EcoTraffic

Lorenzo Esposito, Alessandro Frisone, Francesco Grassi, Giorgio Venezia

April 18, 2025

Version 1.0.0

Contents

1 The Project and the project goals

1.1 The problem

Two urgent global concerns are environmental sustainability and climate change; because of air pollution and greenhouse gas emissions, transportation, especially urban commuting, contributes to worsening those issues.

1.2 The goal

We want to dynamically modify the duration of traffic lights on the main roads in the city depending on the directions from where we observe the main traffic movements. For instance, if, at a certain point in time, we observe that the traffic flow on a certain road A is significantly higher than in the crossing roads, then we may decide to extend, for instance, for one hour, the duration of green lights on A (and, consequently, extend the duration of red lights in the crossing roads).
We want to analyze the daily traffic patterns and identify possible optimizations in terms of one-way roads, traffic lights configuration, and public transport schedule.
We want to collect information about the planning of events attracting large crowds (e.g., important sport events, concerts, fairs) and define event-specific configurations for traffic lights, roads and public transport schedules.

1.3 Stakeholders

- Drivers: for reduced waiting times stuck in the traffic.
- Citizen: for air pollution, public transport.
- Urban Area manager: for optimization of city viability.
- Events planner: for a better management of events attracting large crowds.

2 Requirement Analysis

2.1 Relevant human and non-human actors

2.1.1 Human Actors

- Drivers: benefit from the EcoTraffic system.
- Citizen: benefit from the EcoTraffic system.
- Urban Area manager: approve or rejects modifications to one-way roads, traffic lights configuration, and public transport schedule.

2.1.2 Non-Human Actors

- Traffic Lights: get its state set from the ET system for a determined time period.
- Sensor Infrastructure: send sensor information to the ET system via data bus.
- Public Transport Microservice: sends public transport schedules to the EcoTraffic system via function calls.
- News Channel: transmits city events information to the ET system.

2.2 Use Cases

2.2.1 Scenarios (to modify)

Traffic light duration adjustment

1. During peak traffic hour, cars take more than the necessary to cross a particular intersection coming from a busy road, while the crossing road are less used.
2. The system sensor measures the times taken and publish them on the data bus.
3. EcoTraffic retrieves the data from the data bus and stores them into a database.
4. EcoTraffic tries to catch misbehaviour.

5. If a misbehaviour is detected, then EcoTraffic compute the new routines time for the traffic light which must be modified.
6. EcoTraffic connects to the traffic light control system providing serial numbers of the traffic lights which must modify the routine with the amount of time interval to be changed.
7. EcoTraffic writes into a log file the modifications done.

Daily Analysis and optimization

1. Ten minutes after midnight, EcoTraffic retrieve daily data from the database.
2. EcoTraffic analyse this data to detect daily traffic pattern and to find possible optimization in term of one-way roads, traffic lights configuration, and public transport schedule.
3. EcoTraffic performs the getScheduleByStreet and the getScheduleByLine offered by the microservice.
4. EcoTraffic tries to reorganize the schedules to optimize the public transport.
5. If possible optimizations are found, EcoTraffic store them into the database.
6. Once the UAM access the system, it presents the optimizations found and wait for the answer of the UAM.
7. EcoTraffic writes into a log file the answer for yearly reporting.

Traffic Zone Optimization

1. The Urban Area Manger asks to EcoTraffic to verify if a certain zone is optimized in terms of one-way roads, traffic lights configuration, and public transport schedule.
2. EcoTraffic retrieve from the database the traffic patterns of the zone.
3. EcoTraffic analyses the data retrieved and tries to minimize the medium amount of time taken to cross an intersection into the zone.
4. EcoTraffic performs the getScheduleByStreet and the getScheduleByLine offered by the microservice to get all the bus lines that has a stop into the area to optimize.
5. After the information arrived, EcoTraffic tries to reschedule the timetable of the bus lines in such a way to minimize the traffic. (i.e if in the zone there's a school, then the system could anticipate the timetable of the stops near that to avoid the students to arrive late or could suggest increasing the number of buses in the area).
6. EcoTraffic presents to the UAM the recommendations and waits till the UAM decide to accept or to reject the recommendation and store the decision.
7. EcoTraffic writes into a log file the answer for yearly reporting.

