OpenAPI-based Message Router for Mashup Service Development

Doyoung Lee, Seyeon Jeong, James Won-Ki Hong

Department of Computer Science and Engineering, POSTECH, Korea

{dylee90, jsy0906, jwkhong}@postech.ac.kr

2017. 9. 28

Contents

Introduction

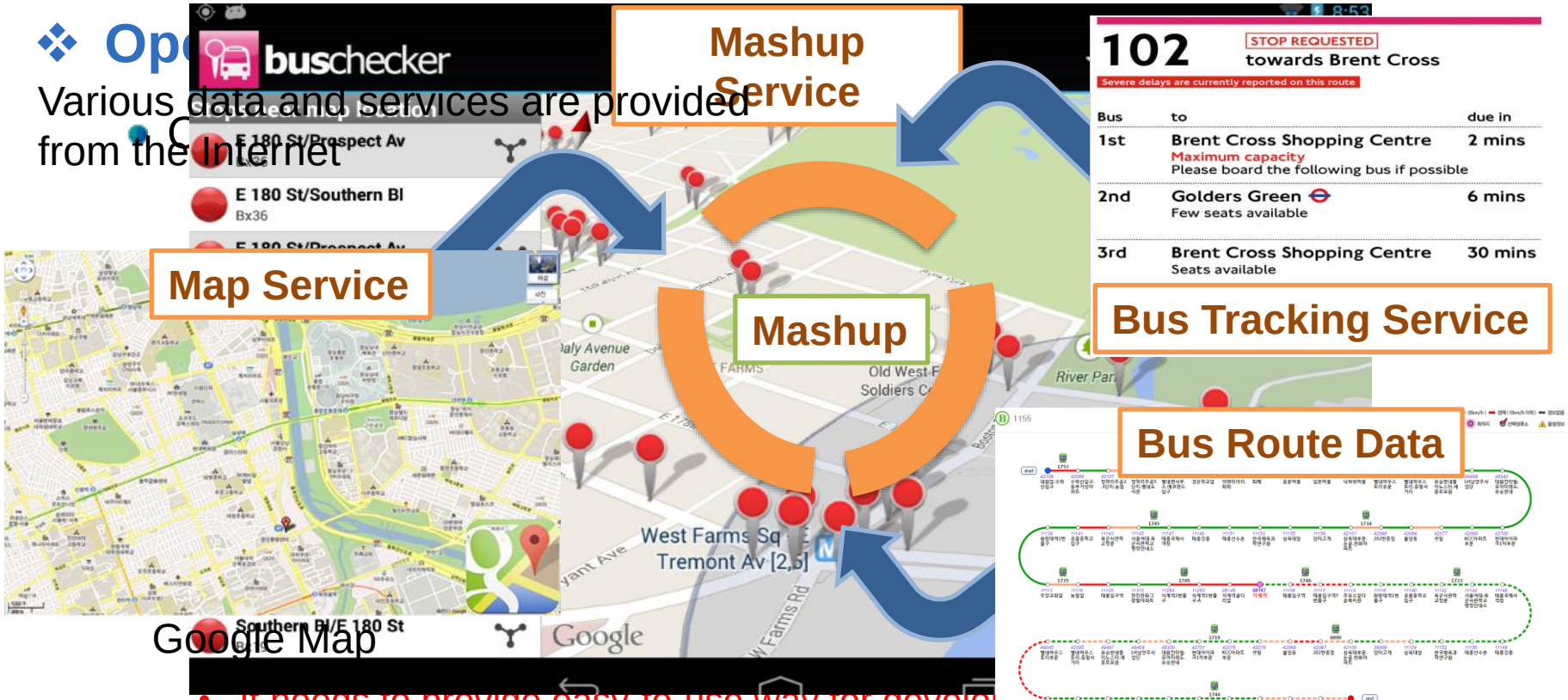
❖ Mashup Service

- Service using data from various sources and existing services
- Burden to construct interfaces for using the data and services

