Project Report
AI-Based Traffic Matrix Prediction for SDN

Submitted by -

Sanskar Bharadia **2024H1030068P**

Aarnav JP **2024H1030075P**

Amir Faizal S **2024H1030073P**

Submitted to -

Virendra Singh Shekhawat

Course Instructor - Advance Computer Network (CS G525)

Index

List of Figures

Abstract

This project focuses on implementing an AI-based Traffic Matrix (TM) prediction system within an SDN environment using the Ryu controller. The work includes the development of a testbed to capture network traffic, generation of realistic traffic patterns, and deployment of recurrent neural network models for time-series traffic prediction. The ultimate goal is to evaluate the predictive accuracy of Long Short-Term Memory (LSTM) networks and their variants, Bidirectional LSTM (BiLSTM) and Gated Recurrent Unit (GRU).

---

1. Introduction

Traffic Matrix (TM) represents the volume of traffic flowing between all source-destination pairs in a network. Accurate TM prediction is crucial for effective network management, enabling tasks such as:

- Dynamic routing to avoid congestion.
- Energy-efficient resource allocation.
- Anomaly detection and Quality of Service (QoS) improvements.

This project leverages the centralized architecture of SDN to capture TM data and uses AI models to predict future traffic patterns, addressing challenges in traditional networks where distributed control limits the availability of traffic data.

2. Problem Statement

The problem addressed in the paper is the challenge of accurately predicting the Traffic Matrix (TM) in Software-Defined Networks (SDNs) to enhance network management tasks like traffic engineering and anomaly detection. Traditional networks struggle with decentralized control and limited local views, leading to inefficient TM predictions. This study applies AI-based models, specifically RNN variants, to improve TM prediction accuracy in SDNs.

3. Objective

The objectives of this project are:

- Build an SDN environment using Mininet and the Ryu controller.
- Implement a TM observation system to capture traffic data.
- Generate diverse network traffic using real-world traffic patterns.
- Evaluate and compare RNN variants—Long Short-Term Memory (LSTM), Bidirectional LSTM (BiLSTM), and Gated Recurrent Unit (GRU)—for TM prediction in SDNs, utilizing two datasets:
    a. A custom dataset generated through a testbed.
    b. GEANT backbone network dataset
- Measure the performance of RNN models based on prediction accuracy (Root Mean Square Error - RMSE) and computational efficiency.
- Provide recommendations on the best model for TM prediction, taking into account the complexity and accuracy trade-offs.

3.1 Key Challenges:

- **Decentralized Networks**: Limited visibility due to distributed control planes.
- **Linear Models' Limitations**: Models like ARIMA cannot handle nonlinear traffic patterns.
- **Computational Scalability**: Large networks require efficient data handling and processing.
- **Controller Compatibility Issues:**
  Initially used the POX controller but faced compatibility issues with newer versions of Python and OpenFlow, leading to a shift to the Ryu controller for its better support and updated APIs.
- **Working with PCAP Files:**
  Understanding the structure of pcap files and using tools like `tcprewrite` and `tcpreplay` to modify and replay traffic was challenging and required significant research.
- **Long-Term Testbed Operation:**
  Running the Mininet testbed continuously for 2–3 days to collect sufficient TM data was challenging due to occasional network crashes and the need to monitor data consistency.

4. Proposed Methodology

This project integrates **SDN-based traffic observation** with **AI models** for prediction. It is structured into two key components:

4.1 Traffic Observation Module in SDN

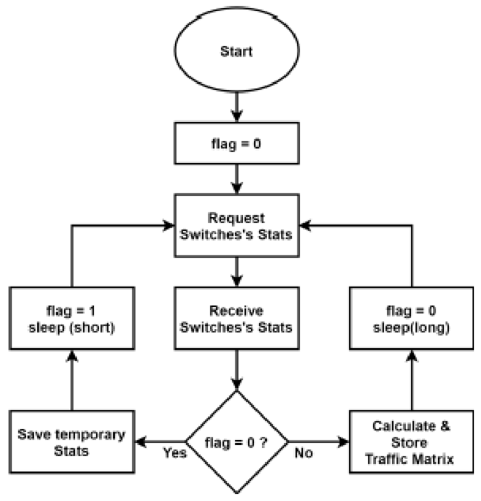

Fig. 1. TM Observation Working Flow

4.1.2 Workflow

- The controller sends OFP_FLOW_STATS_REQUEST messages to each switch.
- Switches respond with flow-level statistics (e.g., byte count, packet count).
- The TM is computed using these statistics: TM=ΔOTVM/Δt
  (where ΔOTVM is the difference between two consecutive Overall Traffic Volume Matrices)

4.2 TM Prediction Module

4.2.1 Model Overview

Three RNN-based models are employed:

- **LSTM**: Captures long-term dependencies in sequential data.
- **BiLSTM**: Enhances prediction by processing input sequences in both forward and backward directions.
- **GRU**: A simplified LSTM alternative with fewer parameters.

4.3 Workflow

Model	Parameters	Advantages	Disadvantages
LSTM	44,501	Handles long-term patterns	Resource-intensive
BiLSTM	89,001	Bidirectional processing	High computational cost
GRU	33,701	Efficient and accurate	Less effective on long patterns

Table 1 - Modal Comparison

- **Data Preprocessing**:
  - TMs are vectorized for feeding into RNNs.
  - Data normalization ensures values fall within [0, 1] for optimal training.
- **Training**:
  - Each model is trained using a sequence of past TMs to predict the next TM.
- **Evaluation**:
  - Root Mean Square Error (RMSE) is used to compare model performance.

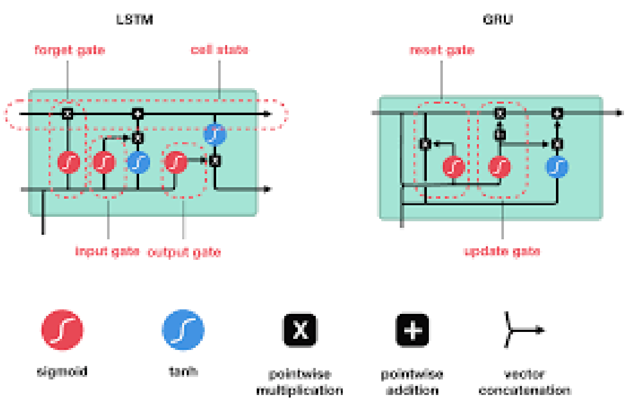

Figure 2 - LSTM and GRU cell architecture

5. Datasets

5.1 GÉANT Dataset

- **Description**: Real-world traffic data from 23 nodes and 38 links.
- **Size**: 10,772 TMs, recorded at 15-minute intervals over four months.
- **Usage**: 80% for training, 20% for testing.

5.2 Simulated SDN Testbed

- **Setup**: Mininet simulation of 14 switches and 18 links.
- **Traffic**: Synthetic patterns generated from modified pcap files.
- **Data Size**: 4123 TMs recorded at 1-minute intervals.

Dataset	Nodes	Links	TMs	Interval
GÉANT	23	38	10,772	15 minutes
SDN Testbed	14	18	4123	1 minute

Table 2- Dataset Summary

6. Work Performed

6.1 Setting Up the SDN Environment

- **Software Tools Used:**
  - **Mininet**: Simulated a software-defined network with 14 OpenFlow-enabled switches and 18 links, following the BSO topology.
  - **Ryu Controller**: Configured as the SDN controller to manage and control the network switches.
  - **Wireshark and Netresec Tools**: Used to analyze and modify traffic data.

- **Network Topology:**
  - Built a hierarchical network structure with switches connected to hosts, each with a unique IP address for easy traffic identification.
  - Configured flow rules and pre-installed routing paths for efficient data transmission.