Event-specific configurations

1. News channel publishes information about upcoming event with the expected attendance.
2. EcoTraffic receives the event information via integration with news channel.
3. EcoTraffic automatically categorizes the event by the attendance.
4. EcoTraffic analyzes historical traffic patterns from similar events.
5. EcoTraffic retrieves public transport schedules via microservice using getScheduleByStreet and getScheduleByLine operations.
6. EcoTraffic generates event-specific configuration recommendations for traffic lights and roads.
7. The UAM accesses to EcoTraffic and reviews the suggestions proposed and decides to accept or to reject them.
8. EcoTraffic writes into a log file the answer for yearly reporting.

System monitors traffic during special event (forse da togliere, altrimenti si deve modificare l'architettura)

1. EcoTraffic detect the presence of a special event happening today.
2. Traffic sensors detect traffic flow.
3. Traffic sensors publish data to message bus.
4. EcoTraffic receives sensor data from message bus.
5. EcoTraffic retrieve the data in the area near the event.
6. EcoTraffic implements temporary traffic light timing changes if needed.
7. EcoTraffic logs all adjustments and their effectiveness
8. EcoTraffic updates event-specific configuration data based on observations for future similar events

Citizen views public traffic reports

1. Citizen accesses EcoTraffic public portal.
2. EcoTraffic presents options for viewing reports.
3. Citizen selects the preferred option.
   (a) Citizen selects "Daily Traffic Reports" and choose the date and time.
      i. EcoTraffic retrieves daily report data from database service.
      ii. EcoTraffic displays report showing:

a. Average traffic flow on main roads.
b. Visualization of peak congestion periods.
c. List of actions taken automatically.
d. Traffic prediction for tomorrow.

(b) Citizen selects "Yearly Reports" option.

i. EcoTraffic displays yearly report options.
ii. EcoTraffic retrieves yearly report data from database service.
iii. EcoTraffic displays comprehensive report showing:
i Suggested actions that were accepted.
ii Suggested actions that were rejected.

UAM views optimization to approve or reject

1. UAM access to EcoTraffic public portal.
2. EcoTraffic displays all the suggested changes not already approved or rejected.
3. The UAM analyse each suggestion and decide to approve or reject choosing an option displayed.
4. EcoTraffic writes into a log file the answer for yearly reporting.

2.2.2 Use case diagrams (to modify)

Traffic light duration adjustment

Actors: Sensors, Traffic lights, database.

Entry condition: After sensors measures new crossing times, new data arrives on the bus.

Flow of events:

- EcoTraffic reads data from the message bus.
- EcoTraffic stores in the database the data received.
- EcoTraffic compares the crossing times of the roads in the crossings.
- If EcoTraffic detects traffic load imbalance, the system computes the green light duration to optimize vehicle flow in the more congested direction.

Exit condition: The system sends the adjusted times to the traffics lights control system.

Daily Analysis and Optimization

Actors: Webservice, database

Entry condition: It's ten past midnight.

Flow of events:

- EcoTraffic queries the database to get all the crossing times measured in the day before.
- EcoTraffic analyses the data retrieved to obtain traffic pattern.
- EcoTraffic analyses the traffic pattern to find possible optimization in terms of one-way roads and traffic lights configuration.
- EcoTraffic retrieves public transport schedules via microservice using getScheduleByStreet and getScheduleByLine operations.
- EcoTraffic tries to find better schedules for the public transport line.

Exit condition: The system write into the database the possible optimization found.

Traffic zone optimization

Actors: Urban area manager, Public Transport Microservice, database

Entry condition: UAM accesses EcoTraffic public portal and decide to request optimization of a certain area.

Flow of events:

- EcoTraffic queries the database to obtain the crossing times of the zone indicated by the UAM
- EcoTraffic analyses the data to find possible optimization in terms of one-way roads and traffic lights configuration.
- EcoTraffic retrieves public transport schedules via microservice using getScheduleByStreet and getScheduleByLine operations.
- EcoTraffic tries to find better schedules for the public transport line.
- The system sends the new scheduling and configurations proposal to the UAM for approval.
- The system waits for the answer.
- When the decision is taken, the answer is written into a log file.

Exit condition: Successful write of the log.

Event-specific configurations

Actors: News channel, UAM, Public transport microservice

Entry condition: News channel publishes information about upcoming events.

Flow of events:

- EcoTraffic categorizes the scale of the event by attendance.
- EcoTraffic investigates the database for historical traffic patterns and past decision tskrn in similar events in previous log files.
- EcoTraffic retrieves public transport schedules via microservice using getScheduleByStreet and getScheduleByLine operations.
- Using the collected data, EcoTraffic generates event-specific configurations that are optimized in terms of one-way roads, traffic lights configuration and public transport schedules.