❖ Ope

Various data and services are provided from the Internet

- C



Google Map

- It needs to provide easy-to-use way for develop

Bus Information Application

Related Work

❖ Approaches for Mashup Service Development

- IoTMaaS (Services Computing (SCC), 2013)
  - Stand for IoT Mashup as a Service
  - Cloud based IoT mashup service model
  - Overcome difficulties of connecting and data processing among heterogeneous devices
  - Satisfy various mashup requirements such as different device vendors and users
- LAMEC (World Symposium on Computer Application & Research (WSCAR), 2014)
  - Lightweight Architecture for Mobile web content access over Enterprise Cloud mashup
  - Utilize mobile devices and cloud computing resources to minimize size of data delivered

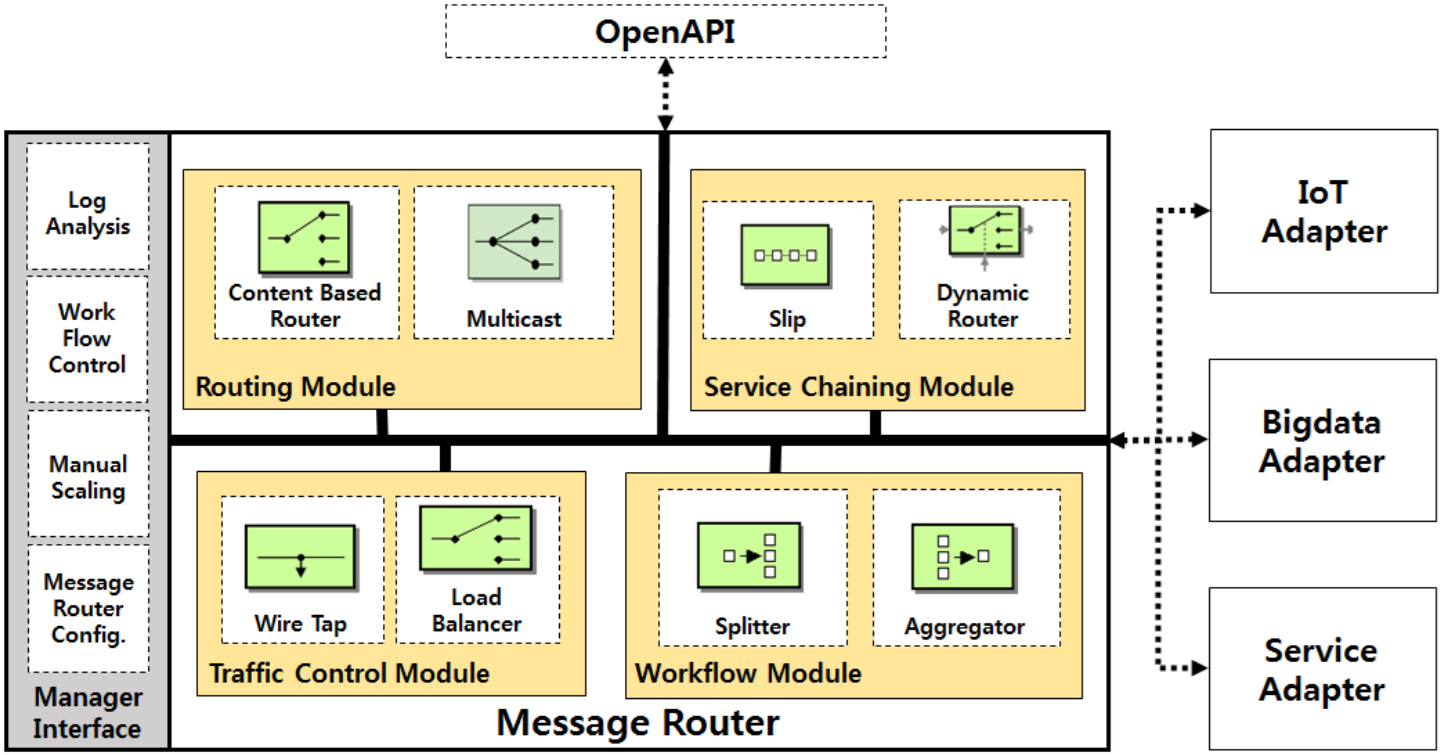
❖ Limitations

- Small size of the number of types of interlinked platforms
- Limited interfaces for linking various platforms

