NIC Gateway Optimization

By Divyalakshmi Varadha Rajan Prem Sudha
CSE 422 Honors Project – Fall 2024

Introduction

This project aims to optimize network performance by dynamically selecting the best Network Interface Card (NIC) based on real-time latency measurements. The system continuously monitors multiple NICs, pings selected target websites, and determines the optimal NIC for network traffic based on the lowest Round-Trip Time (RTT). The implementation includes a real-time dashboard using **Streamlit** to visualize NIC performance.

Design and Architecture

The system consists of the following core components:

1. **Network Interface Detection**
   - The software detects all available NICs on the system using the netifaces Python library.
   - The system retrieves the IP address assigned to each NIC.

2. **Real-Time RTT Measurement**
   - The software performs periodic ping tests to a set of predefined websites:
     - Google.com
     - YouTube.com
     - StackOverflow.com
     - Gmail.com
     - MSU.edu
   - The RTT values from the ping responses are extracted and stored for analysis.

3. **NIC Selection**
   - The software compares the RTT values of each NIC and selects the NIC with the lowest average RTT.
   - The system dynamically updates the default NIC for outgoing traffic using ip route replace default.

4. **Visualisation and Dashboard**
   - A real-time web dashboard is implemented using **Streamlit**.
   - The dashboard displays:
     - A table with NIC, target website, timestamp, and RTT values.
     - A graph showing the RTT trends over time.
     - Summary statistics, including the baseline RTT and optimized RTT after NIC switching.

Implementation Details

Tech Stack

- **Python** for scripting and logic implementation.
- **Netifaces** for NIC detection.
- **Subprocess** for executing ping commands.
- **Streamlit** for real-time web-based visualization.
- **Pandas** for data processing.
- **Matplotlib** for plotting RTT performance over time.

Key Functions

- get_nic_ips(): Retrieves IP addresses of available NICs.
- ping_targets(nic): Measures RTT for each NIC by pinging predefined websites.
- select_nic(): Selects the best NIC based on RTT results.
- test_nic(stage): Measures RTT performance before and after NIC optimization.
- st.dataframe(df): Displays real-time RTT data.
- st.pyplot(fig): Generates and updates the RTT performance graph.

Observations

- Initial measurements showed that different NICs had significantly varying RTT values based on network conditions.
- Some NICs lacked IP assignments, causing them to be ignored in RTT testing.
- The algorithm switched NICs dynamically based on latency measurements, which resulted in noticeable improvements in RTT values after optimization.
- The real-time dashboard correctly displayed RTT values and NIC performance over time.
- The table and graph dynamically updated as new RTT data was collected.
- Some missing RTT values (e.g., due to unreachable targets) were handled without giving errors or halting the program execution.

How to Test the Application

Setup and Execution Steps

1. In a terminal session, SSH into the remote server.
    - ssh firstname@IP_ADDRESS
    - password: PID

2. Change the directory to where the Python file resides.

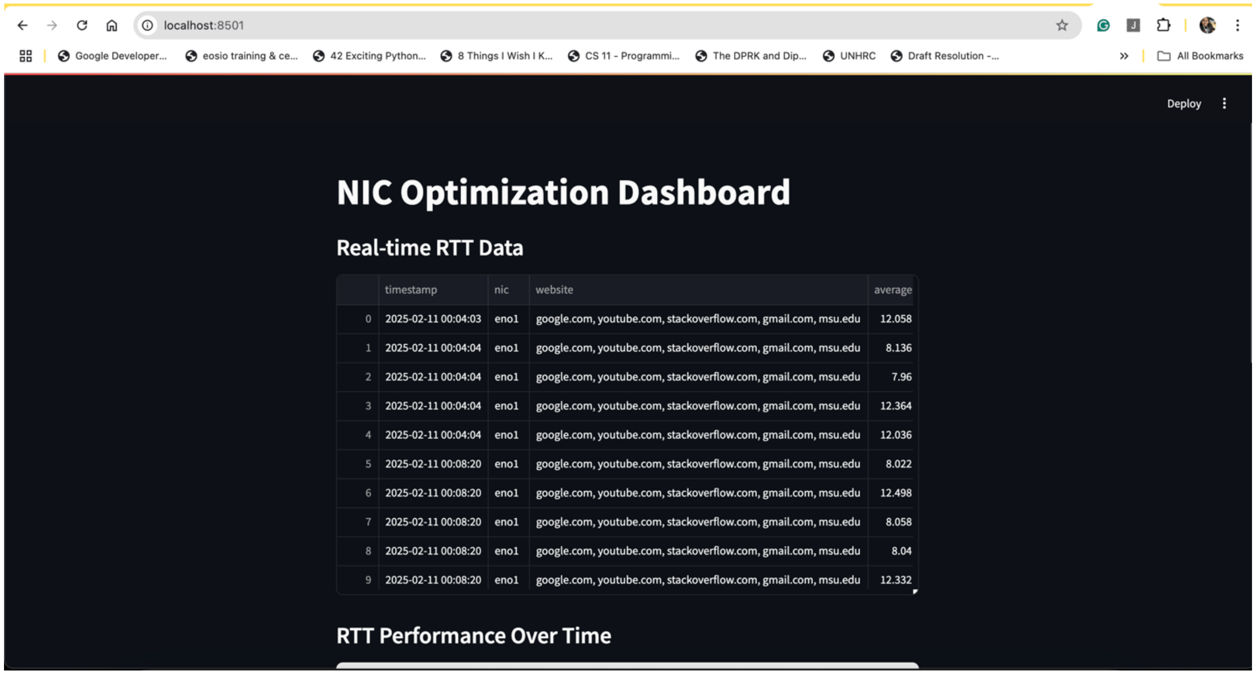
3. Ensure dependencies are installed.
    - sudo pip3 install streamlit pandas netifaces matplotlib