6.2. Implementing the TM Observation System

- **Development of the Observation Module:**
  - Implemented in Ryu using Python and the OpenFlow protocol.
  - Send `OFP_FLOW_STATS_REQUEST` messages to all switches to retrieve flow statistics periodically.
  - Extracted source-destination IP addresses and byte/packet counts from flow tables.
  - Aggregated the collected statistics into an Overall Traffic Volume Matrix (OTVM).
  - Computed the TM by calculating the difference between two consecutive OTVMs and dividing by the time interval.
- **Challenges Overcome:**
  - Ensured minimal overhead on network switches by carefully scheduling flow statistics requests.
  - Verified data accuracy by cross-checking flow statistics using Wireshark captures and xterm.

6.3. Generating Network Traffic

- **Traffic Generation Tools and Methodology:**
    - Modified real-world packet capture (pcap) files using `tcprewrite` to include diverse traffic types such as video streaming, browsing, and intrusion attempts.
    - Played back the pcap files using `tcpreplay` to simulate traffic within the testbed.
    - Generated traffic over two- three days, capturing data at one-minute intervals to ensure sufficient data for training and evaluation.
- **Key Features:**
    - Diverse traffic patterns to test model robustness under various scenarios.
    - Realistic traffic to mimic actual network behavior, enhancing model generalizability.

6.4. Building the TM Prediction Module

- **Preprocessing Traffic Data:**
    - Normalized TM data to scale values between 0 and 1 for efficient model training.
    - Reshaped TM data into time-series vectors suitable for recurrent neural network inputs.
- **AI Models Implemented:**
    - **LSTM**: Captured long-term dependencies in traffic patterns.
    - **BiLSTM**: Utilized bidirectional analysis for more comprehensive feature extraction.
    - **GRU**: Offers a simpler and more resource-efficient alternative to LSTM.
- **Model Training:**
    - Split the dataset into training (80%) and testing (20%) sets.
    - Trained the models for 200 epochs using TensorFlow/PyTorch.
    - Used the Root Mean Square Error (RMSE) metric to evaluate model performance.

6.5. Evaluation and Analysis

- **Root Mean Square Error (RMSE)**: Measures prediction accuracy.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - x_i)^2}$$

- y_i: Predicted value.
- x_i: Actual value.
- n: Number of data points.

- **Datasets Used:**
  - **GEANT Dataset:** A publicly available dataset with 23 nodes and 38 links.
  - **Testbed Dataset:** Generated during the traffic replay process, comprising 4123 TM entries.

6.6 Findings and results

Across both the **GÉANT dataset** and the **SDN Testbed**, the key observations and results are as follows:

- **GRU Performance**:
  - GRU consistently achieved the lowest RMSE, demonstrating superior accuracy and computational efficiency.
  - It excelled in handling stable traffic patterns, making it ideal for scenarios with consistent traffic trends.
- **BiLSTM Performance**:
  - BiLSTM outperformed other models in handling irregular and unsteady traffic patterns, accurately predicting spikes and fluctuations.
  - However, this came at the cost of higher resource consumption and longer computation times.
- **General Observations**:
  - All models performed well for stable traffic but struggled with sudden spikes or abrupt traffic changes.
  - GRU's simplicity and efficiency make it the most practical choice overall, while BiLSTM is better suited for networks with unpredictable traffic dynamics.

By balancing accuracy, computational efficiency, and adaptability to traffic patterns, GRU emerged as the most versatile model, while BiLSTM proved optimal for irregular traffic scenarios where accuracy outweighs resource concerns.

Model	**Dataset**	**RMSE**	**Remarks**
GRU	GÉANT	Low	Best overall
BiLSTM	GÉANT	Medium	Suitable for irregular traffic
GRU	SDN Testbed	Low	Efficient and scalable

