HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

NGUYỄN QUỐC HỮU

NGHIÊN CỨU XÂY DỰNG GIẢI PHÁP PHÂN TÁN ĐỘNG TRONG HỆ THỐNG PHÂN TÍCH MÃ ĐỘC IOT BOTNET DỰA TRÊN KAFKA VÀ KSQL

Chuyên ngành: HỆ THỐNG THÔNG TIN
Mã số: 8.48.01.04

TÓM TẮT LUẬN VĂN THẠC SĨ

HÀ NỘI - NĂM 2022

Luận văn được hoàn thành tại:

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Người hướng dẫn khoa học: TS. NGÔ QUỐC DŨNG

(Ghi rõ học hàm, học vị)

Phản biện 1: TS. Vũ Văn Thỏa

Phản biện 2: PGS.TS. Nguyễn Linh Giang

Luận văn sẽ được bảo vệ trước Hội đồng chấm luận văn thạc sĩ tại Học viện Công nghệ Bưu chính Viễn thông

Vào lúc: 16 giờ 15 ngày 05 tháng 7 năm 2022

Có thể tìm hiểu luận văn tại:

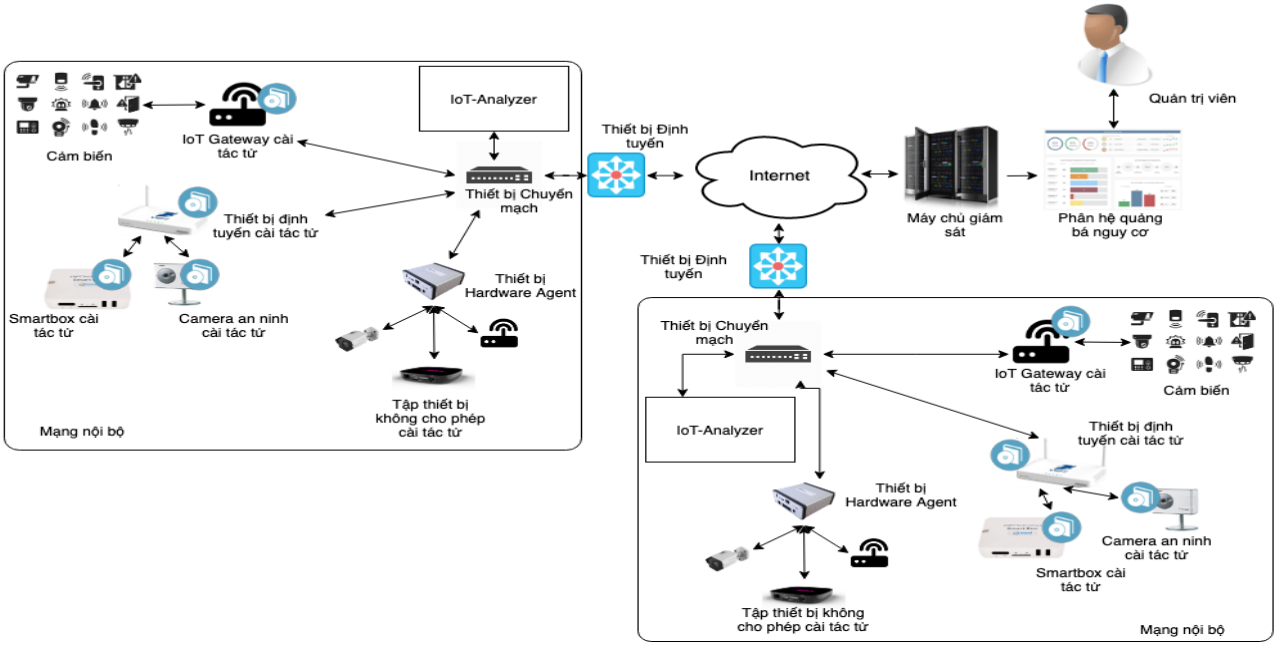
- Thư viện của Học viện Công nghệ Bưu chính Viễn thông.

MỞ ĐẦU

1. Lý do chọn đề tài.

Cách mạng công nghiệp 4.0 mang đến rất nhiều những thay đổi nhanh chóng về công nghiệp trên toàn thế giới, đặc biệt là mở ra các công nghệ của tương lai. Một trong số các công nghệ được phát triển rất mạnh trong cách mạng công nghiệp 4.0 là Internet vạn vật (Internet of thing – IoT). Các thiết bị IoT có đặc điểm là hạn chế về tài nguyên, năng lực xử lý thấp và bộ nhớ nhỏ. Sự thiếu hụt các cơ chế và tiêu chuẩn bảo mật dẫn đến rất nhiều lỗ hổng trên thiết bị IoT được khai thác. Một trong số các kịch bản tấn công phổ biến là kẻ tấn công sẽ sử dụng các thiết bị IoT kiểm soát được trở thành một phần của một IoT botnet. Do đó, việc đảm bảo an toàn thông tin cho các thiết bị IoT đã và đang là một chủ đề được nghiên cứu rất nhiều trên thế giới.

Hệ thống phát hiện và cảnh báo mã độc IoT Botnet đã và đang được phát triển bởi các nhà nghiên cứu cũng như các hãng sản xuất phần mềm/phần cứng trên thế giới. Trong đó TS. Ngô Quốc Dũng đã đưa ra một giải pháp hệ thống gồm tiền xử lý, xử lý, phát hiện cảnh báo và ngăn chặn các cuộc tấn công mạng nhằm vào các thiết bi IoT cỡ nhỏ phù hợp với các thiết bị IoT phổ biến tại Việt Nam.



Mô hình tổng quan hệ thống tự động phát hiện, cảnh báo và ngăn chặn tấn công mạng nhằm vào các thiết bị IoT cỡ nhỏ sử dụng mạng lưới tác tử thông minh.

Bài toán đặt ra khi triển khai hệ thống là tại một mạng nội bộ, nếu một IoT-Analyzer bị quá tải thì mạng nội bộ nó chịu trách nhiệm sẽ có nguy cơ mất an toàn bảo mật. Do đó cần có cơ chế xử lý khi một IoT-Analyzer quá tải và đưa ra hướng giải quyết dựa vào các IoT-Analyzer khác không bị quá tải. Bằng hướng nghiên cứu này, học viên đã chọn đề tài "***Nghiên cứu xây dựng giải pháp phân tán động trong hệ thống phân tích mã động IoT Botnet dựa trên KAFKA và KSQL***" nhằm cải thiện nhược điểm của hệ thống.

2. Tổng quan về đề tài nghiên cứu.

Hiện nay, có rất nhiều các nghiên cứu và khảo sát về hệ thống xử lý phân tán nhằm tang hiệu quả của việc xử lý dữ liệu. HARUNA ISAH và cộng sự [1] đã trình bày nghiên cứu so sánh về các khung phân tích và xử lý luồng dữ liệu phân tán và đánh giá quan trọng về các khung xử lý luồng dữ liệu phân tán mã nguồn mở đại diện (Storm, Spark Streaming, Flink, Kafka Streams) và mã nguồn thương mại (IBM Streams). Supun Kamburugamuve và Geoffrey Fox [2] đã đưa ra đánh giá một hệ thống xử lý luồng trực tuyến trên hai khía cạnh độc lập. Khía cạnh thứ nhất, có một API lập trình để phát triển các ứng dụng xử lý luồng trực tuyến và khía cạnh còn lại là có một công cụ thực thi thực thi ứng dụng xử lý luồng trực tuyến.

Các chương trình mã nguồn mở phổ biến để xử lý luồng dữ liệu trên thế giới như Apache Spark, Apache Flink, Apache Kafka. Apache Kafka là một nền tảng xử lý luồng trực tuyến phân tán. Kafka hỗ trợ cho nhiều kiểu xử lý luồng, giúp giải quyết các vấn đề khó khăn mà ứng dụng gặp phải: xử lý dữ liệu không theo thứ tự, xử lý lại đầu vào khi thay đổi mã, thực hiện tính toán trạng thái, v.v. KSQL là công cụ xử lý luồng cho phép xử lý dữ liệu thời gian thực dựa trên Apache Kafka.

3. Mục đích nghiên cứu.

Nghiên cứu và xây dựng giải pháp phân tán động dựa trên nền tảng các công nghệ mới, cụ thể là Kafka và KSQL nhằm tối ưu hoá khả năng phân tích và phát hiện mã độc IoT Botnet.

4. Đối tượng và phạm vi nghiên cứu.

- Đối tượng nghiên cứu: Hệ thống xử lý phân tán động trên nền tảng công nghệ Kafka, KSQL…

- Phạm vi nghiên cứu: Hệ thống phân tích mã độc IoT Botnet.

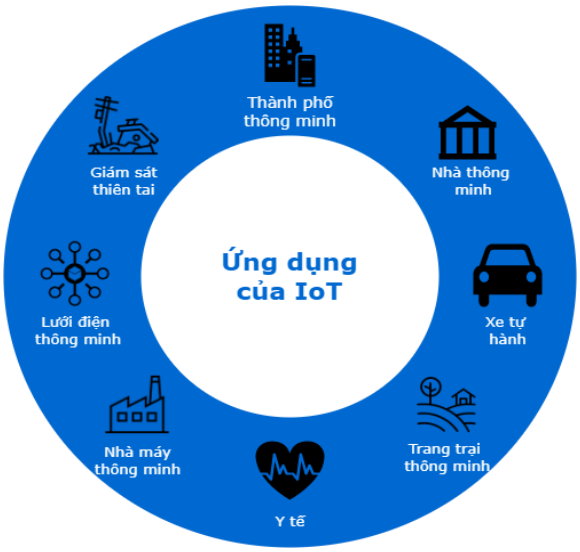
5. Phương pháp nghiên cứu.

- Phương pháp nghiên cứu lý thuyết: Phân tích và tổng hợp các tài liệu liên quan đến đề tài. Phân loại và hệ thống hoá lý thuyết các kết quả nghiên cứu hiện tại như bài báo, kết quả nghiên cứu khoa học về phân tán động, phân tích và phát hiện mã độc IoT botnet.

- Phương pháp nghiên cứu thực tiễn: Sử dụng phương pháp thực nghiệm khoa học để xây dựng, đánh giá giải pháp phân tán động.

CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

Trong sự phát triển mạnh mẽ của nền công nghiệp 4.0, internet vạn vật là một lĩnh vực đã và đang được nghiên cứu và ứng dụng rộng rãi. Trong những năm gần đây, sự bùng nổ về số lượng của thiết bị IoT giúp con người rất nhiều trong đời sống cũng như nhiều ngành công nghiệp. Tuy nhiên bên cạnh ưu điểm vượt trội về ứng dụng, các thiết bị IoT cũng có các đặc điểm liên quan đến bảo mật mà hacker có thể khai thác. Với số lượng thiết bị IoT tăng trưởng từng ngày, việc nghiên cứu các biện pháp bảo mật cho thiết bị IoT đang rất được quan tâm hiện nay. Ngoài việc xây dựng hệ thống phân tích và phát hiện mã độc, tối ưu hiệu năng hệ thống để có thể phát hiện sớm mã độc cũng là một công việc rất quan trọng. Trong chương này, luận văn sẽ trình bày tổng quan về mã độc IoT botnet và mô hình hệ thống phân tích, phát hiện mã độc IoT botnet.