Exit condition: The system write into the database the possible optimization found.

System monitors traffic during special event

Actors: Sensors, traffic lights, Public transport microservice

Entry condition: EcoTraffic detects an event on that specifi day from the News channel.

Flow of events:

- The system looks for historical traffic patterns in the area and at the time before/during/after the event.
- The system retrieves public transport schedule via microservice.
- The system periodically collects data from the bus. todo
- The system periodically compute an optimization for the traffic lights on the roads impacted by the event using the collected data.
- The system updates event-specific configuration data based on observations for future similar events.
- The system logs all adjustments and their effectiveness.

Exit condition: Successful write of the log.

Citizen views public traffic reports

Actors: Citizen

Entry condition: Citizen accesses EcoTraffic public portal.

Flow of events:

- The system presents options for viewing reports.
- The citizen selects the type of information that they want to be displayed.
- The system retrieves daily or yearly data from database service.
- The system elaborates the data to facilitate interpretation.
- The system displays a report with the elaborated data.
- The citizen chooses between seeing more information or exiting the portal.

Exit condition: The citizen exits the portal.

UAM views optimization to approve or reject

Actors: UAM

Entry condition: UAM accesses EcoTraffic public portal and decide to view the optimization waiting for approval.

Flow of events:

- EcoTraffic queries the database for the optimization proposals waiting for approval.
- EcoTraffic elaborates the data to facilitate interpretation.
- EcoTraffic displays all the decision to be taken.
- EcoTraffic waits for the UAM to answer to each request.
- Every time a decision is taken EcoTraffic stores it into a log file.

Exit condition: All the decision have been taken.

2.3 Domain assumption

1. The sensor infrastructure works correctly and with low latency 24/7.
2. The traffic lights are not faulty and set their state correctly in time from the ET system.
3. Drivers behave accordingly to the traffic light state.
4. No car can obstruct the passage in the crossing no matter the reason.
5. Events planners always report to the news channel up to date events in the city.
6. The public Transport Microservice always returns the right timetable given a line or the name of a street.

2.4 Requirements

2.4.1 Functional Requirements (to modify put id of each requirement. Es FR1...)

1. In any circumstance the system must not allow two orthogonal traffic lights to be green at the same time.

2. The system must modify the duration of the green lights to reduce traffic.

3. The system shall process and aggregate traffic data to identify traffic flow patterns.

4. The system should send adjustment commands to the traffic light control system.

5. The system should be able to write to a log file what needed.

6. The system shall generate optimization suggestions for one-way road and traffic light configurations.

7. The system shall generate optimization suggestions for public transport schedules.

8. The system shall continuously receive data from both the message bus and the news channel.

9. The system must gather data from the microservice.

10. The system must assess the potential traffic impact of planned events.

11. The system shall generate event-specific suggestions for public transport adjustments.

12. The system shall present suggestions to urban area managers for review.

13. The system shall record and apply the acceptance or rejection of the proposal by the urban area managers.

14. The system shall generate daily reports on average traffic flow.

15. The system shall generate yearly reports on suggestions proposed and their outcome.

16. The system shall publish reports for public access.

2.4.2 Non-Functional Requirements

1. The system shall process sensor data in real time.
2. The system should be available 24/7.
3. The system shall implement traffic light adjustments in 15 seconds.
4. The system shall generate reports within 1 hour after midnight.
5. The system should maintain data consistency during communication with external systems.
6. The system should be scalable regarding the addition of sensors and bus lines.

