**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

[**1. Project Objectives 3**](#_qw9b74k7txrs)

[1.1. Project Definition 3](#_m0tvoykrd1r)

[1.2. Key Objectives 3](#_1vp6egg8p2ze)

[**2. Background 3**](#_ov41snng4zrk)

[2.1. Existing SDN Architecture 3](#_uw441rf8c1st)

[2.2. Expansion into SDN Load Balancing 3](#_k0rwot15fn68)

[**3. Contributions 4**](#_3504s0w9jaeq)

[3.1. Development of a Dynamic Load Balancer 4](#_ygyobafsmbjr)

[3.2. Real-Time Traffic Management 4](#_y40sa2esfngd)

[3.3. Integration with SDN for Dynamic Reconfiguration 4](#_bq3w4tjf0rev)

[3.4. Scalability & Flexibility 4](#_lyc3fm4usxnr)

[3.5. Capstone Worthiness 4](#_h6o54h6nekgu)

[**4. Broader Impacts 5**](#_5bn6j0hdoewn)

[4.1. Environmental Impact 5](#_vsljzwus3vd4)

[4.2. Social Impact 5](#_frmsxnkvgkgi)

[4.3. Economic Impact 5](#_5igxf7m0kahs)

[4.4. Relevance to Current State of the Art 5](#_pfce6g4r5ex4)

[**5. Approach & Methodology 6**](#_65c89yv3n8ca)

[5.1. Problem Statement & Research 6](#_gg2fgea6t4t0)

[5.2. Design & Architecture 6](#_lcp2w07xtvei)

[5.3. Implementation 7](#_7h6mv04bnnnu)

[5.4. Testing and Evaluation 8](#_clmshmt7f0fg)

[**6. Appendix 8**](#_hg17fu3yd4dt)

[6.1. Definitions 8](#_dvhadhgvzauj)

[6.2. Images 8](#_bhob1pdg7ouk)

# Project Objectives

## 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. *Dynamic Load Balancer* - Design and develop a prototype load balancer that is tailored for cloud-based telemetry systems.
    2. *Integration of Load Balancer with Existing SDN Architecture* - Add the ability to the load balancer to continuously monitor network traffic paths in real-time.
    3. *Scalability & 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.

# Background

## Existing SDN Architecture

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.

## Expansion into SDN Load Balancing

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.

# Contributions

## Development of a Dynamic Load Balancer

The main contribution of this project is the development of a dynamic load balancer that integrates SDN and Kafka technologies. This addresses a gap in the market for network management products by allowing real-time traffic management without the need for costly proprietary software or human intervention.

## Real-Time Traffic Management

Current off-the-shelf solutions struggle with real-time traffic management due to their lack of scalability, flexibility, and adaptability. By leveraging Kafka’s real-time data streaming features, which provide continuous updates on network traffic and resource usage, the load balancer will improve decision-making for traffic management.

## Integration with SDN for Dynamic Reconfiguration

The project will also utilize SDN's centralized control for dynamic network reconfiguration. This allows the load balancer to adapt in real-time to changing network conditions, reducing latency and ensuring optimal resource allocation.

## Scalability & Flexibility

The combination of SDN and Kafka will produce a scalable and flexible load balancing solution that can be applied across a variety of network environments, from small business networks to large cloud-based systems. Unlike proprietary load balancers, this solution can be customized with specific algorithms to meet unique network demands.

## Capstone Worthiness

The complexity of integrating Kafka, SDN controllers, and custom load balancing algorithms into a cohesive system makes this project a compelling capstone. It requires a deep understanding of network systems, event-driven architectures, and real-time decision-making. By addressing this challenge, the project aims to push the boundaries of traditional load balancing and introduce a novel approach to network management.

# Broader Impacts

## Environmental Impact

The successful implementation of this project could have significant environmental benefits. By using dynamic load balancing, networks can allocate resources more efficiently, which reduces energy consumption and operational costs. Idle or overloaded servers contribute to resource waste and higher energy use. This project ties directly to the principles of green computing, encouraging more sustainable and responsible use of technology by reducing the environmental footprint of network operations.