Hình 1.1: Ứng dụng của IoT trong thực tế

1.1 Tổng quan mã độc IoT Botnet

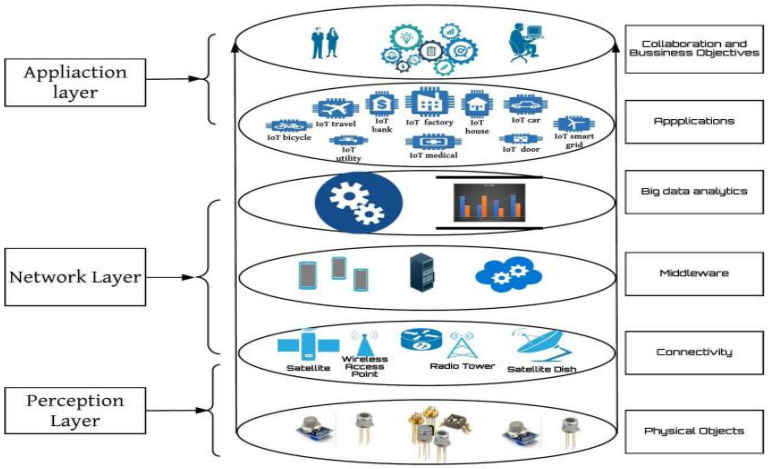
1.1.1 Các khái niệm trong IoT

Hiện nay khái niệm về IoT có rất nhiều cách giải thích khác nhau được các tổ chức đưa ra, một trong các tổ chức uy tín trên thế giới là Liên minh Viễn thông thế giới (ITU – International Telecommunication Union) [2] đã đưa ra một khái niệm tương đối hoàn chỉnh về IoT. Khái niệm IoT theo ITU được phát biểu như sau: "*Internet of things là một cơ sở hạ tầng mang tính toàn cầu cho xã hội thông tin, mang đến những dịch vụ tiên tiến bằng cách kết nối các "đồ vật" (cả vật lý lẫn ảo) dựa trên sự tồn tại của thông tin, dựa trên khả năng tương tác của các thông tin đó và dựa trên các công nghệ truyền thông. Thông qua việc khai thác khả năng nhận biết, thu thập xử lý dữ liệu, công nghệ các hệ thống IoT tận dụng mọi thứ để cung cấp dịch vụ cho tất cả các loại ứng dụng khác nhau, đồng thời bảo đảm tính bảo mật và quyền riêng tư*". Trong khuôn khổ luận văn và phù hợp với nội dung nghiên cứu, học viên đưa ra khái niệm thiết bị IoT như sau:

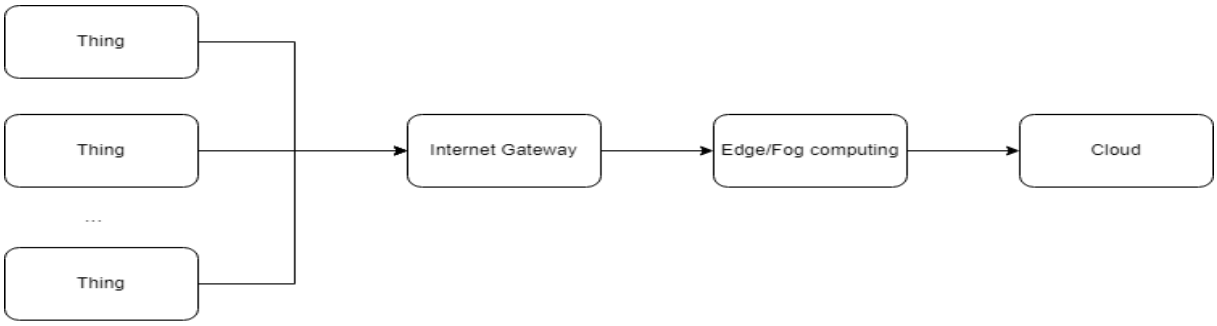
Định nghĩa 1: Thiết bị IoT là những thực thể vật lý hoặc ảo hoá đa kiến trúc, có các đặc điểm cụ thể như tài nguyên hạn chế, nhỏ gọn để phục vụ mục các đích khác nhau, có khả năng xử lý dữ liệu, kết nối và chia sẻ thông tin với các thực thể khác.

a) Kiến trúc của IoT

Kể từ khi những nghiên cứu đầu tiên về IoT được thực hiện, kiến trúc ba lớp đã là mô hình tổng quan cho các ứng dụng IoT. Lớp nhận thức: Bản thân các cảm biến nằm trên lớp này. Dữ liệu có thể được thu thập từ bất kỳ số lượng cảm biến nào trên thiết bị được kết nối. Lớp mạng: Lớp mạng mô tả lượng lớn dữ liệu đang di chuyển trong ứng dụng. Lớp này kết nối các thiết bị khác nhau và gửi dữ liệu đến các dịch vụ back-end thích hợp.Lớp ứng dụng: Lớp ứng dụng là những gì người dùng nhìn thấy. Đây có thể là một ứng dụng để điều khiển một thiết bị trong hệ sinh thái nhà thông minh hoặc một bảng điều khiển hiển thị trạng thái của các thiết bị là một phần của hệ thống.



Hình 1.2: Các lớp trong kiến trúc IoT



Hình 1.3: Kiến trúc của hệ thống IoT theo bốn giai đoạn

Kiến trúc Internet of Things được thể hiện trong hình 1.3 sử dụng cách tiếp cận bốn giai đoạn: (1) Thiết bị (Device/Thing): Các thiết bị này có thể là cảm biến hoặc thiết bị truyền động trong lớp nhận thức (Perceptron). (2) Cổng Internet (Internet Gateway): Cổng Internet sẽ nhận dữ liệu thô từ thiết bị và xử lý trước khi gửi lên đám mây. (3) Điện toán biên hoặc sương mù (Edge/Fog computing): Để xử lý dữ liệu nhanh nhất có thể mà không cần đến các thực thể trong điện toán đám mây. (4) Đám mây hoặc trung tâm dữ liệu (Cloud or datacenter): Các lớp ứng dụng và nghiệp vụ nằm trong giai đoạn này, nơi các phần mềm quản lý có thể được cung cấp thông qua dữ liệu được lưu trữ trên đám mây.

b) Đặc điểm của thiết bị IoT

Thiết bị IoT có các ứng dụng khác nhau, nên chúng cũng sẽ có rất nhiều đặc điểm khác nhau. Tuy nhiên các thiết bị IoT có một số đặc điểm chung được trình bày trong phần này. Khác với những thiết bị trên Internet thông thường, thiết bị IoT có các đặc điểm như sau:

- Tính không đồng nhất

- Tài nguyên hạn chế và thiết bị nhỏ gọn
- Thay đổi động
- Quy mô mở rộng lớn
- Tính kết nối liên thông

1.1.2 Mã độc IoT Botnet

a) Khái niệm mã độc

Trước hết, thuật ngữ mã độc nói chung trong khoa học máy tính được Cisco [5] định nghĩa như sau: *"Phần mềm độc hại (malware), viết tắt của "malicious software", đề cập đến bất kỳ phần mềm xâm nhập nào được phát triển bởi tội phạm mạng (thường được gọi là "tin tặc") để đánh cắp dữ liệu và làm hỏng hoặc phá hủy máy tính và hệ thống máy tính. Ví dụ về phần mềm độc hại phổ biến bao gồm vi rút, sâu, vi rút Trojan, phần mềm gián điệp, phần mềm quảng cáo và ransomware"*.

Botnet có thể được định nghĩa là một tập hợp các thiết bị bị xâm nhập được gọi là bot chạy mã độc và được kiểm soát bởi quản trị viên được gọi là botmaster. Thông thường, một mạng botnet bao gồm ba thành phần chính: kẻ tấn công; cơ sở hạ tầng độc hại; các bot.

b) Mã độc IoT botnet

Mã độc IoT botnet là loại mã độc được tin tặc phát triển trên nền tảng IoT. TrendMicro [6] đưa ra khái niệm IoT botnet *"Mạng botnet IoT là một mạng lưới các thiết bị được kết nối với Internet vạn vật (IoT), điển hình là các bộ định tuyến, đã bị nhiễm phần mềm độc hại (cụ thể là phần mềm độc hại IoT botnet) và rơi vào tầm kiểm soát của các tác nhân độc hại. Các mạng botnet IoT được biết đến với việc được sử dụng để phát động các cuộc tấn công từ chối dịch vụ (DDoS) phân tán vào các thực thể mục tiêu để làm gián đoạn hoạt động và dịch vụ của họ"*.

Có nhiều nghiên cứu về IoT botnet [9]–[11] chỉ ra rằng IoT botnet hoạt động qua ba giai đoạn chính như sau:

Giai đoạn 1: Giai đoạn quét:

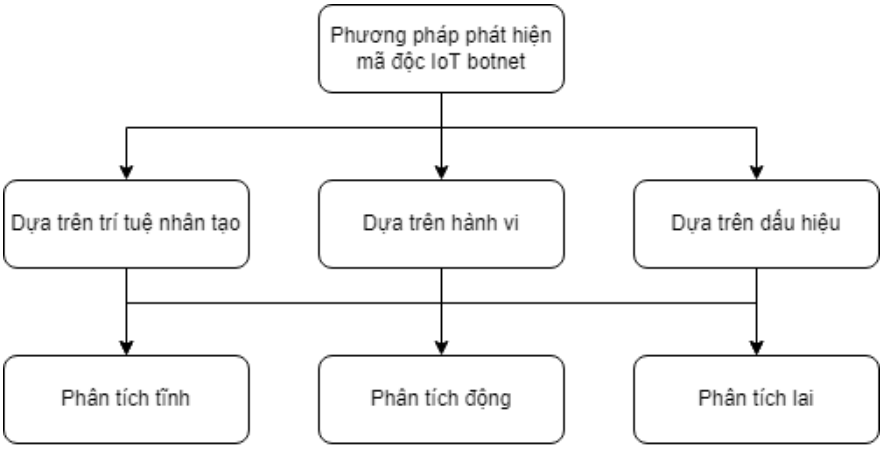
Giai đoạn 2: Giai đoạn lan truyền:

Giai đoạn 3: Giai đoạn tấn công:

1.2 Tổng quan hệ thống phân tích và phát hiện mã độc IoT Botnet

1.2.1 Các phương pháp phân tích và phát hiện mã độc IoT Botnet

Hiện nay, các phương pháp phát hiện mã độc được nghiên cứu và áp dụng trên thế giới được chia thành ba loại chính như sau: (1) các phương pháp phát hiện dựa trên trí tuệ nhân tạo (AI-based), (2) các phương pháp phát hiện dựa trên hành vi (behavior-based) và (3) các phương pháp phát hiện dựa trên dấu hiệu (signature-based).



Hình 1.3: Các phương pháp phát hiện mã độc IoT botnet

Quá trình mổ xẻ mã độc để hiểu cách thức hoạt động, xác định chức năng, nguồn gốc và tác động tiềm ẩn của nó được gọi là phân tích mã độc. Có hai cách tiếp cận cơ bản để phân tích mã độc IoT botnet: (1) phân tích tĩnh, (2) phân tích động và (3) phân tích lai.

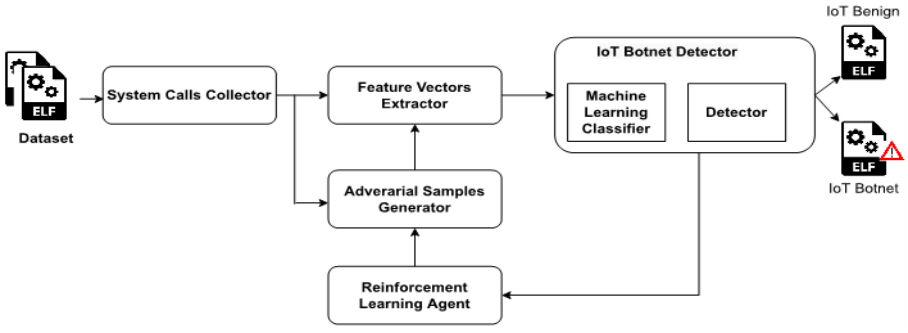
1.2.2 Học máy trong phát hiện mã độc IoT Botnet

Trí tuệ nhân tạo đã và đang được nghiên cứu ứng dụng rộng rãi trong các lĩnh vực, trong đó có bảo mật trong IoT. Các phương pháp học máy được áp dụng để phát hiện các cuộc tấn công sớm dựa trên các dữ liệu đặc trưng của hệ thống hoặc tập tin. Các thuật toán học máy được chia làm 3 loại, gồm:

- Học có giám sát (Supervised learning)
- Học không giám sát (Unsupervised learning)
- Học tăng cường (RL)

1.2.3 Mô hình hệ thống phân tích và phát hiện mã độc IoT Botnet

Ở khuôn khổ luận văn này, học viên trình bày mô hình hệ thống phân tích và phát hiện mã độc do học viên và nhóm của T.S Ngô Quốc Dũng đã nghiên cứu bao gồm chi tiết về giai đoạn tiền xử lý dữ liệu, cơ chế phân loại dựa trên vectơ tính năng và mô hình học tập củng cố để cải thiện phát hiện tấn công zero-day. Mô hình tổng quan hệ thống được minh hoạ tại hình 1.7.