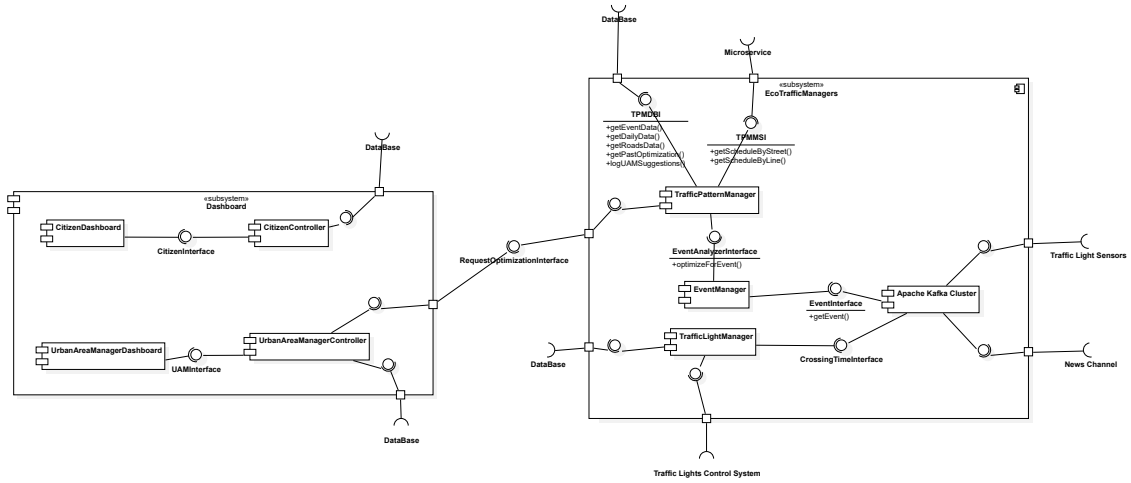
2.4.3 Constraints (to modify, insert something to use the external services)

1. The setLight function must be atomical.
2. The call to the microservice must be done using an API REST.

3 Design

3.1 Architecture

3.1.1 Components diagram



3.1.2 Subsystems and components

The EcoTrafficManagers subsystem includes the component:

- **ApacheKafkaCluster:** the component is based on a framework for event-driven paradigm and it includes primitives to create event producers and consumers and a runtime infrastructure to handle event transfer. This component it's the one responsible to receive the data from the sensors system and from the news channel and to distribute them to the right components who then will process and analyse them. More information could be found at `https://kafka.apache.org/`.

- **TrafficLightManager:** this component receives the data measured by the sensors system from the ApacheKafkaCluster and provides method to analyse this data to find unbalanced traffic loads into a crossing, to compute the new time in which the green light is switched on to reducing traffic congestion, to store into the database of the system the data obtained from the ApacheKafka-Cluster and to connect to the control system of the traffic lights to apply the right modifications.

- **EventManager:** this component receives the event specification from the ApacheKafkaCluster after that the news channel has published them. This component provides methods to retrieve information (such that the event type, the date, the place and the expected attendance) and a method to call the TrafficPatternManager component to optimize the event configuration to reduce traffic load.

- **TrafficPatternManager:** this component could be called by three events. The first is when the EventManager calls it to perform event-based optimization, in this case the component queries the database to find similar event and the daily traffic patterns in the zone where the event takes place, then generate possible optimizations from this data and from the timetables obtained after the getScheduleByStreet() and the getScheduleByLine() calls to the microservice. The second is when the UrbanAreaManagerController asks for an optimization around a specific zone, in this case the component queries the daily traffic patterns in the zone and the timetables and tries to optimize the traffic loads. The third takes place automatically ten minutes after the midnight, in this case the component queries the daily crossing time in all the crossings of the area and finds traffic patterns, after that it tries to optimize in terms of one-way roads, traffic lights configuration, and public transport schedule.

The Dashboard subsystem includes the following components:

- **CitizenDashboard:** this component shows to the citizen an interface from where it can be decided to see the daily reports or the yearly one. After choosing, the component calls the CitizenController to get the reports. When the response arrives, the component shows to the user the reports.

- **CitizenController:** this component receives the request from the citizen dashboard and them queries the database to satisfy the request. After, the data are elaborated and sent to the dashboard.