Table 3 - Model Performance Summary

6.7 Graph: Traffic Prediction

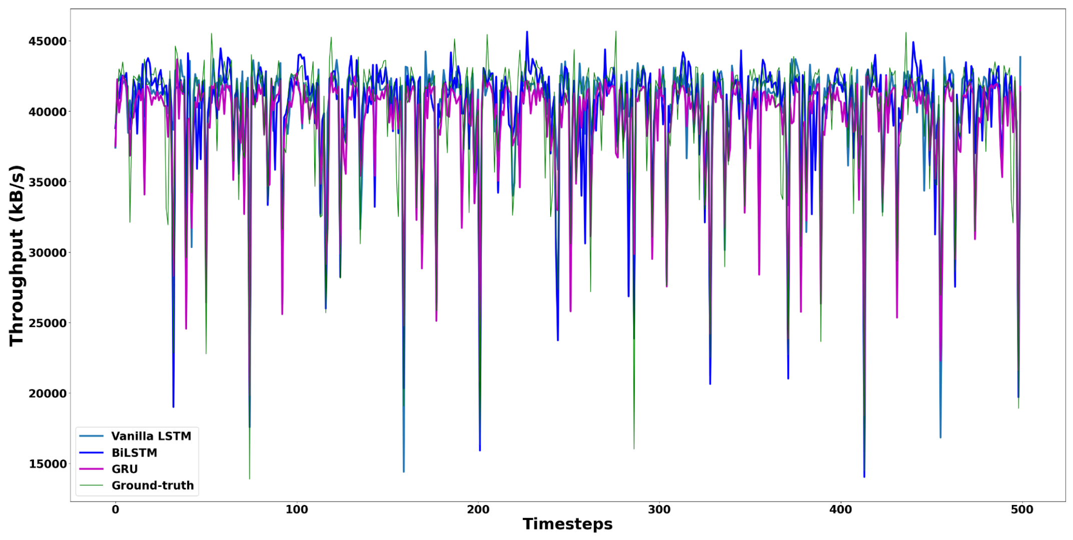

Figure 3 - Simulated Testbed 52nd OD prediction comparison

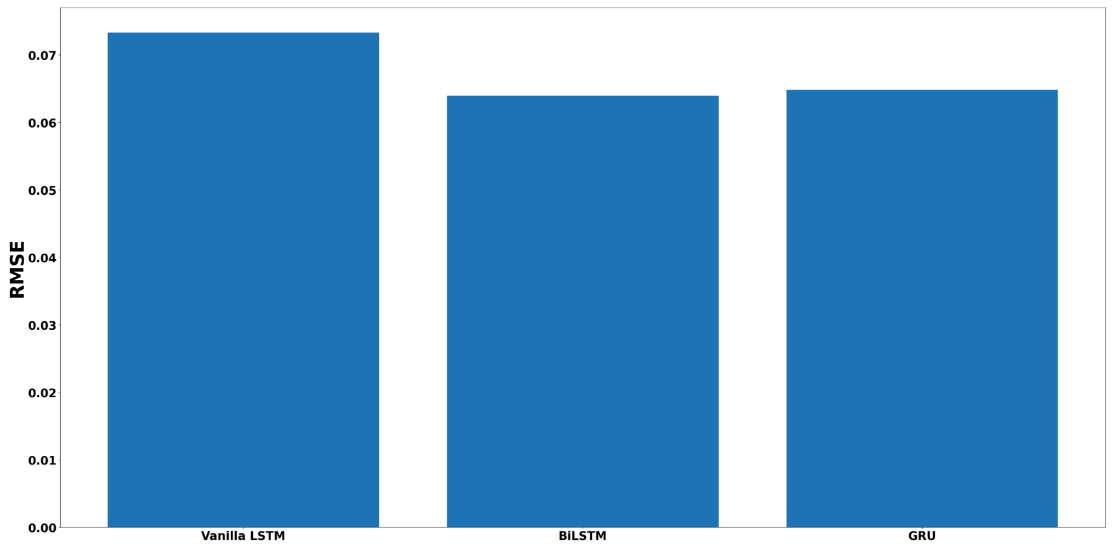

Figure 4 - Simulated Testbed 52nd OD RMSE

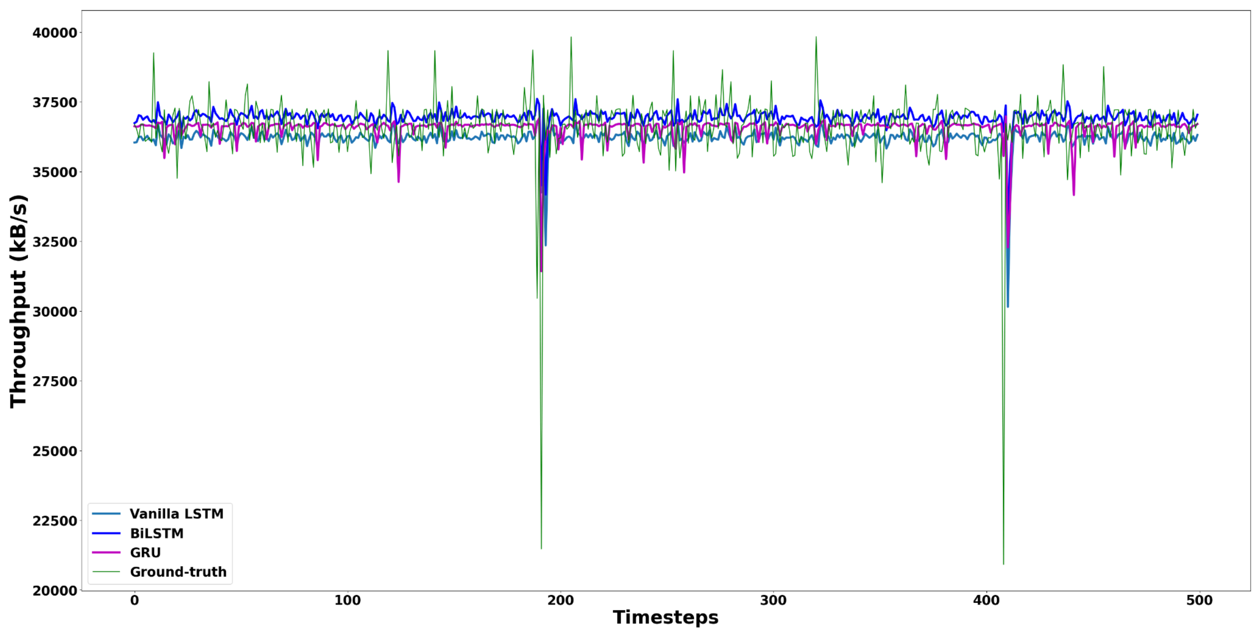

Figure 5 - Simulated Testbed 5th OD prediction comparison

