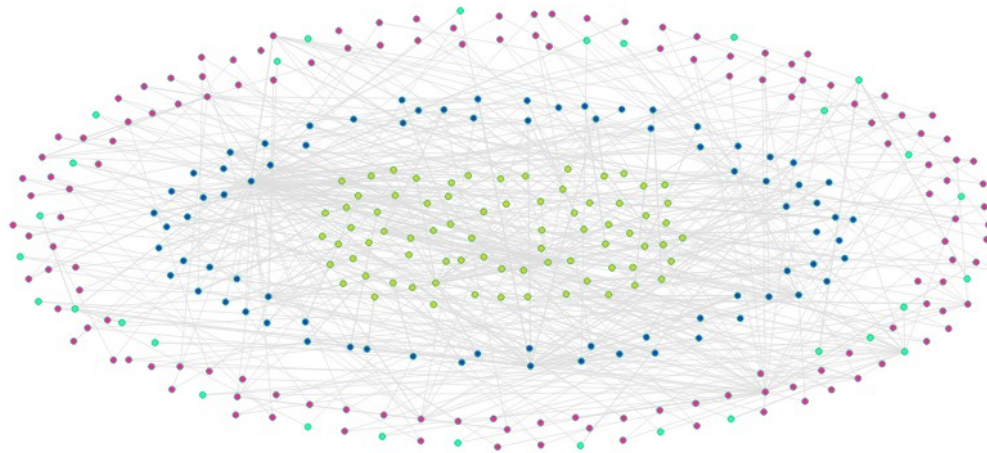


# Real-time Visualization of Analyzed Industrial Communication Network Traffic

**Xiaoru Li, Klevia Ulqinaku, Mario Alberto Gonzalez Ordiano, Philipp Mergenthaler**  
**Advisor: Ankush Meshram**

PSE 2018/19



# Background

- Industrial Network Security aims to understand the traffic in industrial production systems
- Analysis of the traffic to find anomalies
- Real-time visualization to help the user understand
  - Communication behavior
  - Changes in the communication
- Incidents can be detected visually



# Requirements

## ■ 24 Functional Requirements

- User access control, security roles
- Three different diagram types
- Brushing
- Data filter

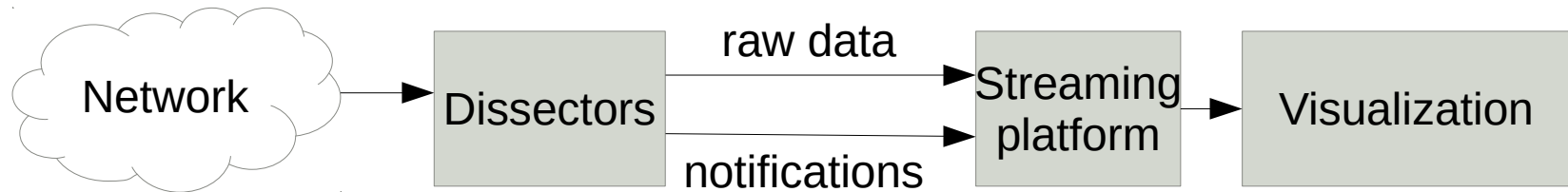
## ■ 12 Non-Functional Requirements

- extensibility



# The Workflow

- Network traffic is recorded
- Traffic data is analyzed (dissected)
- Data is fed to a streaming platform (Kafka)
- A visualization tool displays data and analysis results



# Architecture and Design

## ■ Client-Server Architecture

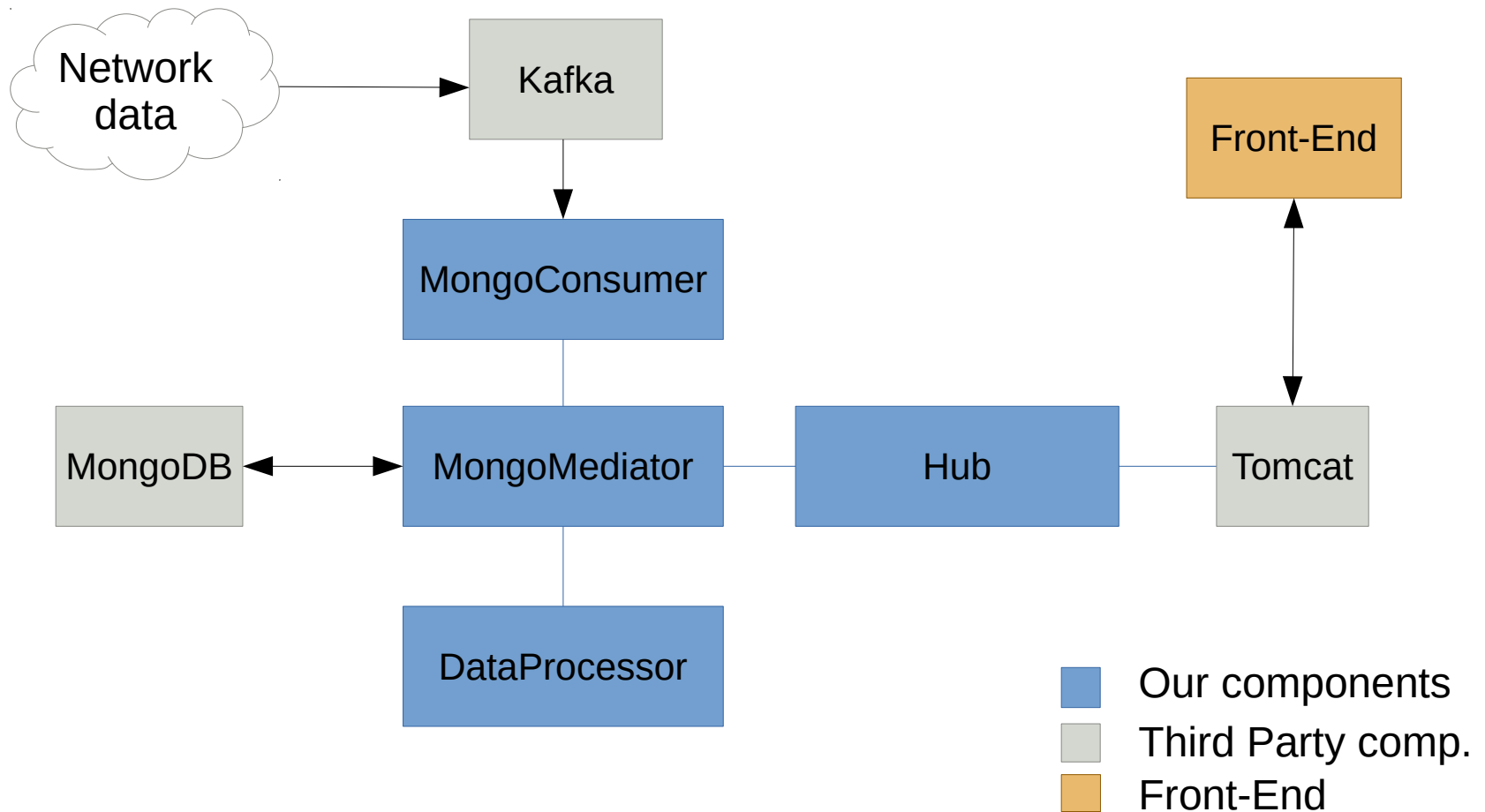
### ■ Front-End:

- Model-View-Controller pattern
- Observer pattern

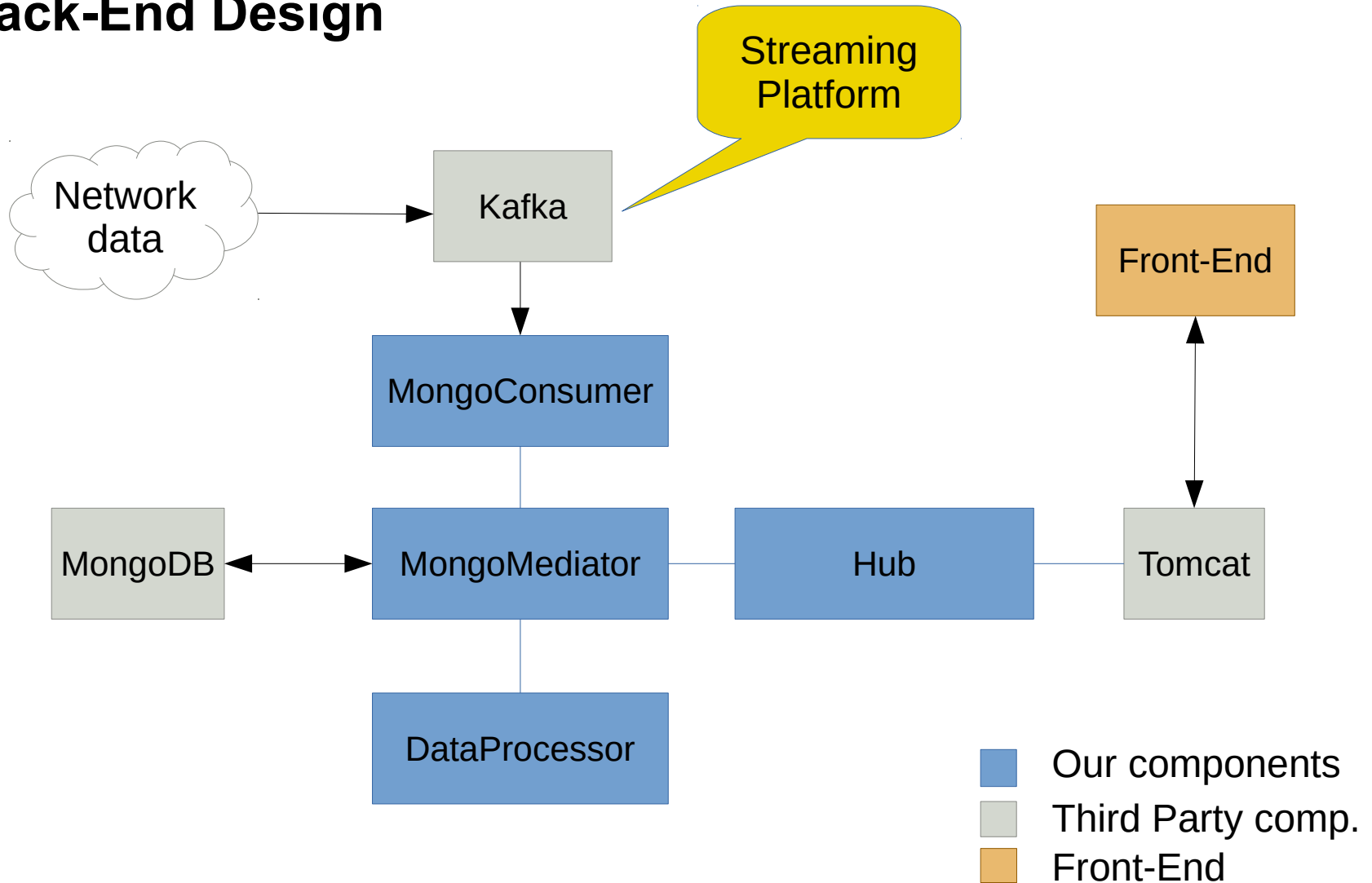
### ■ Back-End:

- Mediator pattern
- Strategy pattern

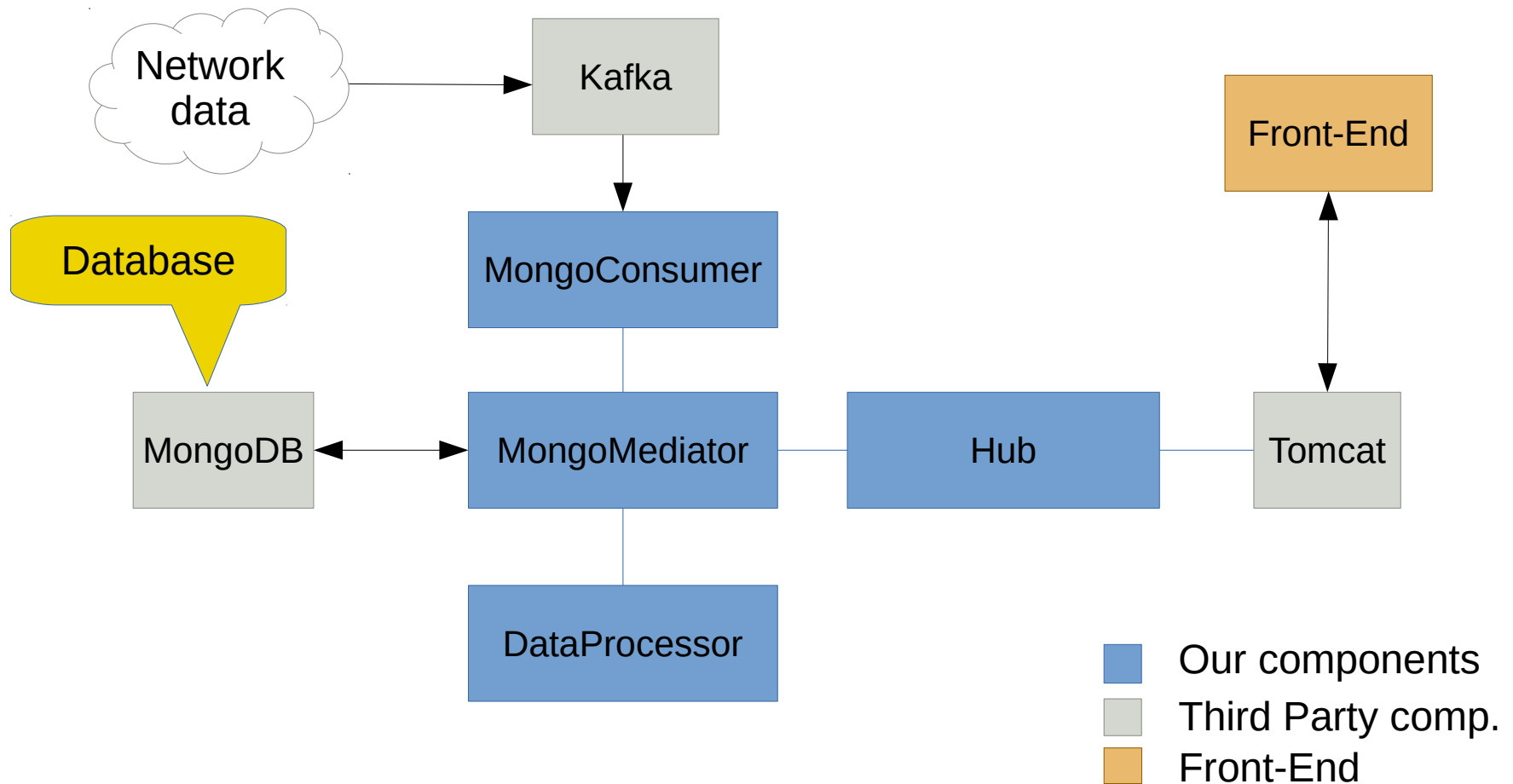
# Back-End Design



# Back-End Design

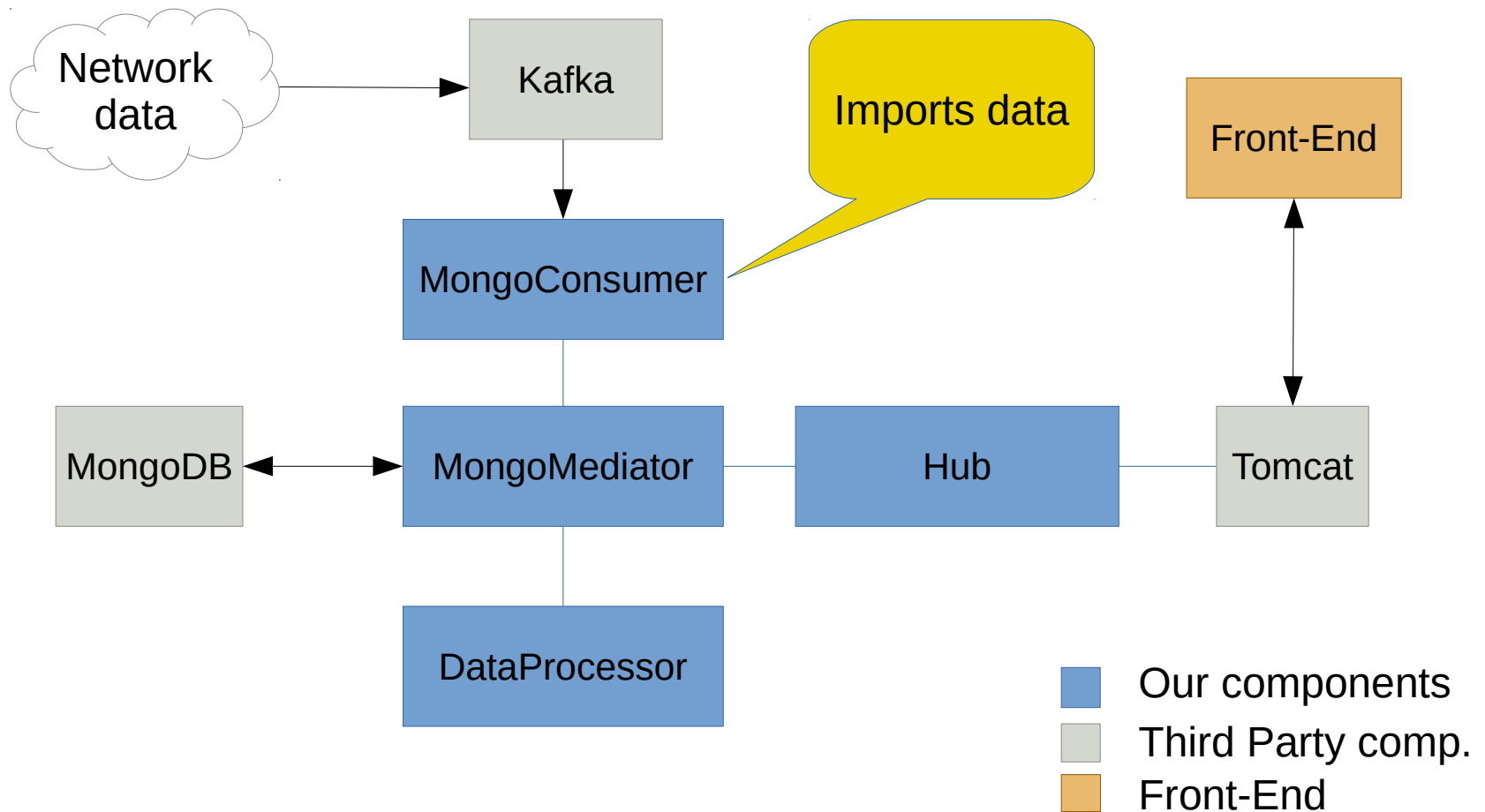


# Back-End Design

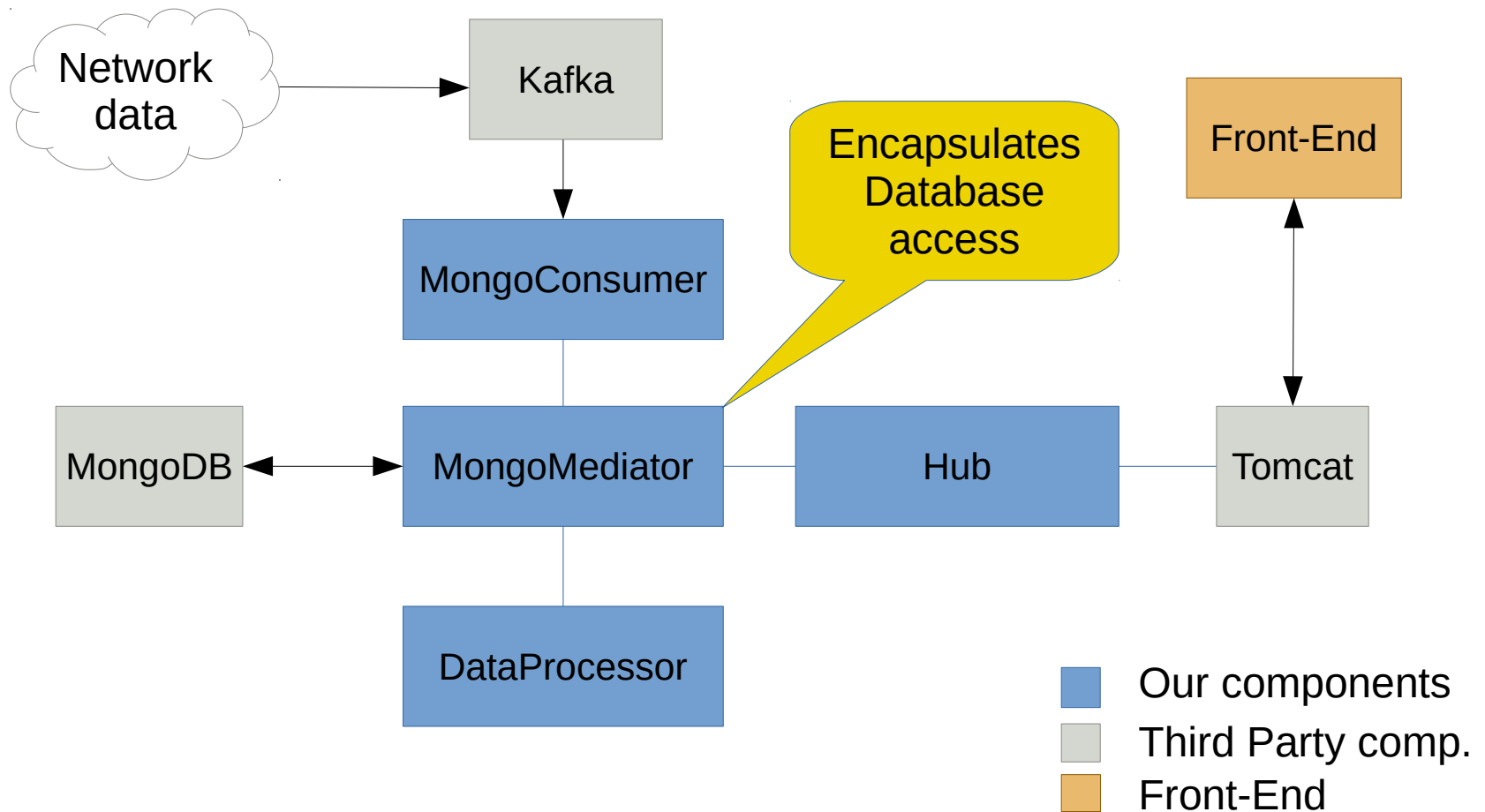




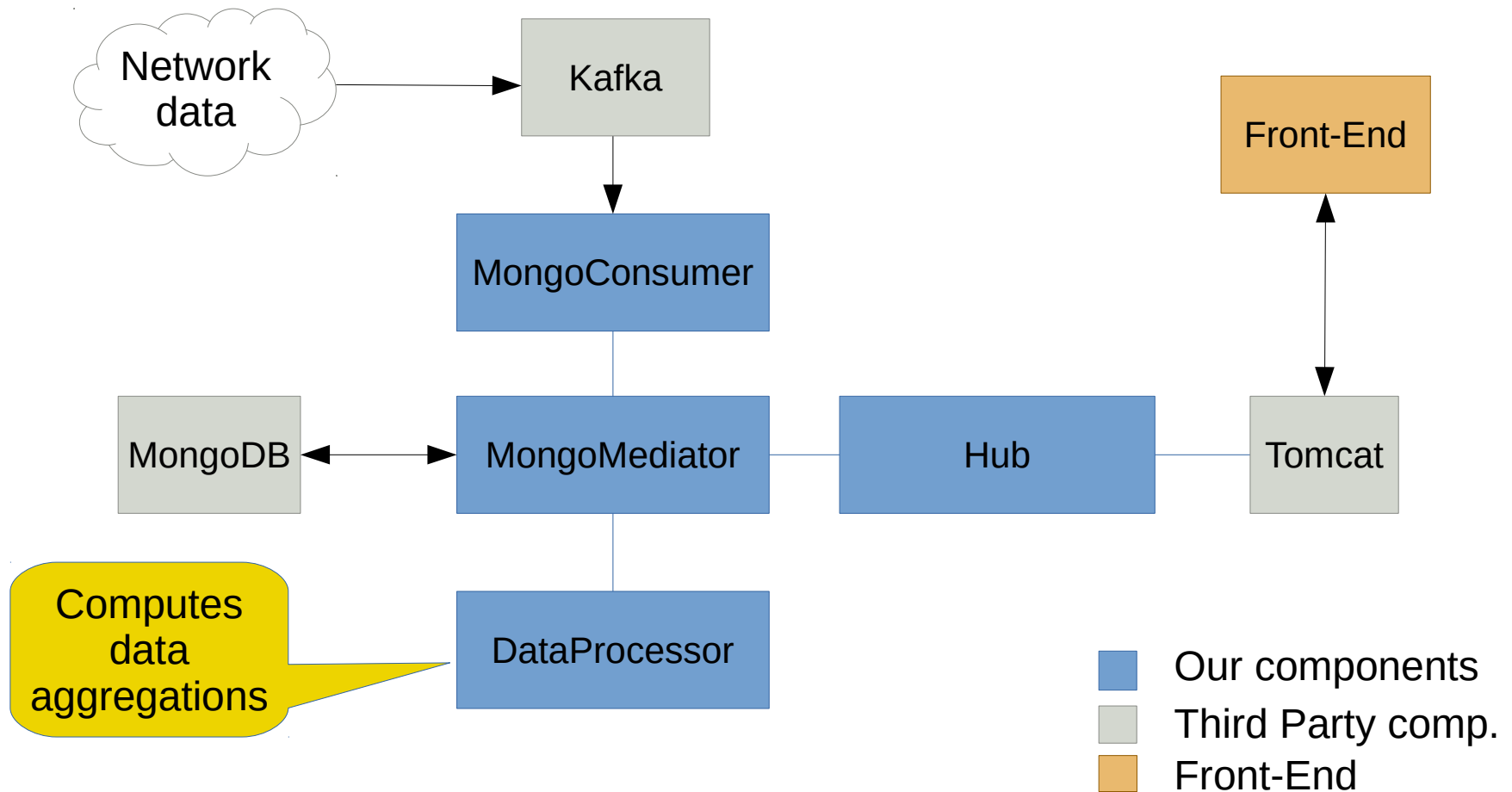
# Back-End Design



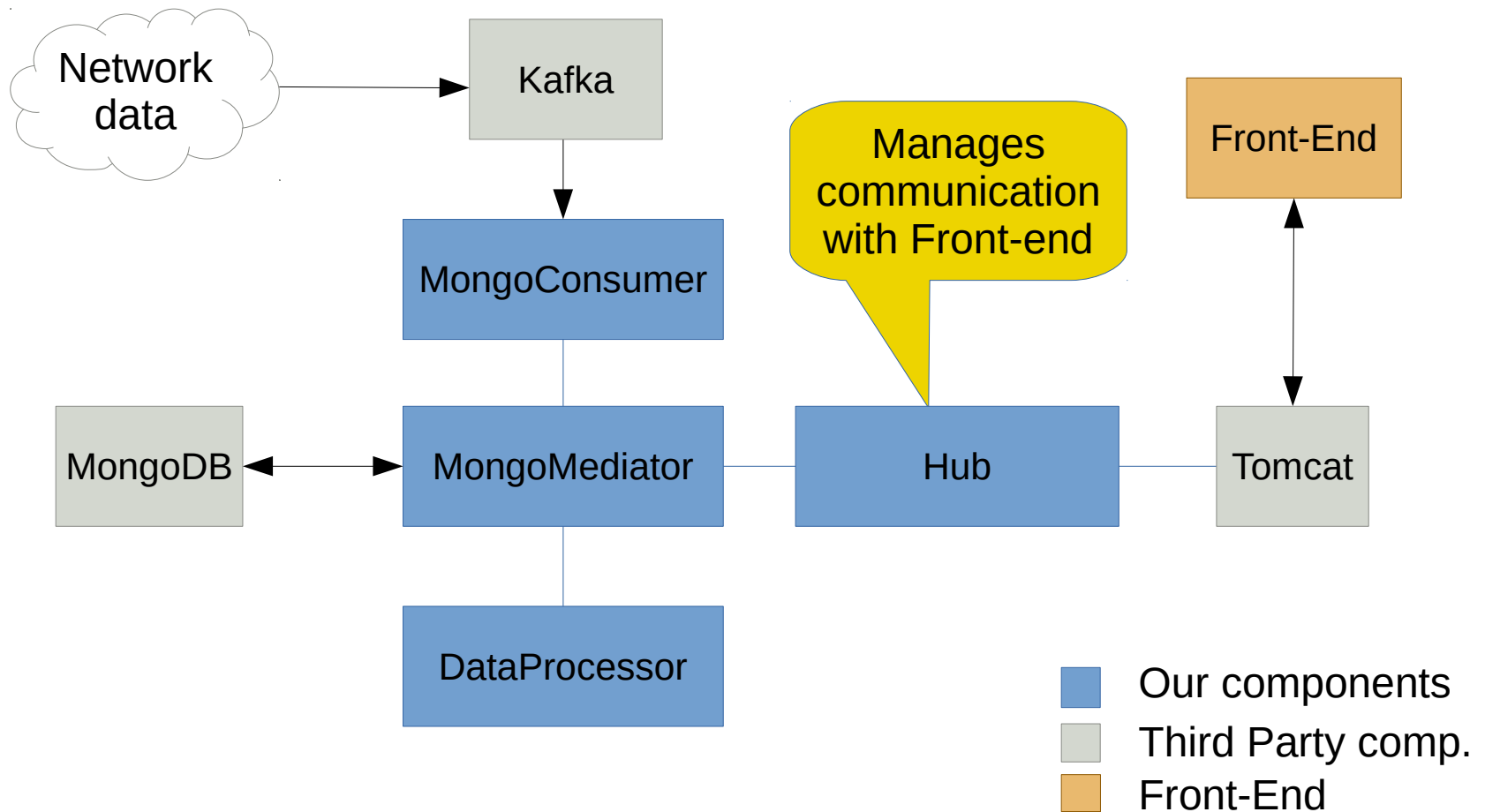
# Back-End Design



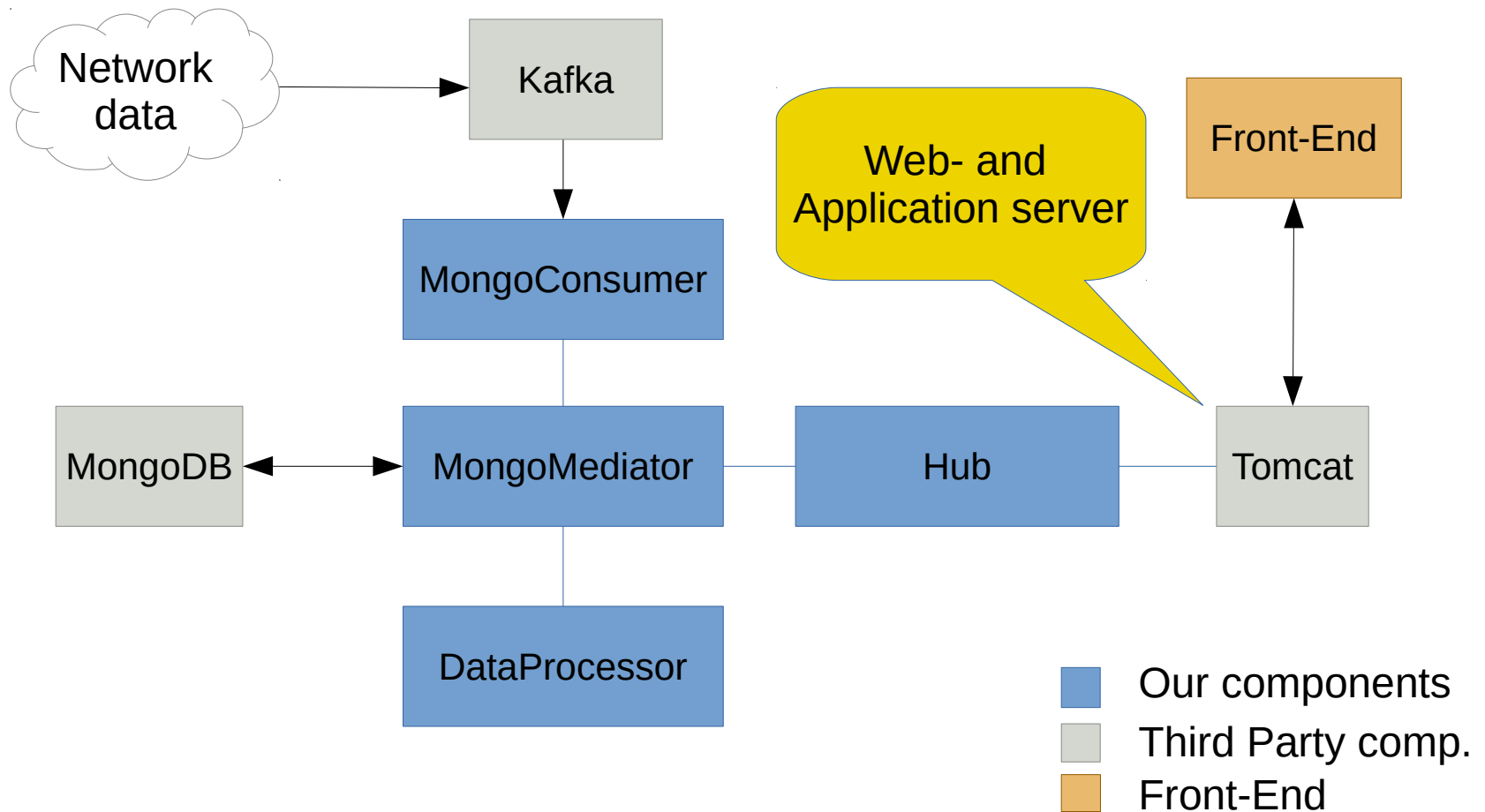
