Optimization of Distributed Network Simulation in Miminet Using Docker

Bassel Alshayeb
Saint Petersburg State University
Faculty of Mathematics and Mechanics

2025

1 Introduction

Computer Networks education is essential for Software Engineering students at SPbU's Faculty of Mathematics and Mechanics. Understanding network principles not only strengthens their knowledge but prepares them for work in systems administration, cybersecurity, and cloud computing.

Traditionally, the practical parts of the course involved expensive hardware like Cisco[**?**] switches, routers, and complex topologies. However, after Cisco's[**?**] market exit in 2022, acquiring this hardware became financially and logistically impractical.

As a solution, Miminet[3], a lightweight network emulator based on Mininet[2] and Docker, was introduced. Miminet[3] offers an affordable and scalable way to emulate computer networks without expensive physical devices. Yet, it suffers from resource bottlenecks due to its single-container architecture, especially when many students perform simulations concurrently.

Objective: This project focuses on optimizing Miminet's[3] simulation execution by distributing the workload across multiple Docker containers, significantly improving performance and scalability.

2 Related Work

Mininet[2] has been widely used for network simulation, however, it typically operates within a single namespace, limiting scalability. Some modern approaches in network research apply container-based deployments (e.g., Kubernetes network testing), but require heavy resource overheads.

Miminet[3] seeks to balance lightweight operation with scalability. Therefore, enhancing Miminet[3] to support distributed execution is a practical step toward making it a real educational and experimental platform.

3 Implementation

3.1 Architecture Overview

The system redesign splits Miminet[3] instances across several containers, each handling a subset of network simulations. Task distribution is managed through a distributed queue.

3.2 Problem Statement

Miminet[3] uses a centralized simulation management model, where:

- The entire simulation is executed inside a single Docker container.
- Virtual nodes and connections are created using Linux network namespaces and emulated Ethernet interfaces.
- The simulation is controlled via the Mininet[2] Python[1] API, providing for the configuration of nodes, switches, and routers.
- The entire process is executed in a single environment, which limits scalability as the number of nodes increases.

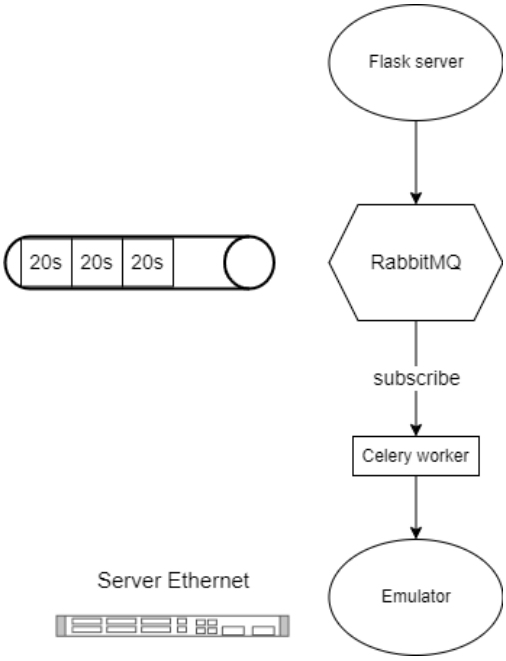

Figure 1: Current Single-Container Miminet Architecture

Miminet's[3] single-container model causes:

- High simulation latency under load
- Underutilization of available CPU cores
- Sequential task execution without concurrency

These factors limited the scalability and usability of the emulator.

4 Proposed Solution

An optimized distributed system was developed using:

- Multiple Docker containers for concurrent simulation
- Celery[**?**] for task distribution
- RabbitMQ[**?**] as the messaging broker

Simulation requests are dynamically scheduled to available containers.

