LSTM-BASED SERVER AND ROUTE SELECTION IN DISTRIBUTED AND HETEROGENEOUS SDN NETWORK

Status	Idea
Date Added	@September 27, 2023 3:25 PM
Live Date	@September 27, 2023
Sponsor	
Tags	Information

🏆 **GOAL** | Khái niệm tổng quan

SCRIPT

Summary

Main contents

Vocabulary

Aa New words	Type	Meaning	Example
benchmarks	noun	tiêu chuẩn	The experimental results demonstrate that our proposal improves link utilization, packet loss rate, response time, and overhead by up to 15%, 10%, 14%, and 25% respectively to benchmarks.

Aa New words	Type	Meaning	Example
propagated	adjective	việc truyền tải, lan truyền hoặc chia sẻ thông tin, dữ liệu hoặc trạng thái mạng từ một nguồn đến các điểm đích khác trong mạng. Nó liên quan đến việc đảm bảo rằng thông tin được đưa ra và được cập nhật ở một nơi có thể được truy cập và sử dụng bởi các thành phần khác trong hệ thống mạng.	The former enables other SDN controllers to update the propagated knowledge based on the REST API platform
stochastic	adjective	A stochastic process or system is connected with random probability.	To deal with the second limitation, we aim to study various evolutionary algorithms (e.g., genetic, ant-colony techniques, etc.) with multi-intelligent agents to make our server and route selection mechanism more adaptive in stochastic environments.

0. Abstract

- East-West interface SINA: adaptively ensure network state consistency in a multi-domain SDN networks
- LSTM-based link cost prediction for the server and route selection mechanism in a distributed and heterogeneous SDN network

💡 **Chú ý:**

1. **LSTM (Long Short-Term Memory):** LSTM là một loại mạng thần kinh nhân tạo được sử dụng trong việc xử lý dữ liệu chuỗi, như dữ liệu thời gian hoặc dữ liệu có liên quan đến thứ tự. Đặc điểm quan trọng của LSTM là khả năng xử lý và học được các phụ thuộc dài hạn trong dữ liệu chuỗi. Điều này làm cho LSTM rất phù hợp cho các ứng dụng liên quan đến dự đoán dựa trên lịch sử và chuỗi thời gian, như dự đoán tình hình định tuyến trong mạng.
2. **Artificial Neural Network (ANN):** ANN, hay còn gọi là mạng thần kinh nhân tạo, là một mô hình toán học được thiết kế để mô phỏng cấu trúc và hoạt động của não người. Nó bao gồm nhiều lớp (layers) của các đơn vị tính toán gọi là neuron, và các liên kết giữa chúng có trọng số. ANN được sử dụng để học từ dữ liệu và thực hiện các nhiệm vụ như phân loại, dự đoán và tối ưu hóa.
3. **SINA (SDN Inter-Domains Network Application):** Đây là một ứng dụng, một phần mềm, hoặc một công nghệ được đề xuất để hỗ trợ việc chia sẻ trạng thái mạng giữa các miền SDN khác nhau, trong một mô hình mạng phân phối và đa dạng. Nhiệm vụ chính của SINA là cho phép các miền SDN khác nhau cập nhật và chia sẻ trạng thái mạng của họ với nhau.
    a. **SINA Listener component:** Đây là một thành phần chạy tương tự như một tiến trình daemon (dịch vụ nền), và nhiệm vụ của nó là lắng nghe và bắt các sự kiện thay đổi trạng thái mạng trong các dịch vụ lõi của mỗi điều khiển. Sau đó, các sự kiện này được truyền đạt cho các điều khiển SDN khác.
    b. **SINA API component:** Đây là một tập hợp các định nghĩa và giao thức được sử dụng để quản lý cơ sở dữ liệu trạng thái mạng. SINA API cho phép các điều khiển SDN làm cho tài nguyên của họ truy cập được. Ví dụ, khi có một sự kiện được ghi lại từ dịch vụ lõi, điều khiển một thông báo cho điều khiển hai bằng cách sử dụng thành phần SINA API. Thành phần này chịu trách nhiệm duy trì và cập nhật cơ sở dữ liệu trạng thái mạng cục bộ.
4. **Giao diện Bắc-Nam (North-South Interface):**
    - **Hướng:** Giao diện Bắc-Nam thường đề cập đến giao tiếp giữa các thiết bị điều khiển SDN (SDN controllers) và các thiết bị mạng truyền thống hoặc máy tính ngoài mạng, tức là từ "bắc" đến "nam" của mạng, điều này liên quan đến truy cập và kết nối mạng từ bên ngoài vào.
    - **Mục tiêu:** Giao diện Bắc-Nam nhằm kiểm soát và quản lý các kết nối, luồng dữ liệu, và chính sách mạng đi vào và ra khỏi hệ thống SDN. Nó quản lý các tài nguyên mạng và tương tác với các ứng dụng và dịch vụ bên ngoài mạng SDN.
5. **Giao diện Đông-Tây (East-West Interface):**
    - **Hướng:** Giao diện Đông-Tây đề cập đến giao tiếp giữa các thành phần hoặc lĩnh vực khác nhau trong cùng một hệ thống SDN hoặc giữa các hệ thống SDN khác nhau, tức là từ "đông" đến "tây" trong mạng, liên quan đến giao tiếp nội bộ trong hệ thống.
        1. **Khía cạnh Đông-Tây trong SDN:** Khi người ta nói về hướng Đông-Tây trong SDN, họ thường đang thảo luận về việc trao đổi dữ liệu hoặc thông tin giữa các phần khác nhau của mạng SDN. Điều này có thể liên quan đến việc chia sẻ trạng thái mạng giữa các miền SDN khác nhau hoặc giữa các thành phần của một miền SDN cụ thể.
            a. **Phía Đông (East):** Trong ngữ cảnh mạng SDN, phía Đông thường biểu thị các phần của mạng chịu trách nhiệm về việc gửi dữ liệu hoặc yêu cầu dịch vụ. Nó liên quan đến các thiết bị hoặc ứng dụng mạng mà gửi yêu cầu và dữ liệu đến mạng để được xử lý hoặc đáp ứng. Phía Đông thường chịu trách nhiệm về việc tạo dữ liệu và yêu cầu trong mạng.
            b. **Phía Tây (West):** Phía Tây thường biểu tượng cho các phần của mạng chịu trách nhiệm về việc xử lý yêu cầu và dữ liệu từ phía Đông. Nó liên quan đến các thiết bị hoặc ứng dụng mạng

mà thực hiện các chức năng xử lý, quản lý tài nguyên, và cung cấp dịch vụ dựa trên yêu cầu từ phía Đông. Phía Tây thường chịu trách nhiệm về việc xử lý và cung cấp dịch vụ trong mạng.