Overall Design (1/5)

❖ Message Router Design (1/2)

- Message router
  - Design of message router using the enterprise integration pattern (EIP)
  - Message transmission and process between interlinked platforms and services
  - Composed functional modules: Service chaining, Routing, Workflow, Traffic control



Overall Design (2/5)

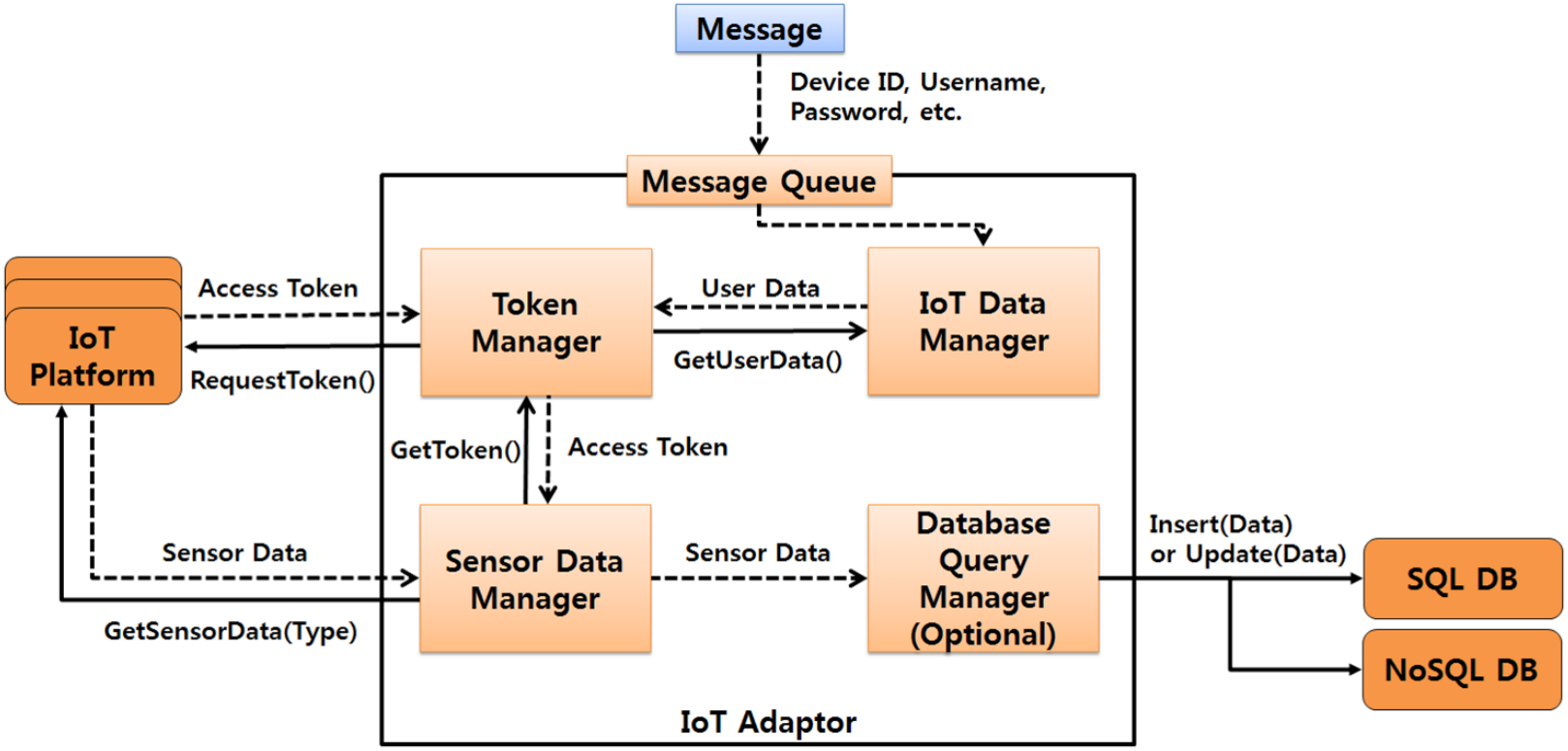
❖ Message Router Design (2/2)

