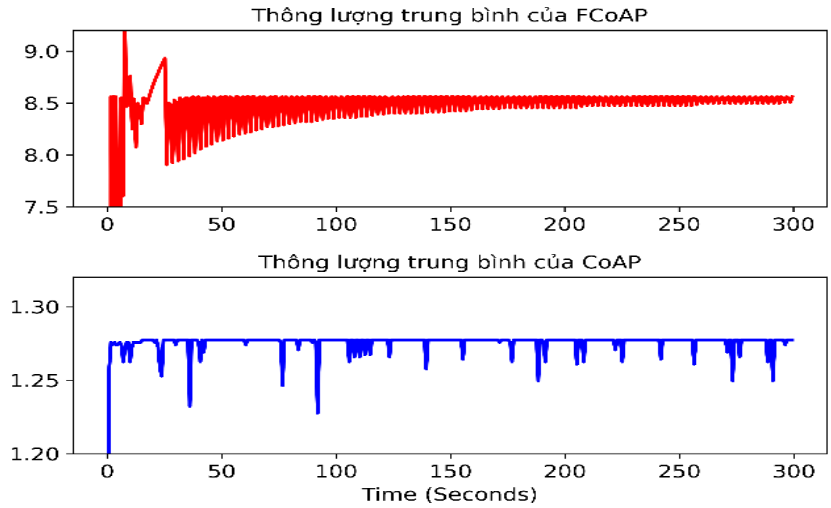
Bảng 3.4: So sánh hiệu năng của FCoAP và CoAP

Tham số hiệu năng	**FCoAP**	**CoAP**
Tổng số gói gửi	2558	389
Số gói phát thành công	2558 (100%)	389 (100%)
Số gói phát lại	1 (0,04%)	0 (0%)
Số gói phát lại đúp	1 (0,04%)	0 (0%)
Số gói bị mất	0 (0%)	0 (0%)
Trễ trung bình	392,43 ms	390,31 ms
Thông lượng trung bình	8,35 Kbps	1,27 Kbps

Hình 3.17 so sánh độ trễ và hình 3.18 so sánh thông lượng của các luồng tin FCoAP và CoAP. Như trên hình, mặc dù mức độ biến thiên khác nhau, song độ trễ lớn nhất vẫn nhỏ hơn *430* ms cho tất cả gói tin của các luồng tin FCoAP và CoAP. Đối với CoAP, độ trễ trung bình là *390,31* ms cho *389* gói tin phát đi trong *300s* với khoảng tin cậy là (*389,46; 391,17*), mức tin cậy 99%. Với FCoAP, độ trễ trung bình là *392,43* ms cho *2558* gói tin phát đi trong *300s*, khoảng tin cậy (*392,02; 392,85*), mức tin cậy 99%. FCoAP truyền thành công nhiều gói tin so với CoAP do FCoAP sử dụng FCS có điều khiển tốc độ phát cao hơn, song vẫn giữ được độ trễ tối đa.



Hình 3.17 Độ trễ của FCoAP và CoAP



Hình 3.18 Thông lượng FCoAP và CoAP

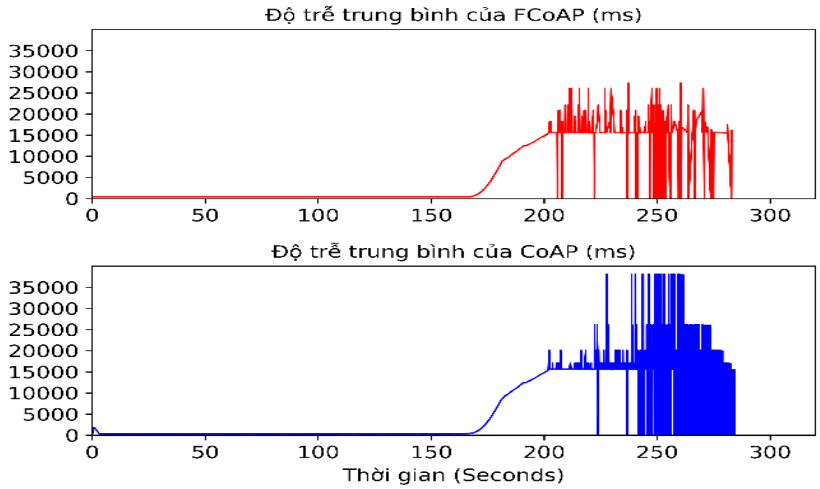
Hình 3.18 so sánh thông lượng của các luồng tin FCoAP và CoAP. Thông lượng trung bình của CoAP là 1,27 Kbps với khoảng tin cậy là (*1,26; 1,28*), mức tin

cậy *99%*. FCoAP có thông lượng trung bình xấp xỉ *8,35* Kbps với khoảng tin cậy (*8,24; 8,50*), mức tin cậy *99%*. Kết quả cho thấy FCoAP cho thông lượng cao và trễ nhỏ hơn CoAP trong cùng điều kiện thí nghiệm. Lý do là FCoAP điều khiển tốc độ phát, cho phép tận dụng băng thông cổ chai, còn CoAP thì không. Vì vậy, số gói phát đi thành công của FCoAP (*2558*) cao hơn so với CoAP (*389*).

3.5.4. Kịch bản 3.3: Hiệu năng FCoAP và CoAP khi có lưu lượng UDP thay đổi

Kịch bản 3.3 sử dụng mạng như hình 2.7 (chương 2) với các tham số như ở bảng 3.2, một luồng UDP có tốc độ thay đổi, 10 luồng FCoAP và 10 luồng CoAP để so sánh hiệu năng. Luồng UDP phát với tốc độ *2* Kbps trong khoảng *0-160s*. Sau đó tốc độ UDP tăng dần theo từng bước *1* Kbps trong khoảng *160s-200s*. Biến đổi tốc độ của UDP tạo ra tình huống tăng tải nhanh dẫn đến nguy cơ tắc nghẽn cao. Từ thời điểm *200s*, luồng tin UDP giảm nhanh tốc độ phát với bước giảm *2* Kbps nhằm nhanh chóng giải phóng các gói tin *inflight* ùn tắc trên mạng.

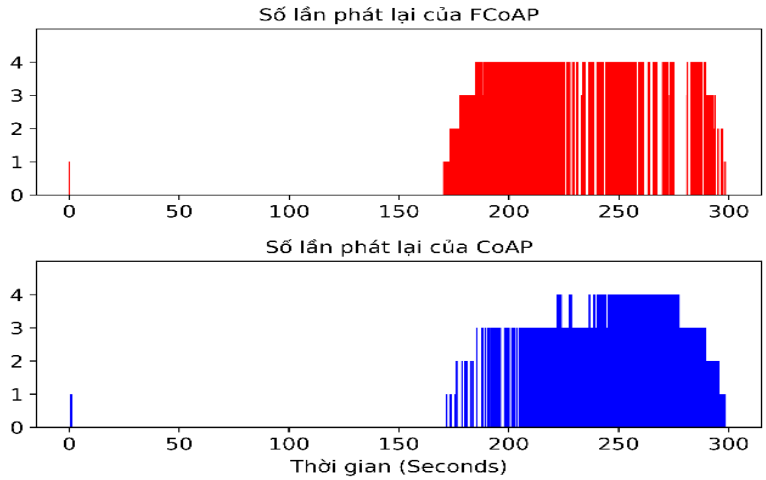
Hình 3.19 biểu thị độ trễ của FCoAP và CoAP, hình 3.20 biểu thị số lần (theo trục hoành) và số gói phát lại (theo trục tung).



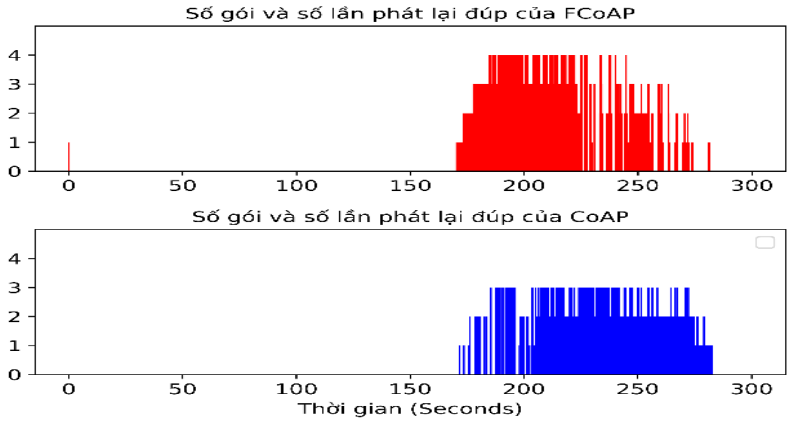
Hình 3.19 Độ trễ của FCoAP và CoAP khi có lưu lượng UDP thay đổi

Trong thời gian từ *0-160s*, cả FCoAP và CoAP đều không có tắc nghẽn. Khi tải lưu lượng tăng do luồng tin UDP tăng tốc độ (từ *160s*), độ trễ của FCoAP và CoAP tăng dần cùng với số lần phát lại. Mất gói xảy ra ở khoảng *180s*. Độ trễ tăng nhanh và biến thiên đạt xấp xỉ *22s-28s* do phải phát lại các gói tin. Độ trễ của FCoAP có mật độ thưa hơn của CoAP, đồng nghĩa FCoAP có ít gói bị trễ hơn so với CoAP.

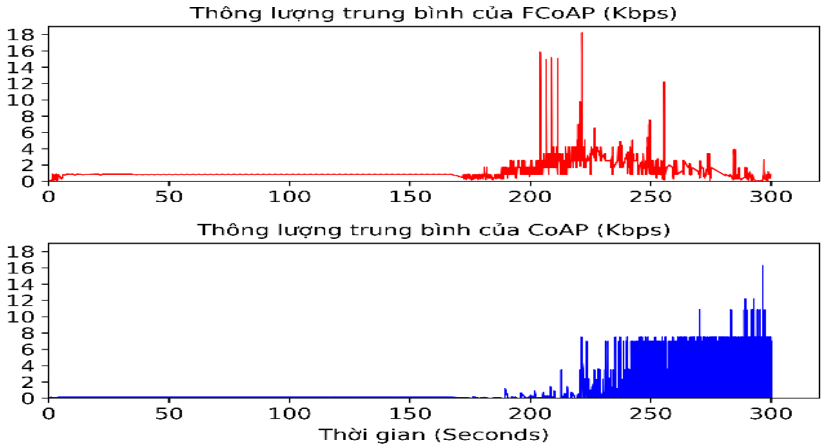
Như trên hình 3.20, nhiều gói tin cần tới *4* lần phát lại ở hai cơ chế, song số lần phát lại của FCoAP là *669* nhỏ hơn so với *2954* gói tin phát lại trong CoAP.



Hình 3.20 Số lần phát lại của FCoAP và CoAP khi có lưu lượng UDP thay đổi



Hình 3.21 Phát lại đúp của FCoAP và CoAP khi có lưu lượng UDP thay đổi



Hình 3.22 Thông lượng của FCoAP và CoAP khi có lưu lượng UDP thay đổi

Hình 3.21 biểu thị số lần và số gói tin phát lại đúp của FCoAP và CoAP. Số lần

phát lại đúp của FCoAP là *571* ít hơn so với *922* gói tin trong CoAP. Hình 3.22 so sánh thông lượng của FCoAP và CoAP. Thông lượng trung bình được tính cho các gói tin có phản hồi ACK, kể cả gói tin phát lại đúp. Do số gói phát lại đúp của CoAP trong khoảng *230s-300s* cao hơn so với FCoAP, mật độ đồ thị ở CoAP dày đặc hơn so với FCoAP. Tuy nhiên, thông lượng thực tế sẽ không bao gồm gói tin phát lại đúp. Bảng 3.5 tổng hợp các chỉ số hiệu năng của FCoAP và CoAP. Kịch bản 3.3 khác với kịch bản 3.2 là có băng thông cổ chai 75 Kbps cao hơn. Mặt khác do kịch bản 3.3 có thêm luồng UDP thay đổi tốc độ, CoAP giành được thêm băng thông khi luồng UDP trả lại trong các khoảng ngừng phát. Vì vậy, số gói tin CoAP phát đi là 3172 với thông lượng là 6,32 Kbps (Bảng 3.5) cao hơn so với kịch bản 3.2 (Bảng 3.4). Tuy nhiên, số gói tin phát lại đúp và phát lại của CoAP rất lớn. Kết quả trong bảng cho thấy, mặc dù CoAP phát đi nhiều gói hơn (bao gồm cả gói phát lại) song số gói thành công là *1840* (*59,01%*) thấp hơn so với *2020* (*95,92%*) của FCoAP. Tỷ lệ mất gói, phát lại, phát lại đúp của FCoAP đều thấp hơn CoAP trong điều kiện thí nghiệm này. Do có tỷ lệ thành công cao hơn nên độ trễ trung bình của FCoAP là *4304,14* ms nhỏ hơn so với *8639,60* ms trong CoAP và thông lượng trung bình của FCoAP là *6,97* Kbps cao hơn so với *6,32* Kbps trong CoAP.