Cụ thể, trong mô hình mạng SDN, phía Đông thường đề cập đến các SDN switches, endpoints, hoặc các thành phần khác mà gửi yêu cầu và dữ liệu đến mạng SDN. Phía Tây thường ám chỉ các SDN controllers và các thành phần quản lý mạng mà thực hiện xử lý yêu cầu từ phía Đông và quản lý tài nguyên mạng.

2. **Khía cạnh Bắc-Nam trong SDN:** Khi người ta nói về hướng Bắc-Nam trong SDN, họ thường đang thảo luận về việc quản lý và triển khai SDN trong hệ thống mạng. Điều này có thể liên quan đến việc xác định các nguồn tài nguyên, quyền truy cập và chính sách cho các thiết bị mạng hoặc ứng dụng SDN.

- **Mục tiêu:** Giao diện Đông-Tây nhằm kiểm soát và quản lý trạng thái mạng, thông tin về tài nguyên, luồng dữ liệu, sự kiện và cập nhật giữa các thành phần hoặc lĩnh vực SDN khác nhau. Nó cung cấp tính linh hoạt và tích hợp trong hệ thống SDN và cho phép các lĩnh vực khác nhau hoạt động cùng nhau một cách hiệu quả.

6. QoS (Quality of Service) metrics là các chỉ số và đánh giá được sử dụng để đo lường và đảm bảo chất lượng của các dịch vụ mạng và ứng dụng. Các QoS metrics quan trọng trong mạng bao gồm:

- **Bandwidth (Băng thông):** Là lượng dữ liệu có thể truyền qua mạng trong một khoảng thời gian cụ thể. Đo lường bằng đơn vị bits per second (bps) hoặc các đơn vị lớn hơn như megabits per second (Mbps) hoặc gigabits per second (Gbps). Đối với các ứng dụng yêu cầu nhiều dữ liệu, băng thông cao là cần thiết.
- **Latency (Độ trễ):** Là thời gian mất đi từ khi gửi một gói tin đến khi nó đến đích. Đo lường bằng đơn vị thời gian như milliseconds (ms). Độ trễ thấp là quan trọng cho các ứng dụng như video gọi và trò chơi trực tuyến.
- **Packet Loss (Tỷ lệ mất gói tin):** Là tỷ lệ gói tin bị mất trong quá trình truyền dữ liệu. Đo lường bằng phần trăm. Tỷ lệ mất gói tin cao có thể làm giảm chất lượng của ứng dụng.
- **Jitter (Độ biến động thời gian):** Là sự biến động trong thời gian mất đi của gói tin. Đo lường bằng thời gian, thường là milliseconds. Độ biến động thời gian cao có thể gây ra các vấn đề trong các ứng dụng yêu cầu đồng bộ hóa như cuộc gọi video.
- **Reliability (Độ tin cậy):** Liên quan đến khả năng mạng hoạt động mà không bị lỗi hoặc gián đoạn trong một khoảng thời gian dài. Đo lường bằng phần trăm hoặc thời gian trung bình giữa các sự cố.
- **Availability (Khả dụng):** Liên quan đến thời gian mạng hoạt động và có sẵn cho việc sử dụng. Đo lường bằng phần trăm thời gian hoạt động trong một khoảng thời gian cụ thể.
- **Throughput (Lưu lượng):** Là lượng dữ liệu thực tế có thể truyền qua mạng trong một khoảng thời gian cụ thể. Tùy thuộc vào mô hình truyền thông, lưu lượng có thể thấp hơn băng thông do các yếu tố như độ trễ và mất gói tin.
- **Packet Delay Variation (PDV):** Đo lường sự biến đổi trong thời gian mất đi của các gói tin. Được sử dụng trong các ứng dụng yêu cầu đồng bộ hóa như âm thanh và video streaming.
- **Blocking Probability (Xác suất chặn):** Đo lường xác suất một cuộc gọi hoặc yêu cầu truyền dữ liệu bị từ chối do tài nguyên không đủ.

7. SARS (SDN Application for Relationship Analysis and Resource Selection) là một ứng dụng trong mạng SDN (Software-Defined Networking) được thiết kế để thực hiện các nhiệm vụ sau:

- **Phân tích Mối quan hệ Lưu lượng Mạng:** SARS được sử dụng để phân tích mối quan hệ giữa các luồng lưu lượng mạng trong mạng SDN. Nó có khả năng xem xét cách các luồng giao tiếp với nhau và làm thế nào chúng ảnh hưởng đến nhau. Việc này có thể giúp hiểu rõ hơn về tình hình mạng và cách các dịch vụ và ứng dụng tương tác với nhau.
- **Lựa chọn Tài nguyên Máy chủ:** SARS cung cấp khả năng lựa chọn tài nguyên máy chủ dựa trên thông tin phân tích. Nó có thể quyết định tài nguyên máy chủ nào là phù hợp nhất để phục vụ các yêu cầu từ các ứng dụng hoặc dịch vụ cụ thể. Điều này có thể bao gồm lựa chọn máy chủ có hiệu năng cao, có sẵn tài nguyên đủ, hoặc có độ trễ thấp nhất để đáp ứng nhu cầu của ứng dụng.
- **Định tuyến Mạng:** SARS có thể giúp quyết định định tuyến lưu lượng mạng dựa trên thông tin phân tích. Nó có thể xác định đường dẫn tối ưu cho các gói tin dựa trên các yêu cầu cụ thể của ứng dụng hoặc dịch vụ, chẳng hạn như đảm bảo chất lượng dịch vụ (QoS) hoặc tối ưu hóa hiệu suất mạng.

⇒ SARS là một ứng dụng quan trọng trong mạng SDN giúp cải thiện hiệu suất mạng, đảm bảo chất lượng dịch vụ và tối ưu hóa sử dụng tài nguyên bằng cách phân tích mối quan hệ của lưu lượng mạng và đưa ra quyết định về lựa chọn tài nguyên và định tuyến.

1. INTRODUCTION

A. Software-defined Network (SDN) network model

- Ability to re-route networks on the fly
- Offering realtime visibility to the whole system
- Providing the real-time ability to automatically re-route or to stand-up new functions and routes without adding any new hardware.