4. Run the Python program to launch the Steamlit app. This takes a while to complete its execution, i.e. pinging all the respective target websites to get the data for the RTTs.
    - streamlit run honorsfinal1.py

5. Open another terminal session side by side. Implement Port Forwarding using the following command:
    - ssh -L [PORT # used by Streamlit]:[IP Address of Remote Server]:[PORT # used by Streamlit]: -N -f [netid]@scully.egr.msu.edu
    - **Example:** ssh -L 8501:35.9.42.236:8501 -N -f varadhad@scully.egr.msu.edu

6. Access the NIC GUI Dashboard by navigating to
    - http://localhost:[PORT # used by Streamlit]
    - **Example:** http://localhost:8501.

GUI Screenshots



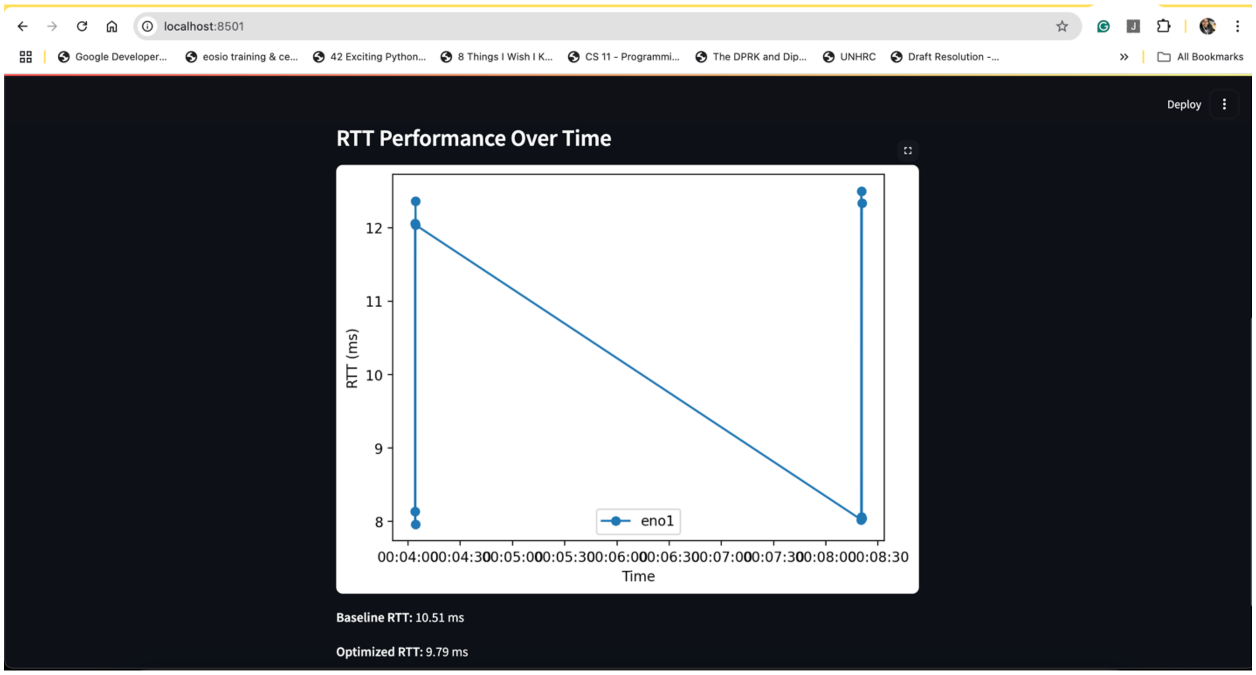
NIC Optimization Dashboard

Real-time RTT Data

	timestamp	nic	website	average
0	2025-02-11 00:04:03	eno1	google.com, youtube.com, stackoverflow.com, gmail.com, msu.edu	12.058
1	2025-02-11 00:04:04	eno1	google.com, youtube.com, stackoverflow.com, gmail.com, msu.edu	8.136
2	2025-02-11 00:04:04	eno1	google.com, youtube.com, stackoverflow.com, gmail.com, msu.edu	7.96
3	2025-02-11 00:04:04	eno1	google.com, youtube.com, stackoverflow.com, gmail.com, msu.edu	12.364
4	2025-02-11 00:04:04	eno1	google.com, youtube.com, stackoverflow.com, gmail.com, msu.edu	12.036
5	2025-02-11 00:08:20	eno1	google.com, youtube.com, stackoverflow.com, gmail.com, msu.edu	8.022
6	2025-02-11 00:08:20	eno1	google.com, youtube.com, stackoverflow.com, gmail.com, msu.edu	12.498
7	2025-02-11 00:08:20	eno1	google.com, youtube.com, stackoverflow.com, gmail.com, msu.edu	8.058
8	2025-02-11 00:08:20	eno1	google.com, youtube.com, stackoverflow.com, gmail.com, msu.edu	8.04
9	2025-02-11 00:08:20	eno1	google.com, youtube.com, stackoverflow.com, gmail.com, msu.edu	12.332

RTT Performance Over Time

Figure 1: The NIC Gateway Dashboard showing real-time RTT Data for five websites.