- Routing module
  - It identifies openAPI messages and delivers them to a destination
    - OpenAPI messages are sent by developers to request data or external services
  - A destination can be another internal module or an external adapter
- Service chaining module
  - It is responsible for providing the service chain function
  - A message can be transferred through multiple destinations
- Workflow module
  - Workflow in the message router is presented as a task flow in terms of time scheduling
  - It is responsible for aggregating and splitting functions for messages
- Traffic control module
  - The overhead of message router needs to be distributed for load balancing
  - It measures the overhead of message router by monitoring messages

Overall Design (3/5)

❖ IoT Adapter

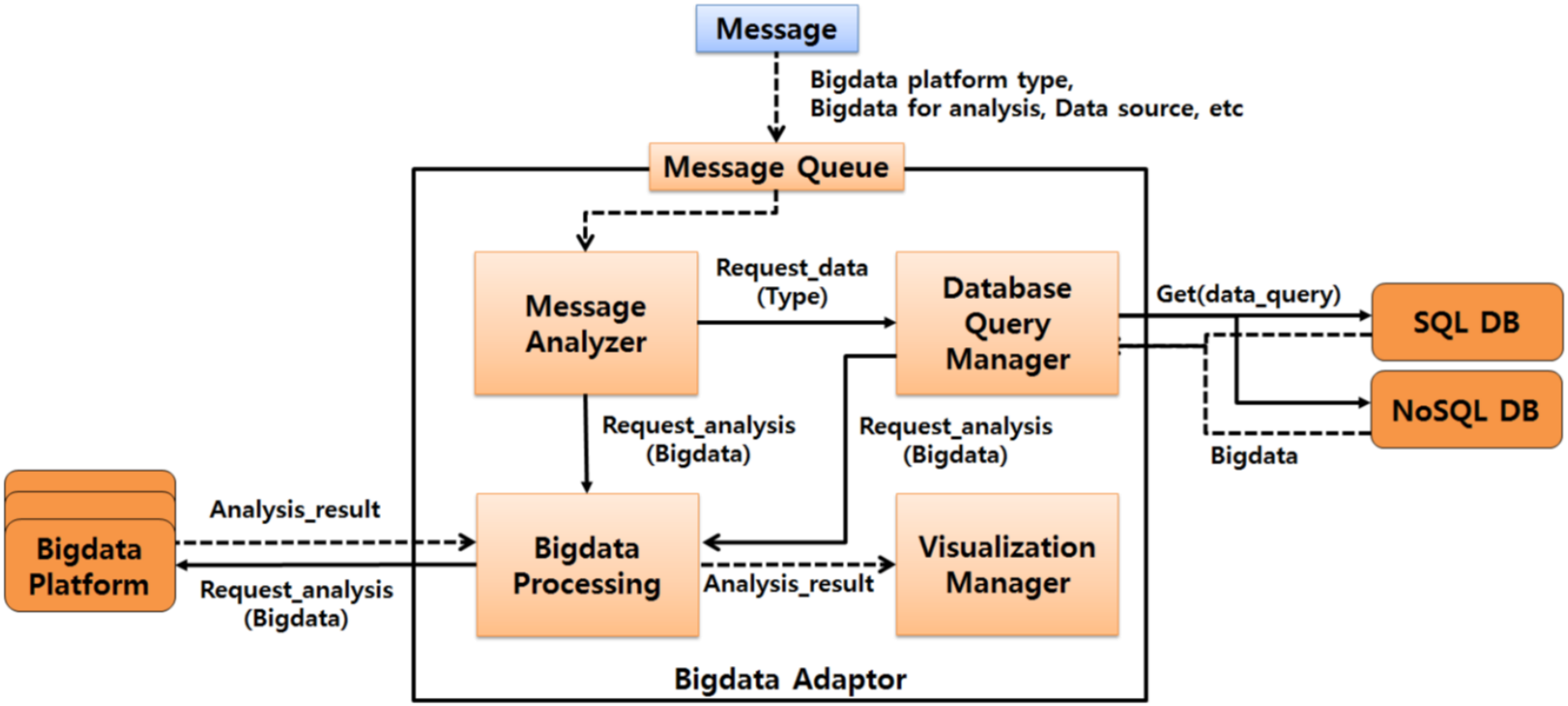
- Interoperability with IoT platforms
  - Ex. Mobius, IoTMakers, Amazon Web Services IoT, Azure IoT Suite, ThingWorx
  - IoT platform is a hub that connects a number of IoT devices and manages them
    - IoT adapter can access IoT data through IoT platforms