Hình 1.4. Mô hình hệ thống phân tích và phát hiện mã độc IoT botnet

Mô hình được đề xuất bao gồm năm thành phần chính: Bộ thu thập lệnh gọi hệ thống (SCC – System Calls Collector), Bộ trích xuất vectơ tính năng (FVE – Feature Vectors Extractor), Bộ phát hiện Botnet IoT (IBD – IoT Botnet Dectector), Bộ tạo mẫu đối nghịch (ASG – Adversarial Samples Generator), Tác nhân học tăng cường (RLA – Reinforcement learning). Hoạt động của mô hình được chia thành hai giai đoạn: (1) Giai đoạn đầu tiên là giai

đoạn phân loại học máy mà IBD sẽ được đào tạo bằng cách sử dụng các bộ phân loại học máy khác nhau. (2) Giai đoạn thứ hai là cải thiện phân loại,

1.3. Tổng quan giải pháp xử lý phân tán động

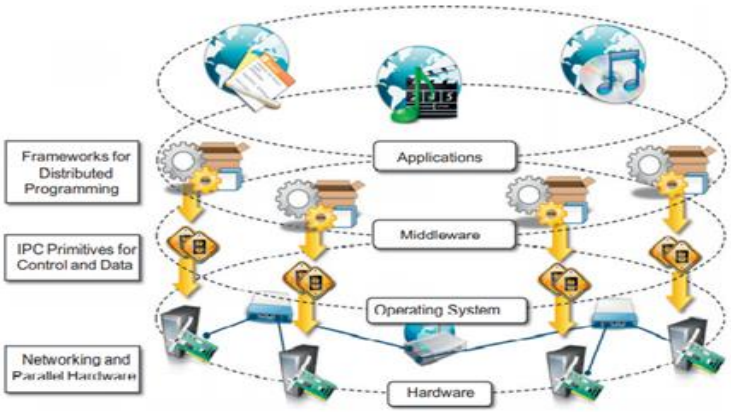
1.3.1. Các định nghĩa về hệ thống phân tán.

Hệ thống phân tán hiện nay được định nghĩa theo nhiều cách, trong đó có một số định nghĩa tương đối tổng quát như của tác giả George Coulouris và cộng sự [17] đã định nghĩa: *"hệ thống phân tán là một hệ thống trong đó các thành phần phần cứng hoặc phần mềm được kết nối mạng được giao tiếp và điều phối tác vụ qua cách truyền các message."*. Tác giả Andrew S. Tanenbaum và cộng sự [18] đã đưa ra định nghĩa về hệ thống phân tán như sau: *"Một hệ thống phân tán là một tập hợp các máy tính độc lập nhưng ở phía người dùng của nhìn nhận như một hệ thống mạch lạc duy nhất."*. Qua các định nghĩa khác nhau về hệ thống phân tán, trong khuôn khổ luận văn, học viên định nghĩa hệ thống phân tán như sau:

Định nghĩa 3: Hệ thống phân tán là hệ thống mà các máy tính được kết nối với nhau, trong đó các tác vụ được điều phối xử lý phân tán giữa các máy và ở phía người dùng thì nhìn nhận như một hệ thống đồng nhất.

1.3.2 Các thành phần trong một hệ thống phân tán

Trong hệ thống phân tán, việc chia sẻ bộ nhớ và cơ chế liên lạc giữa các thành phần trong hệ thống là hai yếu tố quan trọng nhất. Căn cứ theo một số định nghĩa, khái niệm và các đặc điểm của hệ thống phân tán thì cơ chế liên lạc giữa các cấu phần (cả phần cứng và phần mềm) trong hệ thống phân tán là yếu tố quan trọng nhất. Nếu coi mức trừu tượng thấp nhất trong hệ thống phần tán là mức tiến trình (process) thì cơ chế liên lạc giữa các cấu phần của hệ thống phân tán là IPC – Inter Process Communication Hình 1.10 minh hoạ mô hình của một hệ thống phân tán.



Hình 1.5 Mô hình phân lớp của hệ thống phân tán.

Hai mô hình chính thuộc cơ chế liên lạc IPC là mô hình chia sẻ bộ nhớ (shared memory) – với nhiệm vụ định nghĩa, cấp phát và khởi tạo khu vực bộ nhớ lưu trữ chung – và mô hình truyền nhận thông điệp (message passing) – với nhiệm vụ truyền tải thông điệp giữa các tiến trình.

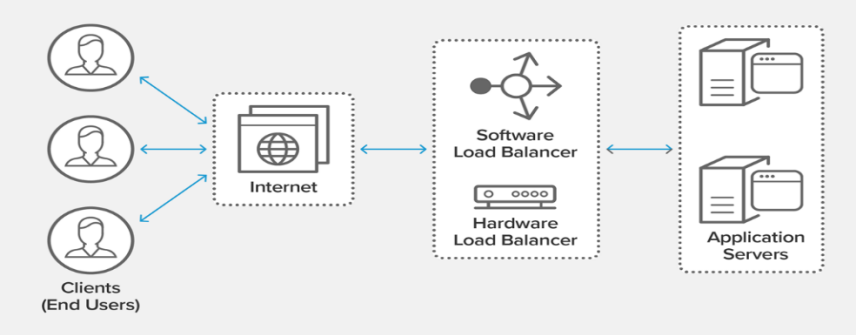
1.3.3. Tổng quan về lý thuyết cân bằng tải.

Load balancing (cân bằng tải) là một phương pháp phân phối khối lượng tải trên nhiều máy tính hoặc một cụm máy tính để có thể sử dụng tối ưu các nguồn lực, tối đa hóa thông lượng, giảm thời gian đáp ứng và tránh tình trạng quá tải trên máy chủ. Lợi ích của cân bằng

tải như sau: Tăng độ tin cậy và khả năng dự phòng cho hệ thốn, Tăng tính bảo mật cho hệ thống, Khả năng mở rộng hệ thống.

a) Cân bằng tải phần cứng và cân bằng tải phần mềm.

Bộ cân bằng tải phần cứng là một thiết bị phần cứng có hệ điều hành chuyên biệt phân phối lưu lượng ứng dụng web trên một cụm máy chủ ứng dụng. Để đảm bảo hiệu suất tối ưu, bộ cân bằng tải phần cứng phân phối lưu lượng truy cập theo các quy tắc tùy chỉnh để các máy chủ ứng dụng không bị quá tải.



Hình 1.6 Bộ cân bằng tải phần cứng và phần mềm.

Cân bằng tải phần mềm Cân bằng tải phần mềm cung cấp chức năng tương tự của cân bằng tải phần cứng, nhưng nó không yêu cầu thiết bị cân bằng tải chuyên dụng. Phần mềm cân bằng tải có thể chạy trên máy chủ thông thường hoặc thậm chí là máy chủ ảo.

b) Cân bằng tải tĩnh: Thuật toán cân bằng tải là "tĩnh" khi nó không tính đến trạng thái của hệ thống để phân phối các tác vụ. Qua đó, trạng thái hệ thống bao gồm các thước đo như mức tải (và đôi khi là quá tải) của một số bộ xử lý nhất định.

c) Cân bằng tải động.: Các thuật toán cân bằng tải động tính đến tải trọng hiện tại của từng đơn vị tính toán (còn gọi là các nút) trong hệ thống.

Thuật ngữ “phân tán động” sử dụng trong luận văn được hiểu như là cách thức phân tán xử lý các tác vụ trong một hệ thống một cách động nhằm tăng tốc độ xử lý và tránh việc tắc nghẽn dữ liệu, quá tải hệ thống.

Kết luận Chương 1

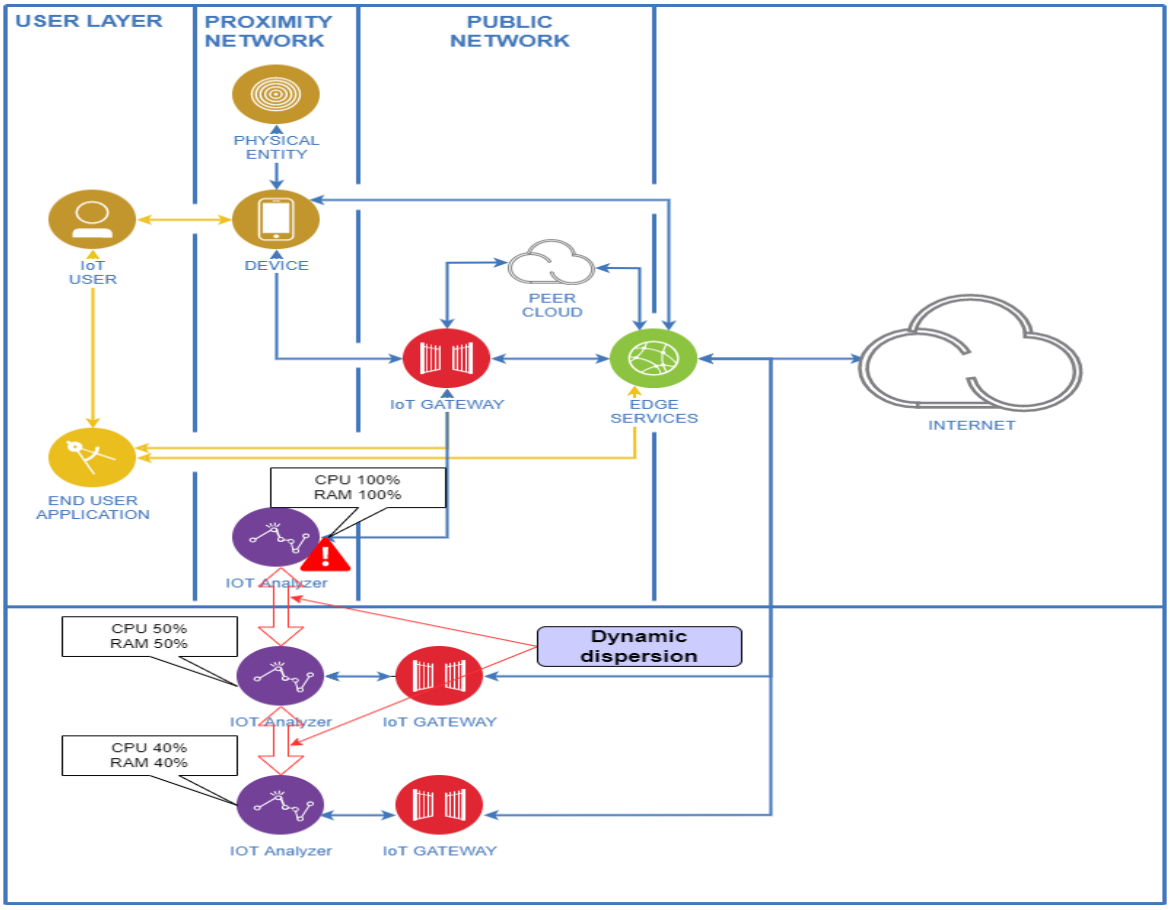
Trong khuôn khổ của luận văn, học viên tập trung nghiên cứu và áp dụng hướng cân bằng tải động để áp dụng cho xử lý phân tán trong IoT Analyzer nhằm nâng cao hiệu quả việc phân tích và phát hiện mã độc IoT botnet.

CHƯƠNG 2. ỨNG DỤNG GIẢI PHÁP PHÂN TÁN ĐỘNG TRONG HỆ THỐNG PHÂN TÍCH MÃ ĐỘC IOT BOTNET.

Trong chương 2, học viên sẽ trình bày về bài toán phân tán động trong hệ thống phân tích mã động IoT botnet nhằm tăng cường hiệu năng xử lý của toàn hệ thống. Từ những kiến thức quan trọng về cân bằng xử lý, học viên đưa ra phân tích, thiết kế và xây dựng mô hình xử lý bài toán sử dụng Apache Kafka và KSQL.

2.1. Phát biểu bài toán

Tại môi trường thực tế, số lượng thiết bị IoT là rất lớn và việc phân tích khối lượng dữ liệu cho toàn bộ các thiết bị IoT đòi hỏi một hệ thống được tối ưu tốt và có khả năng cân bằng năng lực xử lý. Bài toán học viên đặt ra là làm thế nào để IoT Analyzer bị quá tải có thể phân tán các tác vụ cho một thiết bị IoT Analyzer khác với mục đích làm cho hệ thống không bị tắc nghẽn và gián đoạn.



