CSCI 492: Senior Project I

Cloud Based Telemetry System for SDNs

Initial Project Proposal Document

Team Members

1) John Aulich, CSCI

2) Chase Bacon, CSCI

3) Cole Hardy, CSCI

4) Cheydan Hauf, CSCI

5) Adam Roy, CSCI

Project Advisor: Dr. Jun Liu

September 20th, 2024

# 

# Table of Contents

[Table of Contents 2](#_bplpvb7rx6q3)

[1. Key Objectives (summary of goals) 3](#_n2hfbylegpuk)

[1.1. Project Definition 3](#_d4yyyxyqdvdy)

[1.2. Key Objectives 3](#_7xzim0yroajf)

[1.2.1. Prototype a Dynamic Load Balancer 3](#_ym3lhyav874)

[1.2.2. Real-Time Traffic Path Monitoring 3](#_6o49j0vbyt43)

[1.2.3. Improve Network Performance 3](#_ek9wxpjvwrid)

[2. Leveraging Past Success (Background of prior work) 3](#_re0tx9r9p9gv)

[2.1. Summarization of Past Work 3](#_lzgw7ftvk5a2)

[2.2. Building on Past Work 3](#_lzgw7ftvk5a2)

[3. Methodology & Implementation 4](#_5j22rm3q9u6a)

[3.1. Problem Statement & Research 4](#_b99b2wtfy879)

[3.2. Design & Architecture 4](#_h3rxdetagwt)

[3.2.1. Network Architecture 4](#_wrp535cegfkk)

[3.2.2. Data Processing 4](#_p6du7k6bbho1)

[3.2.3. Load Balancing Logic 5](#_nlmf1sy1q7s)

[3.2.4. Considerations for stability 5](#_emhbe0dud8l3)

[3.3. Implementation 5](#_tp7abua12tcc)

[3.3.1. Telemetry Collection and Processing 5](#_o9vuh56gsxri)

[3.3.2. Load balancer implementation and deployment 5](#_c8ngyvie2zny)

[3.3.3. Algorithm development and optimization 5](#_thqmdu5fiqc0)

[3.3.4. Software Resources 6](#_fx8tb4hy5lo2)

[3.3.5. Hardware Resources 6](#_6da764d5mnms)

[3.4. Testing and Evaluation 6](#_1rcn2naa11gp)

[3.4.1. Simulation Environment 6](#_8hjjfmlt9rje)

[3.4.2. Traffic Generation 6](#_8nromo7s3uce)

[3.4.3. Metrics Collection 7](#_dw6cc84ol7rj)

[4. Anticipated Outcomes & Benefits (broad impact) 7](#_5ie6d62s3ly8)

[5. Unique Value Proposition (potential contributions) 7](#_64iwoea5vyl8)

[6. Appendix 7](#_a312cioif8ri)

[6.1. Definitions & Descriptions 7](#_symvm87vdv5e)

[6.1.2. Descriptions 7](#_mce2lufk9cjy)

# Key Objectives (summary of goals)

## Project Definition

The goal of this project is to advance the ongoing research into cloud-based network telemetry infrastructure by creating a load balancer. This simple load balancer will have the ability to dynamically change the paths of traffic flow based on the traffic status and conditions of a network. Managing the paths will increase data throughput and reduce congestion.

## Key Objectives

### 1.2.1. Prototype a Dynamic Load Balancer

Design and develop a prototype load balancer that is tailored for cloud-based telemetry systems.

### 1.2.2. Real-Time Traffic Path Monitoring

Add the ability to the load balancer to continuously monitor network traffic paths in real-time.

### 1.2.3. Improve Network Performance

Utilize the telemetry data that is being collected to identify network performance issues and implement dynamic traffic flow adjustments to increase the data throughput.

# Leveraging Past Success (Background of prior work)

## 2.1. Summarization of Past Work

Last year’s students made significant progress by tracking telemetry data from a cloud environment at UND. They integrated open-source packages, such as Kafka and GoFlow2, into their system to monitor data paths between source and destination containers. Using these tools, the project successfully tracked and reconstructed the exact data paths taken by network traffic. This information is crucial for comparing the actual data paths with the intended paths set by the SDN controller.