# Back-End Design



# Back-End Design

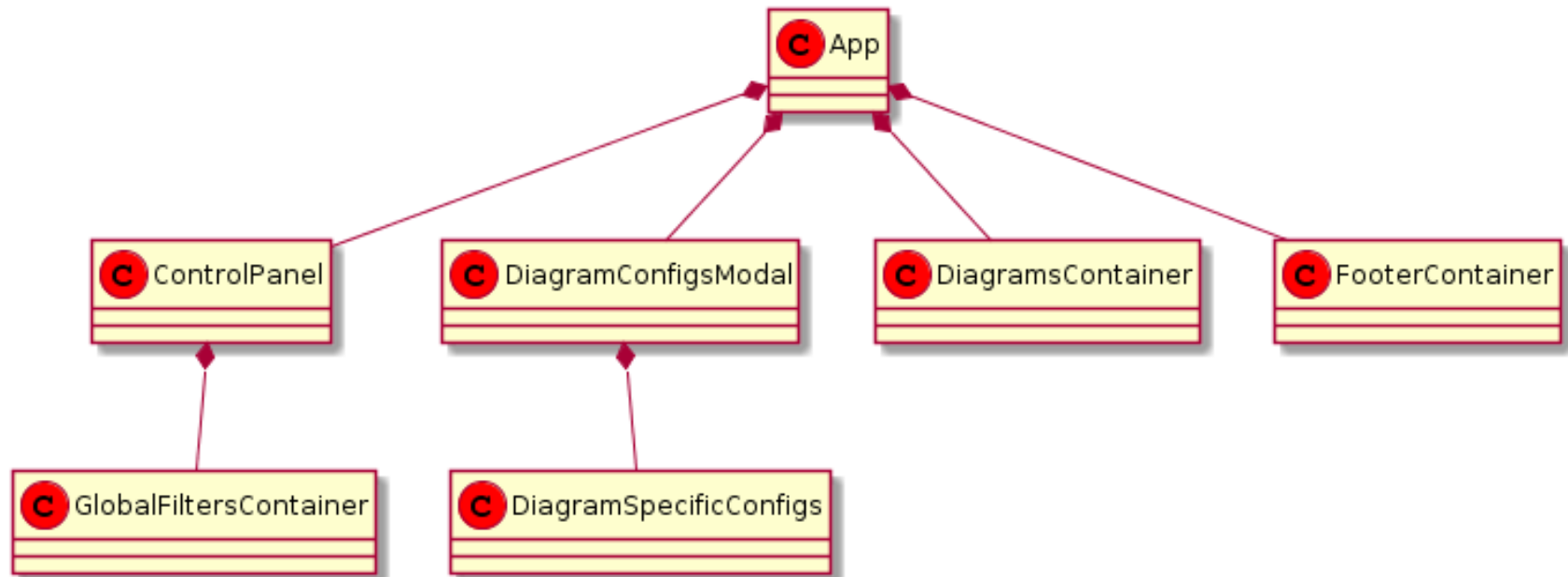


# Back-End Design

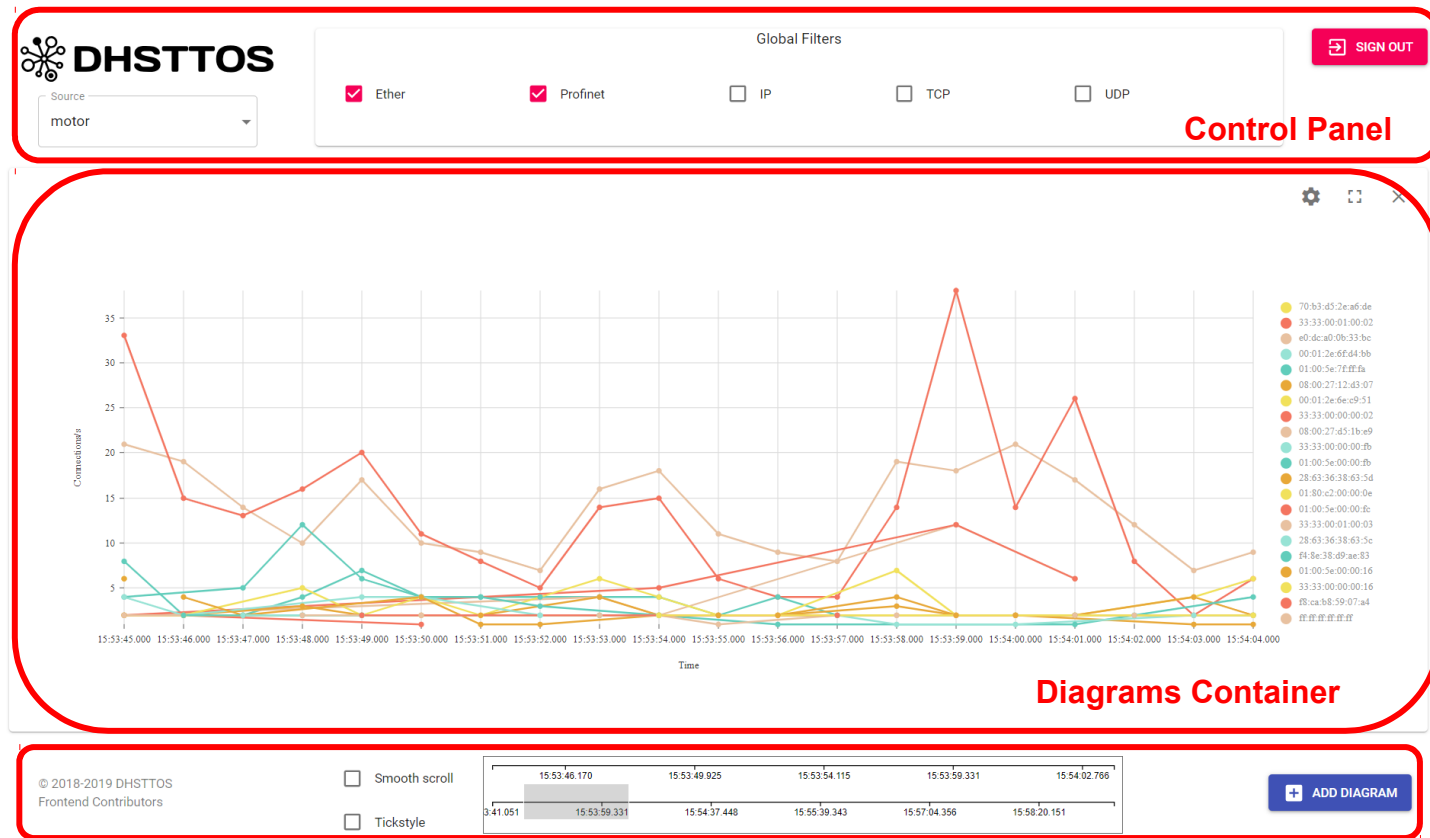


# Front-End Design

## Overview of GUI Elements



# Front-End Design



Overview



Back-end



**Front-end**



Demo

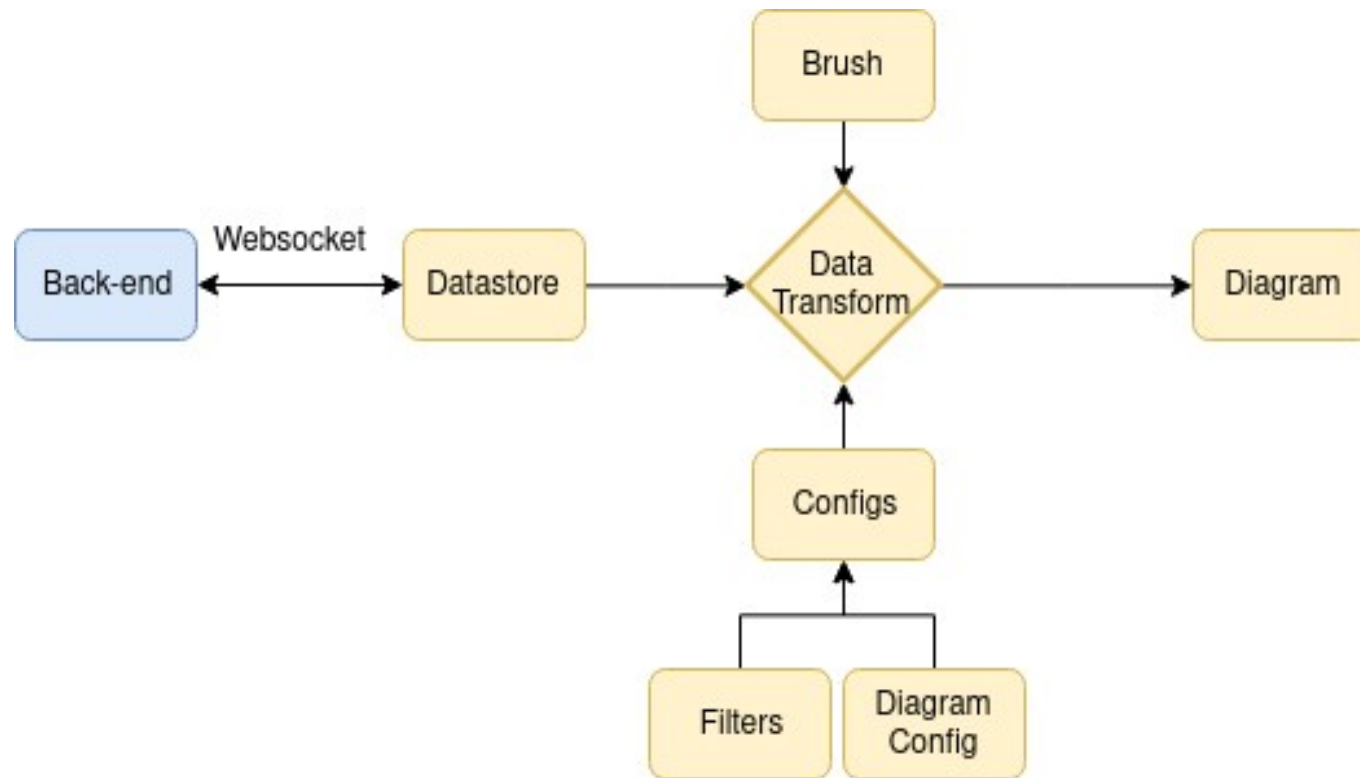


Evaluation



Conclusion

## Front-End Data Flow





# Front-End Components

- Written in Javascript