B. Server and route selection (SARS)

- Common: distributed architecture with multiple domains, each controlled by an SDN controller. → Implementation: Server and route selection (SARS)
- Each domain has only its network state information >< SARS mechanism effective if one domain has the network state information of others. ⇒ need for cross-domain communication.
- **Knowledge-Defined Network framework (KDN):** use East-West interface SINA (control plane) ⇒ allow controllers to exchange network state information and guarantee consistency of network state information across SDN domains.
- **SARS:** not efficient based solely on the current state of the network.
  - Path chosen in current state won't be good solution for future states.
- **Traditional Open Shortest Path First (OSPF)** often selects paths based on min hop counts.
- **SDPredictNet with a neural network**: predict future network states for routing, only implemented on small-scale intra-SDN domains.

C. Contribution

- KDN architecture deployed with three planes: the data plane, the control plane & knowledge plane.
- **SINA in control plane:** guarantee network state consistency, allow network state sharing between different SDN controllers (provide a global view)

- **SINA in top of knowledge plane:** SARS mechanism is implemented after having a global view of the network
- **Long Short-Term Memory Network (LSTM) in KDN:** optimize our SARS.
  - **LSTM:** discover the non-linear nature and uncertainty of traffic flows.
  - Predicted link cost consists of several network metrics: delay, packet loss, link overhead, and link utilization.

2. RELATED WORK

A. SDPredictNet

- Framework including LSTM and artificial neural network (ANN)
- Aim of using LSTM and ANN: help update the flow table for routing, with the delay being reduced after recognizing congested areas in the networks.
- Traffic features: captured by the LSTM model, then fed into the ANN model to recommend path for future packets at the given instant.
- Iperf tool: generate data and obtain traffic parameters
- Given (n–1) packet parameters, LSTM attempts to forecast the nth packet's parameters.
- ANN tries to map a sequence of predicted packets' parameters to the output switches with two values (0 or 1). → Final: sequence of switches labeled as 1

B. Two-phase routing mechanism

- Predict link quality in an SDN-based ad hoc network.
- First phase: XGBoost - library uses an ensemble of decision trees to generate predictive models (used for leaf node score's optimization of each decision tree)
- Second phase: minimum cost tree method generates routes with the highest packet delivery ratio (PDR).

C. QoS in SDN-based mobile ad hoc network

- Routing optimization problem to satisfy user's QoS in the SDN-based mobile ad hoc network
- First stage: wavelet neural network (WNN) - forecast the following timestamp link quality values.
- Second stage: routing problem formulated as the binary knapsack problem → differential search (DS) (meta-heuristic optimization algorithm) used to generate feasible solutions.

1) Wavelet Neural Network (WNN)

- **Wavelet:** Là một phương pháp trong xử lý tín hiệu và học máy, dựa trên việc sử dụng các hàm sóng (wavelet) để biểu diễn tín hiệu. Các hàm sóng cho phép phân tách tín hiệu thành các thành phần tần số khác nhau và có khả năng phát hiện cấu trúc tín hiệu ẩn.
- **Neural Network:** Là một mô hình học máy được lấy cảm hứng từ cách hoạt động của mạng thần kinh trong não người. Mạng nơ-ron nhân tạo bao gồm các lớp nơ-ron kết nối với nhau và được sử dụng để học các mô hình từ dữ liệu.

Wavelet Neural Network (WNN) là sự kết hợp giữa cả hai khái niệm trên. Nó sử dụng các hàm sóng để biểu diễn dữ liệu và sử dụng mạng nơ-ron để học và dự đoán các mô hình từ dữ liệu. Trong ngữ cảnh công trình nghiên cứu được đề cập, WNN được sử dụng để dự đoán giá trị chất lượng liên kết (link quality values) trong mạng di động dựa trên SDN.

2) SDN-based Mobile Ad Hoc Network (Mạng di động không cơ địa SDN)

- **SDN (Software-Defined Networking):** Là một mô hình mạng trong đó việc quản lý và kiểm soát mạng được tách biệt từ thiết bị vật lý bằng phần mềm. SDN cho phép quản trị mạng tương tác với và điều khiển các thiết bị mạng từ một vị trí trung tâm, tạo ra tính linh hoạt và quản lý hiệu quả.
- **Mobile Ad Hoc Network (Mạng di động không cơ địa):** Là một loại mạng không cần sự triển khai cơ địa của cơ sở hạ tầng mạng cố định. Trong mạng này, các thiết bị di động có thể tự kết nối và tạo mạng tạm thời mà không cần một hạ tầng cố định. Điều này thường được sử dụng trong các tình huống như tình huống khẩn cấp hoặc trong các môi trường không có cơ sở hạ tầng mạng.

Mạng SDN-based Mobile Ad Hoc Network là một mô hình mạng di động không cơ địa mà việc quản lý và kiểm soát mạng được thực hiện bằng cách sử dụng SDN, tách biệt quản lý và kiểm soát mạng từ các thiết bị di động tạo thành mạng tạm thời.

2) Binary Knapsack Problem (Bài toán cái túi nhị phân)

- Đây là một bài toán tối ưu hóa phổ biến trong lĩnh vực tối ưu hóa. Trong bài toán này, bạn có một tập hữu hạn các mục hàng, mỗi mục có giá trị và trọng lượng riêng. Nhiệm vụ là chọn một tập con của các mục hàng sao cho tổng trọng lượng của chúng không vượt quá một ngưỡng (cái túi) và tổng giá trị của các mục hàng này là lớn nhất có thể. Mỗi mục hàng có thể được chọn hoặc bỏ (nhị phân).
- Ví dụ, trong bài toán định tuyến mạng, các "mục hàng" có thể biểu thị các tuyến đường hoặc tùy chọn định tuyến khác nhau với giá trị (quality of service, QoS) và "trọng lượng" (chi phí hoặc khả năng của đường truyền).

Bài toán cái túi nhị phân thường được sử dụng để tối ưu hóa việc chọn lọc các tùy chọn hoặc tài nguyên để đáp ứng một hạn chế về trọng lượng hoặc dung lượng.

D. AIER (Artificial Intelligence Enabled Routing)

- **ANN network:** predict traffic congestion before making route decisions.
- **AIER:** solve the inadequacy of learning capacity from precedent experiences to prevent inefficient route selection by the dynamic routing mechanism of the SDN controller.
- **Three primary phases in AIER:**
  - AIER collects the training data, traffic flows in the network, and its label (1 - congestion or 0 - otherwise)
  - ANN model trained before being deployed on the control plane
  - AIER can choose an appropriate path to avoid congestion using the Dijkstra algorithm.