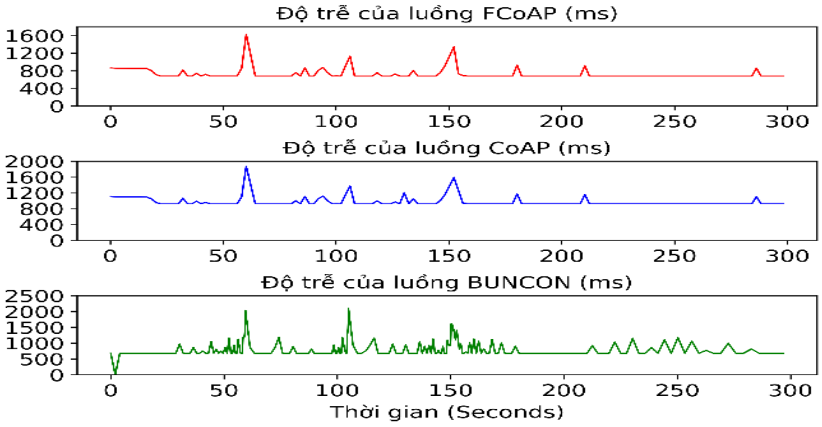
Bảng 3.5: Hiệu năng của FCoAP và COAP khi có lưu lượng UDP thay đổi

Tham số hiệu năng	**FCoAP**	**CoAP**
Tổng số gói gửi	2106	3172
Số gói phát thành công	2020 (95,92%)	1840 (58,01%)
Số gói phát lại	669 (31,77%)	2954 (93,13%)
Số gói phát lại đúp	571 (27,11%)	922 (29,07%)
Số gói bị mất	86 (4,10%)	1332 (41,99%)
Trễ trung bình	4304.14 ms	8639,60 ms
Thông lượng trung bình	6,97 Kbps	6,32 Kbps

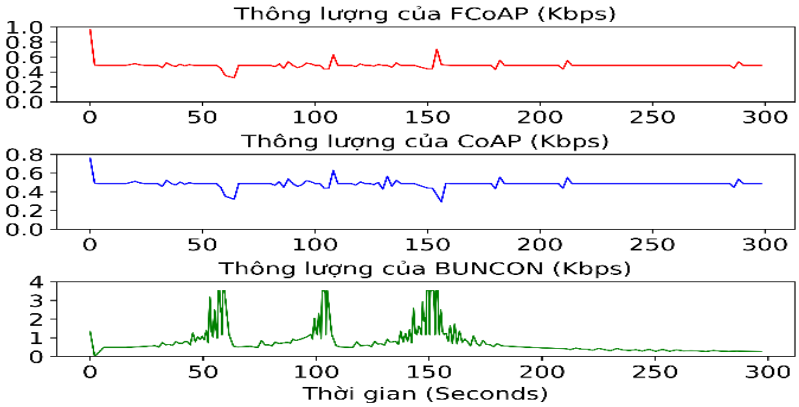
3.5.5. Kịch bản 3.4: Hiệu năng FCoAP và CoAP khi có lưu lượng CoAP hỗn hợp

Kịch bản 3.4 sử dụng mạng như hình 2.7 (chương 2) với các tham số như ở bảng 3.2. Một luồng CoAP không tin cậy, gọi tắt là luồng BUNCON (Basic

Unconfirmable) tương tranh băng thông cổ chai với 10 luồng FCoAP và 10 luồng CoAP để so sánh hiệu năng. Luồng BUNCON bắt đầu với tốc độ phát là *0.5* Kbps, sau đó tăng dần từng bước tới giá trị *25* Kbps. Tiếp đó, tốc độ phát giảm dần từng bước xuống *0.5* Kbps. Quá trình tăng và giảm tốc độ này được lặp lại *3* lần. Cuối cùng, luồng tin BUNCON dừng phát ở *280s*. Kịch bản này là tạo ra một lưu lượng CoAP hỗn hợp có biến đổi động trong mạng.



Hình 3.23 Độ trễ của FCoAP và CoAP khi lưu lượng CoAP hỗn hợp



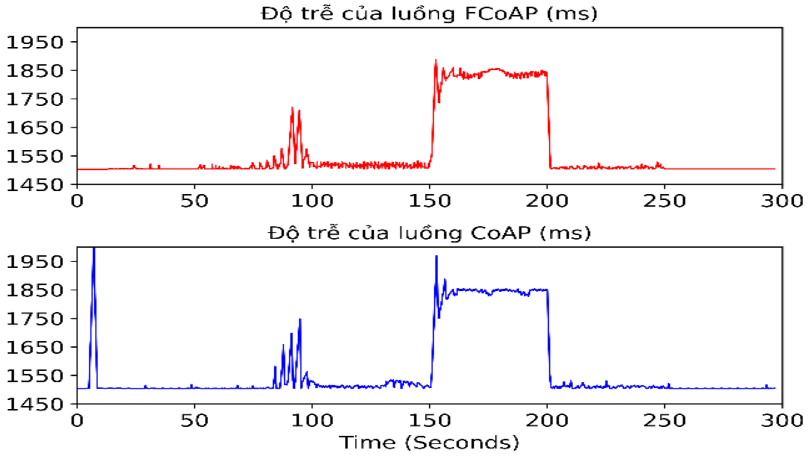
Hình 3.24 Thông lượng của FCoAP và CoAP khi lưu lượng CoAP hỗn hợp

Hình 3.23 hiển thị độ trễ còn hình 3.24 biểu thị thông lượng của các luồng tin. Như trên hình 3.23, độ trễ của các luồng tin tăng trong các khoảng thời gian có tắc nghẽn xảy ra ở *55s*, *110s* và *150s*. Tải lưu lượng cao là do tốc độ phát của luồng tin BUNCON trong các khoảng thời gian này. Độ trễ trung bình của FCoAP là *725,91* ms với khoảng tin cậy (*700,40; 751,43*), mức tin cậy *99%*. Độ trễ trung bình của CoAP là *973,95* ms với khoảng tin cậy (*948,03; 999,95*), mức tin cậy *99%*.

Hình 3.24 so sánh thông lượng của FCoAP, CoAP và BUNCON. Tắc nghẽn xảy ra trong các khoảng *40s-70s*, *80s-110s*, và *140s-180s*. Thông lượng trung bình của FCoAP là *0,49* Kbps với khoảng tin cậy (*0,479, 0,501*), của CoAP là *0,48* Kbps với khoảng tin cậy (*0,477, 0,495*). Các kết quả cho thấy độ trễ và thông lượng trung bình của FCoAP tốt hơn so với CoAP khi tương tranh với lưu lượng CoAP hỗn hợp.

3.5.6. Kịch bản 3.5: FcoAP và CoAP khi có lưu lượng TCP/UDP hỗn hợp

Kịch bản 3.5 sử dụng mạng như hình 2.7 (chương 2) với các tham số như ở bảng 3.2 khi có lưu lượng TCP/UDP hỗn hợp cùng với 10 luồng FCoAP và 10 luồng CoAP để so sánh hiệu năng. Luồng TCP được thiết lập với socket TCP chuẩn trong NS-3 [88] với tốc độ dữ liệu *10* Mbps, kích thước gói dữ liệu là *1040* bit. Tốc độ phát của TCP sẽ biến thiên tùy theo băng thông cổ chai và trạng thái mạng. Luồng tin UDP được thiết lập với socket UDP chuẩn trong NS-3 [88]. Luồng UDP bắt đầu với tốc độ dữ liệu *20* Kbps, sau đó thay đổi thành tốc độ *50* Kbps ở thời điểm *80s*, *150* Kbps ở thời điểm *150s* và tốc độ *5* Kbps ở thời điểm *200s*. Kịch bản này tạo tình trạng tắc nghẽn và biến động mạng với lưu lượng hỗn hợp TCP/UDP.

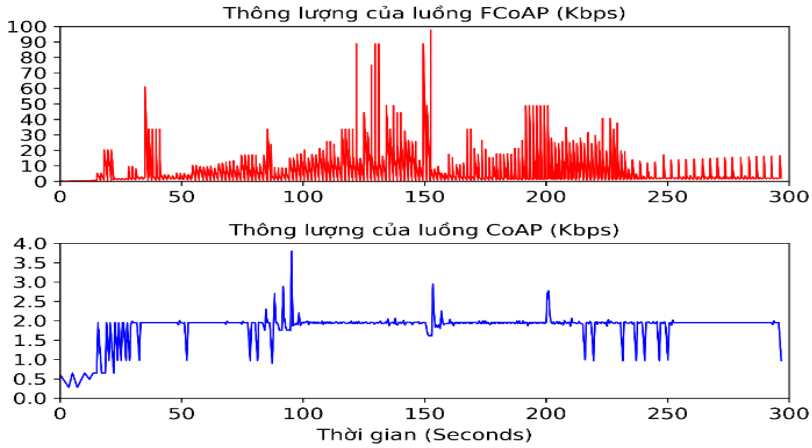


Hình 3.25 Độ trễ của FCoAP và CoAP khi có lưu lượng nền TCP/UDP

Hình 3.25 so sánh độ trễ của FCoAP và CoAP. Như trên hình, độ trễ của FCoAP và CoAP biến thiên theo sự thay đổi tốc độ của các luồng tin UDP và TCP. Độ trễ lớn ở các khoảng từ 80s - 120s và từ 150s - 200s là do UDP phát gói ở tốc độ cao hơn tại các khoảng này. Độ trễ trung bình của CoAP là xấp xỉ *1563,25* ms với khoảng tin cậy (*1550,77, 1575,74*) với các giá trị lớn tới *2007* ms ở 7s và *1970* ms ở

153s. Trong khi đó, độ trễ trung bình của FCoAP là xấp xỉ *1561,26* ms với khoảng tin cậy (*1554,82, 1579,70*) và giá trị lớn nhất là *1888* ms ở *153s* nhỏ hơn so với độ trễ lớn nhất của CoAP.

Hình 3.26 so sánh thông lượng của FCoAP và CoAP. Biến thiên thông lượng của FCoAP phản ánh điều khiển tốc độ phát của FCoAP để tương tranh băng thông khả dụng theo biến thiên của băng thông cổ chai. Trong khi đó, thông lượng CoAP ít thay đổi hơn. Thông lượng trung bình của CoAP là *1,895* Kbps với khoảng tin cậy (*1,863; 1,927*). CoAP có khá nhiều gói tin phải phát lại và phát lại đúp trong khoảng thời gian xảy ra tắc nghẽn. Trong khi đó, thông lượng trung bình của FCoAP xấp xỉ *9,228* Kbps với khoảng tin cậy (*7,907; 10,549*), mức tin cậy *99%*. Kết quả thí nghiệm cho thấy, FCoAP tận dụng được băng thông khả dụng khi chia sẻ băng thông cổ chai với CoAP, TCP và UDP để tăng thông lượng, trong khi CoAP không thể thực hiện điều này. Lý do là FCoAP sử dụng hệ FCS giúp điều chỉnh cập nhật tốc độ phát kịp thời và sát với tình trạng biến động của mạng.