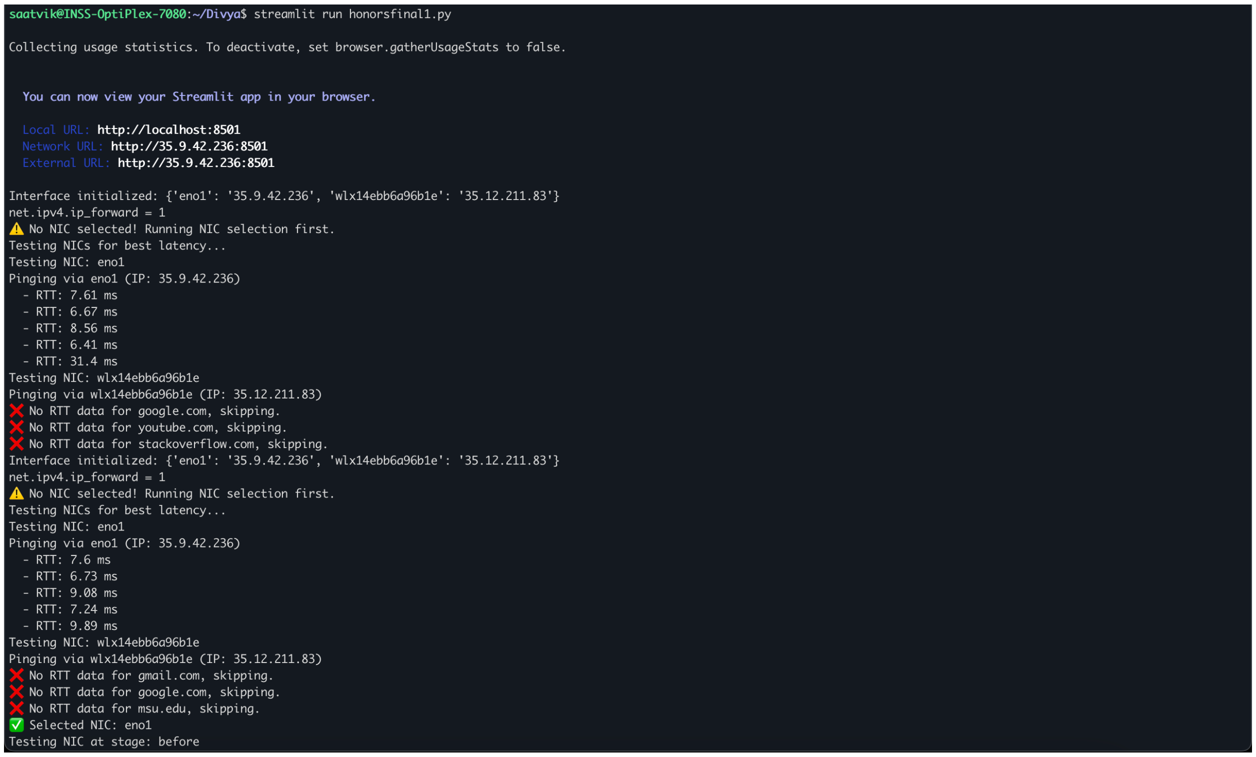


Baseline RTT: 10.51 ms

Optimized RTT: 9.79 ms