E. NeuRoute

- Neural network: optimize the network throughput. → Outperform the traditional routing algorithm in the SDN controller in terms of computational complexity
- Every round: uses the predicted link loads as input before performing the graph search to find optimal paths
- Two main core blocks: first block to predict traffic matrix (LSTM) & second block for traffic routing (feed-forward network)

F. SDVN architecture (Software-defined Vehicular Networks)

- Neuron network: model the traffic flows efficiently in urban areas.
- LSTM network: capture and learn the non-linear nature of the traffic flow.

- SDN into the cloud: optimize computational expenses and storage efficiencies.
- Comprise two significant phases: detecting the congestion-sensitive spots (K-means algorithm) and forecasting traffic congestion trends (LSTM network)

H. NeuTM

- Framework using deep neural network to predict the traffic matrix
- Suitable for learning from data and classifying time series with time intervals of unknown size.
- Traffic matrix: two-dimensional matrix where its (i, j) elements capture the amount of traffic transmitted from node i to node j in a network.
- NeuTM in a GEANT topology, including 23 nodes and 38 links with a POX controller: model's accuracy and prediction measurements outperform traditional methods with better learning of non-linear patterns.

3. IMPLEMENTATION OF KNOWLEDGE-DEFINED NETWORK

- KDN aims to make the distributed large-scale network system more autonomous and smarter with the ability to share knowledge across multi-domains.
- Three layers: the data plane, the control plane, and the knowledge plane. Communication between these layers performed via Northbound and Southbound API.
    - **Data plane** comprises OpenFlow-enable switches and other network devices to accomplish instructions from the control plane (e.g., forwarding, dropping packets, etc.).
    - **Control plane** includes multiple heterogeneous distributed controllers. (ONOS & RYU)
        - ONOS (Open Network Operating System): JAVA, distributed system with high scalability.
        - RYU: open-source SDN controller by Python, implement various SDN applications/ networks
    - Knowledge layer gets all distributed knowledge from multiple SDN domains into storage (or database) before being processed for various applications and services.

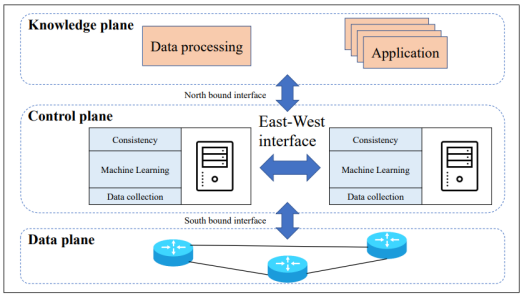


Figure 1: Knowledge-defined heterogeneous architecture

- Multi-threading system like SDN, risk of data leakage when SDN controllers exchange their states: SDN controller should retain its data and exchange its knowledge (ML model produces)
- **East-West interface SINA:** communication bridge, guarantee consistency with two main components(REST API modules and the listener)
    - Listener: listen to the local knowledge generated from a machine learning model before notifying them of other SDN domains

- REST API modules: enables other SDN controllers to update the propagated knowledge based on the REST API platform.

- Consistency mechanism: ensure that the knowledge of distributed SDNs consistent.
  - Includes two major parts: quorum replication and reinforcement-learning techniques.

> 💡 $N_w + N_r > N$ satisfied, this is called the strong consistency mechanism.

  - Reinforcement learning algorithm: learns a set of actions (read and write operations from getting feedback from the environment, our consistency mechanism can adaptively
    select Nw and Nr, ensuring $N_w + N_r \leq N$, and maintaining the balance between controllers communication overhead and the network state consistency.

1) Quorum Replication (Sao chép theo quyền)

1. **Overhead Giao Tiếp:** Mỗi lần một bộ điều khiển gửi hoặc nhận thông tin từ các bộ điều khiển khác, nó phải sử dụng tài nguyên mạng và thời gian để thực hiện các hoạt động giao tiếp. Khi bạn tăng số lượng Nw và Nr (số lần ghi và đọc), điều này có thể dẫn đến một lượng lớn hoạt động giao tiếp, gây ra overhead tăng lên. Overhead quá lớn có thể làm giảm hiệu suất của mạng và làm tăng thời gian phản hồi.
2. **Tính Nhất Quán của Trạng Thái Mạng:** Tính nhất quán là quan trọng để đảm bảo rằng tất cả các bộ điều khiển trong mạng SDN hiểu và sử dụng cùng một thông tin và trạng thái mạng. Tuy nhiên, tính nhất quán không nhất thiết phải đạt được thông qua một số lượng lớn hoạt động giao tiếp. Nếu bạn thiết lập Nw + Nr quá lớn, điều này có thể dẫn đến sự lãng phí về tài nguyên và thời gian giao tiếp mà không cần thiết.

Vì vậy, để đảm bảo rằng mạng SDN vẫn duy trì tính nhất quán trong trạng thái mạng mà không tạo ra quá nhiều overhead giao tiếp, cần phải thiết lập một giới hạn hợp lý cho Nw + Nr sao cho nó không vượt quá tổng số bộ điều khiển (N). Điều này giúp đảm bảo rằng mạng vẫn hoạt động hiệu quả và đáp ứng được yêu cầu của người dùng về tính nhất quán và hiệu suất.