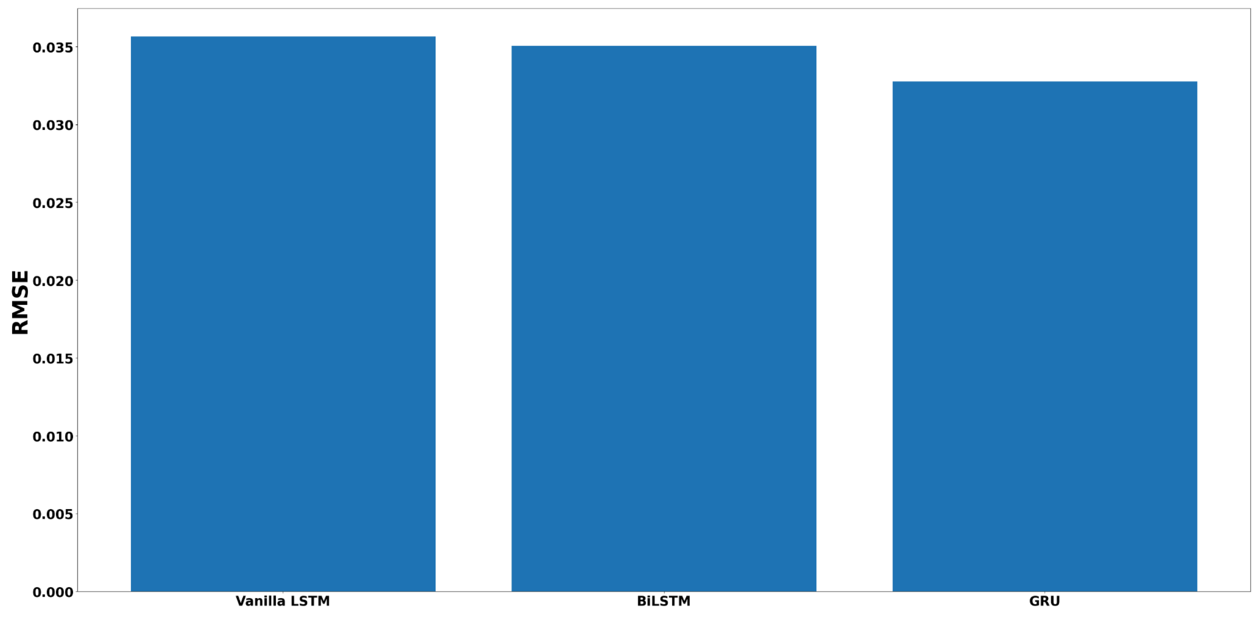

Figure 6 - Simulated Testbed 5th OD RMSE

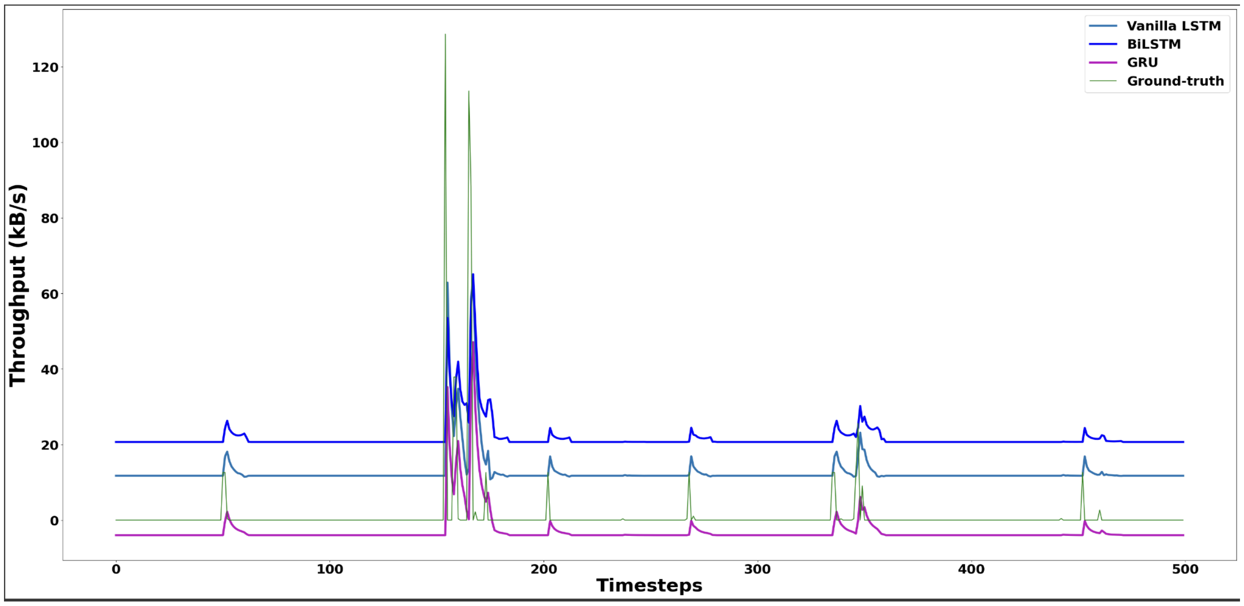

Figure 7 - Geant Testbed 52nd OD prediction comparison

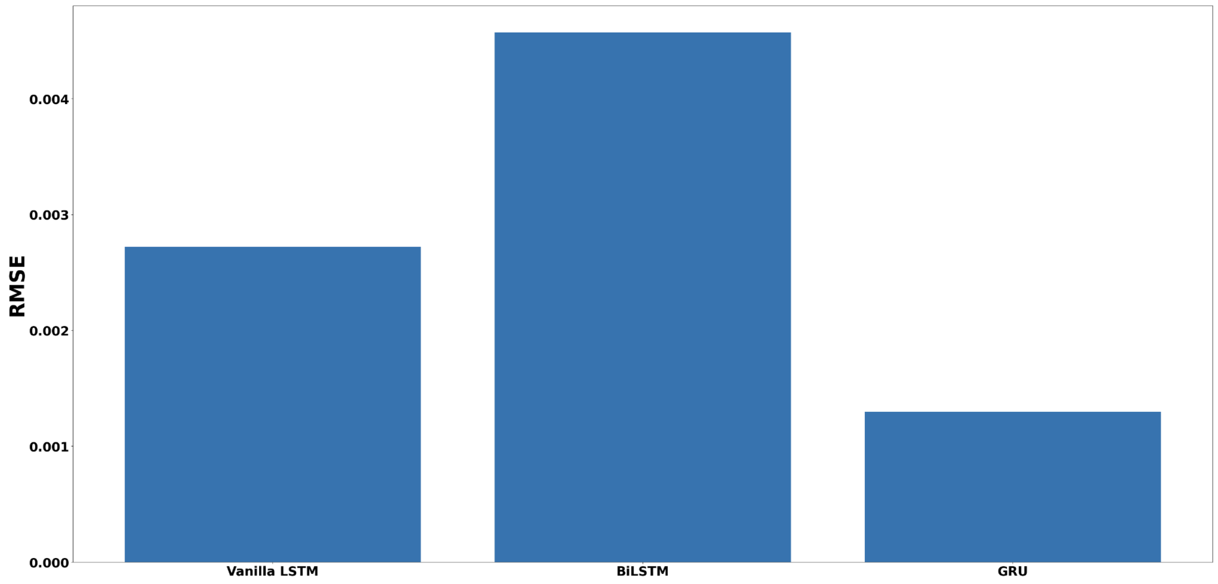

Figure 8 - Geant Testbed 52nd OD RMSE

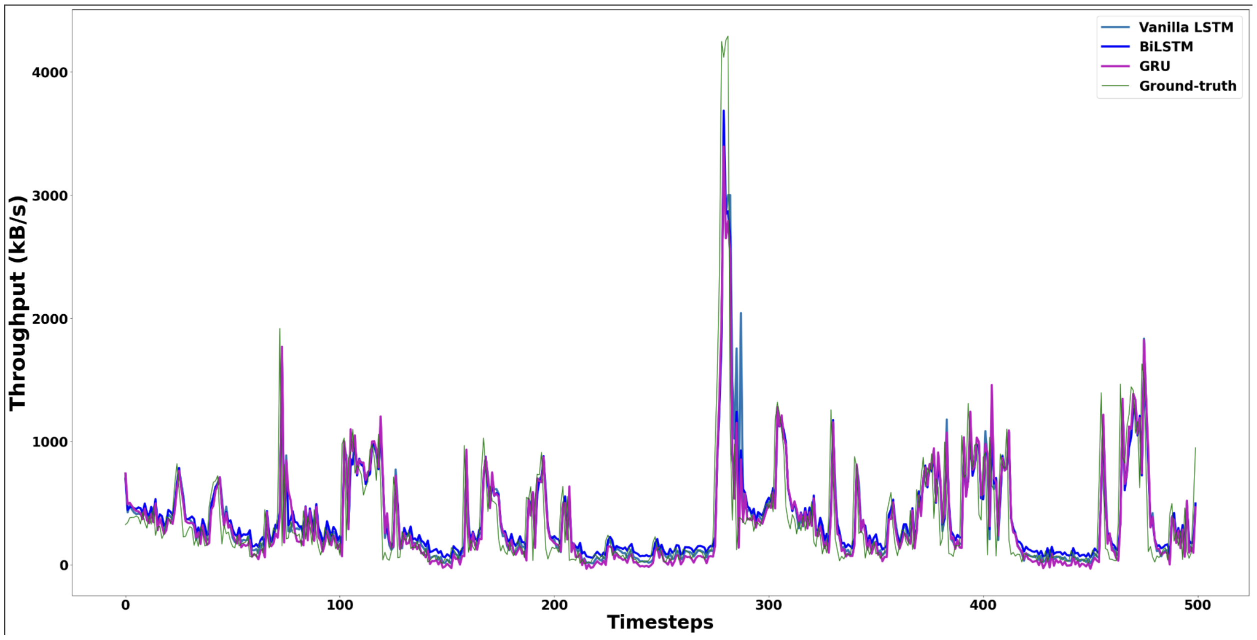

