

Distributed Systems

March 2023

Higher Diploma in Science in Computing

21219842

Chungman Lee

Contents

1. Introduction
2. Service 1: Water Pollution Tracker
   1. Methods
      1. GetWaterPollutionHistory (Server-side streaming RPC)
      2. GetWaterPollutionAlerts(Unary RPC)
      3. MonitorWaterPollution (Bidirectional streaming RPC)
3. Service 2: Air Pollution Tracker
   1. Methods
      1. GetAirPollutionHistory (Server-side streaming RPC)
      2. GetAirPollutionAlerts (Unary RPC)
      3. MonitorWaterPollution (Bidirectional streaming RPC)
4. Service 3: Data Visualization Service
   1. Methods
      1. GetPollutionStatistics (Server-side streaming RPC)
      2. FilterDataByLocation (Bidirectional streaming RPC)
      3. SetFavoriteLocation (Client-side streaming service)
5. Conclusion
6. Git adress

1. Introduction

This smart pollution tracking system aims to help users monitor and analyze the levels of water and air pollution by location. This provides three services: a water pollution tracker, an air pollution tracker, and a data visualization service. The water and air pollution trackers use collect pollution data, while the data visualization service aggregates and displays the data in real-time for users to view.

2. Service 1: Water Pollution Tracker

The water pollution tracker service allows users to track and analyze the levels of water pollution in their environment. The service supports the following RPCs:

2.1 Methods

2.1.1 GetWaterPollutionHistory (Server-side streaming RPC)

**rpc** GetWaterPollutionHistory (WaterPollutionHistoryRequest) **returns** (**stream** WaterPollutionLevel) {}

This retrieves a stream of historical water pollution levels.

2.1.2 GetWaterPollutionAlerts(Unary RPC)

**rpc** GetWaterPollutionAlerts (WaterPollutionAlertsRequest) **returns** (WaterPollutionAlertsResponse) {}

This retrieves a list of pollution alerts triggered by the system when pollution levels exceed a set threshold.

2.1.3 MonitorWaterPollution (Bidirectional streaming RPC)

**rpc** MonitorWaterPollution (**stream** WaterPollutionMonitorRequest) **returns** (**stream** WaterPollutionLevel) {}

This Allows users to continuously monitor air pollution levels and receive real-time updates as pollution levels change.

3. Service 2: Air Pollution Tracker

The air pollution tracker service provides functions tracking and analyzing the levels of air pollution in their environment.

3.1 Methods

3.1.1 GetAirPollutionHistory (Server-side streaming RPC)

**rpc** GetAirPollutionHistory (AirPollutionHistoryRequest) **returns** (**stream** AirPollutionLevel) {}

This RPC retrieves a stream of historical air pollution levels over a specified period.

3.1.2 GetAirPollutionAlerts (Unary RPC)

**rpc** GetAirPollutionAlerts (AirPollutionAlertsRequest) **returns** (AirPollutionAlertsResponse) {}

This RPC retrieves a list of pollution alerts that are made by the system when pollution levels exceed a threshold.

3.1.3 MonitorWaterPollution (Bidirectional streaming RPC)

**rpc** MonitorAirPollution (**stream** AirPollutionMonitorRequest) **returns** (**stream** AirPollutionLevel) {}

This RPC will make users possible to continuously monitor air pollution levels and receive real-time updates as pollution levels change.

4. Service 3: Data Visualization Service

The data visualization service gather the pollution data collected by the water and air pollution trackers and presents it in an interface.

4.1 Methods

4.1.1 GetPollutionStatistics (Server-side streaming RPC)

**rpc** GetPollutionStatistics (PollutionStatisticsRequest) **returns** (**stream** PollutionStatistics) {}

This RPC retrieves a stream of statistics related to water and air pollution levels, such as average pollution levels over a or the highest pollution level recorded.

4.1.2 FilterDataByLocation (Bidirectional streaming RPC)

**rpc** FilterDataByLocation (**stream** LocationFilterRequest) **returns** (**stream** PollutionLevel) {}

This RPC allows users to filter pollution data by their current location and receive real-time updates as pollution levels change in their area.

4.1.3 SetFavoriteLocation (Client-side streaming service)

**rpc** SetFavoriteLocation (**stream** FavoriteLocationRequest) **returns** (**stream** FavoriteLocationResponse) {}

This RPC allows the user to save their favorite locations for tracking pollution. The client can send multiple location requests in a single call, and the server respond with a status message for each location.

5. Conclusion

This smart water/air pollution tracking system will provide users with valuable information about their environment and help them make informed decisions to reduce pollution levels. The system's modular design and use of gRPC and Java will ensure scalability and flexibility in future development.

6. Github: <https://github.com/ChungmanLee/CA_Distributed_System>