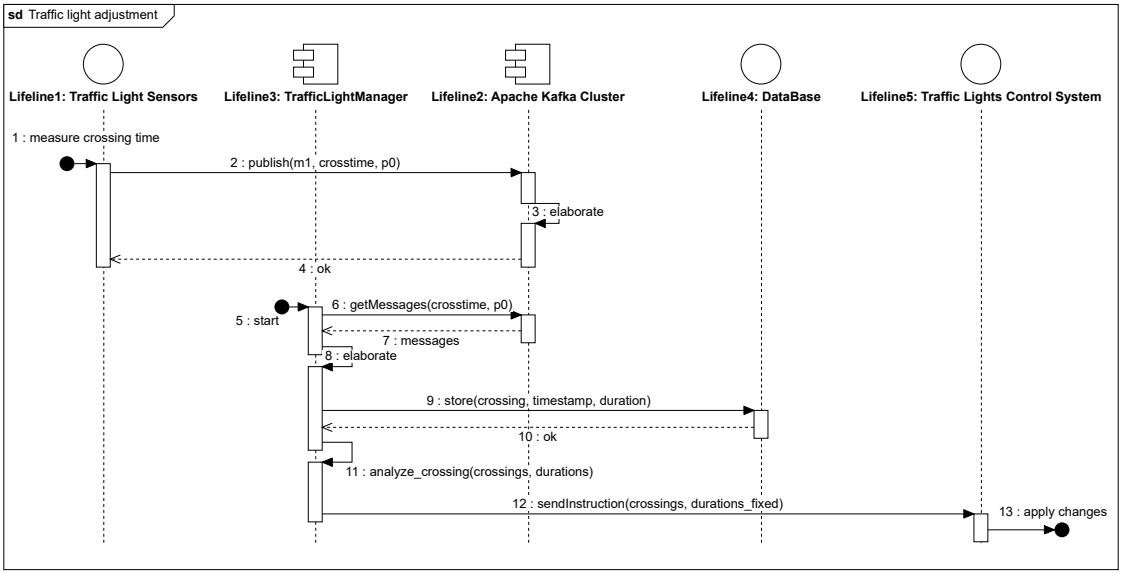
- **UrbanAreaManagerDashboard:** this component shows to the UAM an interface from where it can be decided if to see the pending suggestions or to ask the system to optimize all the configurations in a precise zone. If the first option is taken, the calls the UrbanAreaManagerController to get the pending suggestions. When the response arrives, the component shows to the user the reports and waits till all the decisions are taken. When each decision is taken, the component sends the answer to the controller. If the second option is taken, the dashboard presents the zones in the area and waits for the UAM's decision, after it arrives the components sends to the controller the requests and waits for the answer, after this arrives it displays to the UAM the suggestions, when the decision is taken, the component sends the answer to the controller.

- **UrbanAreaManagerController:** this component is responsible for managing the requests arriving from the dashboard and to redirect them either

to the database if the UAM wants to see the pending suggestions or to the TrafficPatternManager if the UAM wants an optimization for a specific zone. After the responses arrive, the component sends them to the dashboard. If the answer it's an optimization the component waits for the approval or the rejection and stores the decision.

3.2 Sequence Diagrams

Traffic Lights Adjustment



This sequence diagram shows the communication and data flow involved in adjusting traffic light timings based on crossing data.

1. The traffic light sensors system measures a crossing time.
2. The traffic light sensors system publishes the crossing time on the message bus and it's received from the Apache Kafa Cluster.
3. The Apache Kafka Cluster performs the operation as shown in this diagram.
4. The AKC responds to the traffic light sensors system.
5. TrafficLightManager is working.
6. TrafficLightManager tries to get a message.
7. The message, if present, it's sent to TrafficLightManager. The point 6 and 7 are better shown in this diagram.
8. TrafficLightManager elaborates the message received extracting the important information.
9. TrafficLightManager sends a message to the database to make it store the data received.

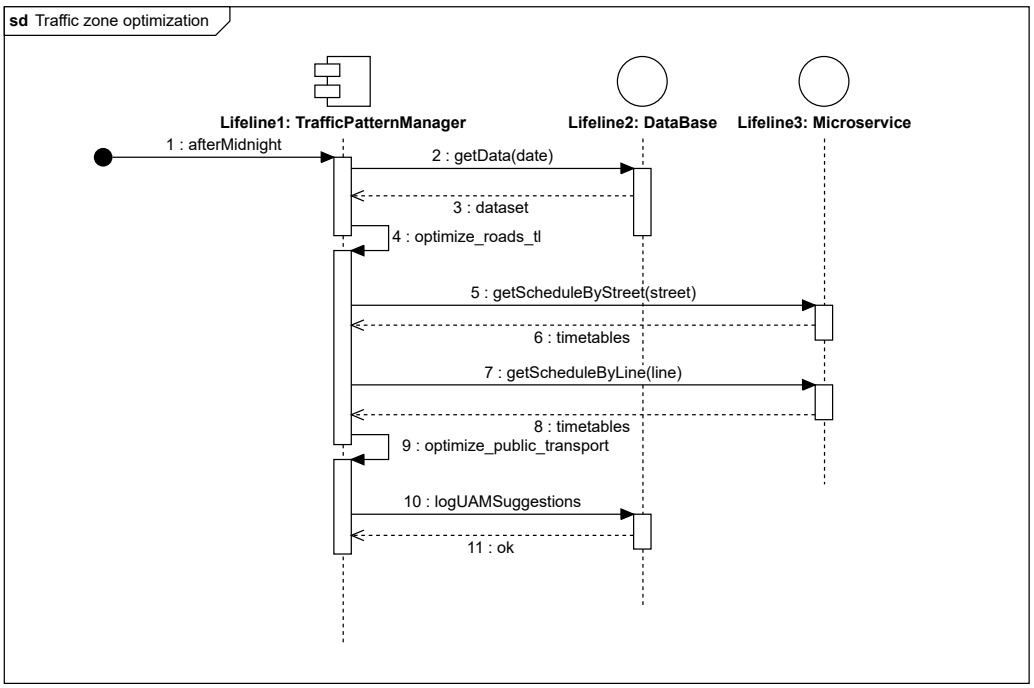
10. Successful store.