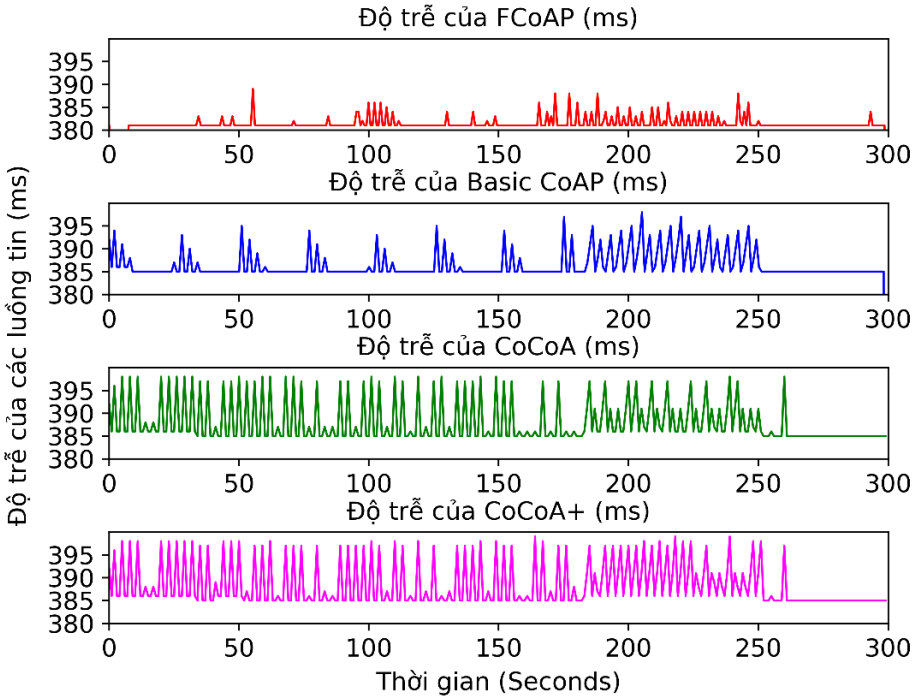


Hình 3.26 Thông lượng của FCoAP và CoAP khi có lưu lượng nền TCP/UDP

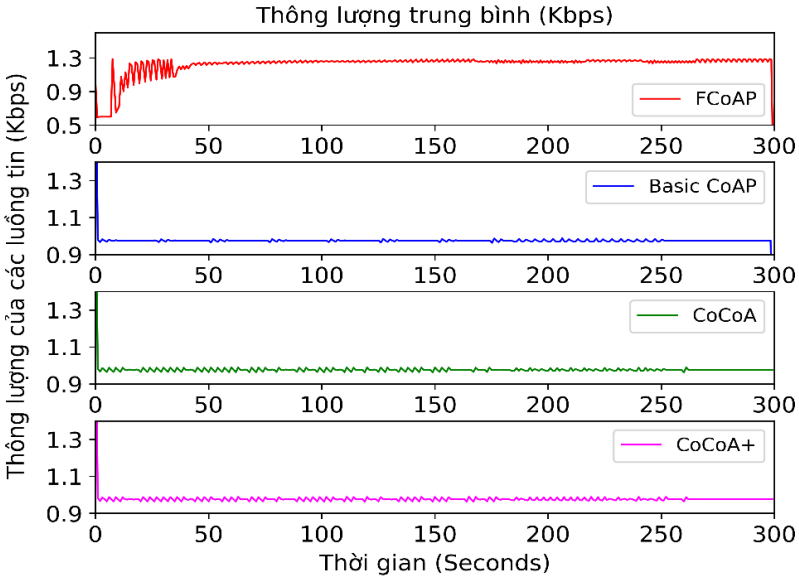
3.5.7. Kịch bản 3.6: So sánh FCoAP với CoAP, CoCoA, CoCoA+

Kịch bản 3.6 sử dụng mạng như hình 2.7 với các tham số như ở bảng 3.2, một luồng UDP có tốc độ phát thay đổi từ 2 Kbps đến 25 Kbps nhằm tạo biến động mạng. Kịch bản này so sánh FCoAP với các giao thức CoAP, CoCoA và CoCoA+. Hình 3.27 biểu thị độ trễ trung bình của các giao thức FCoAP, CoAP, CoCoA và CoCoA+. Độ trễ của FCoAP thấp và có biến thiên nhỏ. Trong khi đó độ trễ của

CoCoA và CoCoA+ lớn hơn và có biến thiên lớn (chi tiết xem bảng 3.6).



Hình 3.27 So sánh độ trễ của FCoAP, CoAP, CoCoA và CoCoA+



Hình 3.28 So sánh thông lượng của FCoAP, CoAP, CoCoA và CoCoA+

Hình 3.28 so sánh thông lượng của FCoAP, CoAP, CoCoA và CoCoA+. Thông lượng của CoAP, CoCoA và CoCoA+ tương đối ngang nhau. So với CoAP, CoCoA và CoCoA+, giao thức FCoAP có thông lượng cao hơn. Phần đầu FCoAP có dao động do FCoAP phát liên tiếp để kiểm tra băng thông cổ chai (chi tiết xem bảng

3.6).

Bảng 3.6: So sánh hiệu năng của FCoAP và các giao thức CoAP khác

Chỉ số hiệu năng	**FCoAP**	**CoAP**	**CoCoA**	**CoCoA+**
Tổng số gói gửi	381	299	300	300
Số gói phát thành công	381 (100%)	299 (100%)	300 (100%)	300 (100%)
Số gói phát lại	0 (0,0%)	0 (0,0%)	0 (0,0%)	0 (0,0%)
Số gói phát lại đúp	0 (0.0%)	0 (0.0%)	12 (4.0%)	7 (2,33%)
Số gói bị mất	0 (0.0%)	0 (0.0%)	12 (4.0%)	7 (2,33%)
Trễ trung bình	380,33 ms	385,18 ms	387,94 ms	388,33 ms
Thông lượng trung bình	1,245 Kbps	0,978 Kbps	0,979 Kbps	0,979 Kbps

Kết quả thí nghiệm trong bảng cho thấy: FCoAP có độ trễ trung bình nhỏ hơn CoAP, CoCoA và CoCoA+. Thông lượng trung bình của FCoAP cao hơn so với CoAP, CoCoA và CoCoA+. FCoAP truyền thành công nhiều gói tin so với CoAP, CoCoA và CoCoA+ trong cùng một điều kiện truyền. Có thể thấy kịch bản 3.6 sử dụng băng thông cổ chai tương tự ở kịch bản 3.2 (bảng 3.4) song có thêm luồng UDP thay đổi tốc độ phát. Vì vậy, số gói do FCoAP phát đi (381 gói ở bảng 3.6) nhỏ hơn so với 2558 gói ở kịch bản 3.2 (bảng 3.4) do luồng UDP giành nhiều băng thông hơn các luồng khác.

3.6. So sánh FCoAP với RCoAP

So với RCoAP, FCoAP có một số thay đổi cải tiến chính như sau:

(1) Thời gian khởi tạo của RCoAP là 2×RTT, còn của FCoAP là 6 chu kỳ RTT cho một phiên kết nối mới. Lý do là FCoAP cần thời gian kết nối ổn định để xác định băng thông cổ chai tương tự [27, 11]. Chi phí này nhỏ so với toàn bộ thời gian truyền tin. Ví dụ, nếu RTT = 100 ms, thời gian kết nối phiên truyền tin là 300s thì thời gian phát sinh của FCoAP do khởi tạo là 0,2%, cao hơn 0,167% so với RCoAP. Phiên kết nối kéo dài thì chi phí cho thời gian khởi tạo càng nhỏ.

(2) Tiêu đề gói tin FCoAP đều có thêm 8 Bytes ở phần tùy chọn so với CoAP [100], trong đó 4 Bytes ghi số thứ tự gói tin phục vụ cho kiểm tra mất gói, 4

Bytes ghi thời gian nhận dùng cho tính thông lượng tức thời và băng thông cổ chai. Số dòng lệnh của FCoAP phát sinh thêm so với RCoAP là 6. Việc xử lý tiêu đề gói khi gửi và khi nhận sẽ phát sinh thêm thời gian so với RCoAP (Phụ lục 4).

(3) Do có thêm phần xử lý cho hệ điều khiển mờ FCS, FCoAP có thêm 37 dòng lệnh so với RCoAP, phát sinh thêm thời gian xử lý so với RCoAP.

Như vậy, FCoAP có thêm chi phí xử lý so với RCoAP. Thời gian xử lý của FCoAP cao hơn so với RCoAP. Tuy nhiên, cả RCoAP và FCoAP đều chỉ sử dụng các vòng lặp đơn trong tính toán. Do đó, độ phức tạp tính toán của cả RCoAP và FCoAP đều là O(n). Trong phạm vi nghiên cứu, luận án không đi sâu vào tính toán độ phức tạp, tối ưu các tham số, chi phí năng lượng tiêu thụ, tác động của lỗi kênh vô tuyến. Đây là những hướng nghiên cứu tiếp trong tương lai.

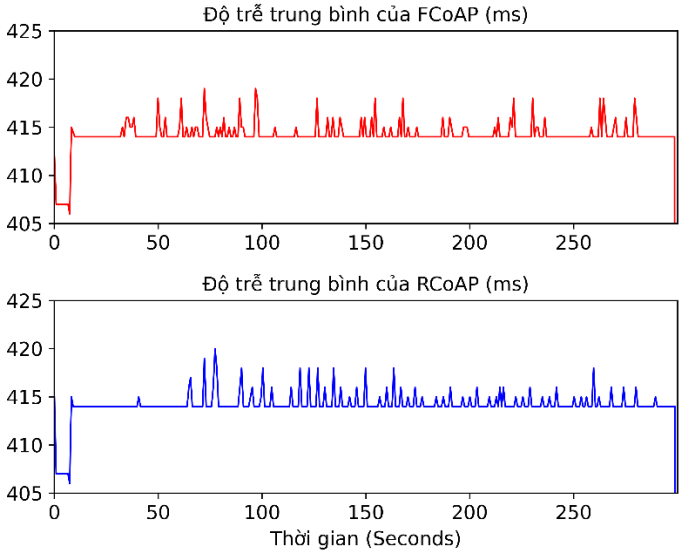
So sánh giữa FCoAP và RCoAP có thể rút ra:

- Cả FCoAP và RCoAP đều có khả năng truyền chuỗi gói tin cậy. Do đó, FCoAP và RCoAP phù hợp với các ứng dụng truyền luồng tin có lượng dữ liệu lớn, yêu cầu thời gian thực như ứng dụng luồng video, mạng camera giám sát, theo dõi y tế và chăm sóc sức khỏe, giám sát công cộng như đã nêu trong khuyến nghị của IETF [101, 102, 103, 55] và ITU [99, 57, 62, 59].
- FCoAP có khả năng xác định băng thông cổ chai và phát hiện sớm tắc nghẽn, sớm điều chỉnh tốc độ phát để hạn chế tắc nghẽn. Tuy nhiên, chi phí thời gian xử lý của FCoAP cao hơn CoAP. Mặt khác, do chu kỳ điều chỉnh tốc độ là một RTT và cần thời gian để tính toán hệ mờ nên FCoAP thích hợp hơn với các ứng dụng có RTT dài gồm nhiều chặng [103, 102, 55].
- So với FCoAP, RCoAP có thêm chức năng phát hiện nguyên nhân mất gói. Chức năng phát hiện nguyên nhân mất gói không cần trong FCoAP vì FCoAP liên tục điều chỉnh tốc độ phát chứ không giảm tốc độ phát đi một nửa khi có tắc nghẽn như ở RCoAP, hơn nữa thêm chức năng này sẽ làm FCoAP phức tạp thêm.
- RCoAP có thể phù hợp hơn cho các ứng dụng truyền chuỗi gói tin theo chặng

ngắn có RTT nhỏ. FCoAP thích hợp hơn cho các ứng dụng truyền chuỗi gói tin có kết nối dài gồm nhiều chặng như các ví dụ nêu trong [103, 102, 55].