Overall Design (4/5)

❖ Big Data Adapter

- Interoperability with data analysis platforms
  - Ex. Apache Spark, Google BigQuery, Amazon Elastic MapReduce
  - In addition, big data should be stored in some databases such as MongoDB and MySQL
    - Big data adapter has an interface to interact with databases



Overall Design (5/5)

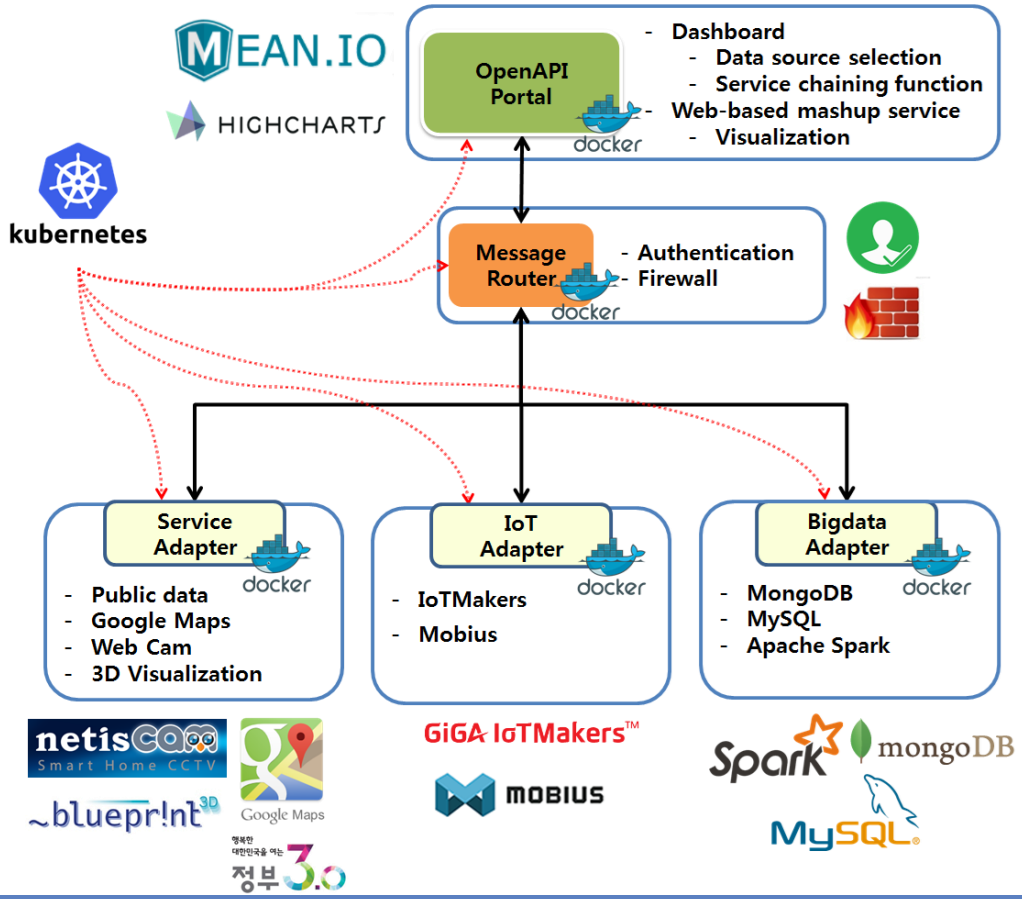
❖ Service Adapter

- An interface to interact with external services and public data
  - Ex. Google Maps, Government 3.0
  - External services and public data can be interlinked through the service adapter
- Access token registration for public data
  - The public data requires an access token which is difficult to remember
  - The service adapter provides a method to register the information in the adapter
  - If a developer requests the same information later, the adapter uses the information
  - It Is not necessary to give the same access token to the developer again

Implementation (1/4)

❖ Message Routers and Adapters

- Interaction with openAPIs and adapters to support development
  - Each component is implemented on each Docker container
  - Kubernetes manages each component for high reliability



Implementation (2/4)

