FlowSensei

Computer Communication Based Software Development Workshop

Project Documentation

Submitters:

- Asaf Koenigsberg
- Ofek Markus
- Itamar Tennenbaum

Mentor:

- Dr. Hadar Binsky

Github: FlowSensei

Project Description

FlowSensei, a SaaS that leverages RouterOS(Linux-based OS for routers), aims to maintain vital network traffic during periods of high demand. By adjusting transport based on real-time conditions and predefined policies, FlowSensei ensures the smooth operation of critical network services and activities.

The Problem

During periods of high network load, non-critical activities can overwhelm crucial services, leading to performance degradation, disruptions and reduced productivity.
Our project seeks to address this by focusing on traffic prioritization and protection during high-demand situations, thereby ensuring uninterrupted operation of critical services.

Approach

Our approach involves a microservices architecture and software defined routers, that can be versatility configured via an API.

The system initializes a priority queue, for each new router, that contains predefined common services. The user can add new services or remove current ones.

When defining the service, the user enters the following parameters:

1. Necessary Parameters:
    1.1. service: The name of the service
    1.2. protocol: The protocol to match (e.g., TCP, UDP).
    1.3. dstPort: The destination port to match.[one or range]
2. Optional Parameters:
    2.1. srcAddress: The source address to match.
    2.2. dstAddress: The destination address to match.
    2.3. srcPort: The source port to match.[one or range]

The router optimizes network performance during high-demand periods by utilizing real-time monitoring and prioritizing traffic based on the priority queue.

It is important to note that FlowSensei is a web app for handling multiple routers globally and not a SDN implementation. Meaning, each router operates without being aware of the others.

Other Approaches

- Static QoS Configurations
  - Description: Predefined prioritization rules. These settings are manually configured and do not adapt to changing network conditions.
  - Comparison with FlowSensei:
    - Flexibility: FlowSensei dynamically adapts to real-time network conditions, unlike static QoS configurations that are rigid and require manual adjustments. This allows for seamless performance optimization without constant intervention.
    - Ease of Use: Leveraging RouterOS, FlowSensei offers a user-friendly interface that simplifies service prioritization. In contrast, static QoS requires technical expertise and manual configuration, making it more complex and time-consuming.
    - Adaptability: FlowSensei automatically adjusts bandwidth limits and prioritization without user input, providing a responsive solution to network changes. Static QoS settings remain fixed unless manually reconfigured, limiting their adaptability.

- Manual Traffic Shaping
    - Description: Manual traffic shaping controls data flow by limiting transmission rates based on predefined policies or rules. It requires regular manual adjustments to stay effective.
    - Comparison with FlowSensei:
        - Flexibility: Manual traffic shaping, like Static QoS, requires constant updates to adapt to network changes, making it inflexible. FlowSensei offers dynamic prioritization based on real-time analysis, eliminating the need for manual adjustments.
        - Ease of Use: FlowSensei simplifies traffic management by automating the process, providing a more user-friendly experience. In contrast, manual traffic shaping requires technical expertise and ongoing management, making it more cumbersome.
        - Scalability: FlowSensei efficiently scales in complex network environments by automatically adjusting to traffic demands. Manual traffic shaping, however, becomes increasingly difficult to manage as the network grows in size and complexity.

- Simple Priority Queuing
    - Description: Simple priority queuing manages traffic by assigning packets to queues based on priority, processing higher-priority packets first. Lower-priority packets may be delayed or dropped as needed.
    - Comparison with FlowSensei:
        - Flexibility: Simple priority queuing is inflexible, relying on fixed priority levels without adapting to network changes. FlowSensei offers greater flexibility by dynamically adjusting prioritization based on real-time conditions and policies.
        - Ease of Use: FlowSensei provides a user-friendly experience, enabling quick adjustments without complex management. Simple priority queuing, though straightforward, still requires manual configuration and lacks the ease of FlowSensei's automation.
        - Adaptability: FlowSensei excels in adaptability by analyzing traffic and adjusting priorities in real-time. Simple priority queuing is static once configured, unable to dynamically respond to network changes.

- Software-Defined Networking (SDN)
  - Description: SDN centralizes network management through software, enabling dynamic and automated control by separating the control plane from the data plane.
  - Comparison with FlowSensei:
    - Flexibility: SDN provides high flexibility with centralized control and dynamic network programming, making it ideal for complex networks. FlowSensei also offers flexibility, particularly in environments using RouterOS, though SDN may be more suitable for large-scale networks.
    - Ease of Use: FlowSensei is designed for ease of use in small to medium networks, where the complexity of SDN might be unnecessary. SDN, while powerful, often has a steep learning curve and requires sophisticated setup, making FlowSensei a more approachable choice for simplicity.
    - Scalability: SDN excels in scalability for large environments with centralized management. FlowSensei scales well for small to medium networks but may not match the scalability of a full SDN solution in very large deployments.

Expected Users

1. Everyday users: Our tool provides them with a simple and intuitive interface to prioritize network activities based on their preferences, ensuring a seamless and enjoyable online experience.
2. Network Admins in commercial small tech companies: Our tool offers an intuitive interface to prioritize critical tasks, enhancing network performance and reliability for seamless business operations.
3. System engineers: Our tool provides them with a user-friendly interface to fine-tune traffic management, optimizing performance and resource allocation for robust system operation.

Main Features

1. Real-Time Monitoring: The system continuously routes internet packets into the ELK stack, storing and presenting meaningful transport data to the user.
2. Policy Configuration: Users can define and modify service priorities to their specific requirements.
3. Critical Service Protection: Ensuring uninterrupted operation of prioritized services during high-demand situations.