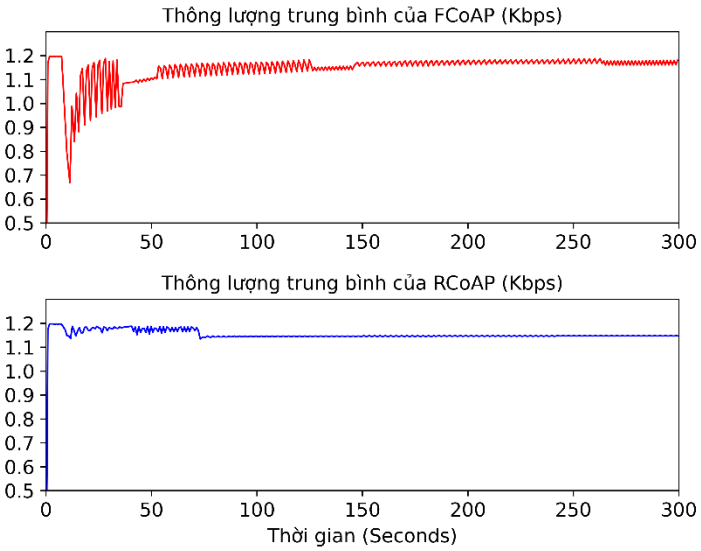
Hai kịch bản mô phỏng được sử dụng để so sánh RCoAP và FCoAP. Các tham số thiết lập mô phỏng tương tự như ở Bảng 3.2. Kịch bản thứ nhất (Kịch bản 3.7) sử dụng mạng hình sao như hình 2.6 để biểu diễn truyền chuỗi gói tin theo chặng ngắn (đơn chặng). Kịch bản thứ hai (Kịch bản 3.8) sử dụng mạng hình 2.7 để biểu diễn truyền chuỗi gói tin có kết nối dài gồm nhiều chặng. Cả hai kịch bản đều sử dụng thêm một luồng UDP có tốc độ phát thay đổi từ 2 Kbps đến 25 Kbps nhằm tạo biến động mạng.

Trong kịch bản 3.7, băng thông cổ chai được mô phỏng tương tự các thí nghiệm ở phần trên với BW = 60 Kbps và D = 300 ms.



Hình 3.29 Độ trễ của FCoAP và RCoAP trong chặng đơn

Hình 3.29 so sánh độ trễ của các luồng tin FCoAP và RCoAP. Như trên hình, do biến động của băng thông cổ chai, độ trễ của FCoAP và RCoAP đều biến thiên, song duy trì ở giá trị trung bình 413 ms đối với FCoAP và 412 ms đối với RCoAP. Hình 3.30 so sánh thông lượng của FCoAP và RCoAP. Thông lượng trung bình của RCoAP là 1,157 Kbps cao hơn so với 1,148 Kbps của FCoAP trong cùng điều kiện thí nghiệm chặng ngắn này. Các kết quả mô phỏng cụ thể được biểu thị trong bảng 3.7.



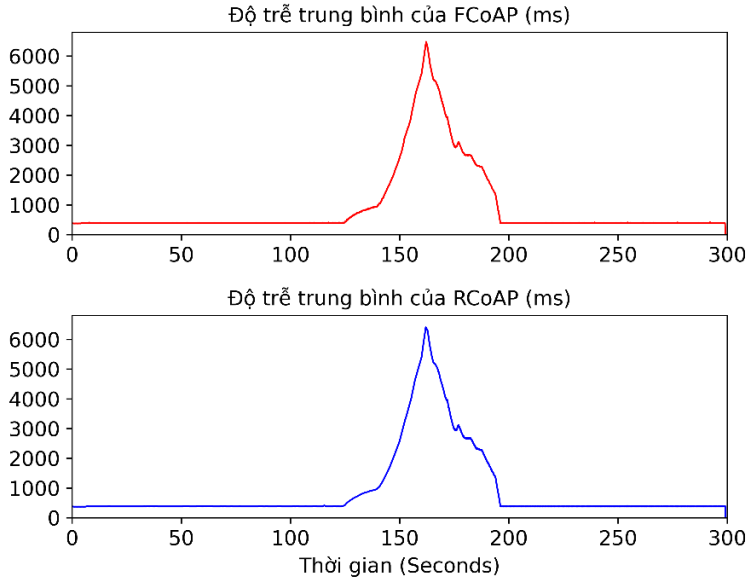
Hình 3.30 Thông lượng của FCoAP và RCoAP trong chặng đơn

Bảng 3.7: So sánh FCoAP và RCoAP trong đơn chặng

Tham số hiệu năng	**FCoAP**	**RCoAP**
Tổng số gói gửi	351	354
Số gói phát thành công	351 (100%)	354 (100%)
Số gói phát lại	0 (0,0%)	0 (0,0%)
Số gói phát lại đúp	0 (0,0%)	0 (0,0%)
Số gói bị mất	0 (0,0%)	0 (0,0%)
Trễ trung bình	413,03 ms	412,99 ms
Thông lượng trung bình	1,148 Kbps	1,157 Kbps

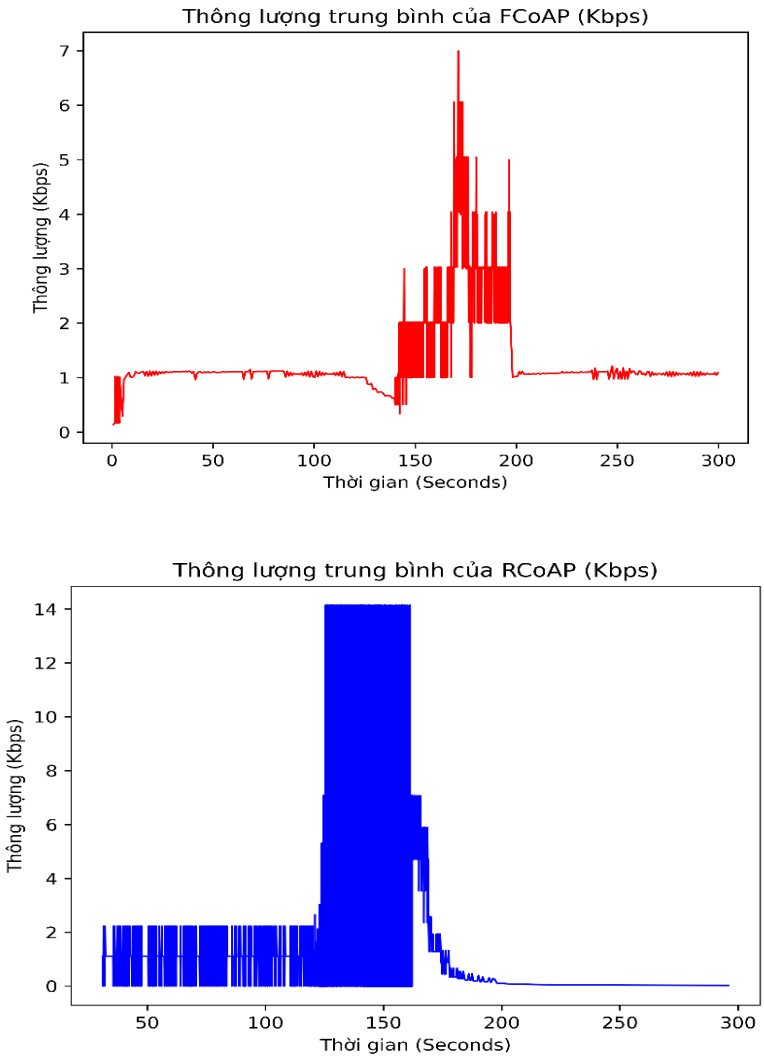
Theo kết quả trong bảng: không có mất gói xảy ra, RCoAP có độ trễ nhỏ hơn FCoAP, thông lượng trung bình của RCoAP cao hơn so với FCoAP. RCoAP truyền thành công nhiều gói tin so với FCoAP. Các kết quả cho thấy RCoAP phù hợp hơn so với FCoAP cho các ứng dụng truyền chuỗi gói tin theo chặng ngắn (đơn chặng). Có thể thấy kịch bản 3.7 sử dụng băng thông cổ chai tương tự kịch bản 3.2 (bảng 3.4) song có thêm luồng UDP thay đổi tốc độ phát tương tự ở kịch bản 3.6 (bảng 3.6). Vì vậy, số gói của FCoAP phát đi (351 gói ở bảng 3.7) nhỏ hơn so với số gói ở kịch bản 3.2 (bảng 3.4) và xấp xỉ số gói ở kịch bản 3.6 (bảng 3.6) do FCoAP tương tranh băng thông với luồng UDP.

Kịch bản 3.8 sử dụng mạng hình 2.7 để biểu diễn truyền chuỗi gói tin có kết nối dài gồm nhiều chặng. Kịch bản này cũng sử dụng các tham số thiết lập mô phỏng tương tự như ở Bảng 3.2 song có băng thông cổ chai lớn hơn được mô phỏng với BW = 75 Kbps và D = 500 ms. Một luồng UDP có tốc độ phát thay đổi từ 2 Kbps đến 25 Kbps trong khoảng từ 100s – 200s nhằm tạo biến động mạng. Hình 3.31 so sánh độ trễ của các luồng tin FCoAP và RCoAP còn hình 3.32 so sánh thông lượng của FCoAP và RCoAP.



Hình 3.31 Độ trễ của FCoAP và RCoAP trong chặng dài

Như trên hình 3.31, do biến động của băng thông cổ chai, tắc nghẽn xảy ra trong khoảng 120 s – 200 s. Độ trễ của FCoAP và RCoAP đều biến thiên và đạt giá trị cao trong khoảng này. Ở các khoảng khác, độ trễ của FCoAP và RCoAP được duy trì ở giá trị trung bình 899 ms đối với FCoAP và 805 ms đối với RCoAP. Hình 3.32 so sánh thông lượng của FCoAP và RCoAP. Thông lượng ổn định khi không xảy ra tắc nghẽn song có biến thiên lớn khi có tắc nghẽn đối với cả FCoAP và RCoAP trong điều kiện thí nghiệm chặng dài này.



Hình 3.32 Thông lượng của FCoAP và RCoAP trong chặng dài

Các kết quả mô phỏng cụ thể được biểu thị trong Bảng 3.8 như sau.

Bảng 3.8: So sánh FCoAP và RCoAP trong chặng dài

Tham số hiệu năng	**FCoAP**	**RCoAP**
Tổng số gói gửi	2916	2754
Số gói phát thành công	2916 (100%)	2754 (100%)
Số gói phát lại	484 (16,60%)	442 (16,05%)
Số gói phát lại đúp	484 (16,60%)	442 (16,05%)
Số gói bị mất	0 (0,0%)	0 (0,0%)
Trễ trung bình	899,82 ms	805,08 ms
Thông lượng trung bình	9,521 Kbps	8,987 Kbps

Theo kết quả trong bảng cho thấy: đối với chặng dài, FCoAP truyền thành công nhiều gói tin so với RCoAP. Mặc dù độ trễ của FCoAP cao hơn chút so với RCoAP song thông lượng trung bình của FCoAP cao hơn RCoAP. Điều này chứng tỏ FCoAP phù hợp hơn so với RCoAP cho các ứng dụng truyền chuỗi gói tin theo chặng dài (đa chặng) như đã nhận xét ở phần trên.

3.7. Kết luận chương 3

Đóng góp thứ hai của luận án là một hệ điều khiển mờ và giao thức FCoAP (Fuzzy CoAP) dựa vào hệ điều khiển mờ để điều khiển tắc nghẽn đã được trình bày trong chương 3. Do các đại lượng tác động đến tắc nghẽn cũng như trạng thái tắc nghẽn là khá mờ, không rõ ràng, có sự chồng lấp, giải pháp sử dụng hệ điều khiển mờ đã được đề xuất nhằm khắc phục các hạn chế (4), (5) và (6) của CoAP và các cải tiến liên quan đã nêu ở chương 1.