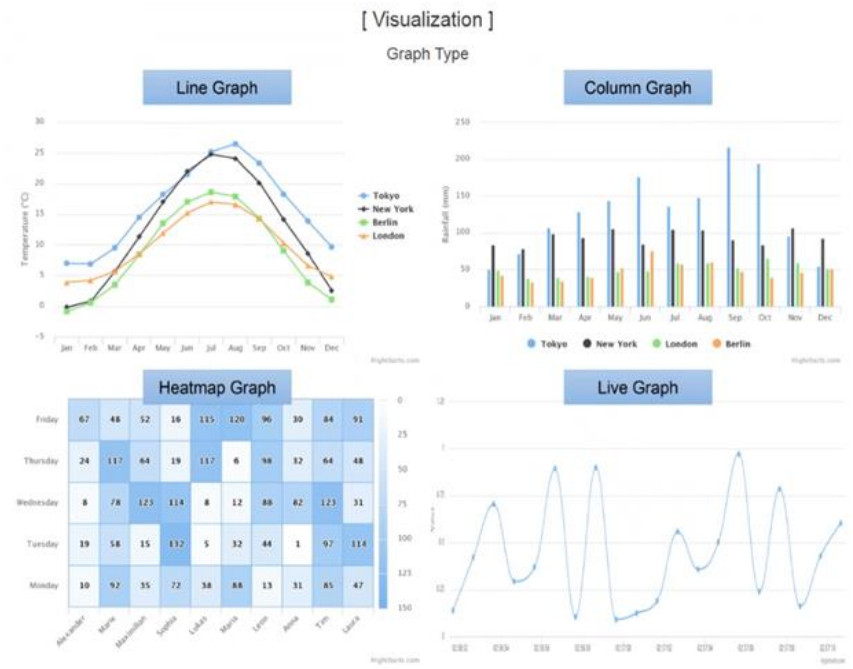
❖ OpenAPI

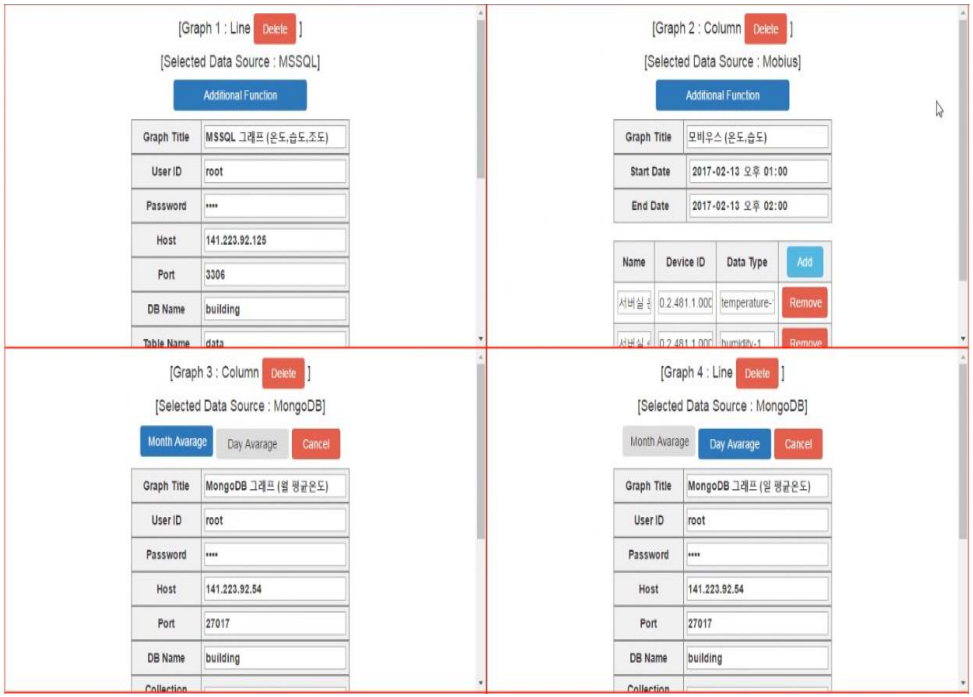
- Interface to handle requests of the developers
  - The message router handles the developers requests through openAPI
- HTTP POST message for openAPI
  - OpenAPI message is defined as a HTTP POST message
  - The request message name and the request function are defined in JSON
- OpenAPI portal to provide openAPIs
  - The openAPI portal is built on MEAN.IO
  - It provides a web-based interface to developers for developing new services
  - Through the portal, developers easily use functions for mashup service
    - Even if they do not know the detailed openAPI format
    - The decisions of the developers are converted to openAPI message by the portal