Future features implementations

- Auto-Suggest Service Prioritization: The system could scan router transport data during inactive hours and suggest prioritization for services that consistently show high demand or criticality during peak periods.
- Machine Learning-Based Traffic Prediction: Integrating machine learning models could predict high-demand periods based on historical data, allowing for preemptive adjustments to network priorities without user intervention. Currently, high-demand periods are detected through bandwidth sampling, but machine learning could enable earlier detection.
- Anomaly Detection: An anomaly detection system could identify unusual traffic patterns or potential threats, dynamically adjusting policies to protect critical services.

Tech Stack

- Each microservice in the server side is built with Express.js
- Frontend app is built with ReactJS
- RouterClient service uses the `'node-routeros'` npm library, which facilitates communication with MikroTik routers. It provides an interface for sending commands and receiving data from RouterOS, allowing microservices to manage and control network configurations effectively.
- In Rabbitmq, each client has a corresponding queue and the data is being routed to queues using exchanges.
  Specifically on FlowSensei:
    - Each microservices has a queue
    - There is only one exchange, to the microservice that handles the RouterOS API calls, named `requests_exchange`.
    - The results return back to the appropriate queues using messages special feature “msg.properties.replyTo” with message ID.
    - Message Formats(JSON-formatted):

      Login Message:

```
{
  "type": "login",
  "username": "<username>",
  "password": "<password>",
  "publicIp": "<public_ip>",
  "routerID": "<router_id>"
}
```

Connection Mark:

```
{
    "type": "connection-mark",
    "chain": "prerouting",
    "connectionMark": "<service_name>",
    "protocol": "<protocol>",
    "dstPort": "<comma_separated_ports>",
    "passthrough": "yes"
}
```

Packet Mark:

```
{
    "type": "packet-mark",
    "chain": "prerouting",
    "connectionMark": "<connection_mark>",
    "packetMark": "<connection_mark>packet",
    "passthrough": "no"
}
```

Update Node Priority

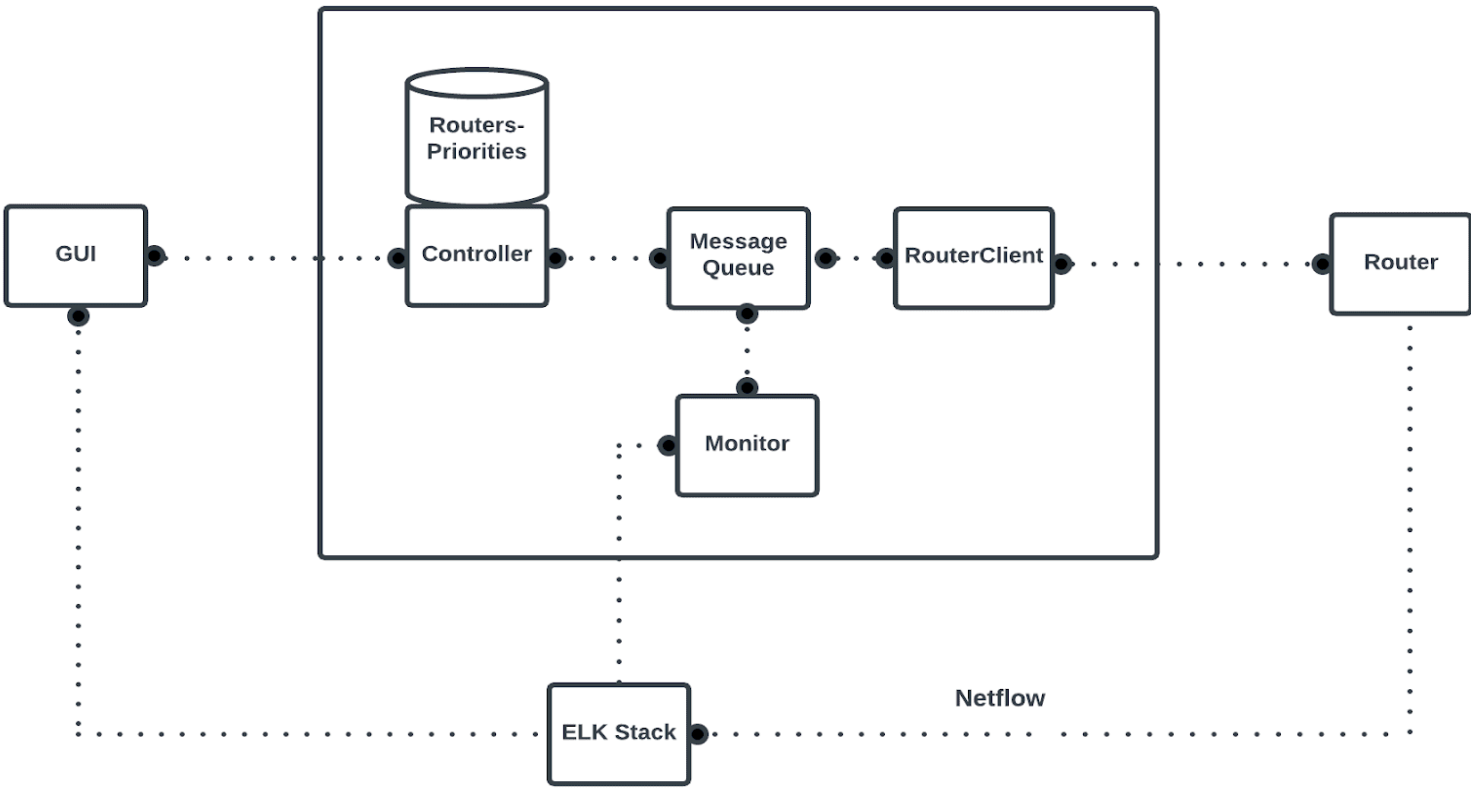
```
{
    "type": "update-node-priority",
    "name": "<service_name>",
    "newPriority": "<new_priority>"
 }
```