2) SARS application

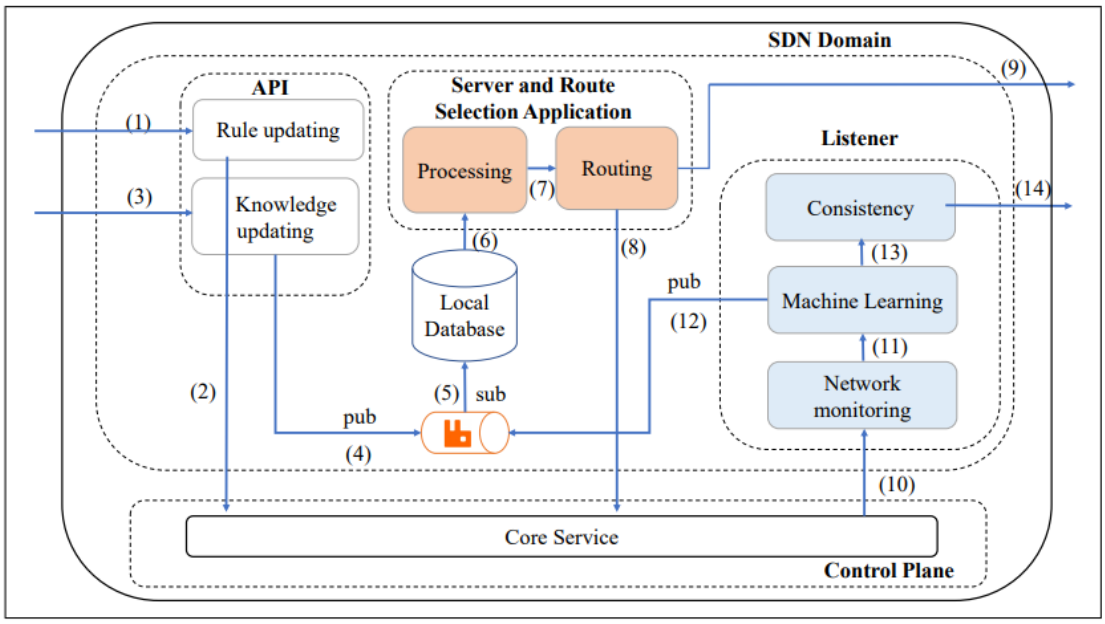


Figure 2: The data flow of the SARS application

- Forwarding rules from other SDN domains are sent to the rule updating module of the current domain before they are passed into the core service (flow 1, 2).
  - Điều này bao gồm việc xử lý và cập nhật luật để đảm bảo tính nhất quán trong cấu hình mạng.
- Current domain also received the network data (QoS metrics) from other domains via the knowledge updating module before these metrics are stored in the local database via a queue message (flow 3, 4, 5).
- Data retreived by the SARS application for analyzing the relationship of the traffic and making the server and routing selection (flow 6, 7).
- Result: transferred into the core service and other SDN domains' API (flow 8, 9)
- Listener component: include network monitoring modules - captures the QoS metrics sent by the controller every 3 seconds (flow 10) , then machine learning - discovers these QoS parameters for finding valuable knowledge and features (flow 11), and consistency module - retreived knowledge sent to the message queue (flow 12) or to the consistency module, where local knowledge is broadcasted to other SDN domains.
- N = {ni|i = 1, 2, ..., n} as the set ofdomains, C = {ci|i = 1, 2, ..., n} as the set of controllers. When traffic and services are
- Traffic&services requested from hosts to servers, network data generated from these connections & collected by each domain controller.
- QoS data stored in local database & analysed before boardcast among N-1 other domains.
- SARS application combines mutiple knowledge from individual domain: make server & routing decisions ⇒ optimize network performance.

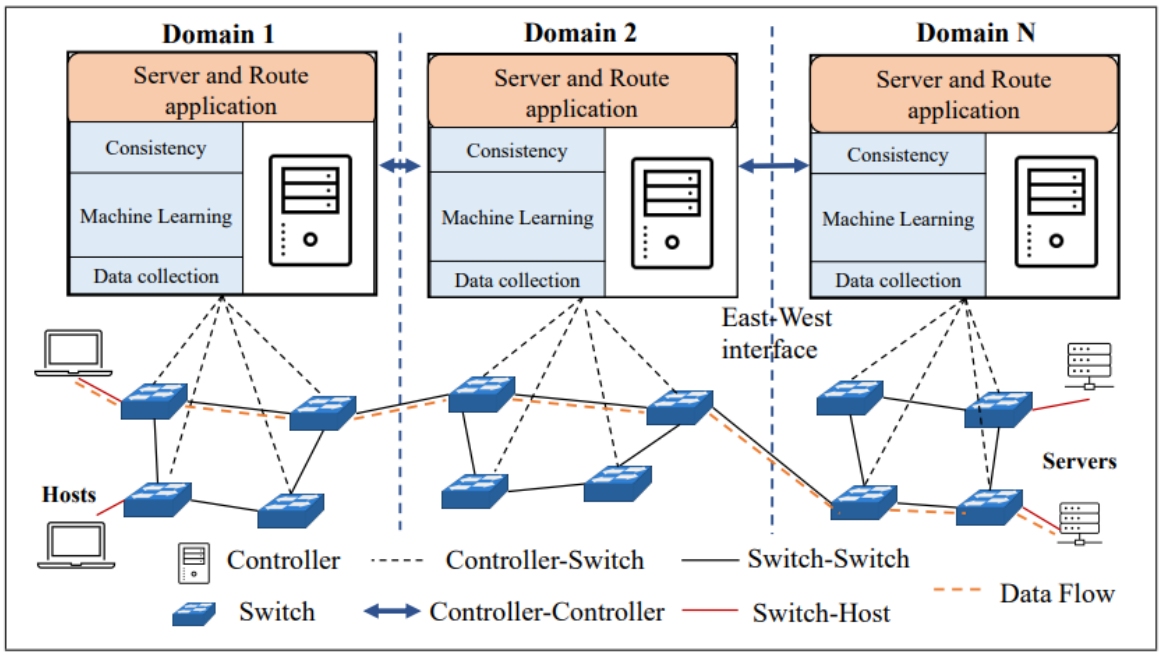


Figure 3: Inter-SDN domains network with KDN architecture

4. SERVER AND ROUTE SELECTION FRAMEWORK

4.1. Problem formalization

- Network topology: undirected weighted graph G=(V,E)
- $G = G_1 \cup G_2 \cup G_3 \cup ... \cup G_l$ with G_i is the graph of ith domain in the whole distributed network.
- $V = v_1, v_2, \ldots, v_n$ is the set of switches.
- $E = e_1, e_2, \ldots, e_m$ is the set of links with each link being assigned a link cost LC_i
- $H = h_1, h_2, \ldots, h_u$ is the set of hosts.
- $S = s_1, s_2, \ldots, s_v$ is the set of servers with each server being assigned a server cost SCi.
- **Input**: host $h_i \in H$, switch $v_i \in V$, link $e_j \in E, (i = 1, 2, ...n), j = (1, 2, ..., m)$
- **Output**: a server s and a path $p = (v_1, v_2), (v_2, v_3), \ldots (v_{p-1}, v_p)$ with $v_1 = h \& v_p = s$

> 💡 Goal: trade-off between the server cost (SC) and the path cost (PC).