Phần đầu chương 3 trình bày giải pháp sử dụng hệ điều khiển mờ với việc tính toán các tham số điều khiển, phương pháp phát hiện sớm tắc nghẽn, phương pháp tính toán băng thông cổ chai và tải lưu lượng chuỗi gói, lựa chọn và tính toán các đầu vào và đầu ra hệ điều khiển mờ. Tiếp đó, luận án đã đưa ra thiết kế hệ điều khiển mờ FCS và đề xuất giao thức FCoAP dựa vào hệ FCS để điều khiển tắc nghẽn với các nội dung: cơ chế điều khiển, các trạng thái giao thức, các thuật toán của giao thức FCoAP. FCoAP có các điểm cải tiến so với RCoAP về khả năng phát hiện sớm tắc nghẽn, điều khiển tốc độ dựa vào tính toán băng thông cổ chai và biến thiên động của các tham số mạng. Phần cuối chương đưa ra tính toán hiệu năng FCoAP và các kết quả mô phỏng thử nghiệm cho FCoAP, thực hiện so sánh FCoAP với CoAP, các giao thức đã chuẩn hóa hiện có là CoCoA, CoCoA+ và với giao thức RCoAP đã đề xuất ở chương 2. Các kết quả mô phỏng cho thấy, FCoAP đạt hiệu quả tốt hơn so với CoAP cũng như các cải tiến CoCoA và CoCoA+ về các tham số hiệu năng như: thông lượng, độ trễ, tỷ lệ mất gói, tỷ lệ phát lại, tỷ lệ phát lại đúp. Luận án đã so sánh FCoAP với RCoAP về các cải tiến, chi phí tính toán và sự phù hợp trong ứng dụng. Nội dung đóng góp trong chương 3 đã được công bố trong [J5] và [J6].

KẾT LUẬN

Mạng IoT đang phát triển mạnh với nhiều ứng dụng đa dạng trong đời sống, tạo điều kiện phát triển một xã hội số với việc kết nối mọi thứ vào Internet. Số lượng thiết bị IoT kết nối mạng tăng nhanh, nhiều, đa dạng. Lưu lượng lớn và đa dạng từ nhiều nguồn ứng dụng.

Tắc nghẽn là một vấn đề thách thức đối với mạng IoT do có nhiều yếu tố khác biệt của mạng IoT so với mạng Internet truyền thống. Lưu lượng trong mạng IoT đa dạng hơn do có nhiều ứng dụng đa dạng mới. Môi trường truyền của mạng IoT có nhiều thách thức kỹ thuật hơn do có những đoạn mạng vô tuyến có băng thông hạn chế và biến đổi động. Các kênh truyền có thể có độ trễ biến động, khả năng mất gói do lỗi và nhiễu kênh vô tuyến. Nhiều ứng dụng thu thập dữ liệu của mạng IoT có nhu cầu truyền chuỗi gói tin với lưu lượng biến thiên và có yếu tố thời gian thực, đòi hỏi độ tin cậy. Các thiết bị IoT thường nhỏ gọn, có tài nguyên hạn chế, nên không thể áp dụng các cơ chế điều khiển phức tạp như của TCP.

Trong số các giao thức cho mạng IoT, CoAP là một giao thức đã được chuẩn hóa cho tầng ứng dụng và rất phổ biến cho truyền dữ liệu tin cậy. Tuy nhiên, cơ chế điều khiển tắc nghẽn của CoAP còn đơn giản, không đáp ứng được yêu cầu phát hiện sớm và điều khiển tắc nghẽn. Một số cải tiến CoAP chủ yếu theo hướng dựa vào mất gói, không có khả năng phát hiện sớm tắc nghẽn, chỉ điều khiển tốc độ phát lại mà không điều khiển tốc độ phát, không thích hợp cho truyền chuỗi gói tin.

Qua nghiên cứu cơ chế điều khiển của CoAP, các cải tiến của nó và các công trình nghiên cứu liên quan, luận án đã phân tích các hạn chế còn tồn tại về khả năng phát hiện sớm tắc nghẽn, khả năng điều khiển tốc độ phát và truyền chuỗi gói. Trên cơ sở đó, luận án đã nghiên cứu đề xuất một mô hình phân tích truyền tin theo chuỗi gói có tin cậy cho CoAP. Dựa vào mô hình này, luận án đã nghiên cứu, đề xuất một giải pháp điều khiển tăng/giảm tốc độ phát CoAP dựa vào phát hiện sớm nguy cơ mất gói, trạng thái mạng và đường truyền nhằm điều khiển tắc nghẽn mạng, nâng cao hiệu quả truyền tin. Luận án đã đề xuất một giao thức mới RCoAP dựa theo mô hình truyền chuỗi gói. Từ việc phân tích, đánh giá các tham số tác động đến tắc

nghẽn, luận án đã chỉ ra khả năng sử dụng một hệ điều khiển mờ để tạo khả năng điều khiển linh hoạt. Trên cơ sở đó, luận án đã đề xuất một giao thức mới FCoAP dựa vào logic mờ. Các kết quả lý thuyết đã được kiểm chứng qua mô phỏng thử nghiệm.

A. Đóng góp mới của luận án

Luận án có các kết quả đóng góp chính như sau:

1) Xây dựng một mô hình phân tích truyền tin theo chuỗi gói cho CoAP và xây dựng giao thức mới RCoAP dựa trên tốc độ để điều khiển tắc nghẽn trong mạng IoT với cơ chế điều khiển tăng giảm tốc độ phát phù hợp với tình trạng tắc nghẽn nhằm đạt được hiệu năng cao về độ trễ, thông lượng, tỷ lệ mất gói, tỷ lệ phát lại và tỷ lệ phát lại đúp so với các cơ chế CoAP chuẩn hóa hiện có. Mô hình phân tích truyền tin theo chuỗi gói được công bố trong [J1, J2], giao thức RCoAP được công bố trong [J3, J4].
2) Xây dựng giao thức mới FCoAP điều khiển tắc nghẽn sử dụng hệ điều khiển mờ theo biến thiên động của tình trạng tắc nghẽn và các tham số mạng nhằm đạt được hiệu năng cao về độ trễ, thông lượng, tỷ lệ mất gói, tỷ lệ phát lại và tỷ lệ phát lại đúp so với các cơ chế CoAP chuẩn hóa hiện có trong điều kiện mạng biến thiên động, kể cả khi có tắc nghẽn nghiêm trọng. Kết quả được công bố trong [J5, J6].

B. Hướng phát triển tiếp

Các hướng có thể phát triển tiếp của luận án gồm:

- Nghiên cứu khả năng tối ưu các tham số điều khiển cho RCoAP và FCoAP.
- Nghiên cứu đánh giá độ phức tạp của RCoAP và FCoAP so với CoAP.
- Nghiên cứu đánh giá RCoAP và FCoAP trong môi trường mạng ứng dụng cụ thể.

DANH MỤC CÁC CÔNG TRÌNH KHOA HỌC ĐÃ CÔNG BỐ

[J1] **Le Thi Thuy Duong**, Hoang Dang Hai, Pham Thieu Nga (2022), "A control mechanism for reliable burst data transfer in IoT networks", *Journal of Science and Technology on Information and Communications*, ISSN 2525-2224, CS.01, No. 2, 2022. pp. 38-49.

[J2] Hoàng Đăng Hải, **Lê Thị Thùy Dương**, Phạm Thiếu Nga (2019), "Giải pháp điều khiển chống tắc nghẽn trong mạng IoT", *Journal of Science and Technology on Information and Communications*, ISSN 2525-2224, No. 1, CS.01, 2019. pp. 7-18.

[J3] **Le Thi Thuy Duong**, Hoang Dang Hai, Pham Thieu Nga (2023), "Avoiding congestion for CoAP burst traffic", **EAI endorsed Transactions on Internet of Things**, Vol. 9, No. 1, p. e2, Mar. 2023.

[J4] Hoang Dang Hai, **Le Thi Thuy Duong** (2021), "RCOAP: A Rate Control Scheme for Reliable Bursty Data Transfer in IoT Networks", **IEEE Access, SCI/Q1**, Vol. 9 (2021), doi: 10.1109/ACCESS.2021.3135435. pp. 169281-169298.

[J5] **Lê Thị Thùy Dương**, Hoàng Đăng Hải, Phạm Thiếu Nga (2023), "Điều khiển mờ hỗ trợ giao thức CoAP nhằm chống tắc nghẽn mạng Internet vạn vật", Tạp chí Nghiên cứu Khoa học Công nghệ Quân sự, số 88, tháng 6/2023, trang 22-33.

[J6] T. N. Pham, D. H. Hoang, **T. T. Duong Le** (2022), "Fuzzy Congestion Control and Avoidance for CoAP in IoT Networks," **IEEE Access, SCI/Q1**, Vol. 10 (2022), doi: 10.1109/ACCESS.2022.3211296. pp. 105589-105611.

TÀI LIỆU THAM KHẢO

Tiếng Việt

[1]. Hồ Thuần, Đặng Thanh Hà, [2007], *Logic mờ và ứng dụng*, Nhà xuất bản Đại học Quốc gia Hà Nội.

[2]. Nguyễn Trọng Thuần, [2000], *Điều khiển Logic và ứng dụng*, Nhà xuất bản khoa học và Kỹ thuật.

Tiếng Anh

[3]. M. Ahmad, M. Hussain, et.al (2018), "End-to-End Loss Based TCP Congestion Control Mechanism as a Secured Communication Technology for Smart Healthcare Enterprises", *IEEE Access*, Vol. 6, pp. 11641-11656.

[4]. P. Aimtongkham, T. G. Nguyen, et.al (2018), "Congestion control and prediction schemes using Fuzzy logic system with adaptive membership function in wireless sensor networks," *Wireless Comm. Mobile Computing*, pp. 1-19.

[5]. P. Aimtongkham, P. Horkaew, C. So-In (2021), "An Enhanced CoAP Scheme Using Fuzzy Logic with Adaptive Timeout for IoT Congestion Control," *IEEE Access*, vol. 9, 2021, pp.58967-58981.

[6]. A. Al-Fuqaha, M.Mohammadi, et.al (2015), "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications", *IEEE Communications Survey & Tutorials*, Vol. 17, No. 4, pp. 2347-2376.

[7]. HA. Al-kashoash, H. Kharrufa, et.al. (2018), "Congestion control in wireless sensor and 6LoWPAN networks: toward the Internet of Things", *Wireless Networks*, https://doi.org/10.1007/s11276-018-1743-y, 2018

[8]. M. Allman, V. Paxson, W. R. Stevens (1999), "TCP congestion control," *RFC 2581*, pp. 1–14. [Online]. Available: https://doi.org/10.17487/RFC2581

[9]. E. Ancillotti E., S. Boletrtieri, R. Bruno (2018), "RTT-based Congestion Control for the Internet of Things", *Proc. of 16th IFIP WG 6.2, Internl Conference, WWIC 2018*, Boston, USA, June 18-20, 2018, pp.3-15.

[10]. E. Ancillotti, R. Bruno, et.al (2018), "Design and Evaluation of a Rate–Based Congestion Control Mechanism in CoAP for IoT Applications," in *Proc. 19th IEEE Int. Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, Greece, Jun. 2018, pp.14–15.

[11]. E. Ancillotti, R. Bruno, "BDP–CoAP: Leveraging Bandwidth-Delay Product for Congestion Control in CoAP," in Proc. of 5th IEEE World Forum on Internet of Things (WF-IoT), Ireland, Apr. 2019, pp. 656-661.

[12]. M. Arora, S. Upadhyaya, N. Kashyap (2018), "Flexible congestion control using fuzzy logic for Wireless Sensor Networks", *Intl. Journal of Computer Sciences and Engineering*, Vol, 6, Iss. 5, May 2018.