Hình 2.1. Bài toán phân tán xử lý tác vụ trong kịch bản một thiết bị IoT Analyzer quá tải.

2.1.1 Phân tích bài toán.

Trong phần phân tích bài toán, học viên đưa ra một số nhiệm vụ cần phải giải quyết trong bài toán, từ đó đưa ra các yêu cầu cụ thể để tìm kiếm giải pháp phù hợp.

Bảng 2.1 Các yêu cầu bài toán.

Các bước	Yêu cầu
Xử lý đầu vào	- Giám sát, thu thập thông tin tải của thiết bị IoT Analyzer. - Định nghĩa ngưỡng quá tải của thiết bị IoT Analyzer. - Tự động kích hoạt khi bị quá tải.

Xử lý dữ liệu	- Phân tán tĩnh hoặc phân tán động - Hỗ trợ gửi dữ liệu giữa các thiết bị IoT Analyzer - Hỗ trợ kích hoạt và dừng gửi dữ liệu giữa các thiết bị IoT Analyzer
Đầu ra	- Hệ thống không bị gián đoạn. - Tăng năng lực xử lý của hệ thống

2.1.2 Khảo sát, đánh giá các giải pháp.

Các thuật toán hỗ trợ cân bằng năng lực xử lý động trên thế giới đã và đang được nghiên cứu và ứng dụng cụ thể. Qua khảo sát các năm gần đây, một số giải pháp công trình khoa học đã được công bố được trình bày trong phần dưới.

a) Thuật toán Biased Random Sampling

b) Thuật toán Active Clustering

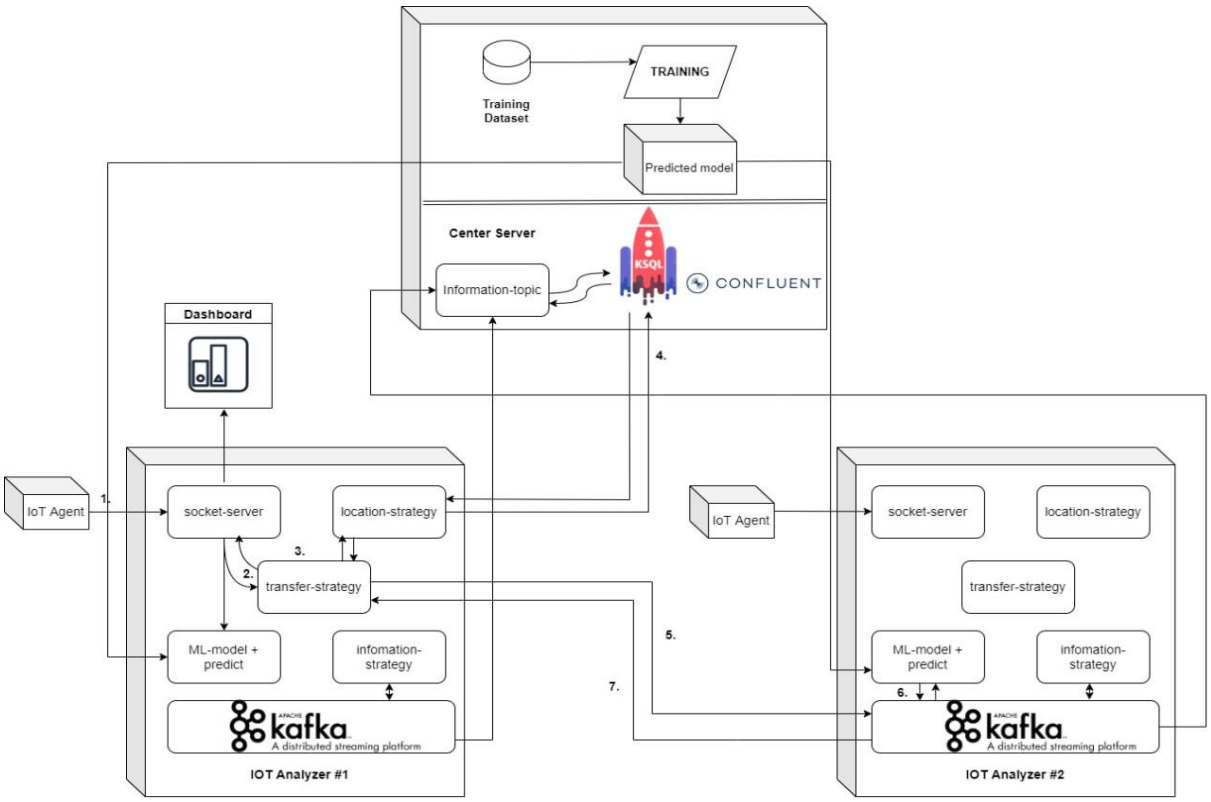
c) Thuật toán Honey Bee Foraging [23]

d) Thuật toán Ant Colony Optimization (ACO)

Các thuật toán trình bày bên trên đều giải quyết được bài toán cân bằng tải với các cách thức hoạt động và xử lý khác nhau. Tuy nhiên các thuật toán này khi áp dụng vào bài toán phân tán động áp dụng trong hệ thống phân tích và phát hiện mã độc IoT thì chưa giải quyết triệt để việc phân tán động. Trong phần tiếp theo, học viên đề xuất một mô hình trong đó, khi một nút bị quá tải nó sẽ là nút chủ và chủ động chọn các nút có tải thấp hơn để chia sẻ tác vụ.

2.2. Mô hình đề xuất

Để đảm bảo các yêu cầu bài toán, học viên đưa ra mô hình đề xuất được minh hoạ tại hình 2.2. Mô hình gồm 2 thành phần chính gồm Center Server và các IoT Analyzer. Các thành phần và hoạt động của hệ thống được mô tả chi tiết ở phần sau.



Hình 2.2 Mô hình đề xuất

2.2.1. Các thành phần trong mô hình

Từ những yêu cầu chức năng trong mô hình, có thể liệt kê ra các hoạt động chính của mô hình và các chủ thể đối tượng người dùng tương ứng. Các đối tượng chính tham gia vào hệ thống này gồm: Thiết bị phân tích cục bộ (IoT Analyzer); Máy chủ giám sát trung tâm (Center Server): Dữ liệu hành vi luồng mạng thu thập từ các thiết bị IoT cỡ nhỏ đã được chuẩn hoá.

2.2.2. Đặc tả yêu cầu chi tiết

a) Yêu cầu chức năng

1) Chức năng "Cấu hình giới hạn tài nguyên hệ thống tại máy chủ trung tâm"
2) Chức năng "Thu thập dữ liệu tài nguyên hệ thống tại máy chủ cục bộ"
3) Chức năng "Gửi yêu cầu hỗ trợ từ máy chủ bị quá tải về máy chủ trung tâm"
4) Chức năng "Cập nhật danh sách tài nguyên hệ thống tại máy chủ trung tâm"
5) Chức năng "Xử lý yêu cầu hỗ trợ quá tải tại IoT Analyzer"
6) Chức năng "Gửi dữ liệu chưa xử lý tới máy chủ hỗ trợ"
7) Chức năng "Xử lý dữ liệu nhận từ máy chủ bị quá tải"

b) Yêu cầu phi chức năng

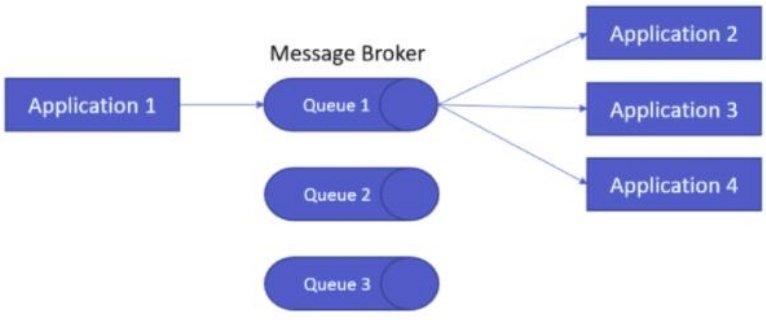
Yêu cầu đặt ra là phải phát triển được hệ thống thỏa mãn các yêu cầu phi chức năng sau:

- Hỗ trợ đọc các tệp dữ liệu lưu trữ theo bảng mã Unicode.
- Hỗ trợ đa nền tảng hệ điều hành phổ biến dành cho máy chủ phân tích bao gồm: Windows, Linux, Unix,…
- Hỗ trợ khả năng sao lưu, khôi phục dữ liệu lưu trữ khi gặp sự cố.

2.3. Ứng dụng Kafka và KSQL trong xây dựng giải pháp xử lý phân tán động

2.3.1 Khảo sát một số công cụ message broker có sẵn.

Để thực hiện được công việc xử lý bản tin (message), học viên đưa ra một số tiêu chí lựa chọn công cụ trong luận văn như sau: Mã nguồn mở, khả năng lưu trữ và xử lý dữ liệu lớn tốt, có khả năng chịu lỗi tốt, có khả năng mở rộng, hiệu năng tốt.



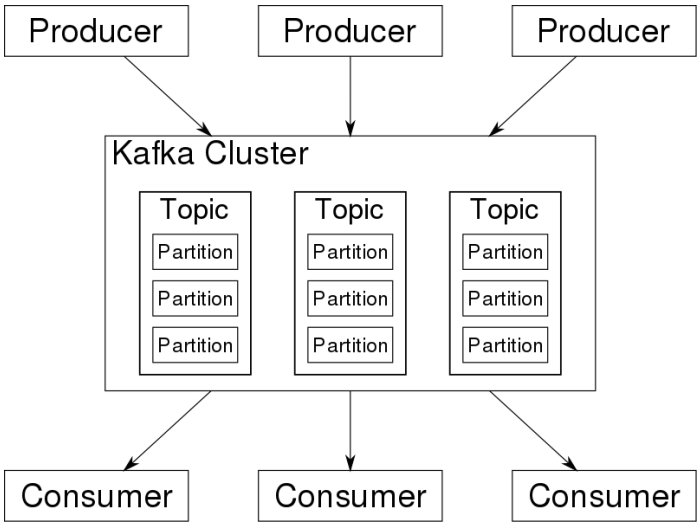
Trong khuôn khổ luận văn, học viên đánh giá 3 công cụ này và chọn một công cụ phù hợp nhất để ứng dụng vào giải pháp cân bằng tải động.

Công cụ	Khả năng mở rộng	Bộ nhớ	Phương thức giao tiếp
RabbitMQ	Dựa trên cấu hình và tài nguyên, khả năng xử lý khoảng 50 nghìn message/giây.	Hỗ trợ cả lưu tin nhắn lâu dài (persistent message) và nhất thời (transient message)	Hỗ trợ cả one-to-one và one-to-many
Redis	Có thể xử lý lên đến 1 triệu message/giây.	Dữ liệu được lưu tại bộ nhớ trong (in-memory database).	Hỗ trợ cả one-to-one và one-to-many
Kafka	Có thể xử lý lên đến 1 triệu message/giây.	Hỗ trợ cả lưu tin nhắn lâu dài (persistent message) và nhất thời (transient message)	Hỗ trợ one-to-many

Dựa vào bảng so sánh bên trên cho thấy công cụ Kafka là một công cụ phù hợp nhất so với các công cụ còn lại. Kafka có khả năng xử lý dữ liệu lớn tốt, khả năng lưu trữ tin nhắn lâu dài, ngoài ra Kafka có khả năng hỗ trợ rất tốt xử lý bất đồng bộ với ngôn ngữ python. Một điểm lợi thế của Kafka đó là hỗ trợ KSQL, một công cụ hỗ trợ truy vấn thông tin luồng dữ liệu (data stream) tương tự như truy vấn cơ sở dữ liệu quan hệ. Do đó học viên lựa chọn công cụ Kafka và KSQL để ứng dụng giải pháp phân tán động trong luận văn. Phần tiếp theo sẽ mô tả chi tiết công cụ Kafka và KSQL.