- α and β are proportional coefficients and Ps,h is all available paths from host s to server d.
- path p^*: minimize the total cost function (TC)

$$TC(SC, PC) = \alpha \cdot SC + \beta \cdot PC$$
$$p^* = \min_p \{TC | p \in P_{s,h}\},$$

- F: set of unidirectional flows
- x_{ij}:binary coefficient, equal to 1 if a path is going through the link (i, j) and equal to 0 otherwise

$$x_{ij} = \begin{cases} 1\ if\ flow\ k\ uses\ link(i,j) \\ 0\ otherwise. \end{cases}$$

- β_k is the bandwidth of flow k
- b_{ij}: available bandwidth between switch i and j, σ_k is the maximum requested cost of flow k
- LC_{ij}: cost of the link between switch i and j.

$$\sum_{k \in F} x_{ij}\beta_k \leq b_{ij}$$
$$\sum_{(i,j) \in E} x_{ij} LC_{ij} \leq \sigma_k$$

4.2. Link cost prediction

- LSTM-based link cost prediction model (LCP)
- Four stages: data collection, preprocessing, construction, and model deployment (SARS mechanism)
- Data collection: network traffic obtained once every three seconds (data layer)
  - Four selected features of the SARS problem: delay, link utilization, link overhead, and packet loss rate.
- Data preprocessing: uses various techniques such as filling in missing values (mean interpolation method) or data normalization (map the QoS data range to the range [0, 1] ⇒ reduce the impact of imbalanced data units (too large or too small).
- Construction: LCP model is constructed by the LSTM networks and fully connected neural networks (FNN).
  - LSTMs will learn the long-term non-linear relationship between our target-link cost values and time-series QoS metrics.
  - FNNs apply various linear transformations to learn the matrix weights that can predict future link cost values

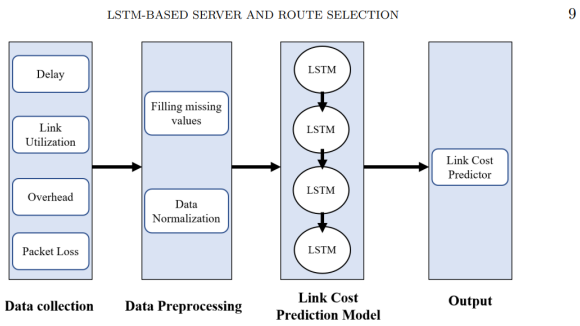


Figure 4: The process of building pretrained LSTM-based link cost prediction model

4.3. Server and route selection mechanism

- **SARS algorithm:** balancing link traffic distribution & distributing the load equally among the servers.
  - **Path cost:** LC values are predicted by our LCP models (sum of all LC values of the crossed links over the path p)

$$PC(p) = \sum_{i \in p} LC_i$$

- **Server cost:** ST - current server traffic, SB - server bandwidth capacity

$$SC(s) = \frac{ST}{SB}.$$

- Proposed total cost TC can be reached by taking a linear combination (weighted sum) of the SC and PC
  - Sum of α and β equals one

$$TC(SC, PC) = \alpha \cdot SC + \beta \cdot PC.$$

- First loop: New server connection is requested, the application pulls these predicted values from a local database with a global view of all knowledge among multiple SDNs using the ACM
- Second loop: iterates over servers and finds each server's shortest path by calculating its path cost and total cost

Algorithm 1 QoS-based SARS algorithm

Input: Graph $G = (V, E)$, set N of SDN domains, set C of controllers, set DB of databases, set M of LCP models
Output: Server $s \in S$ and Path $p = \{(v_1, v_2), (v_2, v_3), ...(v_{p-1}, v_p)\}$

```
In every time period:
for each SDN domain in N domains do
    Predict link cost values using the LCP model
    Compute SC values using (6)
end for
if (a new server connection is requested) then
    Get predicted link costs from the set of local databases using a consistency mechanism
    for each server s ∈ S do
        Compute PC values using (5)
        Compute TC value using (7)
    end for
    Choose path p, server s that minimizes TC value
    for each controller c ∈ C do
        Insert forwarding rules to OpenFlow switches for the selected path
    end for
end if
```

5. LSTM-BASED LINK COST PREDICTION MECHANISM (Cơ chế Dự đoán Chi phí Kết nối dựa trên LSTM)

- LSTM-based link cost prediction mechanism: predict link cost values for the SARS mechanism.

5.1. Data collection

- Collect the network features generated from the traffic flows of communication between hosts and servers in the SDN inter-domains.
- API PortStatistics: extract delay (DL), link utilization (LU), packet loss rates (PLR) and link overhead (LO) in each SDN domain ⇒ Round Robin algorithm
- Having collected data, we can consider a training set $D = (x_1, y_1), \ldots, (x_T, y_T)$ consisting of T inputs $x_t \in R^4$ and corresponding link cost $y_t \in R, t = 1, 2, ..., T$. Each x_t can be treated as a 4-dimensional feature vector of an edge e_{i-j} at time t.

$$LO = \frac{(ByteSent + ByteReceived) - ByteThreshold}{ByteThreshold}$$

- The relationship between the feature vector xt and the link cost yt is defined as

$$yt = x_{Delay_t}\alpha_{Delay} + x_{LinkUtilization_t} \cdot \alpha_{LinkUtilization} + x_{PacketLossRate_t} \cdot \alpha_{PacketLossRate} + x_{LinkOverHead_t} \cdot \alpha_{LinkOverHead}$$

- where the sum of$\alpha_{Delay}, \alpha_{LinkUtilization}, \alpha_{PacketLossRate}, \alpha_{LinkOverHead}$ is equal to 1.

5.2. Data preprocessing (Tiền xử lý dữ liệu)

- QoS parameters are captured at the data link layer to build the real dataset
- Noise values generated by the monitoring tools (e.g., link layer discovery protocol) are removed from each of the 18 datasets

$$X\hat{} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

5.3. LSTM-based link cost prediction (Dự đoán chi phí kết nối dựa trên LSTM)

- Two main phases of the LCP module: the training and testing phases.
- LCP module takes collected features from the previous data collection and data preprocessing modules as input to the deep learning algorithm.
- Each dataset in each domain is then split into three sets: the training, the validation, and the testing sets, with a percentage of 70%, 15%, and 15%, respectively.

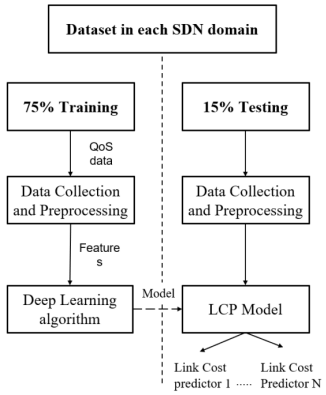


Figure 5: The deep learning-based link cost prediction method

- LCP model, two essential layers:
  - **LSTM layer:** There are three LSTM layers in this architecture. The first one receives the input as a 5 × 4 feature vector where the first dimension is the number of time steps, and the second dimension is the number of features (DL, LU, PLR, and OH). It is worth mentioning that the time steps is selected after the overall performance of the LCP model is assessed. The output of this layer is a 5 × 128-dimensional vector (or time steps × LSTM neurons dimensional space). In the second LSTM layer, the 5 × 128-dimensional vector from the previous step is regarded as an

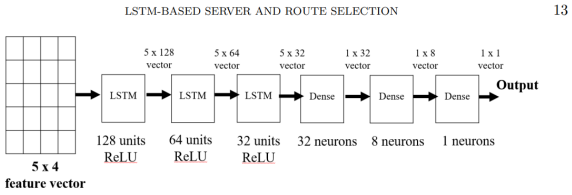


Figure 6: Architecture of the deep learning-based link cost prediction model

input, generating a 5 × 64-dimensional vector. Likewise, the third LSTM layer gets the 5 × 64-dimensional vector and produces a 5 × 32-dimensional vector as an output.

