Tokyo Traffic Lights

Khalil Ouald Chaib

January 2025

1 Introduction

Tokyo has a vast network of traffic lights—approximately 150,000 in operation. However, synchronization issues have led to increased traffic congestion and slower pedestrian crossings, resulting in widespread dissatisfaction among drivers and pedestrians.

To address this challenge, I plan to implement a solution based on reinforcement learning. Assuming access to traffic data for Tokyo, this data will be used to train a reinforcement learning model. The model's agent will receive real-time information on the number of cars and pedestrians at specific locations and times. Based on this input, the agent will determine the optimal traffic light timings to minimize congestion and improve traffic flow.

The primary objective of the model is to significantly reduce traffic congestion while enhancing the efficiency of pedestrian crossings. Since no real traffic data is currently available, I will work under the assumption that a pre-trained model is already integrated into the system for demonstration purposes.

2 Software And System Architecture

2.1 Trafic Lights with camera's and edge devices

Each traffic light in the system will be equipped with two dedicated edge devices, each performing distinct yet complementary roles to ensure optimal traffic management.

The first edge device is responsible for receiving updated traffic light timers from the central Traffic Optimization Service. This device also incorporates a default timer to ensure uninterrupted operation in the event of communication failures. It acts as the controller for the traffic lights, seamlessly adjusting the signals based on the latest optimization data while maintaining a fallback mechanism to handle potential disruptions.

The second edge device is designed specifically for processing the data captured by the camera at each intersection. This device uses advanced computer vision algorithms to count the number of vehicles and pedestrians passing through the intersection. It processes the video feed locally, eliminating the need for raw data to be transmitted to central systems, thereby reducing latency and bandwidth usage.

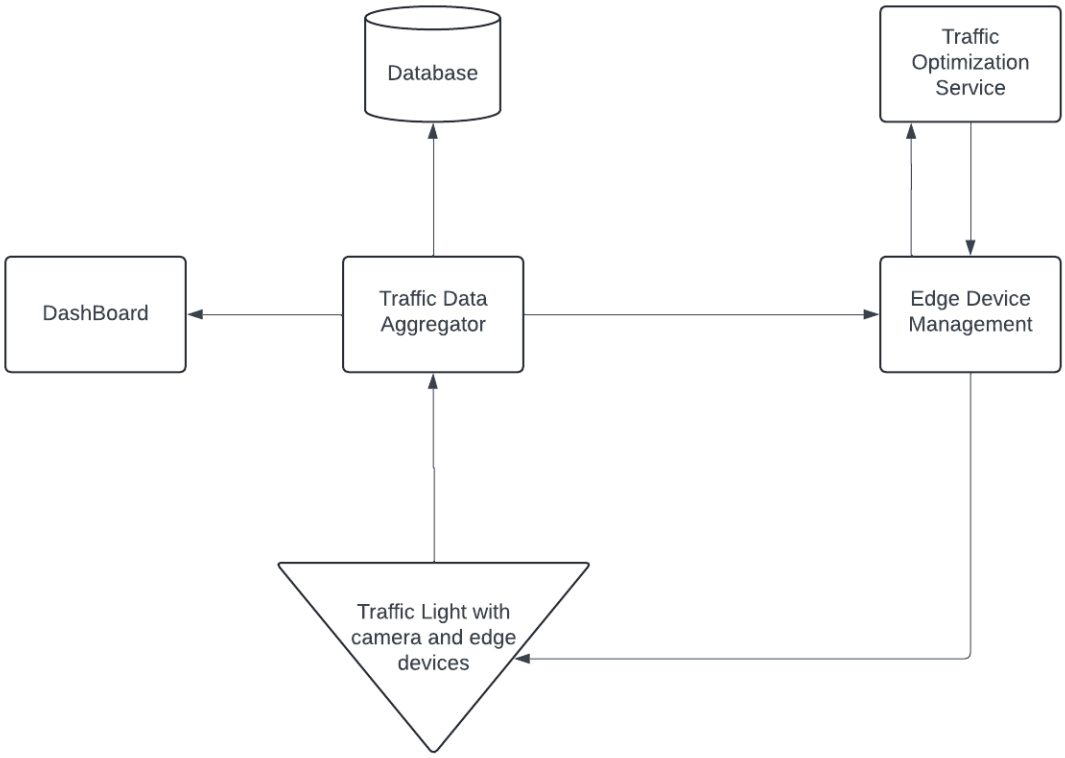

Figure 1: Architecture

The edge device for data processing continuously generates key metrics, including the number of pedestrians and vehicles observed, a timestamp of when the data was collected, and the location of the intersection. This aggregated information is transmitted at regular intervals, typically every few seconds, to the central Traffic Data Aggregator Service.