Logout

```
{
   "type": "logout"
}
```

- FlowSensei supports any router that runs on RouterOS, allowing for dynamic traffic control and effective policy enforcement, making it an ideal choice for adaptive network management.
- ELK(Elasticsearch, logstash and Kibana) are used together for monitoring, storing and visualizing the transport.

System Components



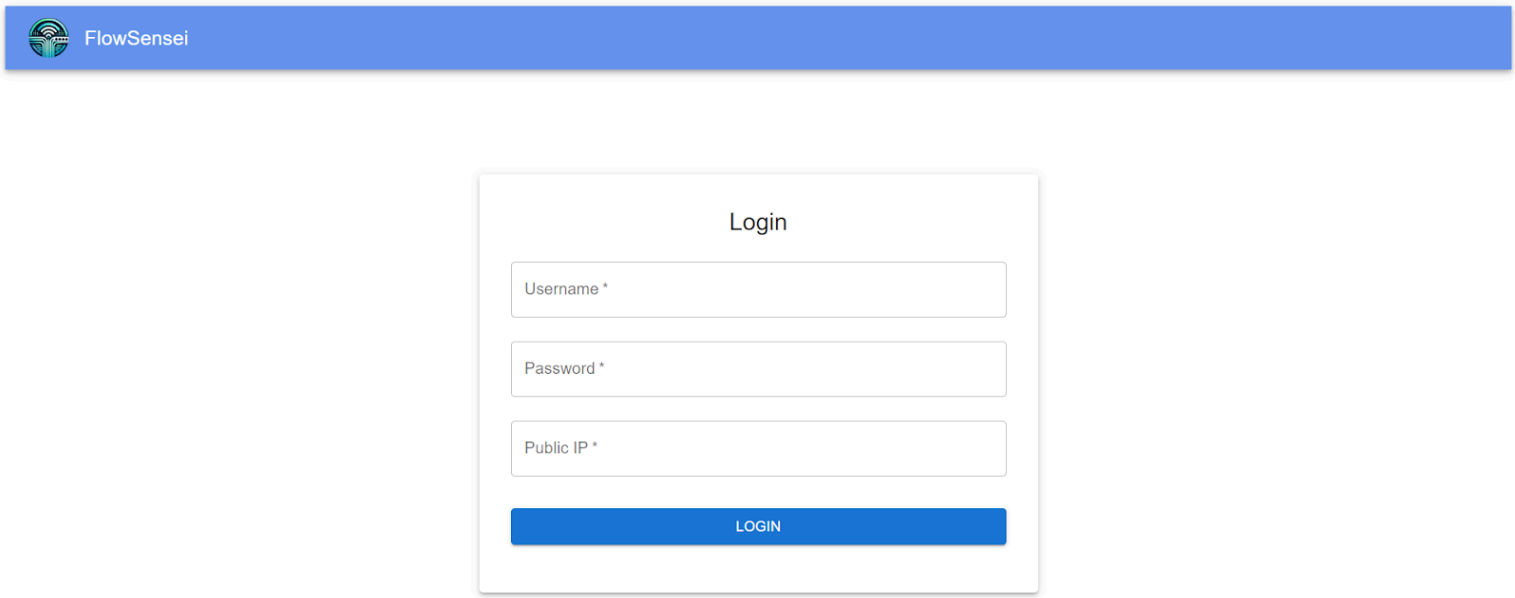
- GUI
  - ReactJS app with an easy to use interface
  - Views
    - Login page
    - Visualization page- useful visualizations in Kibana
    - Services page- A stack of services that represent the priority hierarchy