- **ReLU activation function:** Each output vector from each LSTM layer is passed into a ReLU activation function. The ReLU maps every element of a 5 × n-dimensional vector into new values:

$$f(x) = \begin{cases} 1 \; if \; x < 0 \\ x \; otherwise. \end{cases}$$

- Dense layer: There are three dense layers (or fully connected layers) with 32, 8, and 1 neurons

6. EXPERIMENTAL RESULTS

6.1. Experimental setup

- Distributed system: one master & 18 slave machines.
- 18 slave machines: 8 RYU controllers & 10 ONOS controllers in charge of collecting network states, predicting LCs, and exchanging predicted values with one another.
- Mininet: emulate the network topology

Dynamic network	Parameter
Link	Random Delay [25, 50, 75, 100, 125]ms
Switch	Random Loss [0.1, 1, 2, 5]%
Bandwidth	100MB
Traffic generation tool	Simple HTTP server
Traffic range	[10, 60] MB

Table 3: Dynamic network configurations

6.2. Tuning hyperparameters of the link cost prediction model

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(y\tilde{}t - yt)^2.$$

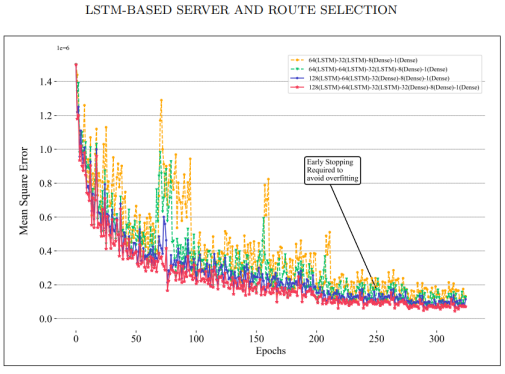


Figure 7: Effect of LSTM architecture on convergence in learning

6.3. Benchmarks

- **Techniques:** Round robin with strong consistency (RR-WSC), QoS-based algorithm with strong consistency (QSARSA-WSC), link cost prediction with strong consistency (LCP-WSC), and QoS-based algorithm with adaptive consistency (QSARSA-WAC).

6.3.1. Benchmarks with strong consistency

- **RR-WSC:** RR-WSC benchmark selects servers following a circular queue, fixed with two parameters Nr and Nw as 1 and 18
- **QSARSA-WSC:** similar to RR-WSC, difference is selects paths using Dijkstra's algorithm with real-time link cost values
- **LCP-WSC:** works on LSTM models that are already trained to predict the link cost values but uses a strong consistency mechanism.

6.3.2. Benchmark with adaptive consistency

- **QSARSA-WAC:** same path selecion mechanism with the second benchmark QSARSA-WSC, but takes advantage of an adaptive consistency mechanism with Nr and Nw being adaptively updated according to the state of a network.

6.4. Performance analysis

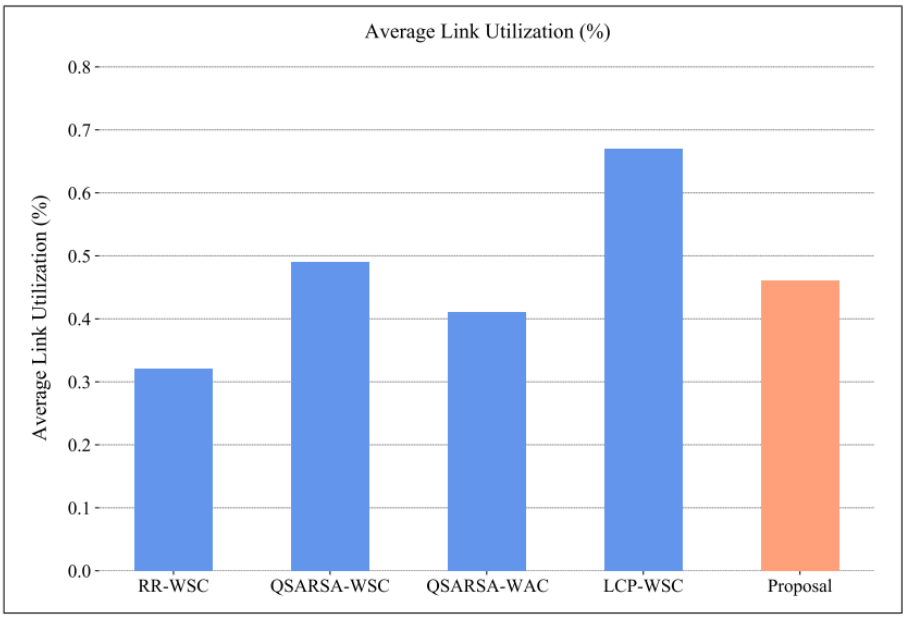


Figure 8: Comparison of Average Link Utilization between the proposal and benchmark methods

6.4.1. Comparing the QoS performance

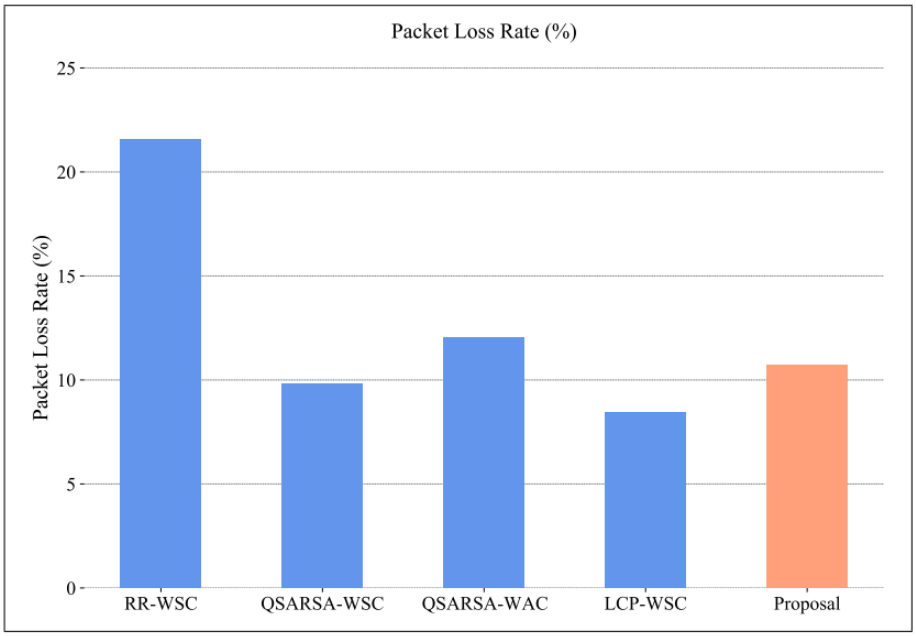


Figure 9: Comparison of Packet Loss Rate (%) between the proposal and benchmark methods