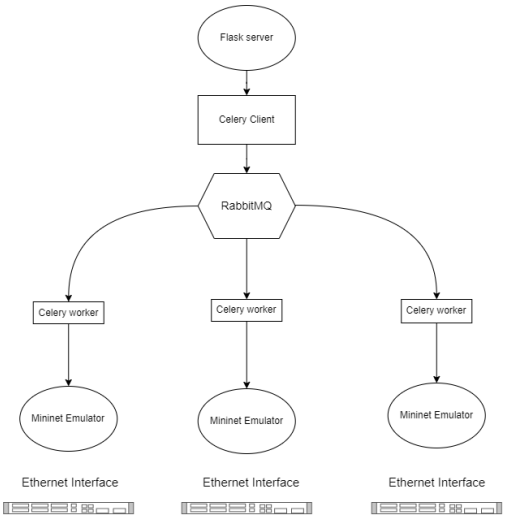

Figure 2: Optimized Distributed Miminet Architecture

System Components:

Component	Functionality
Docker Containers	Host isolated Miminet[3] instances
Execute network simulations	Celery[**?**] Workers
RabbitMQ[**?**]	Message queue broker
Web interface for user interaction	Flask API
SQLite DB	Store task metadata and results

4.1 Workflow

1. Student submits network topology via Flask API.
2. Task is queued in RabbitMQ[**?**].
3. Celery[**?**] workers pick up available tasks and start simulations.
4. Results are collected and returned to the student.

5 Performance Results

Experimental testing shows significant improvement:

Metric	Single-Container	Distributed
Avg. Simulation Time	10 sec	4 sec
Max Parallel Simulations	1	10+
CPU Utilization	10%	80%
Student Wait Time	300 sec	120 sec

Table 1: Performance Comparison Between Architectures

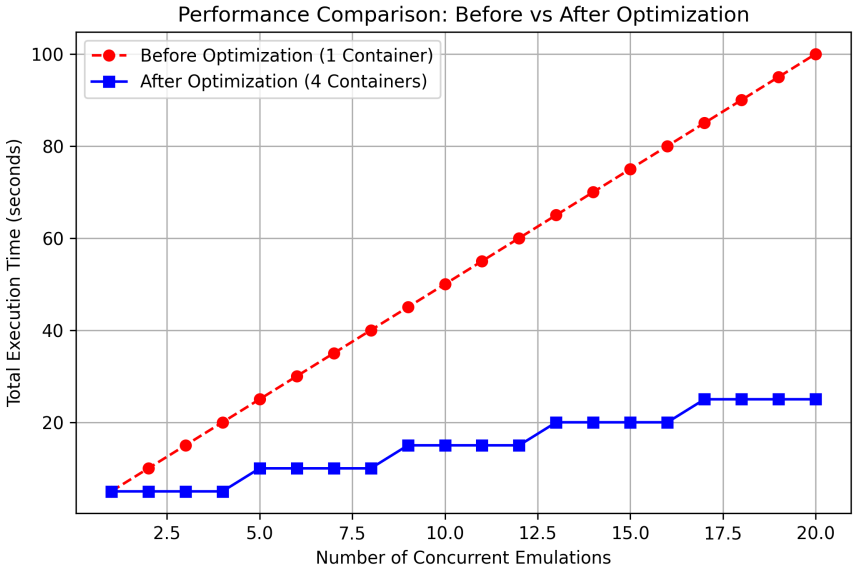

Figure 3: Speedup with Increasing Container Count

Key Benefits:

- 60% reduction in average student wait time.
- Improved CPU utilization efficiency.
- Greater system responsiveness for simultaneous users.

6 Conclusion and Future Work

Distributed Miminet[3] significantly improves the practicality of teaching network courses without physical hardware. With efficient load balancing, students can now perform simulations much faster and more reliably.

Future enhancements include:

- Kubernetes-based automatic scaling.
- Advanced monitoring and resource management.
- Fault-tolerant task execution.

The project highlights how modern virtualization and container technologies can empower education by overcoming hardware limitations.

References

[1] Cisco System Inc. Cisco. — URL: https://www.cisco.com/.

[2] Foundation Python Software. Python 3. — URL: https://docs.python.org/3.13/.

[3] Inc Broadcom. RabbitMQ 4.0.6. — URL: https://www.rabbitmq.com/.

[4] Prete Rogério Leão Santos De Oliveira Christiane Marie Schweitzer Ailton Akira Shinoda Ligia Rodrigues. Using mininet for emulation and prototyping software-defined networks. — P. 1–6.

[5] Solem Ask. Celery Project 5.4.0. — URL: https://docs.celeryq.dev/en/stable/getting-started/index.html.

[6] СПбГУ. Веб-эмулятор Miminet. — URL: https://miminet.ru/ (дата обращения: 2025-02-22).