## Social Impact

From a social standpoint, the enhanced reliability and efficiency provided by the dynamic load balancer could greatly improve the functionality of critical infrastructure networks. This technology has potential applications in emergency response systems, public services, and hospitals, where network reliability is crucial. By ensuring that these networks operate smoothly, the load balancer could lead to improved quality of service and even save lives by ensuring critical systems remain available and responsive when they are needed most.

## Economic Impact

The economic implications of this project are significant as well. By providing an open-source and flexible load balancing solution, small and medium-sized enterprises (SMEs) could adopt cloud-based architectures or distributed networks without needing to invest in expensive proprietary solutions. This would make advanced network management more accessible, enabling these businesses to scale safely and efficiently. The resulting increase in innovation and competition within the field could drive broader adoption of more cost-effective network solutions.

## Relevance to Current State of the Art

This project is closely related to the current state-of-the-art in network management, particularly in terms of green computing and distributed network architectures. By contributing a scalable and adaptable load balancing solution that can handle real-time traffic, it advances both sustainability goals and network optimization efforts. The completion of this project will not only benefit the field of network management but also provide a valuable, cost-effective alternative to proprietary solutions. Its relevance to businesses and institutions that rely on cloud-based architectures and dynamic network systems makes this work highly significant.

# Approach & Methodology

## 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

* + 1. *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.
    2. *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.
    3. *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.
    4. *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

* + 1. *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.
    2. *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.
    3. *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.
    4. *Software Resources*

1. *Go* - The primary language for implementing the load balancer and processing telemetry data
2. *Kafka* - A distributed streaming platform for handling real-time telemetry data
3. *GoFlow2* - A telemetry tool for collecting flow records from network switches.
4. *Docker* - Containerization technology for deploying the load balancer and other microservices.
5. *Mininet* - A network simulator for emulating the SDN environment and generating test traffic.
6. *OpenFlow* - The protocol used by the SDN controller to manage flow tables in OVS instances.
   * 1. *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

* + 1. *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.
    2. *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.
    3. *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.

# Appendix

## Terminology

* Cloud-Based Telemetry: A system for collecting and transmitting data from cloud-based infrastructure to monitor performance, resource usage, and network traffic.
* Load Balancer: A tool that distributes incoming network traffic across multiple servers or paths to ensure optimal resource use, prevent overload, and reduce latency.
* Software-Defined Networking (SDN): A networking architecture that separates the control plane from the data plane, allowing centralized control of network traffic through a programmable SDN controller.
* Kafka: A distributed streaming platform used for real-time data streaming, commonly employed for managing and processing high volumes of telemetry data.
* GoFlow2: An open-source tool for collecting flow records (telemetry data) from network devices, particularly Open vSwitch (OVS) instances.
* Open vSwitch (OVS): A virtual switch designed for network automation and integration with SDN environments, allowing programmatic control of network traffic.
* Telemetry: The process of collecting data remotely for monitoring or analysis, particularly in cloud or networking contexts to track system performance and network conditions.
* Round Robin Algorithm: A simple load balancing method that distributes requests evenly among available servers by rotating through each one in sequence.
* Least Connection Algorithm: A load balancing algorithm that directs traffic to the server or path with the fewest active connections to ensure balanced resource distribution.
* Least Response Algorithm: A load balancing algorithm that routes traffic based on server response times, sending requests to the server that responds fastest.
* Source IP Hash: A load balancing method that uses the source IP address of incoming traffic to determine the server or path to which the traffic should be routed.
* Docker: A platform that allows developers to package applications into containers, ensuring consistency across different environments for deployment and testing.
* Mininet: A network simulator that creates virtual networks for testing SDN applications and network performance.
* OpenFlow: A protocol used in SDN environments to control the flow of traffic through network switches, allowing for dynamic path adjustments.
* Latency: The time it takes for data to travel from the source to the destination in a network. Lower latency improves the responsiveness of network services.
* Packet Loss: The failure of packets to reach their destination across a network, typically due to congestion or faulty network infrastructure.
* SDN Controller: The centralized software application that manages flow control in an SDN environment, determining the paths that data should take through the network.
* Flow Table: A data structure in SDN switches (such as OVS) that dictates how packets are forwarded through the network, based on the rules installed by the SDN controller.

## Images