- Additional third party components (open source):

 React library

 D3 graphics library

 nivo diagram components

 MobX state management

# Development Tools Used

## ■ Back-end development:

 eclipse Eclipse

 Maven

JUnit  Junit

## ■ Front-end development:

 Visual Studio Code

 Parcel.js

 Netlify (CD)

## ■ Common tools:

 Git

 GitHub

 Slack

L<sup>A</sup>T<sub>E</sub>X Latex

# Implementation

- User access control
- Data source selection
- Multiple diagram types
- Brushing
- Modular structure
- 19 of 24 functional requirements
- Node-link diagram (partial)
- Filtering (partial)
- ✗ Data selection
- ✗ Display notifications



# Unexpected Difficulties and Challenges

- Only four team members
- Larger Scope than expected
- Many different technologies
  - Javascript and the libraries make use of multiple programming paradigms
  - Complexity of D3
  - Nivo components have inconsistent features
  - MongoDB idiosyncracies



# Lessons Learned

- Design more thoroughly
  - Especially data structures and protocols
- Plan and schedule more strictly
- Evaluate third party components more thoroughly
- Waterfall model didn't work.



# Best Practices

- Overall design was viable
- Good commit practices
- Frequent team communication
- Flexibility
- Learning from each other



# Conclusion

- We produced a working system
- Usable as a good and extensible base for future work
- Underestimated the amount of work required
- Gained experience with teamwork
- Gained understanding of technologies