11. TrafficLightManager analyses the data received to find eventually unbalanced traffic loads. If it finds them, then it computes also the new green light time for a particular traffic light.

12. TrafficLightManager sends a message to the traffic light control system with the modification to perform.

13. Traffic light control system applies the changes

Traffic Zone Optimization



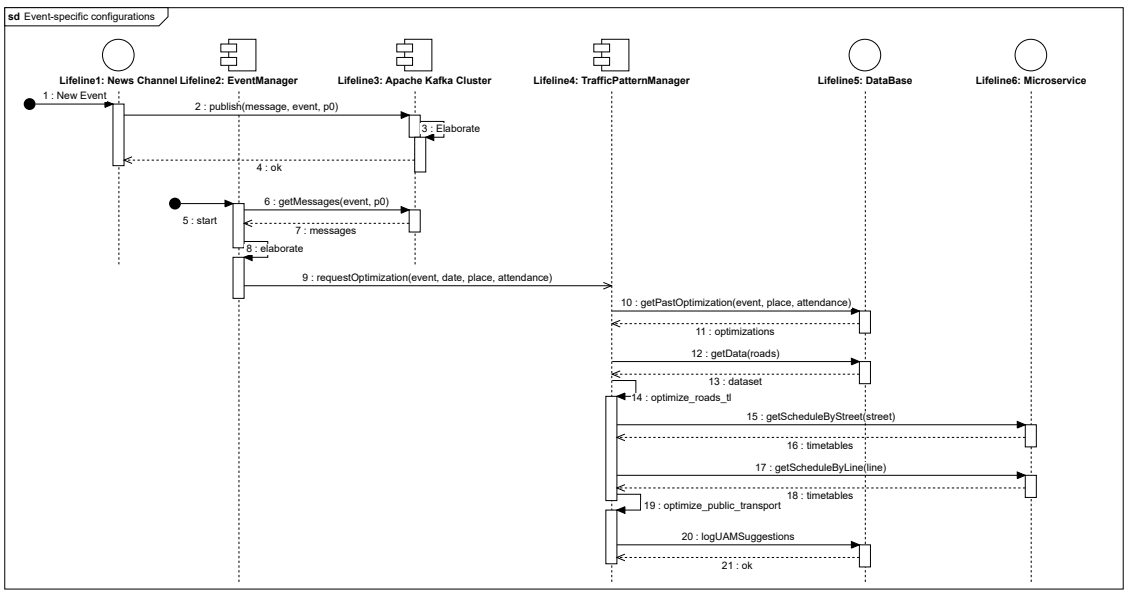
This sequence diagram illustrates the data flow between TrafficPatternManager, database, and Microservice components for optimizing traffic zones, including both road traffic and public transportation optimization

1. The process starts when the TrafficPatternManager receives an "afterMidnight" trigger or signal.

2. TrafficPatternManager requests data from the database collected in the date passed as argument.

3. The database responds by returning the data to the TrafficPatternManager.
4. TrafficPatternManager analyzes the data to identify traffic patterns.
5. TrafficPatternManager stores the patterns into the database.
6. TrafficPatternManager tries to optimize the road traffic lights to reduce the traffic loads.
7. TrafficPatternManager asks for the timetables of the streets to the microsystem provided by the state.
8. The microsystem returns the street timetables.
9. TrafficPatternManager asks also for the timetables of the lines to the microsystem.
10. The microsystem returns the line timetables.
11. TrafficPatternManager tries to optimize public transportation schedules.
12. TrafficPatternManager logs the suggestions and stores the, into the database.
13. The Microservice responds with an "ok" confirmation message, completing the sequence.