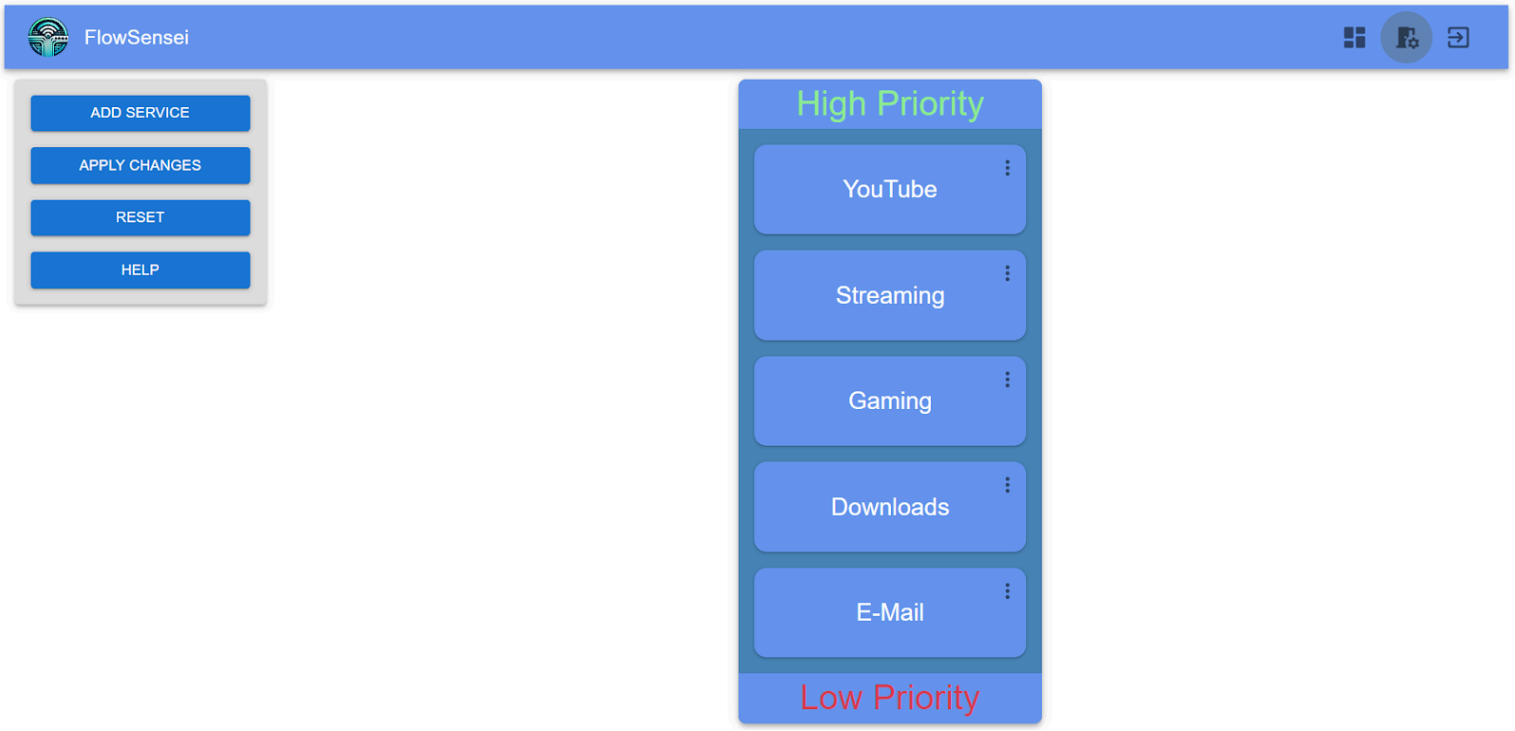
- Controller
  - An Express.js microservice for handling user login, requests and dynamic prioritizing
  - After the user successfully logs in, the service signs a jwt and returns it for further secured requests
  - Endpoints
    - signup[POST]: passes connection request to the router API
    - login[GET]: restoring priority preferences and signing a security token for further activity
    - logout[POST]: deleting the security token
    - service[POST]: adds new service to the priority queue
    - service[PUT]: updates existing service priority
- Message queue
    - Standard RabbitMQ broker for allowing async processing of the requests by RouterClient, allowing more flexibility and improve performance
- RouterClient
    - An Express.js microservice for Handling API calls to routers
    - Uses an internal cache for keeping active sessions without the need for storing credentials.
    - Every 3 hours, the service automatically adjusts the prioritized queue's bandwidth limit based on current usage, ensuring critical services maintain performance during high demand without starving other services. Burst functionality allows temporary bandwidth increases during spikes, with limits on duration and extent, providing flexibility while maintaining overall network control.

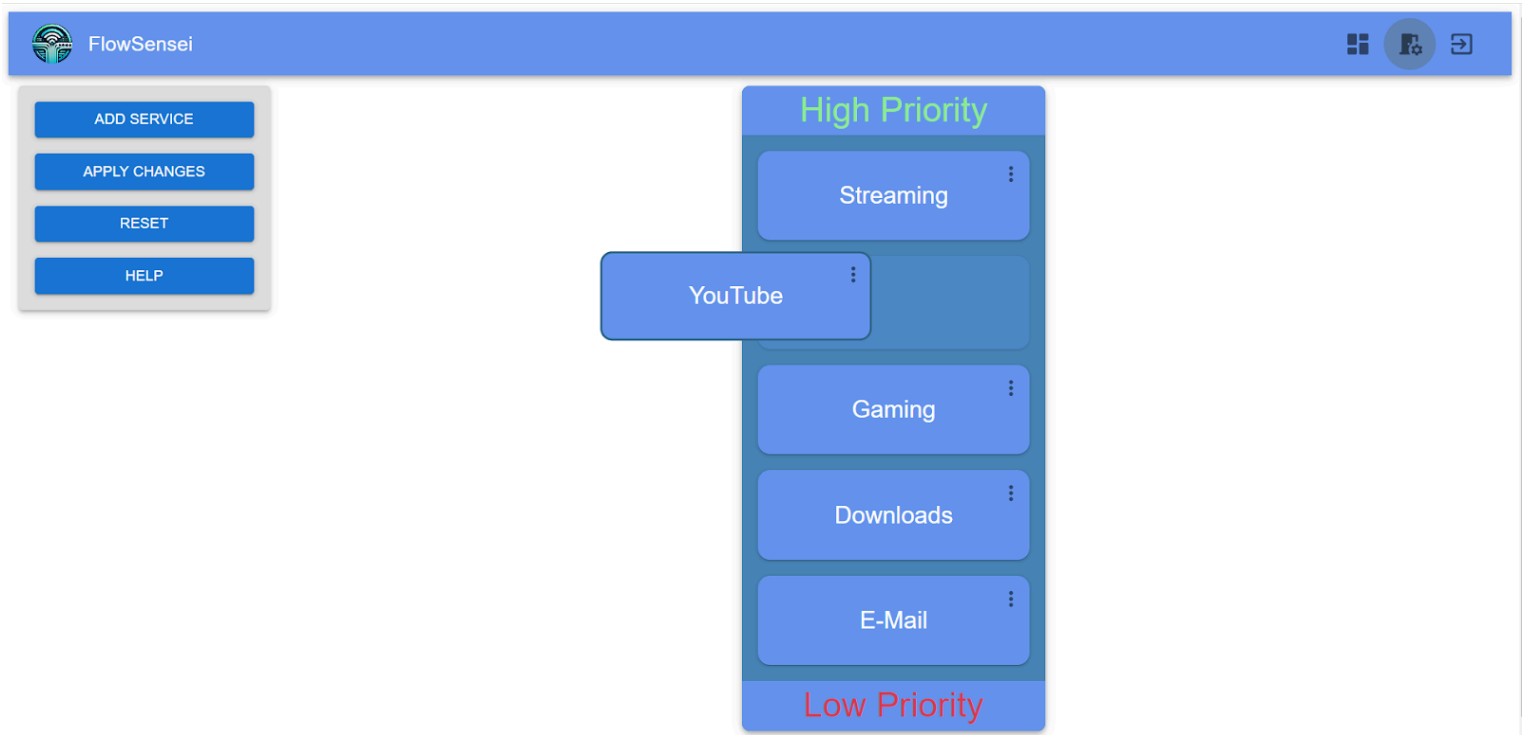
- Monitor
    - An Express.js microservice designed to automate the setup of ELK (Elasticsearch, Logstash, Kibana) for new routers in a network. It programmatically creates Elasticsearch index templates, Kibana index patterns, visualizations, and dashboards for each router. Upon receiving a new router's details via an API request, the service configures the necessary ELK components to monitor and visualize the router's network traffic. This enables seamless integration of new routers into the existing ELK stack without manual intervention.