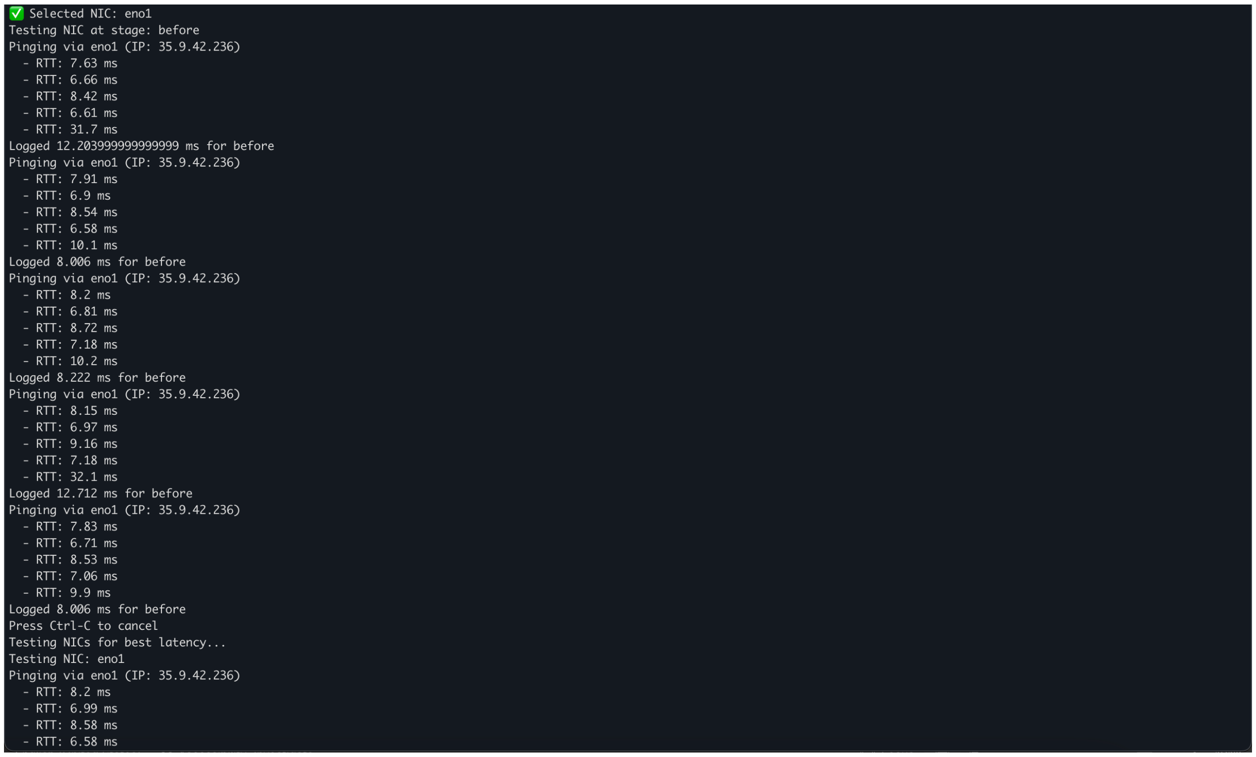
Figure 2: The NIC Gateway Dashboard showing graph of RTT over time and optimized RTT.