Event-specific Configuration



This sequence diagram shows the communication and data flow involved in adjusting traffic light timings based on crossing data.

1. Ten minutes past midnight, an automatic process is started in the system and it is managed by the Traffic Pattern Manager.
2. The Traffic Pattern Manager retrieves data from the DB by passing the current date.

3. The DB then returns a dataset to the Traffic Pattern Manager.

4. The TPM then optimizes the traffic flow for the whole system of traffic lights.

5. Then the TPM asks for the timetables of the streets to the microsystem provided by the state.

6. The microsystem returns the street timetables.

7. The TPM asks also for the timetables of the lines to the microsystem.

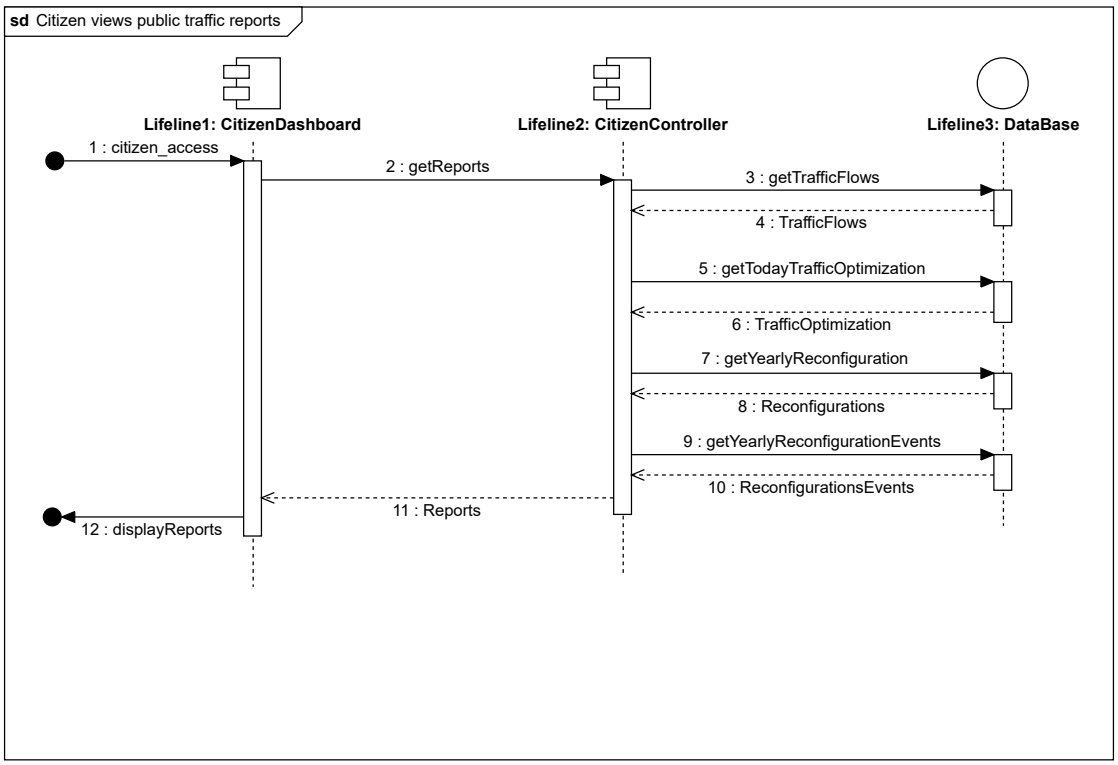
8. The microsystem returns the line timetables.