2.3.1 Lựa chọn công cụ Kafka và công cụ KSQL.

Nền tảng trao đổi thông điệp Apache Kafka: Apache Kafka [28] là một nền tảng xử lý stream mã nguồn mở được phát triển bởi Apache Software Foundation được viết bằng ngôn ngữ lập trình Scala và Java. Kafka cho hệ thống truyền tải dữ liệu phân tán. Kafka lưu trữ thông điệp trong các chủ đề (topic) được phân vùng và sao chép qua nhiều broker trong một cụm (Cluster). Publisher (nguồn gửi thông tin) gửi thông điệp đến các topic mà nguồn nhận thông tin (consumer) đã đăng ký để đọc thông điệp.



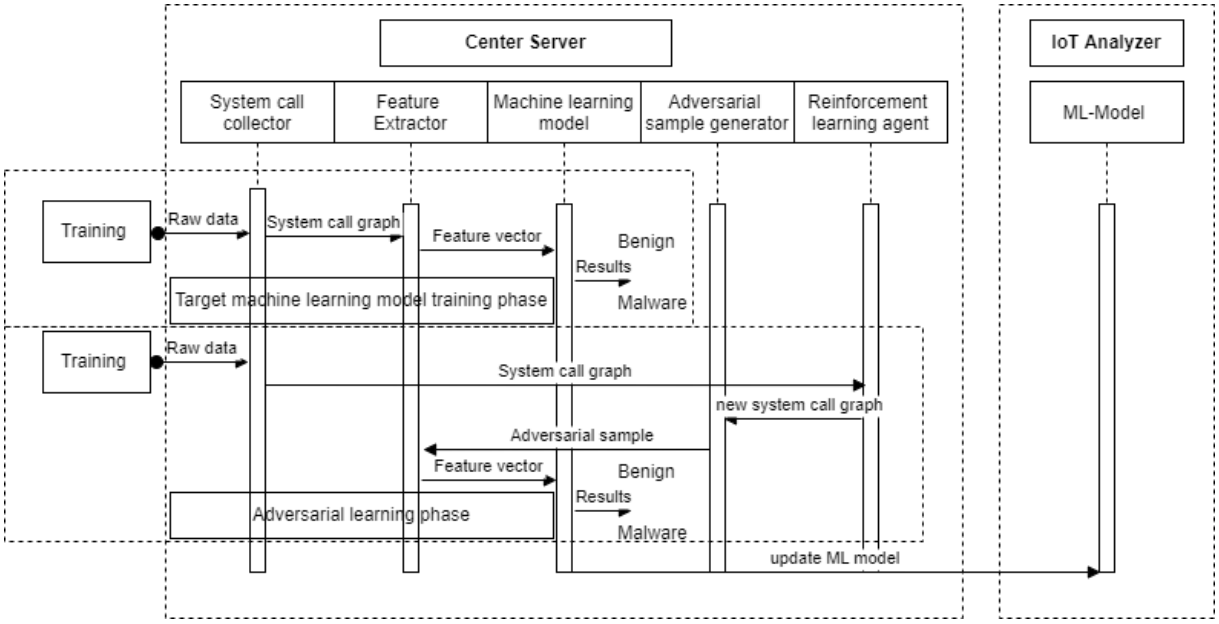
Hình 2.3. Kiến trúc truyền tải thông tin của Apache Kafka

KSQL cho phép xử lý và phân tích dữ liệu truyền trực tuyến theo thời gian thực có trong nền tảng Apache Kafka. Nói cách khác, KSQL cung cấp một khung tương tác để thực hiện các hoạt động Xử lý Luồng như Tổng hợp dữ liệu, Lọc, Kết hợp, Sessionization, Windowing

2.3.2 Hoạt động của mô hình.

a) Máy chủ trung tâm

Máy chủ trung tâm có hai nhiệm vụ chính là (1) thực hiện đào tạo mô hình học máy để phát hiện mã độc IoT Botnet và (2) theo dõi, lưu trữ thông tin tải của tất cả hệ thống.



Hình 2.4 Mô hình huấn luyện học máy phân tích và phát hiện mã độc IoT botnet

Mô đun theo dõi, lưu trữ thông tin tải của hệ thống được coi là trái tim của hệ thống. Mô đun này được sử dụng platform Confluent [29] trong đó tích hợp sẵn công cụ Kafka và KSQL.

b) Thiết bị phân tích IoT Analyzer

Cấu trúc của IoT Analyzer được minh họa tại gồm 5 thành phần chính: ***socket-server*** chịu trách nhiệm nhận/gửi dữ liệu. ***ML-model + predict*****:** là mô hình học máy xử lý các dữ liệu nhận được từ IoT Agent. ***Information-strategy*** là mô đun thu thập thông tin trạng thái của hệ thống. ***Location-strategy*** là mô đun xử lý truy vấn lên máy chủ tập trung để yêu cầu chia sẻ tác vụ khi bị quá tải. ***Transfer-strategy*** là mô đun truyền nhận dữ liệu chưa được xử lý từ một IoT Analyzer quá tải sang một IoT Analyzer khác được chọn từ location-stategy.

c) Thiết bị IoT Agent

Thiết bị IoT Agent là các thiết bị IoT có khả năng thu thập thông tin hoạt động của nó và gửi lên cho IoT Analyzer. Các thông tin được sử dụng là các lệnh gọi hệ thống (system call) được gửi lên cho IoT Analyzer bằng một TCP socket.

d) Cơ chế cân bằng tải.

Thuật toán cân bằng tải bao gồm một số thành phần tương tác theo nhiều cách khác nhau để phân phối lại các công việc đã gửi của người dùng giữa các nút của hệ thống phân tán. Các bước gồm: (1) Đo tải cục bộ: Tại mỗi nút, một cơ chế phải được cung cấp để đưa ra ước tính tốt về tải cục bộ hiện tại. (2) Chính sách thông tin: Thành phần này chịu trách nhiệm

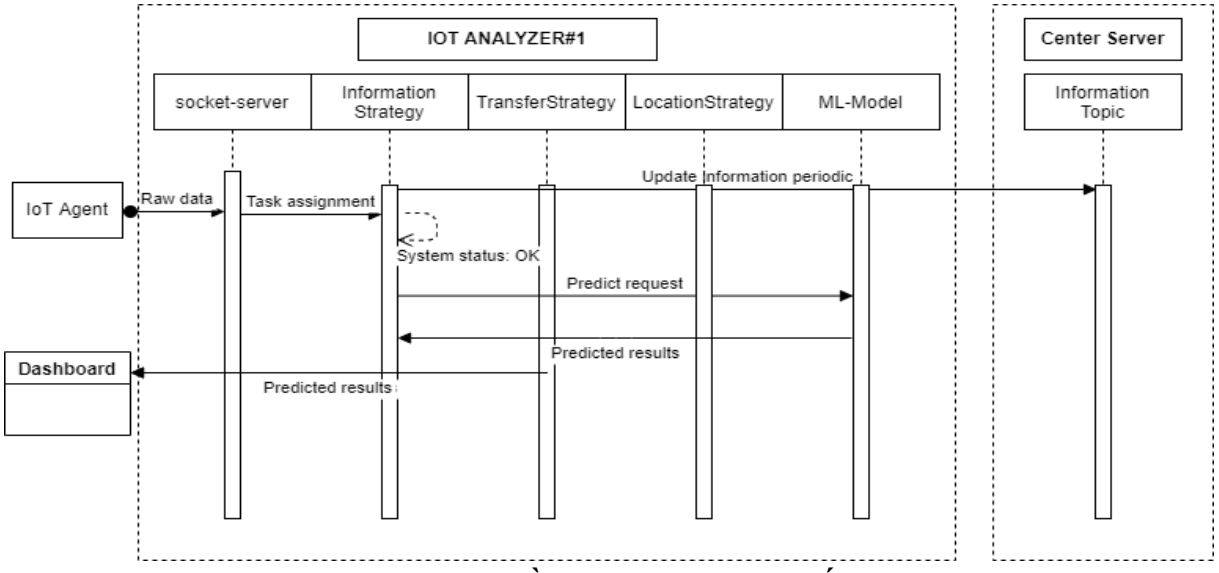
trao đổi và duy trì thông tin cục bộ hoặc nhóm nút khác như mức tải, khối lượng công việc hoặc tải trung bình trên toàn bộ hệ thống. (3) Chính sách chuyển giao: Thành phần này quyết định thời điểm có lợi khi chuyển một quy trình từ khối lượng công việc cục bộ và chọn quy trình nào để chuyển các tác vụ. (4) Chính sách đàm phán: Khi một nút quá tải quyết định rằng một nút khác là nút chuyển giao phù hợp

Khi IoT Analyzer hoạt động trong trạng thái không bị quá tải (load < threshold), luồng công việc được mô tả trong hình 2.5. Cụ thể các bước thực hiện như sau:

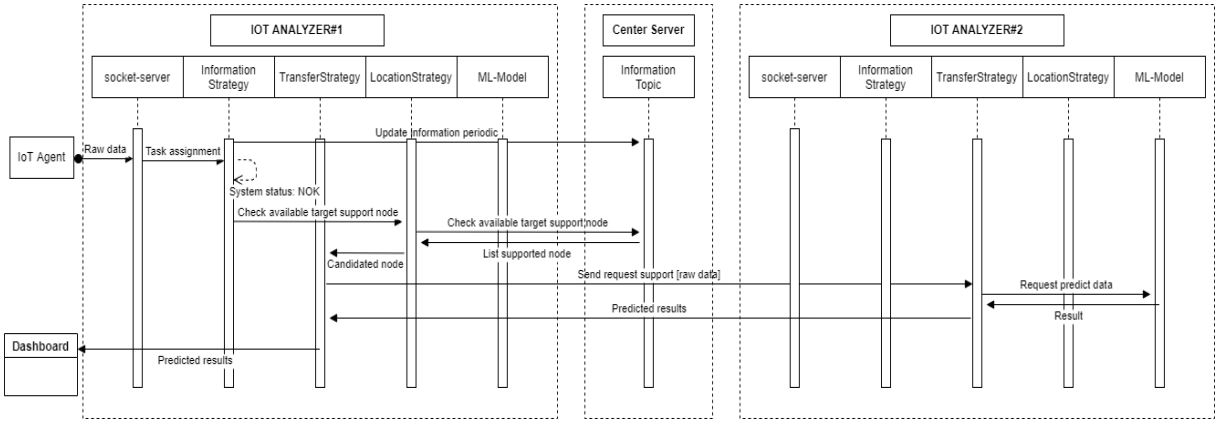
- Bước 1: IoT Analyzer cập nhật trạng thái hệ thống của nó thường xuyên cho máy chủ trung tâm qua Information Strategy
- Bước 2: IoT Agent gửi dữ liệu chưa được xử lý lên cho IoT Analyzer. Dữ liệu này là các tệp thực thi lệnh gọi hệ thống được xử lý thành các đồ thị. (system call graph).
- Bước 3: IoT Analyzer nhận dữ liệu bằng socket-server và thông báo rằng hệ thống cần xử lý dữ liệu.
- Bước 4: Information Strategy sẽ kiểm tra trạng thái hệ thống, kết quả trả về hệ thống đang ở trạng thái không bị quá tải. Điều kiện để hệ thống ở trạng thái không bị quá tải:

$$\left(CPU_{usage} < threshold \middle| RAM_{usage} < threshold\right)$$

  Trong đó threshold = 90%.
- Bước 5: Mô đun học máy sẽ thực hiện phân tích và đưa ra dữ liệu nhận được có phải mã độc hay không và phản hồi lại.
- Bước 6: IoT Analyzer sẽ gửi kết quả đến cho Dashboard.



- **Hình 2.5 Lưu đồ hoạt động của hệ thống ở trạng thái trong tải**



Hình 2.6 Lưu đồ hoạt động của hệ thống khi một node bị quá tải

Khi một nút bị quá tải (load>threshold), luồng công việc sẽ được thực hiện như sau: information-stategy chạy trên từng IoT Analyzer duy trì việc cập nhật tình trạng tài nguyên hệ thống trên máy chủ trung tâm.