Important Note: The microservices system can handle multiple users for multiple routers.

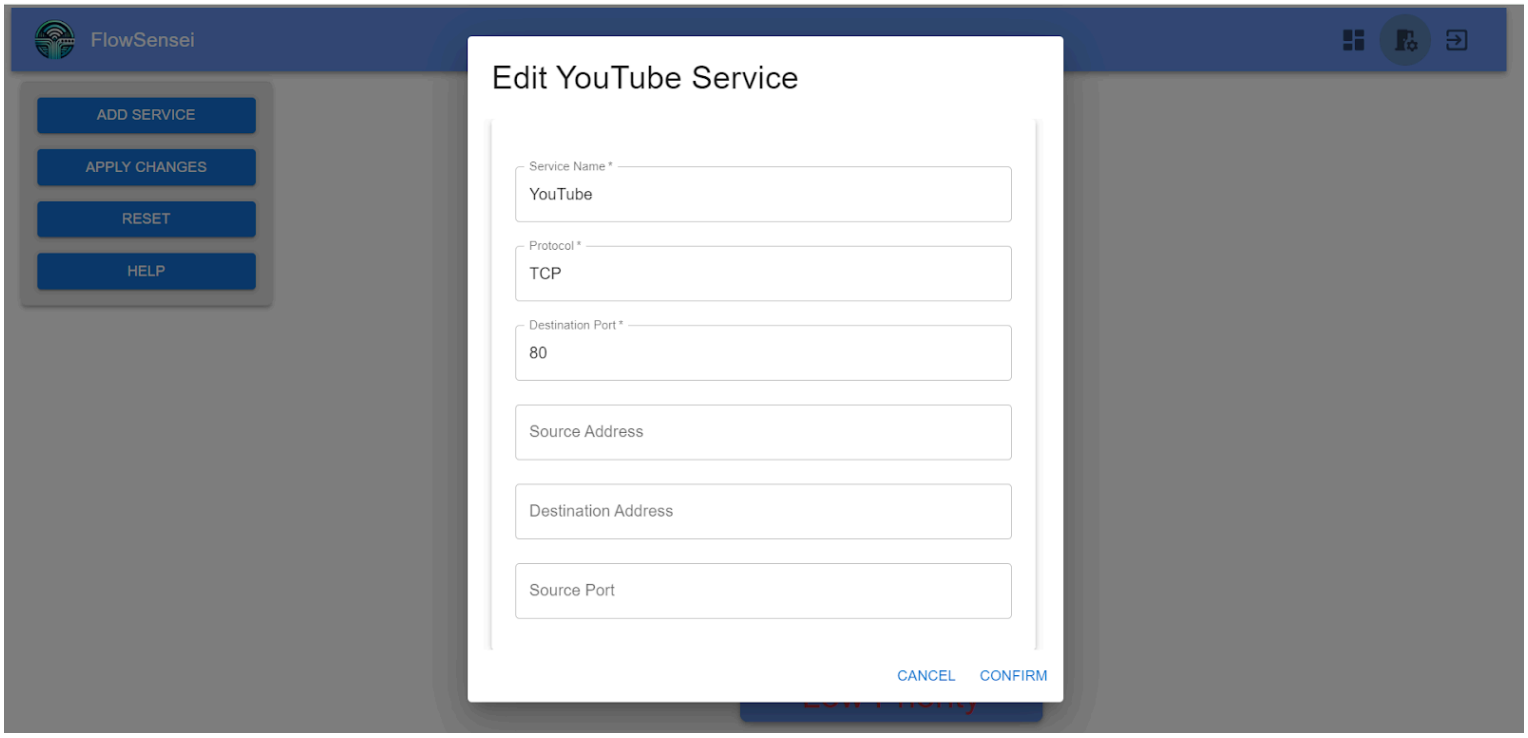