[13]. A. Betzler, C. Gomez, I. Demirkol, M. Kovatsch (2015), "Congestion Control for

CoAP Cloud Services," *Proc of 2015 IEEE Emerging Technology and Factory Automation (ETFA).*

[14]. A. Betzler, C. Gomez, I. Demirkol, J. Paradells (2016), "CoAP congestion control for the internet of things", *IEEE Communications Magazine,* 54(7), July 2016. ISSN 0163-6804, pp. 154–160.

[15]. A. Betzler, C. Gomez, et.al (2016), "CoCoA+: An advanced congestion control mechanism for CoAP," *Ad Hoc Networks,* 33, pp.126-139.

[16]. N. Benamar, J. Harri, J. Lee, T. Ernst (2019), "Basic Support for IPv6 Networks Operating Outside the Context of a Basic Service Set over IEEE Std. 802.11," *RFC 8691,* https://tools.ietf.org/html/rfc8691.

[17]. R.E. Bellman, L.A. Zadeh (1970), "Decision-making in a fuzzy environment," *Management Science,* 17(4), pp. 141–164.

[18]. R. Bhalerao, S. S. Subramanian, J. Pasquale (2016), "An analysis and improvement of congestion control in the CoAP Internet-of-Things protocol", *Proc. 13th IEEE Annual Consumer Communications Networking Conference (CCNC),* pp. 889–894.

[19]. S. Biaz, N. H. Vaidya (2005), "De-Randomizing" congestion losses to improve TCP performance over wired-wireless networks," *IEEE/ACM Transactions on Networking*, vol. 13 (3), pp. 596-608.

[20]. S. Biaz, N.H. Vaidya (1998), "Distinguishing congestion losses from wireless transmission losses: a negative result," In *Proc. 7th Intern. Conf. on Computer Comm. and Networks*, pp. 722-731.

[21]. S. Bolettieri, G. Tanganelli, et.al (2018), "pCoCoA: A precise congestion control algorithm for coap", *Ad Hoc Networks*, 80, pp. 116 – 129.

[22]. C Bormann, A Betzler, C Gomez, I Demirkol (2014), "Coap simple congestion control/advanced", *RFC draft: draft-bormann-core-cocoa-02,* 2014.

[23]. C. Bormann, A. Betzler, C. Gomez, I. Demirkol (2018), "CoAP Simple Congestion Control/Advanced", *RFC-Draft, [Online]. Available: https://tools.ietf.org/id/draft-bormann-core-cocoa-03.txt.*

[24]. C. Bormann, Z. Shelby (2021), "Block–Wise Transfers in the Constrained Application Protocol (CoAP)", *RFC 7959,* [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7959.

[25]. M. Boucadair, J. Shallow (2021), "Constrained Application Protocol (CoAP) Block-Wise Transfer Options Supporting Robust Transmission," *RFC 9177,* [Online]: https://datatracker.ietf.org/doc/html/rfc9177.

[26]. G. Buchholcz, T. Ziegler, T. Van Do (2005), "TCP-ELN: on the protocol aspects and performance of explicit loss notification for TCP over wireless networks," in *Prof. 1st Internl. Conf. on Wireless Internet (WICON'05),* pp. 172-179.

[27]. N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, V. Jacobson (2016), "BBR: Congestion-Based Congestion Control," *Queue*, 14 (5), pp. 20–53.

[28]. S. Cen, P. C. Cosman, G. M. Voelker (2003), "End-to-end differentiation of congestion and wireless losses," in *IEEE/ACM Transactions on Networking*, vol. 11

(5), pp. 703-717.

[29]. W. Chang, P. Chen, C. Yang (2013), "Robust Fuzzy Congestion Control of TCP/AQM Router via Perturbed Takagi-Sugeno Fuzzy Models," in *Intl. Journal of Fuzzy Systems*, Vol.15, No.2, pp.203-213.

[30]. A. Chaudhary, S. K. Peddoju, K. Kadarla (2017), "Study of internet-of-things messaging protocols used for exchanging data with external sources", In *IEEE 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, pp. 666–671.

[31]. S.H. Chen (1985), "Ranking fuzzy numbers with maximizing set and minimizing set," *Fuzzy Sets and Systems*, 17, pp. 113–129.

[32]. Y. Chen, L. Lu, X. Yu, X. Li (2019), "Adaptive Method for Packet Loss Types in IoT: An Naive Bayes Distinguisher," *Electronics*, 8(2), pp.134-1.

[33]. C. Chrysostomou, A. Pitsillides, L., Rossides, et al (2003), "A.: Congestion Control in Differentiated Services Networks using Fuzzy-RED," in *IFAC Control Engineering Practice (CEP) Journal* 11(19), pp. 1153–1170.

[34]. M. Collina, M. Bartolucci, et.al (2014), "Internet of things application layer protocol analysis over error and delay prone links", In *7th Advanced Satellite Multimedia Systems Conf and the 13th Signal Processing for Space Comm. Workshop (ASMS/SPSC)*, pp. 398–404.

[35]. G. Cui, H. Wang, et.al (2021), "NDN congestion control based on fuzzy comprehensive evaluation algorithm," *IOP Publishing, Journal of Physics, CIBDA ser*. 1883 012008 (2021), pp.1-9.

[36]. PK. Donta, SN. Srirama, etal. (2022), "Survey on recent advances in IoT application layer protocols and machine learning scope for research directions", Digital Communication and Networks 8 (2022), pp.727-744.

[37]. S. Deshmukh, V.T. Raisinghani (2020), "AdCoCoA–Adaptive Congestion Control Algorithm for CoAP," in Proc. of 11th IEEE Int. Conf. on Computing, Communication and Networking Technologies (ICCCNT), Kharagpur, India, Jul. 2020, pp. 1-7.

[38]. P.K. Donta, S.N. Srirama, et al., "iCoCoA: intelligent congestion control algorithm for CoAP using deep reinforcement learning," in Journal Ambient Intell Human Computer, Vol. 14 (2023), pp. 2951–2966

[39]. D.H. Esawi, G. Attiya, G. Allam (2021), "Fuzzy Controller based TCP-Vegas Enhancement for Congestion Control," in Menoufia Journal of Electronic Engineering Research (MJEER), 30 (2), Jul. 2021, pp.39-44.

[40]. ETSI TR 103 375 (2016), "Smart M2M, IoT Standards landscape and future evolutions", Technical Report, ETSI Standard organisation, 10/2016, pp.1-87.

[41]. F. Filali (2005), "Link-Layer Fragmentation and Retransmission Impact on TCP Performance in 802.11-based Networks," Proc. of IFIP mobile and wireless communications networks conference (MWCN), 2005.

[42]. S. Floyd, R. Gummadi, S. Shenker (2001), "Adaptive RED: An Algorithms for Increasing the Robustness of RED's Active Queue Management," Report No. 301, Internl. Computer Science Institute, Berkeley, CA 94704, August 2001, https://www.icsi.berkeley.edu/icsi/ node/2032

[43]. S. Floyd, M. Handley (2008), "TCP Friendly Rate Control (TFRC): Protocol Specification", Network Working Group, RFC 5348.

[44]. S. Floyd, M. Handley, J. Padhye, J. Widmer (2000), “Equation-based Congestion Control for Unicast Applications”, ACM SIGCOMM Computer Communication Review, 30(4), pp 43–56.
[45]. C. Gomez, A. Arcia-Moret, J. Crowcroft (2018), “TCP in the Internet of Things: from ostracism to prominence”, IEEE Journal of Internet Computing, 22(1), Jan/Feb 2018, pp.29-41.
[46]. A. Gurtov, T. Henderson, S. Floyd (2004), “The NewReno Modification to TCP’s Fast Recovery Algorithm,” RFC 3782.
[47]. S. Ha, I. Rhee, L. Xu (2008), “CUBIC: A new TCP-friendly High-speed TCP variant,” in SIGOPS Oper. Syst. Rev. 42 (5), pp. 64–74.
[48]. H. Haile, K. Grinnemo, S. Ferlin, et.al. (2021), “End-to-end congestion control approaches for high throughput and low delay in 4G/5G cellular networks,” Computer Networks, Vol. 186, Feb. 2021.
[49]. T. Henderson, S. Floyd, A. Gurtov, Y. Nishida (2012), “The NewReno Modification to TCP's Fast Recovery Algorithm,” RFC 6582, https://tools.ietf.org /html/rfc6582.
[50]. H. Hirotakaster (2022), “CoAP client, server library for Spark Photon, Spark Core,” Available: https://github.com/hirotakaster/CoAP. Accessed date: 16/08/2022.
[51]. Dang Hai Hoang (2003), "Quality of Service Control in the Mobile Wireless Environments", PETER LANG Europäischer Publisher, Frankfurt/Main, Berlin, Bern, Bruxelles, New York, Oxford, Wien. ISBN 3-631-50578-7. US – ISBN 0-8204-6402-3. 2003.
[52]. M.H. Homaei, S. Sgamshirband, et al. (2020), “An Enhanced Distributed Congestion Control Method for Classical 6LowPAN Protocols Using Fuzzy Decision System,” IEEE Access, Vol. 8, 2020, doi: 10.1109/ACCESS.2020.2968524, pp. 20628-20645.
[53]. A. Hornsby, R. Walsh (2010), “From instant messaging to cloud computing, an xmpp review”, In IEEE International Symposium on Consumer Electronics (ISCE 2010), 2010. doi: 10.1109/ISCE.2010.5523293, pp. 1–6.
[54]. IEEE (2015), “Towards a definition of the Internet of Things IoT”, Microsoft Report, Revision 1 (by R.Minerva, et.al), 27 May 2015, 86 pages.
[55]. IETF RFC 8763, IETF (2020), “Deployment Considerations for Information-Centric Networking (ICN)”, IETF, available: https://rfc-editor.org/info/rfc8763.
[56]. International Telecommunication Union (ITU) (2012), “Y.2060: Overview of the Internet of Things”, ITU-T Y.2060. Series Y, 2012.
[57]. International Telecommunication Union (ITU) (2016), “Y.4113: Requirements of the network for the Internet of things Overview of the Internet of Things”, ITU-T Y.4113. Series Y, 09/2016.
[58]. International Telecommunication Union (ITU) (2022), “Y.3080: Information-centric networking in networks beyond IMT-2020: Requirements and mechanisms of the transport layer”, ITU-T Y.3080. Series Y, 09/2022.
[59]. International Telecommunication Union (ITU) (2016), “YSTR-M2M-UCC: OneM2M – Use case collection”, ITU-T Technical Report, 15/7/2017.
[60]. International Telecommunication Union (ITU) (2014), “M2M service layer: APIs and protocols overview”, ITU-T Technical Report, 04/2014.
[61]. International Telecommunication Union (ITU) (2021), “Requirements and capabilities of network connectivity management in the Internet of things”, Series Y:

Global information infrastructure
[62]. International Telecommunication Union (ITU) (2015), "Y.4413/F.748.5: Requirements and reference architecture of machine-to-machine service layer", Series Y: Global information infrastructure
[63]. V. Jacobson (1988), "Congestion avoidance and control," Proc on Communications Architectures and Protocols, SIGCOMM '88, ACM, New York, NY, USA, pp. 314–329.
[64]. R. Jain (1989), "A delay-based approach for congestion avoidance in interconnected heterogeneous computer networks," ACM SIGCOMM Computer Communication Review, 19 (5), pp 56–71.
[65]. S. Jaiswal, A. Yadav (2013), "Fuzzy based adaptive congestion control in wireless sensor networks," in Proc. of 6th International Conference on Contemporary Computing, pp. 433-438.
[66]. I. Jarvinen, I. Raitahila, Z. Cao, M. Kojo (2018), "FASOR Retransmission Timeout and Congestion Control Mechanism for CoAP", In IEEE Global Communications Conference (GLOBECOM), pp. 1–7.
[67]. I. Jarvinen, M. Kojo, I. Raitahila, Z. Cao (2020), "Fast-Slow Retransmission Timeout and Congestion Control Algorithm for CoAP," RFC-draft, https://tools.ietf.org/id/draft-jarvinen-core-fasor-02.html
[68]. H. Jiang, Q. Li, G. Shen, et.al. (2021), "When Machine Learning Meets Congestion Control: A Survey and Comparison," Computer Networks, Vol. 192, 108033, June 2021, pp. 1-23.
[69]. J.H. Jung, M. Gohar, S.J. Koh (2020), "CoAP–Based Streaming Control for IoT Applications," Electronics, 9 (8), 1320, pp. 2-19.
[70]. S. Keshav (1991), "A Control-theoretic Approach to Flow Control," in ACM SIGCOMM, Computer Communication Review, 21 (4), Sept. 1991, pp. 3–15.
[71]. S. Keshav (1991), "The packet pair flow control protocol," Tech. report TR-91-028, Internl. Computer Science Institute (ICSI), Berkeley, Available at: http://www.icsi.berkeley. edu/pubs/techreports/tr-91-028.pdf.
[72]. K. Kharb, B. Sharma, T.C. Aseri (2016), "Reliable and Congestion Control Protocols for Wireless Sensor Networks", Intl Journal of Engineering and Technology Innovation, 6 (1), 2016, pp. 68-78.
[73]. J.A. Khan, M. Shahzad, AR. Butt (2018), "Sizing Buffers of IoT Edge Routers", Proc of 1st Internl Workshop on Edge Systems, Analytics and Networking, EdgeSys'18, 10-15 June 2018, Munic, Germany, pp.55-60
[74]. L. Kleinrock (2018), "Internet congestion control using the power metric: Keep the pipe just full, but no fuller", Ad Hoc Networks, 2018, 1-16.
[75]. R.K. Lam, KC. Chen (2013), "Congestion Control for M2M Traffic with Heterogeneous Throughput Demands", Proc of IEEE Wireless Communications and Networking Conference (WCNC) 2013, pp.1452-1457.
[76]. J. J. Lee, S. M. Chung, B. Lee, et.al (2016), "Round trip time based adaptive congestion control with CoAP for sensor network", In Internl. Conf. on Distributed Computing in Sensor Systems (DCOSS16), pp. 113–115.
[77]. J.J. Lee, K.T. Kim, H.Y. Youn (2016), "Enhancement of congestion control of Constrained Application Protocol/Congestion Control/Advanced for Internet of Things environment," Int. J. of Distributed Sensor Networks, 12 (11), pp. 1-13.
[78]. S. Li, N. Zhang, S.Lin, etal. (2018), "Joint Admission Control and Resource

Allocation in Edge Computing for Internet of Things", Edge Computing for the Internet of Things. IEEE Network, Vol. Jan/Feb. 2018, pp. 72-79.

[79]. C. Lim (2019), "A Survey on Congestion Control for CoAP over UDP," Int. Journal of Internet, Broadcasting and Comm., 11 (1), pp.17-26.

[80]. J. Lin, W. Yu, N. Zhang, et.al (2017), "A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications", IEEE Internet of Things Journal, 4(5), pp. 1125–1142.

[81]. S. Liu, T. Basar, R. Srikant (2008), "TCP-Illinois: A loss- and delay-based congestion control algorithm for high-speed networks," Performance Evaluation, 65 (6), pp. 417-440.

[82]. S. Maesoser (2022), "A partial CoAP implementation with mDNS support, multicast", Available: https://github.com/maesoser/ns3-coap/

[83]. E.H. Mamdani (1974), "Applications of fuzzy algorithms for simple dynamic plant," in Proceedings of IEEE 121(12), pp. 1585–1588.

[84]. A. Mishra (2018), "Performance Analysis of TCP Tahoe, Reno and New Reno for Scalable IoT network clusters in QualNet@Network Simulator", Intl Journal of Computer Sciences and Engineering, 6 (8), pp.347-355.

[85]. N. Misha, L. P. Verma, et.al. (2018), "An Analysis of IoT Congestion Control Policies", Intl Conference on Computational Intelligence and Data Science (ICCIDS 2018), pp. 444-450.

[86]. J. Misic, VB. Misic (2018), "Lightweight data streaming from IoT devices", Proc of IEEE Intl. Conf. on Comm. (ICC 2018), May 2018, pp.1-8.

[87]. R. Mittal, V. T. Lam, N. Dukkipati, et.al. (2015), "TIMELY: RTT-based Congestion Control for the Datacenter," SIGCOMM Comput. Commun. Rev., 45(4), August 2015, pp. 537-550.

[88]. NS-3 Network Simulator, NS3.36, available: https://www.nsnam.org/.

[89]. M.Sharief, A. Oteafy, S. Hassanein (2018), "IoT in the Fog: A Roadmap for Data-Centric IoT Development", IEEE Comm. Magazine, Mar. 2018, pp. 157-163.

[90]. J. Padhye, J. Hurose, D. Towsley, R. Koodi (1998), "A Model Based TCP-Friendly Rate Control Protocol", Proc. Intl. Wrkshp. on Network and Operating System Support for Dig. Audio and Video (NOSSDAV), pp.1-20.

[91]. J.H. Park, J-H. Kim, S-K. Lee (2015), "A Study on the Enhanced Congestion Control Mechanism for Multimedia Traffic in Sensor Networks", Intl Journal of Multimedia and Ubiquitous Engineering, 10(8), pp.391-400.

[92]. S. Quincozes, T. Emilio, J. Kazienko (2019), "Mqtt protocol: Fundamentals, tools and future directions", IEEE Latin America Transactions, 17(09) 2019, pp. 1439–1448.

[93]. W.U. Rahman, Y.S. Choi, K. Chung (2019), "Performance Evaluation of Video Streaming Application Over CoAP in IoT," IEEE Access, Vol. 9, Apr. 2019, pp.39852-39861.

[94]. K. Ramakrishnan, S. Floyd (1999), "A Proposal to add Explicit Congestion Notification (ECN) to IP," RFC 2481, Available: https://tools.ietf.org/html/ rfc2481.

[95]. V. Rathod, N. Jeppu, S. Sastry, et.al (2019), "CoCoA++: Delay gradient based congestion control for Internet of Things," Futur. Gener. Comput. Syst., Vol. 100, Nov. 2019, pp. 1053–1072.

[96]. Y.N. Reddy, P.. Srinivas (2018), "A Combined TCP-friendly Rate control with WFQ Approach for Congestion Control for MANET", Intl. Journal Computer

Network and Information Security, Vol.6, 2018, pp.52-59.
[97]. R. Rejaie, M. Handley, D. Estrin (1999), “RAP: An end-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet,” in Proc. of IEEE INFOCOM '99, Vol.3, 1999, pp. 1337-1345.
[98]. A. Rezaee, F. Pasandideh (2018), “A Fuzzy Congestion Control Protocol Based on Active Queue Management in Wireless Sensor Networks with Medical Applications,” Wireless Personal Comm., 2018, pp. 815–842.
[99]. RFC draft -irtf-icnrg-icniot-03 (2019), “Design considerations for applying ICN to IoT”, IETF ICN RG, 05/2019.
[100]. RFC 7252 , “The Constrained Application Protocol (CoAP)”, IETF, available: https://tools.ietf.org/html/rfc7252
[101]. RFC draft, Workgroup CoRE (2023), “Everything over CoAP”, ITEF RFC draft, 13/3/2023, available: https://datatracker.ietf.org/doc/draft-amsuess-core-coap-kitchensink/
[102]. RFC 7297, “Information-Centric Networking (ICN) Research Challenges,” IETF, available: https:// tools.ietf.org/html/rfc7297
[103]. RFC draft, Workgroup CoRE (2019), “Adaptive RESTful Real-time Live Streaming for Things (A-REaLiST)”, IETF, available:
[104]. RFC 6298, “Computing TCP's Retransmission Timer,” IETF, available: https://rfc-editor.org/info/rfc6298.
[105]. RFC 793, “Transmission Control Protocol”, IETF, https://tools.ietf.org/html/rfc793
[106]. RFC 8312, “CUBIC for Fast Long-Distance Networks” IETF available: https://rfc-editor.org/info/rfc8312.
[107]. F. Righetti, et al. "Investigating the CoAP Congestion Control Strategies for 6TiSCH-Based IoT Networks," in IEEE Access 11 (2023), pp. 11054-11065.
[108]. T.J. Ross (2010), “Fuzzy Logic with Engineering Applications,” *Wiley Publisher, 3rd Edition*, ISBN-10:047074376X, 2010
[109]. T. Saedi, H. El-Ocla (2021), “TCP CERL+: revisiting TCP congestion control in wireless networks with random loss,” Wireless Networks, Vol. 27, 2021, pp. 423–440.
[110]. P. Sethi, S.R. Sarangi (2017), “Internet of Things: Architectures, Protocols, and Applications”, Hindawi Journal of Electrical and Computer Engineering, Vol 2017, Article ID 9324035, 25 pages.
[111]. Z. Shelby, S. Chakrabarti, E. Nordmark, C. Bormann (2012), “Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs),” IETF RFC 6775, https://tools.ietf.org/html/rfc-6775.
[112]. W. Shang, Y. Yu, R. Droms, L. Zhang (2016), “Challenges in IoT Networking via TCP/IP Architecture”, Intl. J. of Science & Research, UCLA TR NDN-0038, 2016, available: http://named-dât-net/techreports.html, pp.1-7.
[113]. C. Sonmez, O.D. Incel, et.al. (2014), “Fuzzy-based congestion control for wireless multimedia sensor networks,” in EURASIP Journal on Wireless Communications and Networking, No.63, pp. 1-17.
[114]. R. Sukjaimuk, Q.N. Nguyen, T. Sato (2018), “A Smart Congestion Control Mechanism for the Green IoT Sensor-Enabled Information-Centric Networking”, Sensors, 18(9) 2889, 2018, pp. 1-19.
[115]. C. Suwannapong, C. Khunboa (2019), “Congestion Control in CoAP Observe Group Communication,” Sensors, 19 (15), 3433, 2019, pp. 1-14.
[116]. T. Szigeti, J. Henry, F. Baker (2018), “Mapping Diffserv to IEEE 802.11,” IETF

RFC 8325, available: https://tools.ietf.org/html/rfc8325.
[117]. M.A. Tariq, M. Khan, M.T. Khan, D. Kim (2020), "Enhancements and Challenges in CoAP–A Survey," Sensors, Vol. 20 (2020), 6391, pp. 1-29.
[118]. P. Thubert (2020), "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Selective Fragment Recovery," IETF RFC 8931, available: https://tools.ietf.org/html/rfc8931.
[119]. Y. Tobe, Y. Tamura, A. Molano, S. Ghosh, H. Tokuda (2000), "Achieving moderate fairness for UDP flows by path-status classification," in Proc. 25th Annu. IEEE Conf. Local Computer Networks (LCN 2000), Tampa, FL, Nov. 2000, pp. 252–261.
[120]. X. Vilajosana, K. Pister, T. Watteyne (2017), "Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration," IETF RFC 8180, available: https://tools.ietf.org/html/rfc8180.
[121]. G. White, A. Palade, et.al (2018), "IoTPredict: Collaborative QoS Prediction in IoT", IEEE Intl. Conf. on Pervasive Computing and Comm. (PerCom), Athens, Greece, 2018, pp. 1-10.
[122]. A. Yang, X. Yang, J. Chang, et.al (2018), "Research on a Fusion Scheme of Cellular Network and Wireless Sensor for Cyber Physical Social Systems", IEEE Access, Vol 6, 2018. pp. 18786-18794.
[123]. L.A. Zadeh (1969), "Toward a theory of fuzzy systems," NASA Report No. CR-1432, 1969, pp. 1–35.
[124]. L.A. Zadeh (1973), "Outline of a new approach to the analysis of complex systems and decision processes," IEEE Transactions on Systems, Man, and Cybernetics 3(1), pp. 28–44.
[125]. M. Zarei, A. M. Rahmani, R. Farazkish (2011), "CCTF: congestion control protocol based on trustworthiness of nodes in wireless sensor networks using fuzzy logic," Int. J. Ad Hoc Ubiquitous Comput., Vol.8, 2011, pp. 54–63.
[126]. H.J. Zimmerman (2001), "Fuzzy set theory - and its applications," Kluwer Academic Publishers, Springer Science, 4th Edition, 2001.