Figure 9 - Geant Testbed 102nd OD prediction comparison

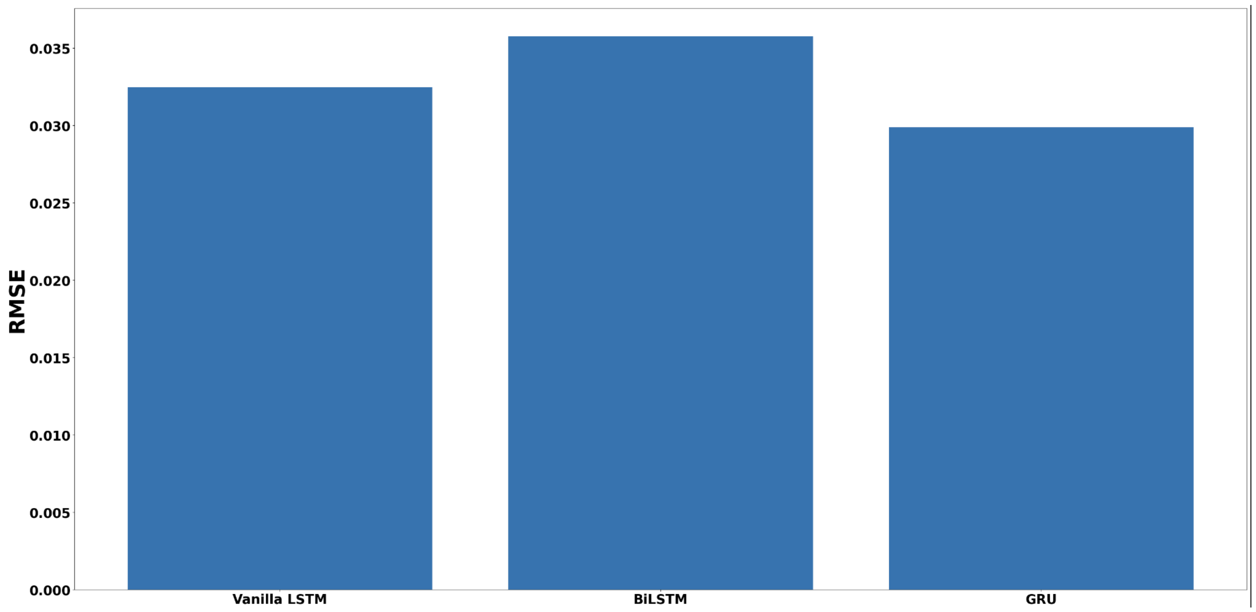

Figure 10 - Geant Testbed 102nd OD RMSE

7. Conclusion

This project successfully simulated an AI-based TM prediction framework using the Ryu controller. The combination of SDN's centralized control and ML's predictive capabilities offers significant potential for advanced network management. GRU emerged as the most effective model, balancing prediction accuracy and resource usage.

8. Future Work

- **Enhancing Models**: Combine RNNs with attention mechanisms for better handling of irregular patterns.
- **Dynamic Routing**: Integrate predictions into SDN controllers for real-time routing adjustments.
- **Scalability**: Extend to larger networks and explore distributed SDN architectures.

9. References

1. Mininet : http://mininet.org/
2. Ryu - https://ryu-sdn.org/
3. Netresec.- https://www.netresec.com/?page=PcapFiles
4. https://wiki.wireshark.org/SampleCaptures#sample-captures
5. https://ieeexplore.ieee.org/document/9500331