- Bước 1: IoT Analyzer cập nhật trạng thái của nó định kỳ cho máy chủ trung tâm qua Information Strategy.
- Bước 2: IoT Agent gửi dữ liệu chưa được xử lý về nút IoT Analyzer. Information Strategy kiểm tra trạng thái hệ thống có bị quá tải hay không. Điều kiện để hệ thống không bị quá tải như sau:

$$\left(CPU_{usage} < threshold \middle| RAM_{usage} < threshold\right)$$

  Trong đó threshold = 90%.
- Bước 3: socket-server gọi location-stategy để tìm kiếm nút đáp ứng yêu cầu tính toán.
- Bước 4: location-strategy thực hiện tìm kiếm nút đủ điều kiện chuyển giao đáp ứng được nhu cầu tính toán (dựa trên RAM, CPU và độ trễ mạng). Location-strategy sử dụng KSQL API để truy vấn dữ liệu trong topic information-stategy nằm trên máy chủ trung tâm và nhận về danh sách các nút có thể đáp ứng được.
- Bước 5: IoT Analyzer bị quá tải tiến hành lệnh ping tới các nút trong danh sách trên, nút có kết quả ping thấp nhất sẽ được chọn để phân phối tác vụ.
- Bước 6: transfer-stategy truyền thông tin từ nút quá tải sang nút từ xa sử dụng Kafka.
- Bước 7: Nút từ xa xử lý dữ liệu và tính toán, sau đó trả về nút quá tải.

Kết luận chương 2.

Chương 2 học viên đã trình bày về bài toán phân tán động cần xử lý trong hệ thống phân tích mã độc IoT botnet, từ đó đưa ra những phân tích, yêu cầu của hệ thống. Học viên cũng đã đề xuất một mô hình phân tán động nhằm giải quyết bài toán nêu trên. Cụ thể trong chương 2 đã trình bày:

- Khảo sát, đánh giá các thuật toán hỗ trợ phân tán động và cân bằng tải động.
- Phân tích, thiết kế chức năng phân tán động giữa các thiết bị IoT Analyzer.
- Lựa chọn sử dụng các công nghệ mới để tích hợp giải pháp hỗ trợ cân bằng tải động và phân tán năng lực xử lý giữa các thiết bị IoT Analyzer.

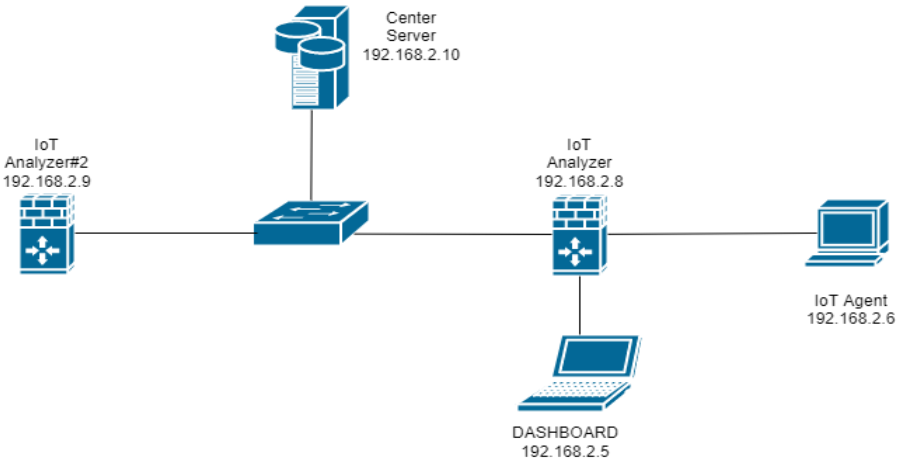
CHƯƠNG 3. THỬ NGHIỆM VÀ ĐÁNH GIÁ.

Trong chương 3, học viên sẽ trình bày về cách thức xây dựng môi trường thử nghiệm và kết quả đánh giá mô hình. Các kịch bản và thông số đánh giá cũng được định nghĩa để đưa ra ưu nhược điểm của mô hình đề xuất. Cuối cùng của chương sẽ trình bày kết quả thử nghiệm và kết luận.

3.1. Môi trường thử nghiệm

3.1.1 Hệ thống thử nghiệm

Hệ thống thử nghiệm bao gồm Máy chủ trung tâm, hai máy chủ đóng vai trò IoT Analyzer, 1 máy trạm đóng vai trò như IoT Agent và 1 máy trạm đóng vai trò như máy chủ Dashboard. Tất cả các máy đều sử dụng chung một hệ thống mạng và được kết nối với nhau thông qua một bộ chuyển mạch. Trên máy chủ trung tâm sẽ được cài đặt Confluent platform cho Kafka và KSQL, trên các máy IoT Analyzer được cài đặt Kafka để truyền các thông tin tải và dữ liệu. Mô hình thử nghiệm được minh hoạ trong hình 3.1.



Hình 3.1 Mô hình thử nghiệm

Các thông số phần cứng của hệ thống như sau: Cấu hình máy chủ trung tâm gồm CPU Intel Xeon E5-2689V1 @2.6GHz + 32GB RAM + GPU RX570 4GB GDDR5 Ubuntu 20.04 LTS, cấu hình máy IoT Analyzer gồm CPU Intel Core I5 8300H @2.3GHz + 16GB RAM + GPU GTX1050Ti 4GB GDDR5 Ubuntu 20.04 LTS.

3.1.2 Quá trình triển khai giải pháp đề xuất

Tại máy chủ trung tâm: Cài đặt platform confluent trong đó tích hợp công cụ Kafka và KSQL.

Bước 1: Cài đặt công cụ Docker và Docker compose

Bước 2: Tạo file docker-compose.yml mô tả các container dịch vụ của Kafka.

Bước 3: Truy cập vào giao diện GUI và tạo các Topic.

Tại IoT Analyzer: Thực hiện chạy các module cho giải pháp cân bằng tải. Khi một IoT Agent gửi file đến IoT Analyzer, socket server sẽ nhận file và bắt đầu thực hiện xử lý.

Bước 1: Copy source code đến IoT Analyzer.

Đường dẫn source code: https://github.com/huunguyen95/distributed-model-thesis

Bước 2: Chạy module Information Strategy để gửi thông tin tài nguyen cho máy chủ trung tâm bằng câu lệnh: *"nohup python3 InformationStrategy.py &"*

Bước 3: Chạy module socket server bằng câu lệnh: *"nohup python3 socker_server.py &"*

Bước 4: Chạy module TransferStategy bằng câu lệnh: "*TransferStategy faust -A data_process_app worker -l info"*

Tại IoT Agent: Khởi chạy module gửi file bằng câu lệnh: *"python3 socket_client.py".* Dữ liệu được gửi chứa trong client_data/test.

3.2. Kịch bản thử nghiệm và bộ tiêu chí đánh giá

3.2.1 Kịch bản thử nghiệm

Tại phần thử nghiệm, học viên sử dụng 250 tệp lệnh gọi hệ thống dưới dạng đồ thị (system call graph). Các tệp này được gửi từ phía IoT Agent cho IoT Analyzer, có định dạng ".adjlist" và nặng tối đa 9.9 MB.

a) Kịch bản với trường hợp thử nghiệm chưa áp dụng phân tán tác vụ.

Kịch bản 1: Khi IoT Analyzer làm việc ở trạng thái không quá tải.

Bước 1: Thực hiện đẩy dữ liệu chưa được xử lý từ IoT Agent.

Bước 2. Thực hiện phân tích trên IoT Analyzer.

Bước 3: Thu thập kết quả.

Kịch bản 2: Khi IoT Analyzer quá tải CPU và RAM.

Bước 1: Tạo môi trường giả lập đầy CPU sử dụng thư viện stress.

```
apt install stress-ng
stress-ng --vm-bytes $(awk '/MemAvailable/{printf "%d\n", $2 * 0.9;}' < /proc/meminfo)k --vm-keep -m 1
```

Bước 2: Thực hiện đẩy dữ liệu chưa được xử lý từ IoT Agent.

Bước 3. IoT Analyzer bị quá tải nhưng vẫn phải xử lý tác vụ.

Bước 4: Thu thập kết quả.

b) Kịch bản với trường hợp thử nghiệm áp dụng mô hình phân tán tác vụ.

Kịch bản 3: Khi IoT Analyzer quá tải CPU và RAM.

Bước 1: Tạo môi trường giả lập đầy CPU sử dụng thư viện stress.

```
apt install stress-ng
stress-ng --vm-bytes $(awk '/MemAvailable/{printf "%d\n", $2 * 0.9;}' < /proc/meminfo)k --vm-keep -m 1
```

Bước 2: Thực hiện đẩy dữ liệu chưa được xử lý từ IoT Agent.

Bước 3. IoT Analyzer bị quá tải sẽ phân tán tác vụ cho IoT Analyzer khác rảnh rỗi thông qua mô hình.

Bước 4: Thu thập kết quả.

3.2.2 Tiêu chí đánh giá

Mục đích của bộ cân bằng tải có 2 tiêu chí chính đó là:

- Đảm bảo tính sẵn sàng của hệ thống.
- Đảm bảo được việc xử lý lưu lượng traffic lớn của ứng dụng mà không bị mất dịch vụ.

Với mục đích nêu trên, có rất nhiều các tiêu chí để đánh giá sự hiệu quả của bộ cân bằng tải như đo đạc các thông số về tài nguyên sử dụng (resource ultilization), độ trễ (latency), chi phí (cost), ….

Tuy nhiên, trong khuôn khổ của luận văn, để chứng minh sự hiệu quả của giải pháp cân bằng tải động có thể giải quyết được bài toán đã nêu ở mục 2.1 không, học viên đánh giá sự hiệu quả của giải pháp theo một số thông số như sau:

- Tổng thời gian xử lý một lượng công việc giống nhau (R): Giá trị của tổng thời gian phản hồi khi thực hiện các tác vụ trên IoT Analyzer.

$$R = T_1 + T_2 + \cdots + T_n$$

Trong đó: T_i là thời gian xử lý tác vụ thứ i

- Khả năng xử lý của hệ thống (S): Được tính là khối lượng dữ liệu được xử lý trong một khoảng thời gian.

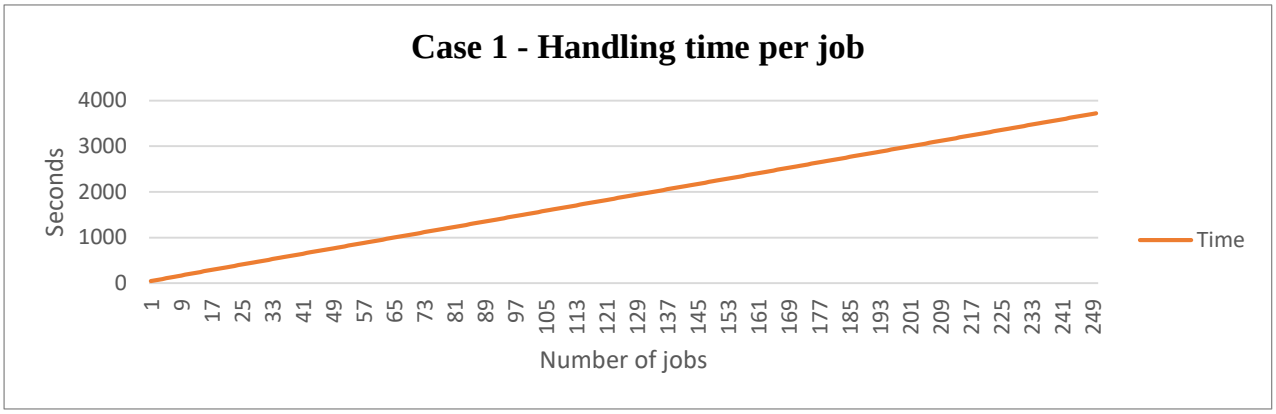
$$S = \frac{Task\ (MB)}{Time\ (s)}$$

- Hai giá trị này này sẽ được ghi lại và đánh giá qua các kịch bản khác nhau.