6.4.2. Comparing the response time and overhead

- SLA threshold is set up at approximately 170,000 ns: upper bound to estimate the servers' response time

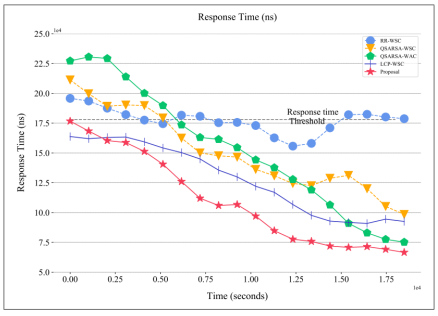


Figure 10: Comparison of Response Time between the proposal and benchmark methods

Table 4: Comparison of average response time and average link overhead between proposal and benchmarks.

Benchmark	Response time (ns)	Improvement of response time	Overhead	Improvement of overhead
RR-WSC	177,282	38%	0.70	75%
QSARSA-WSC	150,737	27%	0.40	57%
QSARSA-WAC	152,897	28%	0.35	51%
LCP-WSC	128,000	14%	0.23	25%
Our proposal	**110,089**	-	**0.17**	-

7. CONCLUSIONS

- LSTM-based approach for selecting servers and routes in a distributed SDN network to enhance long-term network operation with high QoS.
- Approach: addresses the single point of failure issue in intra-SDN domains by introducing KDN.
- Method: exploits link cost values predicted by Long Short-Term Memory network (LSTM) to collaborate and share predicted knowledge among multiple domains. ⇒ utilize links and prevent bottlenecks
- Limitations:
  - **Imbalanced dataset situation:** sample size in one domain significantly larger than in other domains, prevent the model from learning knowledge across multiple network domains. ⇒ explore Transfer Learning (TL)
  - **Route and selection mechanism:** not adaptive in complex environments because routing decisions cannot be updated if unexpected congestion occurs during the routing process. ⇒ study various evolutionary algorithms (e.g., genetic, ant-colony techniques, etc.) with multi-intelligent agents

RESEARCH AND NOTES

Bài báo (survey):

https://husteduvn-my.sharepoint.com/:b:/g/personal/trang_lh205234_sis_hust_edu_vn/EQNA13UuF3tPj5RItI96fYsBF1QGxdz9641wc4UvQ0eu6Q?e=rw7u87

Mo phong thuc hanh :

Setting Up ONOS Controller with Mininet

Hello there! My name is Yu-Wei Chang, and you may call me ernie. I see programming as a hobby so I would spend some effort

https://ernie55ernie.github.io/sdn/2020/07/21/setting-up-onos-controller-with-mininet.html

Mô phỏng SDN bằng Mininet và ONOS - Phần 1: Cài đặt môi trường và công cụ

Tổng quan về SDN SDN hay mạng điều khiển bằng phần mềm (Software Defined Networking) được dựa trên cơ chế tách riêng việc kiểm soát một luồng mạng với luồng dữ liệu (control plane và data plane). SDN...

https://viblo.asia/p/mo-phong-sdn-bang-mininet-va-onos-phan-1-cai-dat-moi-truong-va-cong-cu-XL6lAoMNKek

SDN-IP Tutorial - ONOS - Wiki

Have questions? Stuck? Please check our FAQ for some common q… …swers.

https://wiki.onosproject.org/display/ONOS/SDN-IP+Tutorial

GitHub - ArnoTroch/ONOS-Tutorial: Tutorial on how to use ONOS as a robust SDN controller with mininet

Tutorial on how to use ONOS as a robust SDN controller with mininet - GitHub - ArnoTroch/ONOS-Tutorial: Tutorial on how to use ONOS as a robust SDN controller with mininet

https://github.com/ArnoTroch/ONOS-Tutorial#accessing-the-gui

Mininet:

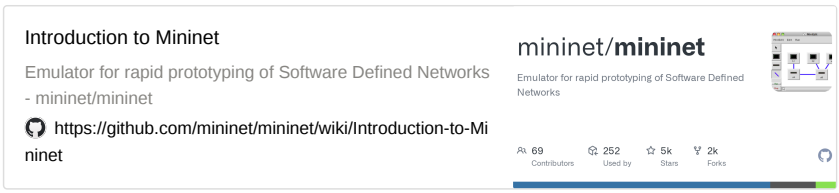
sudo mn --switch=ovs,protocols=OpenFlow14 --controller=remote,ip=127.0.0.1 --custom mininet-demo.py --topo tp

https://github.com/mininet/mininet/releases/

Mininet Walkthrough - Mininet

Mininet Walkthrough This walkthrough demonstrates most Mininet commands, as well as its typical usage in concert with the Wireshark dissector. The …

http://mininet.org/walkthrough/#part-1-everyday-mininet-usage



FEEDBACKS

Gợi ý của thầy

- He thong SDN: S-Computing Network (mang bien) ⇒ Domain tham gia: Access network, Cac cum may chu
- 1 controller dieu khien 1 domain
- Build ma nguon onos: can thiep va ung dung cua onos

Công việc hoàn thiện

- [x] ~~Cài đặt mininet: Giả lập một mạng thiết lập giữa mininet với onos~~
- [x] ~~Đọc bài báo về triển khai mạng, cách định tuyến routing~~
- [ ] Cach lam chu ma nguon onos
- [ ] Xay dung 1 ung dung trong onos: Dung mininet thuc hien mang nhieu nut, chia domain tung domain co 1 controller (onos)

Thắc mắc

- Tại sao Nw + Nr sao cho nó không vượt quá tổng số bộ điều khiển (N) vẫn đảm bảo tính nhất quán (Nw + Nr > N mới đảm bảo tính nhất quán mạnh)