User Interface Screenshots

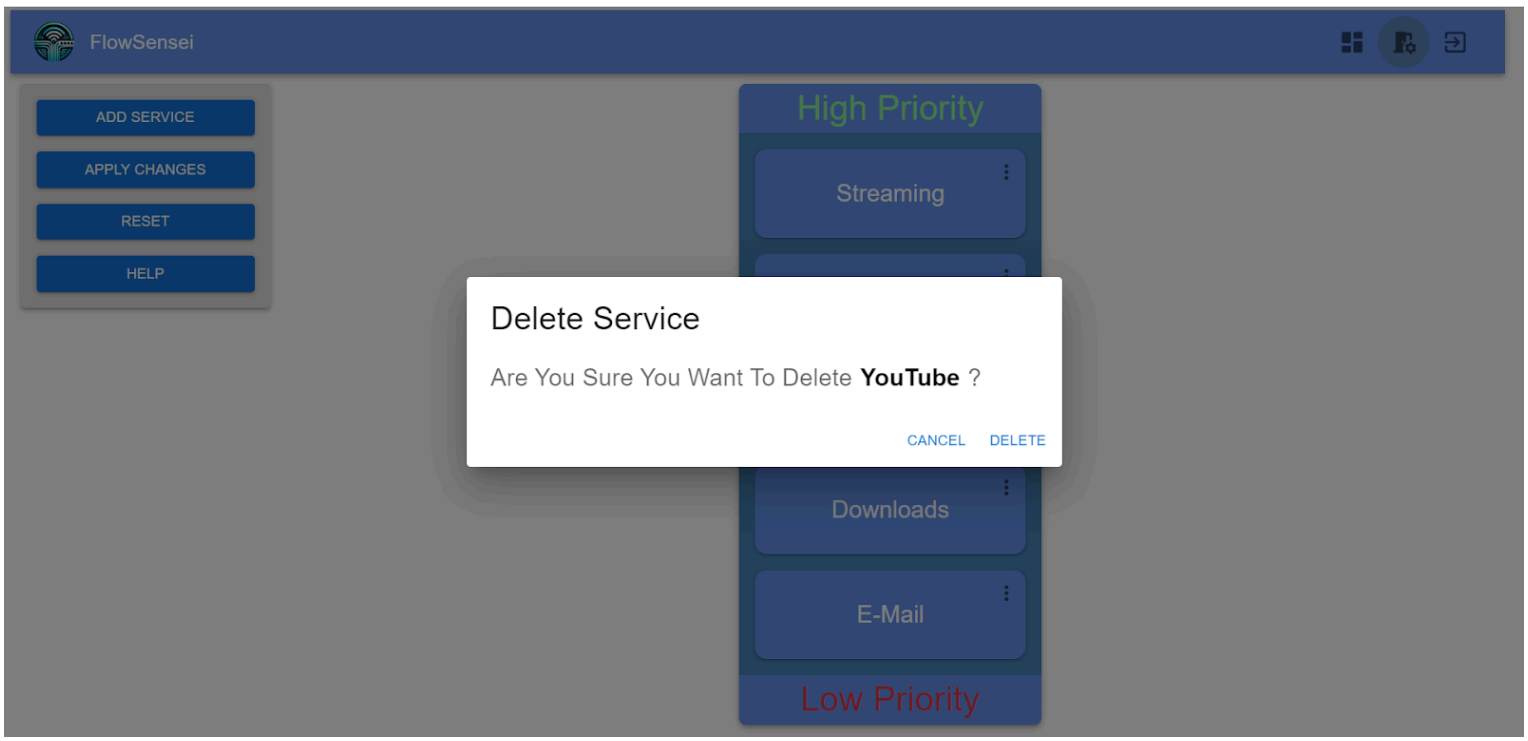


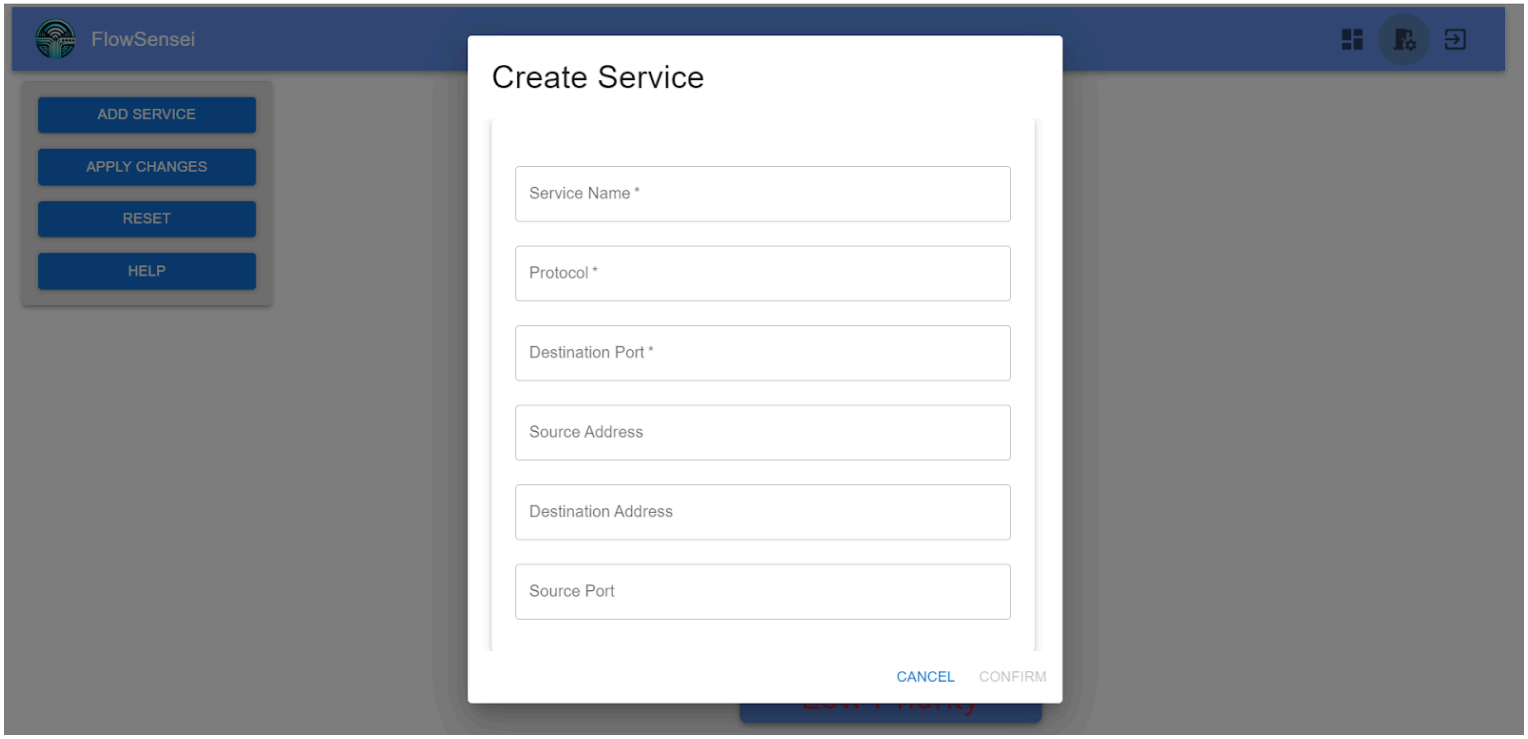


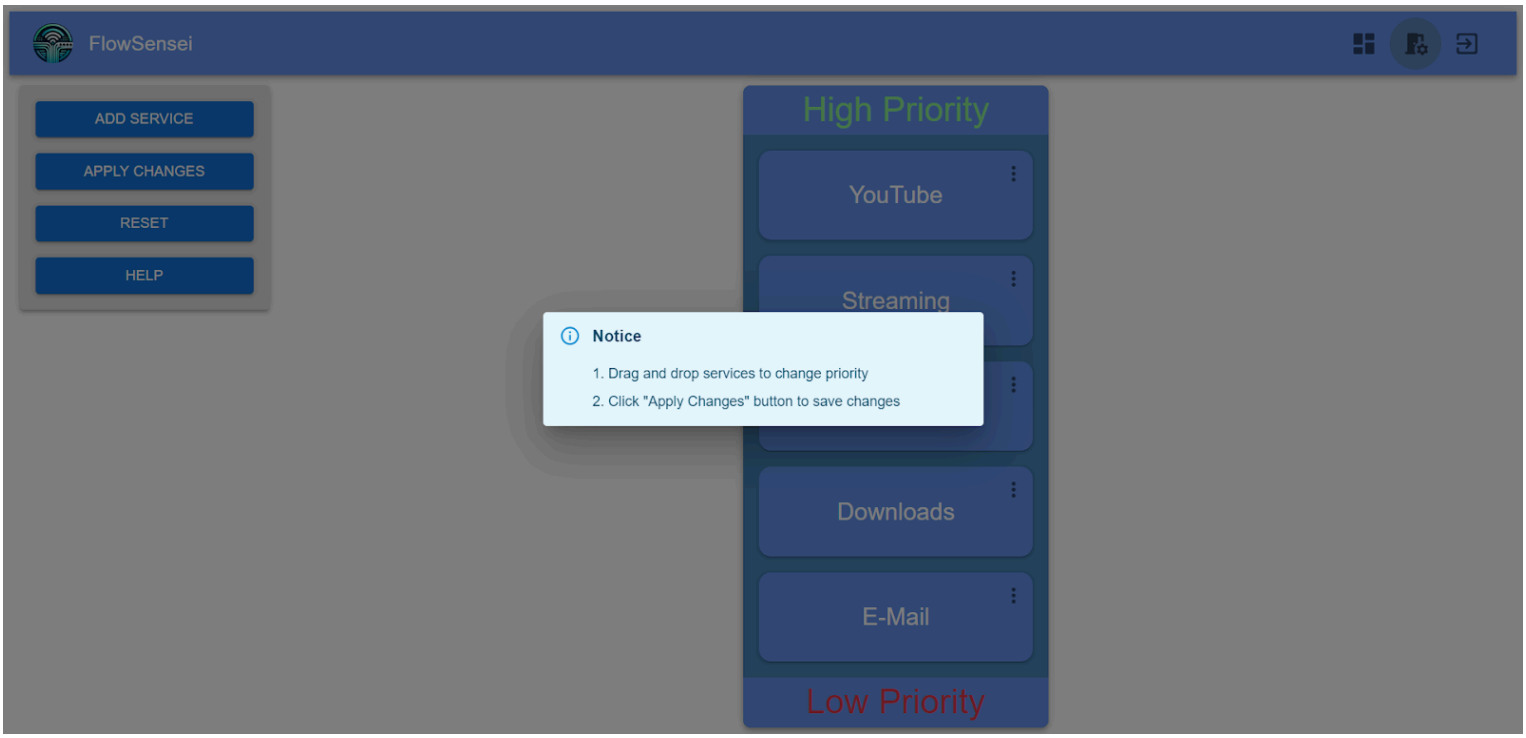

FlowSensei
ADD SERVICE
APPLY CHANGES
RESET
HELP
High Priority
Streaming
YouTube
Gaming
Downloads
E-Mail
Low Priority