9. Then the system proceeds to optimize the public transport tables.

10. In the end the TPM logs the suggestions into the database.

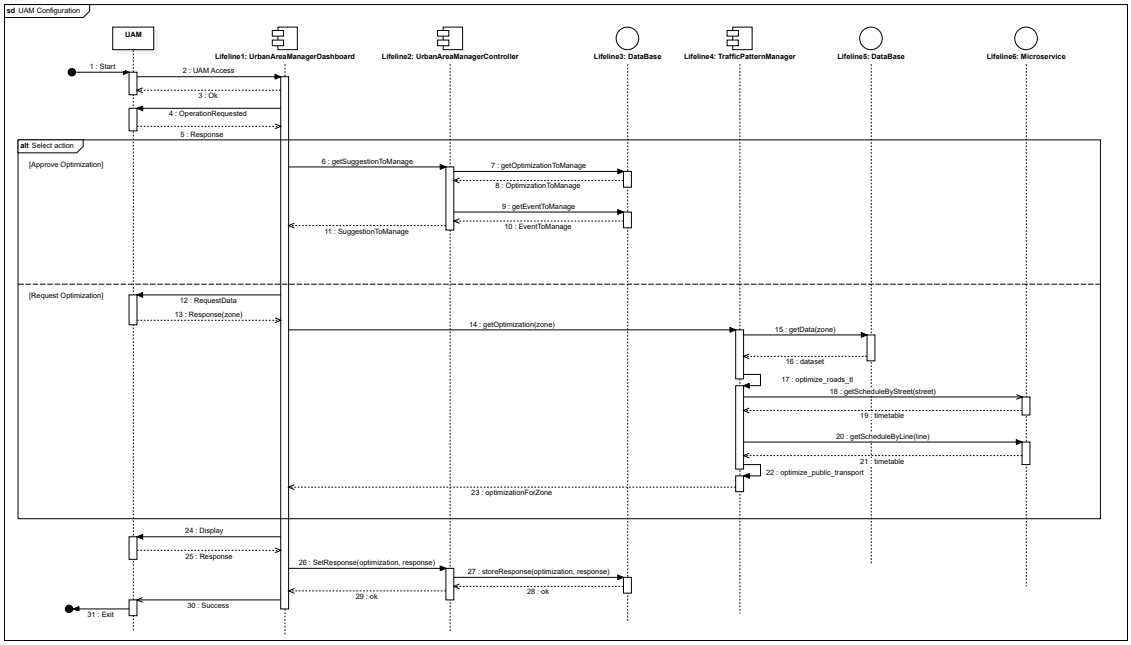
11. The DB returns an ack message in case of successful write.

Citizen views public reports



This sequence diagram shows the communication and data flow involved in adjusting traffic light timings based on crossing data.

UAM configuration



This sequence diagram shows the communication and data flow involved in adjusting traffic light timings based on crossing data.

3.3 Critical points and design decisions

- **Critical point:** Analyzing traffic patterns requires complex processing of historical and real-time data.
- **Design decision:** Our architecture is designed to separate the concerns of data collection, processing, and presentation. The `TrafficPatternManager` is specialized in creating algorithms to identify common traffic patterns and propose optimizations, while the `TrafficLightManager` handles the actual traffic light adjustments.

- **Critical point:** A scalable and efficient communication system is required to handle high-frequency data from sensors and from the news channel.
- **Design decision:** We decided to use the `ApacheKafkaCluster` to handle event-driven communication, allowing real-time and asynchronous data processing of a large volume requests.

- **Critical point:** The system must be able to provide adaptive responses to city events with a minimum latency.
- **Design decision:** The `EventManager` component processes the event data and classifies the event based on the expected attendance. It then triggers the `TrafficPatternManager`, which will provide an appropriate optimization

routine.

- **Critical point:** The system must be able to provide a user-friendly interface for both citizens and urban area managers.
- **Design decision:** The `CitizenDashboard` and `UrbanAreaManagerDashboard` components are designed to provide intuitive interfaces for users to access reports and make decisions. The dashboards are separated from the core logic of the system to ensure modularity and maintainability.

- **Critical point:** Human stakeholders must be involved in taking decisions for critical changes.
- **Design decision:** On the `UrbanAreaManagerDashboard` pop up suggestions for optimizations retreived directly from the unified databasse by the `UrbanAreaManagerContro` The human manager then decides whether to accept or reject the suggestions. When a response is received, the system logs the decisions in the database for future reference.

- **Critical point:** Supporting future scalability and integration of additional sensors or data sources.
- **Design decision:** We designed our architecture to be modular, allowing an extension of the sensor net or for new data sources. The `ApacheKafkaCluster` can easily integrate new actors, ensuring that the system can adapt to future needs without significant modifications.

- **Critical point:** The system must be able to handle data consistency and coherence during communication with external systems.
- **Design decision:** We implemented a robust error handling and data validation mechanism in the `TrafficLightManager` and `TrafficPatternManager` components to ensure that only valid data is processed and stored. This helps maintain data integrity and coherence across the system.

- **Critical point:** The system must be able to report real-time updates to the citizens.
- **Design decision:** The `CitizenDashboard` is designed to provide real-time updates on traffic optimizations and possible changes on viability. The system uses a push mechanism to notify citizens of significant changes, ensuring they are always informed.