Terminal Outputs



```
saatvik@INSS-OptiPlex-7080:~/Divya$ streamlit run honorsfinal1.py

Collecting usage statistics. To deactivate, set browser.gatherUsageStats to false.


  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://35.9.42.236:8501
  External URL: http://35.9.42.236:8501

Interface initialized: {'eno1': '35.9.42.236', 'wlx14ebb6a96b1e': '35.12.211.83'}
net.ipv4.ip_forward = 1
⚠️ No NIC selected! Running NIC selection first.
Testing NICs for best latency...
Testing NIC: eno1
Pinging via eno1 (IP: 35.9.42.236)
  - RTT: 7.61 ms
  - RTT: 6.67 ms
  - RTT: 8.56 ms
  - RTT: 6.41 ms
  - RTT: 31.4 ms
Testing NIC: wlx14ebb6a96b1e
Pinging via wlx14ebb6a96b1e (IP: 35.12.211.83)
❌ No RTT data for google.com, skipping.
❌ No RTT data for youtube.com, skipping.
❌ No RTT data for stackoverflow.com, skipping.
Interface initialized: {'eno1': '35.9.42.236', 'wlx14ebb6a96b1e': '35.12.211.83'}
net.ipv4.ip_forward = 1
⚠️ No NIC selected! Running NIC selection first.
Testing NICs for best latency...
Testing NIC: eno1
Pinging via eno1 (IP: 35.9.42.236)
  - RTT: 7.6 ms
  - RTT: 6.73 ms
  - RTT: 9.08 ms
  - RTT: 7.24 ms
  - RTT: 9.89 ms
Testing NIC: wlx14ebb6a96b1e
Pinging via wlx14ebb6a96b1e (IP: 35.12.211.83)
❌ No RTT data for gmail.com, skipping.
❌ No RTT data for google.com, skipping.
❌ No RTT data for msu.edu, skipping.
✅ Selected NIC: eno1
Testing NIC at stage: before
```

Figure 3: Terminal Outputs showing dynamic NIC selection based on real-time RTT data.



```
✅ Selected NIC: eno1
Testing NIC at stage: before
Pinging via eno1 (IP: 35.9.42.236)
  - RTT: 7.63 ms
  - RTT: 6.66 ms
  - RTT: 8.42 ms
  - RTT: 6.61 ms
  - RTT: 31.7 ms
Logged 12.203999999999999 ms for before
Pinging via eno1 (IP: 35.9.42.236)
  - RTT: 7.91 ms
  - RTT: 6.9 ms
  - RTT: 8.54 ms
  - RTT: 6.58 ms
  - RTT: 10.1 ms
Logged 8.006 ms for before
Pinging via eno1 (IP: 35.9.42.236)
  - RTT: 8.2 ms
  - RTT: 6.81 ms
  - RTT: 8.72 ms
  - RTT: 7.18 ms
  - RTT: 10.2 ms
Logged 8.222 ms for before
Pinging via eno1 (IP: 35.9.42.236)
  - RTT: 8.15 ms
  - RTT: 6.97 ms
  - RTT: 9.16 ms
  - RTT: 7.18 ms
  - RTT: 32.1 ms
Logged 12.712 ms for before
Pinging via eno1 (IP: 35.9.42.236)
  - RTT: 7.83 ms
  - RTT: 6.71 ms
  - RTT: 8.53 ms
  - RTT: 7.06 ms
  - RTT: 9.9 ms
Logged 8.006 ms for before
Press Ctrl-C to cancel
Testing NICs for best latency...
Testing NIC: eno1
Pinging via eno1 (IP: 35.9.42.236)
  - RTT: 8.2 ms
  - RTT: 6.99 ms
  - RTT: 8.58 ms
  - RTT: 6.58 ms
```

Figure 4: Terminal Outputs showing dynamic NIC selection based on real-time RTT data.

Challenges Faced

Since the NICs were set up on a remote server, testing required SSH access to the remote server. I used SSH and port forwarding to view the dashboard locally. The server connections did time out at times on port 22 (SSH) or get refused. Some Python dependencies were missing on the server, requiring manual installation, so I installed required dependencies via pip install and ensured correct permissions for sudo execution. Some NICs did not have an assigned IP, which caused ping failures, so I updated the script to skip NICs without valid IPs and avoid unnecessary errors. I also didn't have required permissions to SSH into the remote server with my first name and PID, so I tried accessing the remote server through my classmate (Saatvik's) credentials on the remote server to test my code.

Future Plans

- **Extend Performance Metrics:** Include **packet loss, jitter, and bandwidth** alongside RTT measurements.
- **Enhance Visualization:** Add **real-time alerts** and a comparison mode for historical NIC performance.
- **Mobile-Friendly UI:** Optimize the Streamlit dashboard for viewing on mobile devices.
- **Predicting RTT with AI:** Using linear regression (or other models) to predict RTT based on history for a particular destination IP address.

Conclusion

The project successfully implemented a real-time NIC optimization system that dynamically selects the best NIC for outgoing network traffic. The software provides a visual representation of network performance and adapts dynamically to changing network conditions. The Streamlit-based dashboard makes real-time network monitoring easy and accessible.

References

- Python Netifaces Documentation: https://pypi.org/project/netifaces/
- Streamlit Documentation: https://docs.streamlit.io/
- Linux Networking Commands: https://linux.die.net/man/