Component	**Quantity**	**Details**
Traffic Lights	150,000	One per intersection.
Cameras	150,000	One per traffic light, paired with both edge devices.
Primary Edge Devices	150,000	Handles light control and communication.
Dedicated Counting Devices	150,000	Processes video feeds for vehicle/pedestrian counts.

Power Supply	150,000	Grid connection with solar backup for uninterrupted power.

2.2 Traffic Data Aggregator

This micro-service will receive the data, save them in the database for future improvements, it will also forward these data to the dashboard and the Edge Device Management service.

Component	**Description**	**Quantity/Resources Required**
API Gateway	Handles data ingestion from traffic light edge devices. Secures endpoints and validates data.	2 instances (auto-scalable), 2 vCPUs, 4GB RAM each
Traffic Data Aggregator	Processes, normalizes, and forwards traffic data to downstream services.	4 instances (auto-scalable), 4 vCPUs, 8GB RAM each
Database	Stores historical and real-time traffic data for analytics and optimization.	1 managed PostgreSQL instance, 32 vCPUs, 128GB RAM, **10TB storage (scalable)**
Message Queue	Ensures asynchronous, reliable communication to Dashboard and Edge Device Management.	1 managed Kafka cluster, 3 brokers, 8 vCPUs, 16GB RAM each
Monitoring & Metrics	Tracks service performance and key metrics (e.g., traffic volume, latency).	1 managed Prometheus & Grafana stack, 8 vCPUs, 16GB RAM
Load Balancer	Distributes incoming traffic among Traffic Data Aggregator instances.	1 Load Balancer (managed, auto-scaling)

Table 2: System Architecture for Traffic Data Aggregator

2.3 Edge Device Management service

The Edge Device Management Service acts as an intermediary between edge devices (traffic lights) and the Traffic Optimization Service. It forwards real-time traffic data to the Traffic Optimization Service, which processes the data and responds with the most optimal traffic light timings to reduce congestion and improve flow. Based on these optimized timings, the Edge Device Management Service sends updated timing signals to the respective edge devices (traffic lights) at specific locations, ensuring a dynamic and efficient traffic management system.

Component	**Description**	**Quantity/Resources Required**
API Gateway	Manages data forwarding and ensures secure communication between services.	2 instances (auto-scalable), 2 vCPUs, 4GB RAM each
Edge Device Management Service	Processes timing updates and manages communication with edge devices.	4 instances (auto-scalable), 4 vCPUs, 8GB RAM each

Component	Description	Quantity/Resources Required
Message Queue	Ensures reliable delivery of timing signals to edge devices.	1 managed Kafka cluster, 3 brokers, 8 vCPUs, 16GB RAM each
Monitoring & Metrics	Tracks system performance and ensures uptime.	1 Prometheus/Grafana stack, 8 vCPUs, 16GB RAM

Table 3: System Architecture for Edge Device Management Service

2.4 Traffic optimization service

The Traffic Optimization Service addresses the traffic synchronization issue by employing a reinforcement learning (RL) model. The decision network of this model will be trained on historical and real-time traffic data from Tokyo to understand patterns and optimize traffic flow. Using this trained model, the service will process the data received from the Edge Device Management Service and compute the most efficient traffic light timings. These optimized timings are then sent back to the Edge Device Management Service to implement real-time changes at specific traffic lights.

Component	Description	Quantity/Resources Required
API Gateway	Handles incoming traffic data and ensures secure communication.	2 instances (auto-scalable), 2 vCPUs, 4GB RAM each
Traffic Optimization Service	Core microservice running the RL inference engine and orchestrating responses.	4 instances (auto-scalable), 8 vCPUs, 16GB RAM each
Model Training Pipeline	Prepares and trains the RL model on historical traffic data.	Cloud GPU/TPU cluster, 16 GPUs, 128GB RAM per node
Database	Stores training data, logging data, and results.	1 managed PostgreSQL instance, 32 vCPUs, 128GB RAM, 10TB scalable storage
Load Balancer	Distributes inference requests across Traffic Optimization Service instances.	1 load balancer (managed, auto-scaling)
Monitoring & Metrics	Tracks system health, latency, and model performance.	1 Prometheus/Grafana stack, 8 vCPUs, 16GB RAM

Table 4: System Architecture for Traffic Optimization Service