  - The developed mashup service is provided as a web-based service
    - After all the requests are processed, the result is presented through a web page

Implementation (3/4)

❖ Building Management System (1/2)

- A building management system as a mashup service
  - It is useful mashup service for demonstrating the effectiveness of the message router
  - It interacts with IoT platforms, big data platforms, and other external services
  - Various graphs present building information by developer's requests
    - The openAPI portal provides interfaces to apply the requests

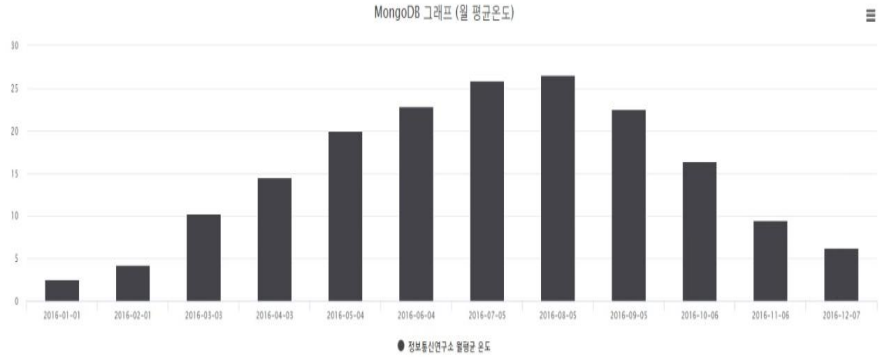


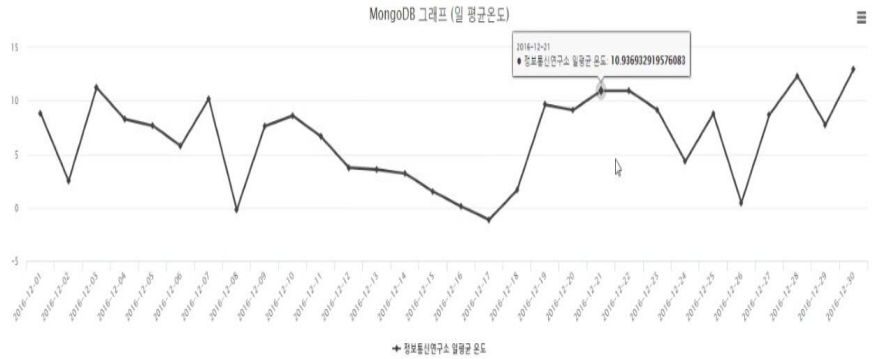


Implementation (4/4)