FlowSensei
ADD SERVICE
APPLY CHANGES
RESET
HELP
High Priority
Streaming
YouTube
Edit
Delete
Gaming
Downloads
E-Mail
Low Priority


FlowSensei
ADD SERVICE
APPLY CHANGES
RESET
HELP
Edit YouTube Service
Service Name *
YouTube
Protocol *
TCP
Destination Port *
80
Source Address
Destination Address
Source Port
CANCEL
CONFIRM


FlowSensei
ADD SERVICE
APPLY CHANGES
RESET
HELP
High Priority
Streaming
Delete Service
Are You Sure You Want To Delete YouTube ?
CANCEL
DELETE
Downloads
E-Mail
Low Priority


FlowSensei
ADD SERVICE
APPLY CHANGES
RESET
HELP
Create Service
Service Name *
Protocol *
Destination Port *
Source Address
Destination Address
Source Port
CANCEL
CONFIRM


FlowSensei
ADD SERVICE
APPLY CHANGES
RESET
HELP
High Priority
YouTube
Streaming
Downloads
E-Mail
Low Priority
Notice
1. Drag and drop services to change priority
2. Click "Apply Changes" button to save changes

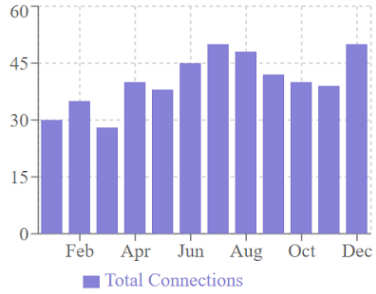
FlowSensei

Total Devices
50

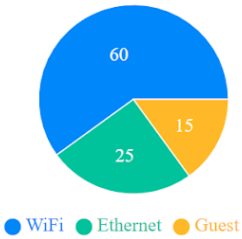
Active Connections
35

Data Usage (GB)
120

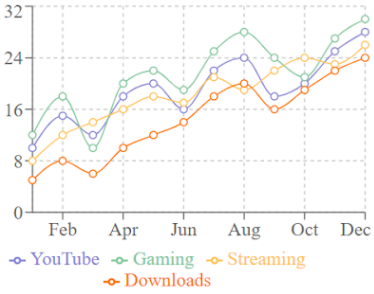
Monthly Active Connections



Connection Types Distribution

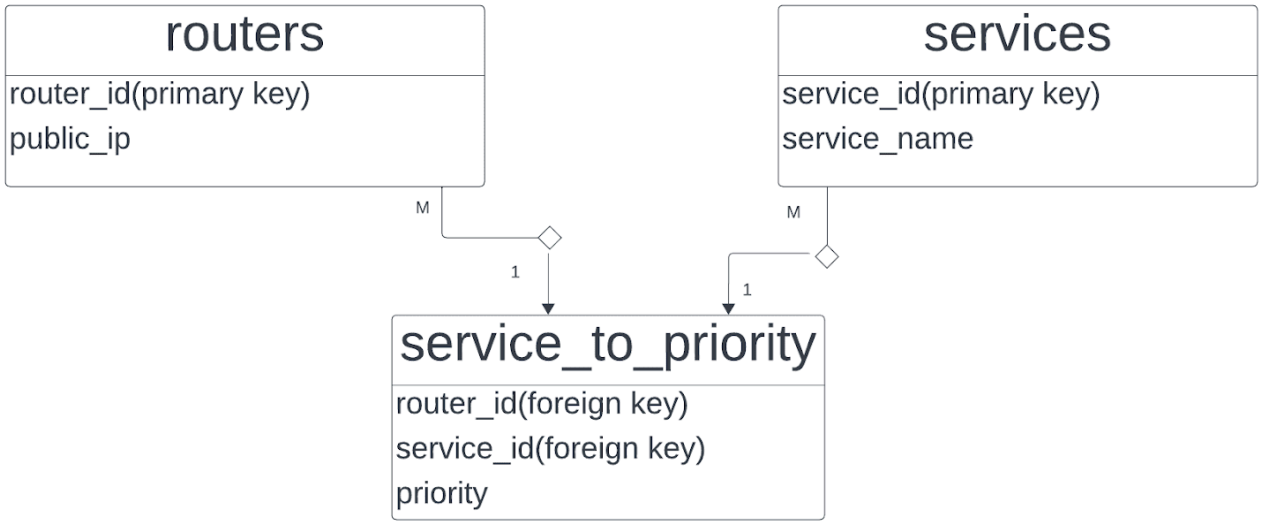


Monthly Data Usage By Service (GB)



Database

- Router-priorities Database: a postgres DB with the following tables



- Sessions Cache: a redis DB is deployed to save active sessions from multiple users.
    - Key: RouterID(assuming each router has a single administrator)
    - Value: The session itself