PHỤ LỤC

A1. Phụ lục 1. Mô phỏng RCoAP và FCoAP trên NS-3

NS-3 là bộ công cụ mô phỏng mạng theo sự kiện rời rạc, hỗ trợ xây dựng và phát triển các giao thức truyền tin. Phiên bản sử dụng trong luận án là NS-3.36 (xem NS-3 Network Simulator, có trên https://www.nsnam.org/) được cài đặt trên hệ điều hành Ubuntu phiên bản 22.04. Toàn bộ code của RCoAP, FCoAP được đặt trong thư mục ns3/src/applications/. Các kịch bản mô phỏng được đặt trong thư mục ns3/scratch/.

Danh mục các tệp chương trình RCoAP và FCoAP gồm:

- RCoAP:
  - Các tệp RCoAP.cc và RCoAP.h trong thư mục ns3/src/applications/model
  - Các tệp RCoAP-helper.cc và RCoAP-helper.h dùng để hỗ trợ thiết lập liên kết cấu hình trong NS-3 đặt trong thư mục ns3/src/applications /helper.
- FCoAP:
  - Các tệp FCoAP.cc và FCoAP.h trong thư mục ns3/src/applications/model
  - Các tệp FCoAP-helper.cc và FCoAP-helper.h dùng để hỗ trợ thiết lập liên kết cấu hình trong NS-3 đặt trong thư mục ns3/src/applications /helper.

Ngoài ra, trong thư mục ns3/src/applications/model có thêm các tệp chương trình CoAP.cc, CoAP.h, CoCoA.cc, CoCoA.h, CoCoA+.cc, CoCoA+.h và các tệp helper.cc tương ứng để hỗ trợ phục vụ cho việc mô phỏng các giao thức CoAP, CoCoA và CoCoA+.

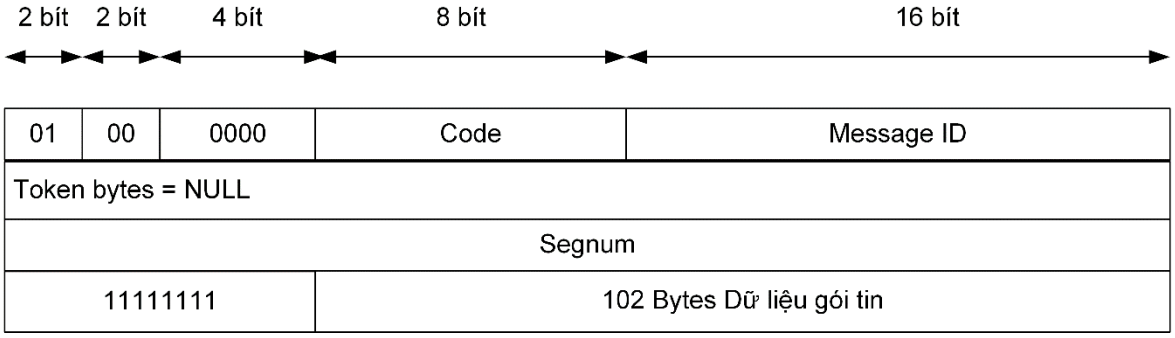
Thiết lập các tệp kịch bản kichban_x.cc cho các mô phỏng khác nhau trong thư mục ns3/scratch/, thực hiện dòng lệnh NS-3 sau để chạy từng kịch bản mô phỏng:

./ns3 run scratch/kichban_x.cc

A2. Phụ lục 2. Tiêu đề gói tin RCoAP

Hình P1 là cấu trúc gói tin RCoAP cho gói tin CON và ACK theo chuẩn RFC 7252 [100]: 2 bit đầu = 01 là phiên bản RCoAP, 2 bit tiếp theo = 00 nếu là gói CON, = 02 nếu là gói ACK theo chuẩn. 4 bit tiếp là chiều dài Token TKL (0000 → TKL = NULL). 8 bit Code là mã lệnh (0.01 = GET, 0.03 = PUT). MessageID là

định danh gói tin được CoAP tạo theo hàm ngẫu nhiên. Số Token bytes tùy theo chiều dài TKL, thử nghiệm không cần dùng Token (đặt TKL=NULL) nên Token Bytes = NULL. RCoAP ghi thêm 4 bytes Segnum vào phần tùy chọn tiêu đề Option 1 so với CoAP chuẩn [100] để ghi số thứ tự gói tin phục vụ cho việc kiểm tra nguyên nhân mất gói. Phần Option 1 là theo chuẩn RFC 7252 [100]. Mã 11111111 đánh dấu bắt đầu phần thân gói tin RCoAP, phần dữ liệu sử dụng 102 bytes trong mô phỏng.



Hình P1. Cấu trúc gói tin RCoAP cho gói tin CON và ACK

Đoạn mã thêm vào NS-3 để tạo tiêu đề gói tin RCoAP như sau:

```
// make RCoAP packet
RCoapPacket packet;
uint8_t seqnum;
seqnum = ++mSeqNO;
TsSqBuf []  = seqnum;    // ghi Seqno vao header
packet.options[1].buffer = (uint8_t *)TsSqBuf; // ghi sqn vao RCoAP Header
```

A3. Phụ lục 3. Các hàm chính của RCoAP

Phần mềm RCoAP được xây dựng trong NS-3 dựa trên một số mô đun phần mềm cơ bản của Maesor [82] và Take [50]. Các hàm chính của RCoAP như sau.

```
namespace ns3 {
class RCoap: public Application {
  virtual void RStartApplication(void);                 // hàm khởi tạo kết nối
  virtual void RStopApplication(void);                  // hàm kết thúc kết nối

// States
  void RInitState();              // hàm trạng thái khởi tạo
  void REndInitState();                   // hàm kết thúc khởi tạo, xác định tốc độ phát ban đầu
  void RSteady();              // hàm trạng thái ổn định: tăng giảm tốc độ phát
  void RDetected();            // hàm trạng thái phát hiện tắc nghẽn
```

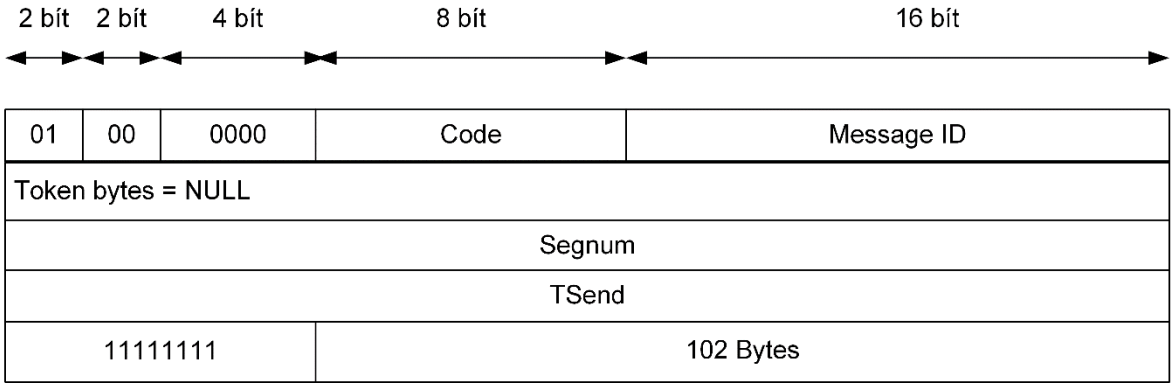
```
  void RBackoff();                // hàm trạng thái lùi để giảm tắc nghẽn
// Timer functions
  void RDataSendTimer();                      // hàm định thời phát gói theo nhịp
  void RTO_TimeoutR (uint32_t seqnom);    // hàm kích hoạt định thời RTO

// Send/Receive functions      // các hàm cho gửi gói và hàm sự kiện nhận gói
  void SendRCoap(RCoapPacket& packet, Ipv4Address ip, int port);
  uint16_t sendRCoapPacket(IpAddress ip, int port, char *data, COAP_TYPE type,
            COAP_METHOD method, uint8_t *token, uint8_t tokenlen, uint8_t
            *datapayload, uint32_t payloadlen);
  uint16_t SendCache_RCoAP(uint32_t seqnom);    // phát lại gói tin đã mất
  void ReceiveRCoAP_Packet(Ptr<Socket>socket);  // hàm sự kiện nhận gói

 // Cache for Retransmissions          // hàm phát lại gói tin đã lưu trong bộ nhớ
  bool RCoAP_Cache(uint32_t seqnom, uint32_t msgid, double mts, double ipg);
}
```

A4. Phụ lục 4. Tiêu đề gói tin FCoAP

Hình P2 là cấu trúc gói tin FCoAP cho gói tin CON và ACK theo chuẩn RFC 7252 [100]. Các bit trong tiêu đề gói tin tương tự như hình P1, ngoại trừ có thêm 4 bytes ở phần tùy chọn tiêu đề Option 2 để ghi thời gian phát gói tin CON phục vụ cho việc tính thông lượng tức thời và băng thông cổ chai của FCoAP.



Hình P2. Cấu trúc gói tin FCoAP cho gói tin CON và ACK

Đoạn mã thêm vào NS-3 để tạo tiêu đề gói tin FCoAP như sau:

```
// make FCoAP packet
FCoapPacket packet;
uint8_t seqnum;
uint8_t TSend;
seqnum = ++mSeqNO;
TsSqBuf1 [ ]  = seqnum; // ghi Seqno vào FCoAP header
TsSqBuf2 [ ]  = TSend;   // ghi thời gian phát vào FCoAP header
```

```
    packet.options[1].buffer = (uint8_t *)TsSqBuf; // ghi sqn vao FCoAP Header
    packet.options[2].buffer = (uint8_t *)TsSqBuf; // ghi TSend vao FCoAP Header
```

A5. Phụ lục 5. Các hàm chính của FCoAP

Phần mềm FCoAP được xây dựng dựa trên RCoAP trong NS-3 song có thay đổi một số hàm chính như sau.

```
namespace ns3 {
class FCoap: public Application {
  virtual void RStartApplication(void);                    // hàm khởi tạo kết nối
  virtual void RStopApplication(void);                     // hàm kết thúc kết nối

// States
  void F_InitState();              // hàm trạng thái khởi tạo
  void F_EndInitState();           // hàm kết thúc khởi tạo, xác định băng thông cổ chai và
                                   // tốc độ phát ban đầu
  void F_Steady();                 // hàm trạng thái ổn định: điều chỉnh tốc độ phát theo FCS
  void F_Backoff();                // hàm trạng thái lùi để giảm tắc nghẽn
  double FIS(double inp1, double inp2);        // hàm tính C_degree với hệ điều khiển mờ
  void CalMFXX(double xx);                     // Tính tập mờ đầu vào x
  void CalMFYY(double yy);                     // Tính tập mờ đầu vào y

// Timer functions
  void F_DataSendTimer();                      // hàm định thời phát gói theo nhịp
  void RTO_Timeout_F (uint32_t seqnom);  // hàm kích hoạt định thời RTO

// Send/Receive functions      // các hàm cho gửi gói và hàm sự kiện nhận gói
  void Send_FCoap(FCoapPacket& packet, Ipv4Address ip, int port);
  uint16_t send_FCoap_Packet(IpAddress ip, int port, char *data, COAP_TYPE type,
              COAP_METHOD method, uint8_t *token, uint8_t tokenlen, uint8_t
              *datapayload, uint32_t payloadlen);
  uint16_t SendCache_FCoAP(uint32_t seqnom);              // phát lại gói tin đã mất
  void ReceiveRCoAP_Packet(Ptr<Socket>socket);            // hàm sự kiện nhận gói

 // Cache for Retransmissions           // hàm phát lại gói tin đã lưu trong bộ nhớ
  bool FCoAP_Cache(uint32_t seqnom, uint32_t msgid, double mts, double ipg);
}
```