❖ Building Management System (2/2)

- Real-time graph
  - Message router gets the latest data from the selected data sources
- Non-real-time graph
  - Message router gets result of developer's request (Big data analysis, data query)

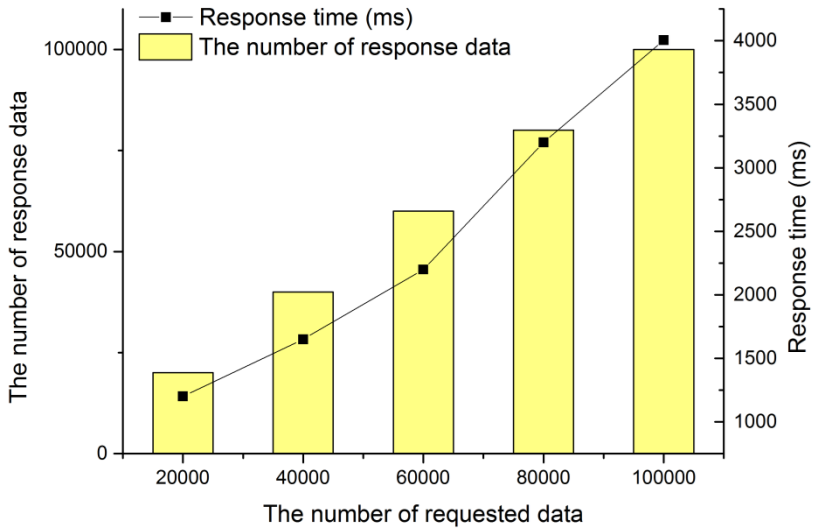




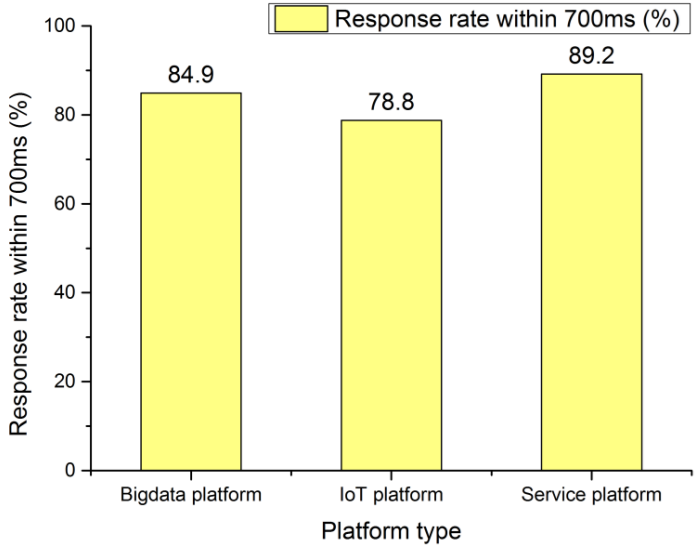
Evaluation

❖ Data Reliability

- The reliability of non-real-time and real-time data transmissions
  - Non-real-time data transmission
    - A measure of the response time and missing data when a large amount of data is requested
    - Request IoT sensor data stored in MongoDB through the big data platform
  - Real-time data transmission
    - How quickly the requested data is delivered rather than the amount of data
    - Generate 1,000 data requests for each platform



<Non-real-time data transfer reliability>



<Real-time data transfer reliability>

Conclusion

❖ Summary

- Proposed an openAPI-based message router
  - For supporting mashup service development
  - It helps developers reduce their burden of mashup service development
  - It provides traffic control, service chain management, and workflow management
- Built an easy-to-use openAPI portal for the message router
  - It shows how to use functionalities of the message router
  - It provides easy-to-use interfaces to interwork various platforms
  - Building management system is implemented as a proof-of-concept

❖ Future Work

- Considering a general interface design for platform interworking
- Defining additional openAPI formats to implement useful functions

Q&A