2.2. Building on Past Work

As previously described, last year’s students focused on the implementation of data path reconstruction. Our aim is to extend this project by introducing load balancing capabilities that leverage those aforementioned telemetry streams. More specifically, the load balancer will use the telemetry data to monitor network conditions in real-time and reroute traffic as necessary. This will ensure reduced latency and better resource allocation while minimizing packet loss. This proposed project will also lay the groundwork for future expansion into more complex network optimization strategies.

# Methodology & Implementation

## Problem Statement & Research

Our capstone project aims to develop a network load balancer specifically tailored to the unique demands of our cloud-based telemetry network. Traditional load balancing approaches often struggle to handle the high data volumes, real-time requirements, and distributed nature of such networks, leading to suboptimal performance and potential service disruptions. By leveraging modern technologies like containerization and software-defined networking (SDN), we seek to create a load balancer that can dynamically adapt to changing network conditions and optimize the performance of our telemetry network.

## Design & Architecture

### Network Architecture

The proposed network load balancer architecture is centered around OvSwitch (OVS) nodes, managed by a Software Defined Networking controller. GoFlow2 will be employed to collect telemetry data from each OVS switch.

### Data Processing

Kafka will be employed as the data streaming platform to process incoming telemetry data and stream it to the load balancer. The entire load balancing process will be implemented in the Go programming language. Go will consume telemetry data from the Kafka broker in real time, leveraging its concurrency features to efficiently handle multiple streams of telemetry data concurrently to provide quick traffic-balancing decisions.

### Load Balancing Logic

Various algorithms will be implemented in Go to dynamically adjust traffic flows based on telemetry data and the balancing algorithm will be able to make decisions informed by current network conditions. Go’s concurrency model will facilitate efficient processing of multiple telemetry streams enabling real-time decision making.

### Considerations for stability

Fault tolerance can be implemented on critical components like the SDN controller and Kafka brokers. The system can be built for scalability in both horizontal and vertical aspects to accommodate increased workloads. Monitoring and management practices can be utilized in order to collect and visualize key aspects of the system and provide the tools to better manage the load balancing system.

## Implementation

### Telemetry Collection and Processing

GoFlow2 will be integrated into the SDN system to collect flow records from OVS instances. This telemetry data will be forwarded to Kafka for processing and a Go-based load balancer will consume the telemetry data from Kafka and process the data concurrently.

### Load balancer implementation and deployment

The load balancer will be developed as a Go microservice and it will be containerized using docker to facilitate flexible deployment and testing in various environments. The load balancer will listen to the telemetry data from Kafka and adjust network traffic flows based on the selected load balancing algorithm.

### Algorithm development and optimization

The load balancer will implement various load balancing algorithms in Go, such as Round Robin, Least Connection, Least Response, and Source IP Hash. The routing decisions will be based on real-time telemetry data and each algorithm will be tested for effectiveness under diverse traffic conditions.

### Software Resources

* Go: The primary language for implementing the load balancer and processing telemetry data
* Kafka: A distributed streaming platform for handling real-time telemetry data
* GoFlow2: A telemetry tool for collecting flow records from network switches.
* Docker: Containerization technology for deploying the load balancer and other microservices.
* Mininet: A network simulator for emulating the SDN environment and generating test traffic.
* OpenFlow: The protocol used by the SDN controller to manage flow tables in OVS instances.

### Hardware Resources

Cloud-based virtual machines or physical servers on the UND network will be utilized to host the various components, including OVS switches, SDN controllers and the Go-based load balancer. The user of VMs will allow us to simulate a distributed cloud environment with containers acting as PCs and OVS instances managing traffic within and between VMs

## Testing and Evaluation

### Simulation Environment

The system will be deployed in a simulated SDN environment using Mininet and Docker containers for the Go-based microservices and Go-based load balancer.

### Traffic Generation

Network traffic will be generated to simulate various system loads and the network load balancer will be tested and evaluated on its ability to dynamically redistribute traffic across OVS.

### Metrics Collection

Key metrics while testing such as latency reduction, link utilization and packet loss will be tracked to evaluate the effectiveness of the load balancer in optimization of network traffic. The telemetry system’s performance will also be analyzed in terms of its ability to handle real time data processing.

# Anticipated Outcomes & Benefits (broad impact)

# Unique Value Proposition (potential contributions)

# Appendix

## Definitions & Descriptions

* + 1. Definitions  
       SDN: Software defined network

### Descriptions