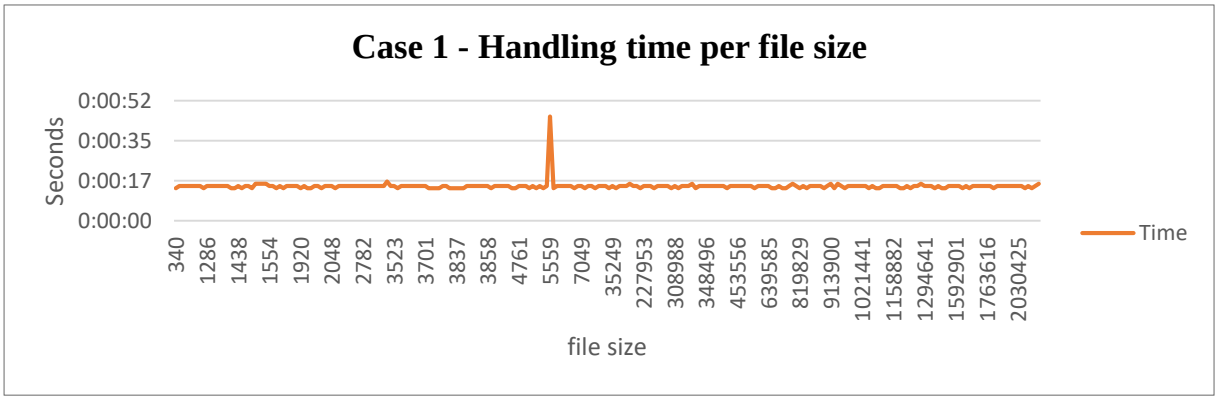
3.3. Kết quả và đánh giá

3.3.1. Kịch bản 1

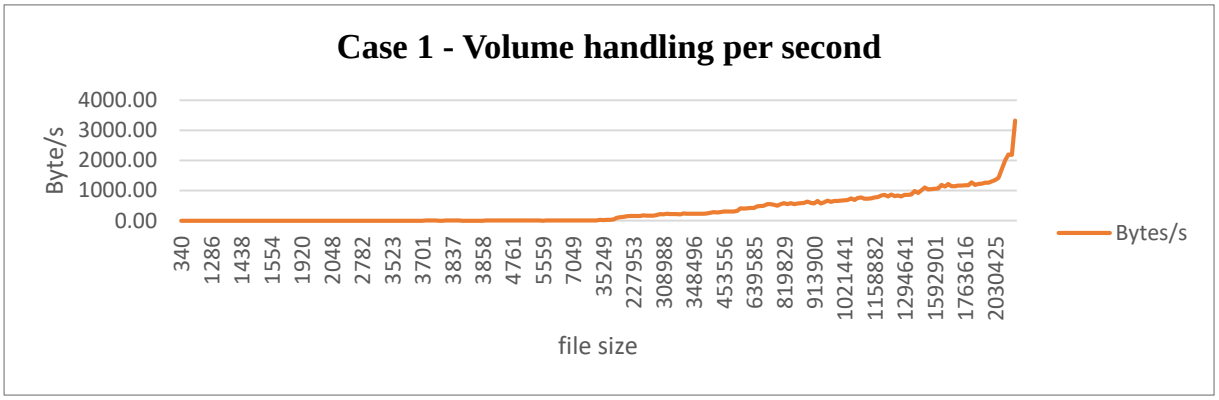
Tại kịch bản 1, kết quả cho thấy khi một nút IoT Analyzer ở trạng thái làm việc bình thường xử lý 250 tệp tin mất 3721 giây (tương đương 1 giờ 2 phút 1 giây). Thời gian xử lý trung bình một tệp tin vào khoảng 15 giây. Hình 3.2 mô tả thời gian xử lý của một IoT Analyzer với số lượng các tệp tin đầu vào. Hình 3.3 mô tả thời gian xử lý một tệp tin với kích thước khác nhau. Cuối cùng hình 3.4 cho thấy khối lượng dữ liệu hệ thống xử lý trong 1 giây khi không bị quá tải.



Hình 3.2 Kịch bản 1: Thời gian xử lý trên số lượng các tệp tin đầu vào.



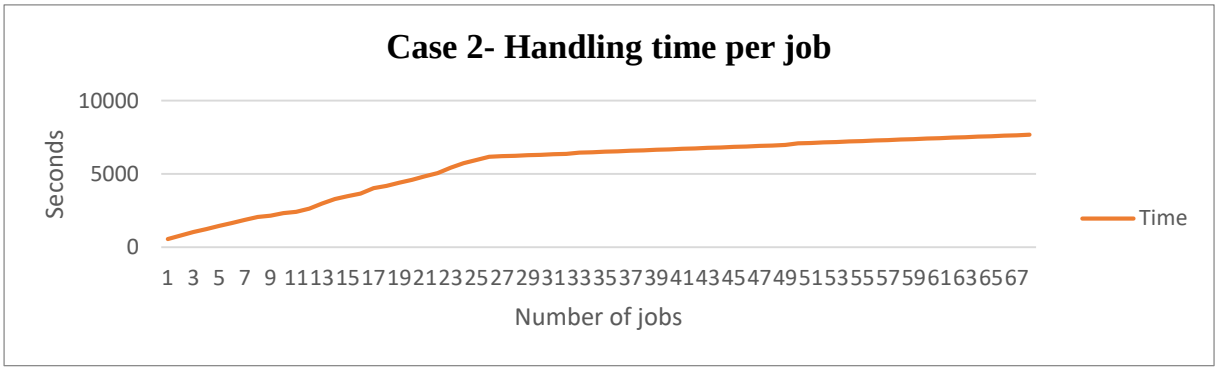
Hình 3.3 Kịch bản 1: Thời gian xử lý ứng với kích cỡ của dữ liệu đầu vào.



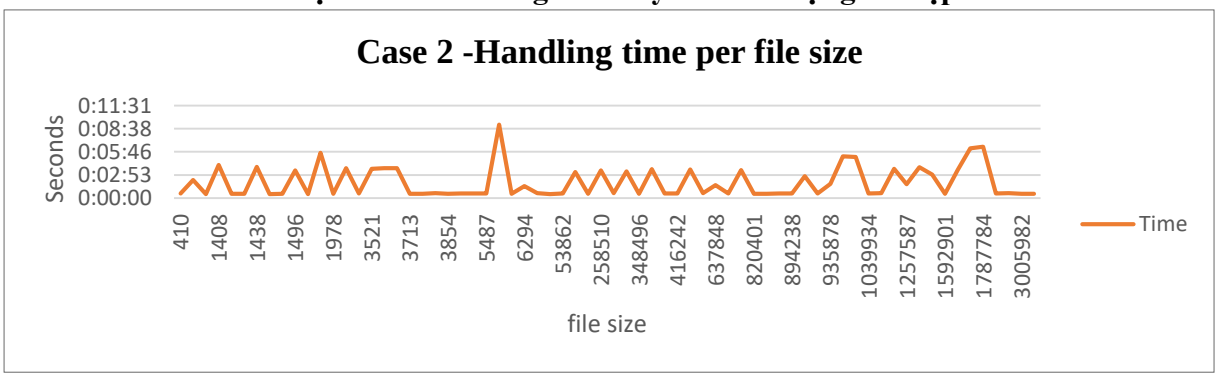
Hình 3.4 Kịch bản 1: Khối lượng xử lý ứng với kích cỡ tệp dữ liệu đầu vào.

3.3.2. Kịch bản 2

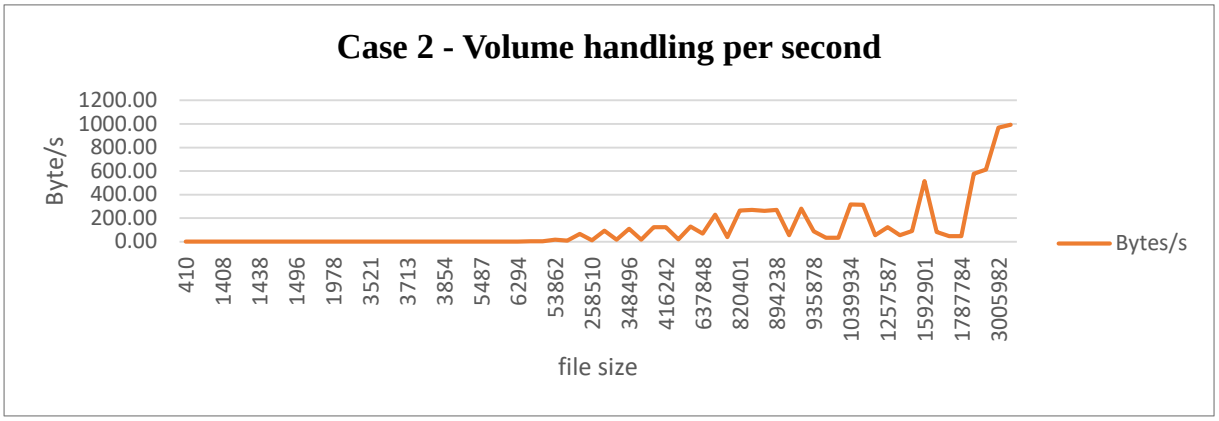
Tại kịch bản 2, khi hệ thống bị quá tải cho thấy kết quả xử lý chậm đi rất đáng kể. Tải của hệ thống khi đang xử lý: CPU 100%, RAM 92%. Khi hệ thống xử lý chậm là mối tiềm tàng về bảo mật trong hệ thống. IoT Analyzer ở trạng thái quá tải xử lý 69 tệp tin mất 7674 giây (tương đương 2 giờ 7 phút 54 giây) gần gấp đôi so với kịch bản 1 và hiệu suất kém rất nhiều. Thời gian xử lý trung bình một tệp tin vào khoảng 1 phút 56 giây. Hình 3.5 mô tả thời gian xử lý của một IoT Analyzer với số lượng các tệp tin đầu vào. Hình 3.6 mô tả thời gian xử lý một tệp tin với kích thước khác nhau. Cuối cùng hình 3.7 cho thấy khối lượng dữ liệu hệ thống xử lý trong 1 giây khi bị quá tải.



Hình 3.5 Kịch bản 2: Thời gian xử lý trên số lượng các tệp tin đầu vào.



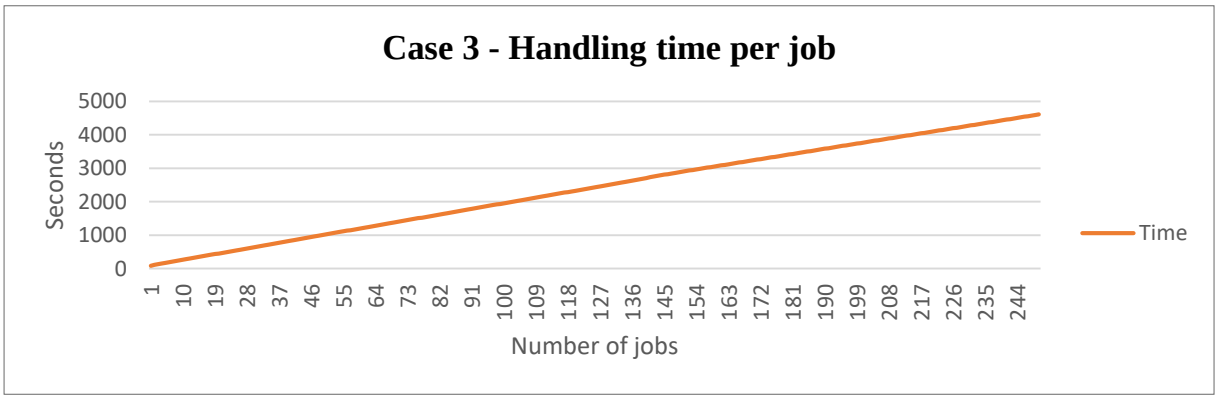
Hình 3.6 Kịch bản 2: Thời gian xử lý ứng với kích cỡ của dữ liệu đầu vào.



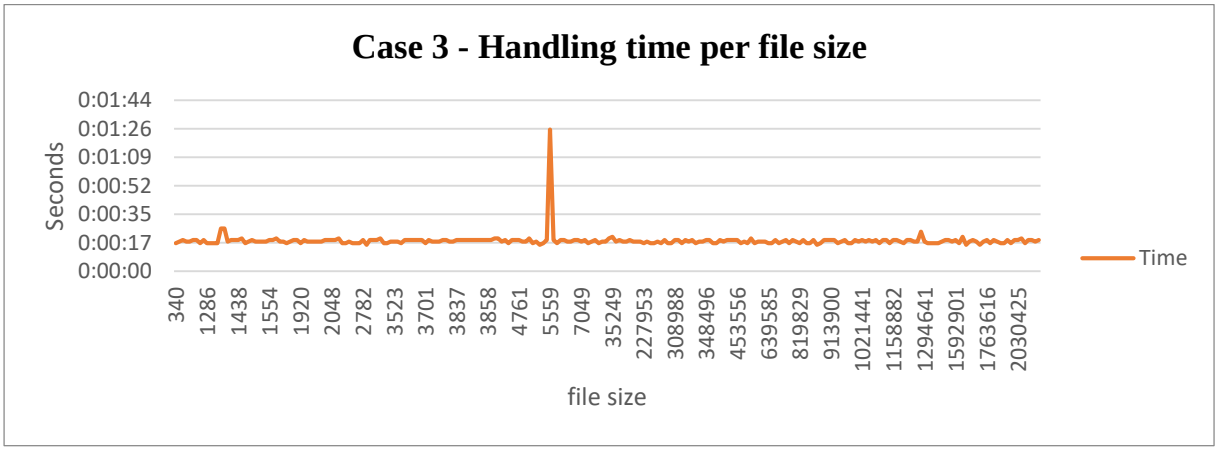
Hình 3.7 Kịch bản 2: Khối lượng xử lý ứng với kích cỡ tệp dữ liệu đầu vào.

3.3.3. Kịch bản 3

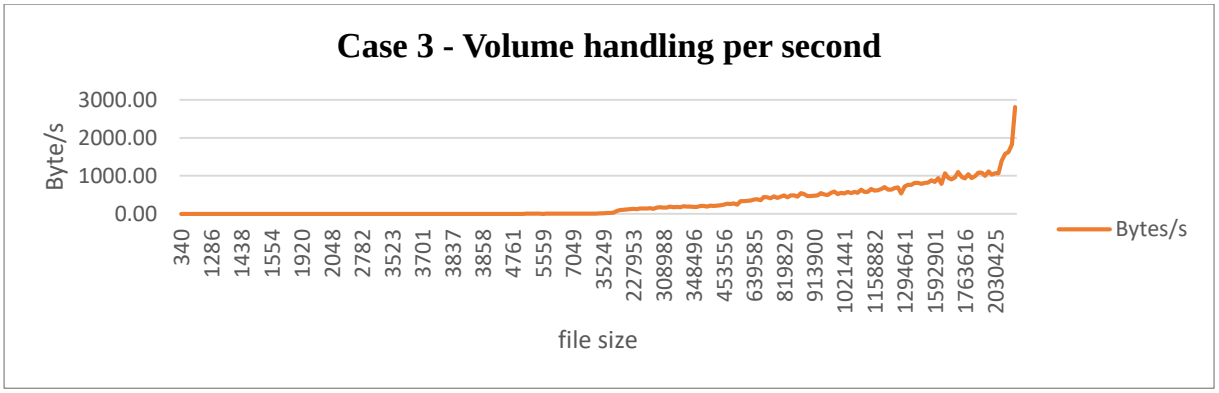
Tại kịch bản 3, khi hệ thống bị quá tải đã đẩy các tác vụ cho một nút IoT Analyzer khác. Việc này không làm gián đoạn quá trình phân tích và phát hiện mã độc IoT botnet của hệ thống, mặt khác có thể tối ưu tài nguyên rảnh rỗi. Tải của hệ thống khi đang xử lý: CPU 100%, RAM 92%, nút IoT Analyzer đích không bị quá tải có CPU, RAM < 90%. IoT Analyzer ở trạng thái quá tải xử lý 250 tệp tin mất 4608 giây (tương đương 1 giờ 16 phút 46 giây). Thời gian xử lý trung bình một tệp tin vào khoảng 18 giây. Hình 3.8 mô tả thời gian xử lý của một IoT Analyzer với số lượng các tệp tin đầu vào. Hình 3.9 mô tả thời gian xử lý một tệp tin với kích thước khác nhau. Hình 3.10 cho thấy khối lượng dữ liệu hệ thống xử lý trong 1 giây khi bị quá tải. Cuối cùng Hình 3.11 là biểu đồ so sánh giữa các kịch bản, cho thấy hiệu quả của việc phân tán tác vụ trong hệ thống.



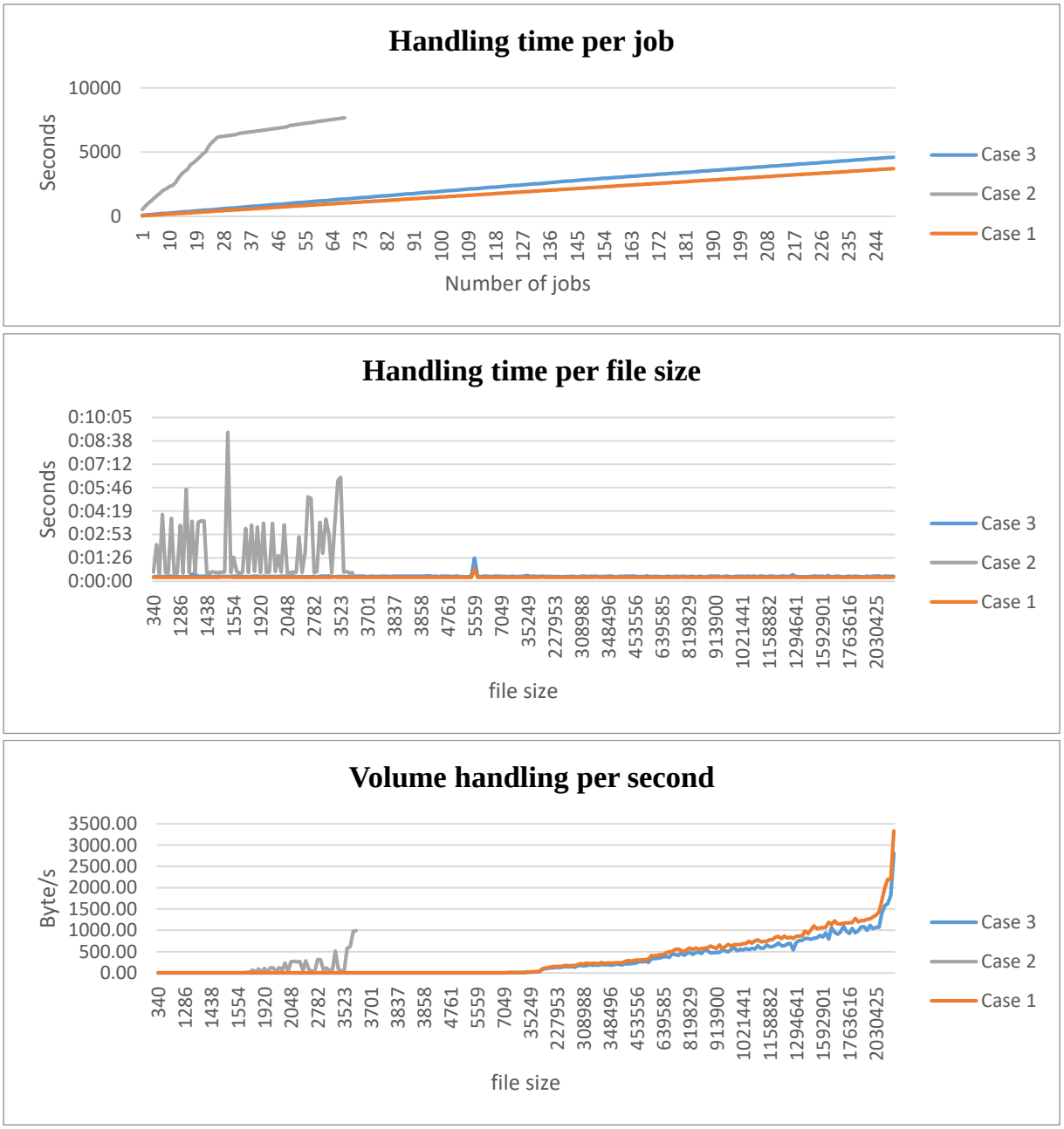
Hình 3.8 Kịch bản 3: Thời gian xử lý trên số lượng các tệp tin đầu vào.



Hình 3.9 Kịch bản 3: Thời gian xử lý ứng với kích cỡ của dữ liệu đầu vào.



Hình 3.10 Kịch bản 3: Khối lượng xử lý ứng với kích cỡ tệp dữ liệu đầu vào.



Hình 3.11 So sánh hiệu quả giữa khi áp dụng giải pháp phân tán động (kịch bản 2) và khi không áp dụng giải pháp phân tán động.

Bảng 3.1 So sánh kết quả giữa áp dụng giải pháp phân tán động (kịch bản 2) và khi không áp dụng giải pháp phân tán động.

Kịch bản	Số lượng tệp tin xử lý	Tổng thời gian xử lý	Trung bình thời gian xử lý	Số byte xử lý trung bình trong 1 giây
1	250	1:02:01	0:15	33573.13
2	69	2:07:54	1:56	10940.48
3	250	1:16:48	0:19	27383.72

Kết luận chương 3.

Trong chương này, học viên đã đưa ra môi trường thử nghiệm sát với thực tế nhất đối với mô hình đề xuất. Ngoài ra, học viên đã đưa ra một số kịch bản thực nghiệm trong trường hợp không có cơ chế cân bằng tải và có có chế cân bằng tải. Đồng thời đưa ra các tham số để đánh giá khả năng phân tán tác vụ của hệ thống. Kết quả đạt được khi thử nghiệm tương đối khả quan khi áp dụng cơ chế phân tán tác vụ nhằm tăng cường khả năng phát hiện mã độc một cách nhanh chóng. Điều này cũng cho thấy tiềm năng của mô hình và có thể được tối ưu, sửa đổi và phát triển trong tương lai.

KẾT LUẬN

Qua thời gian nghiên cứu, tìm hiểu, tiếp cận các tri thức về lĩnh vực bảo mật trong IoT, đặc biệt là phát hiện mã độc IoT Botnet, luận văn đã đưa ra mô hình phát hiện mã độc IoT Botnet, được công bố tại hội nghị Computer Science On-line Conference thứ 11 năm 2022. Đặc biệt, để nâng cao khả năng của hệ thống khi thiết bị IoT Analyzer bị quá tải, luận văn đã đưa ra mô hình phân tán động nhằm phân tán tác vụ và thực hiện phân tích mã độc một cách nhanh chóng nhất. Luận văn "Nghiên cứu xây dựng giải pháp phân tán động trong hệ thống phân tích mã độc IoT botnet dựa trên kafka và ksql" đã được học viên hoàn thành và đạt được các mục tiêu nghiên cứu. Các vấn đề được trình bày bao gồm:

- Tổng quan cơ sở lý thuyết mã độc trong IoT, mã độc IoT botnet, từ đó đưa ra một mô hình phân tích và phát hiện mã độc IoT Botnet.
- Tổng quan về giải pháp phân tán nói chung, đặc biệt là hệ thống phân tán động được áp dụng rất nhiều trong việc tránh quá tải ngày nay.
- Trình bày về bài toán cần xử lý trong mô hình phân tích và phát hiện mã độc IoT botnet và đưa ra một mô hình giải pháp phân tán động tác vụ dựa trên các công nghệ mới nhất hiện nay là Kafka và KSQL.
- Cuối cùng, luận văn trình bày môi trường thử nghiệm hệ thống, xây dựng chi tiết các kịch bản đánh giá cũng như bộ tham số đánh giá.

Qua kết quả thử nghiệm cho thấy, hiệu năng của hệ thống phân tích và phát hiện mã độc được đảm bảo khi sử dụng cơ chế phân tán tác vụ động. Từ đó có thể nâng cao độ tin cậy và khả năng phân tích, phát hiện mã độc IoT botnet trên các thiết bị IoT. Thử nghiệm cho kết quả tương đối khả quan và cho thấy tiềm năng ứng dụng của mô hình trong tương lai. Tuy nhiên, mô hình trong luận văn cũng cần cải thiện khi dữ liệu IoT mở rộng ra rất lớn và mô hình học máy cần được cập nhật. Trong tương lai, học viên sẽ tiếp tục nghiên cứu và tối ưu mô hình bằng cách áp dụng các giải pháp xử lý dữ liệu lớn cũng như mô hình học máy hiện đại để đem lại những hiệu quả tốt hơn.

Trên đây là một số kết luận và hướng nghiên cứu của luận văn. Học viên xin chân thành cảm ơn các quý thầy, cô đã hướng dẫn khoa học và phản biện để giúp học viên hoàn thiện những thiếu sót trong luận văn cũng như định hướng cho việc phát triển đề tài trong tương lai.