Karlsruher Institut für Technologie

Real-time Visualization of Analyzed Industrial Communication Network Traffic

FAKULTÄT FÜR INFORMATIK

The Background

- Industrial Network Security – want to understand the traffic
- Analysis of the traffic
- Real-time visualization to help the user understand
- Incidents can be detected visually

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

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

Requirements

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

- 12 Non-Functional Requirements

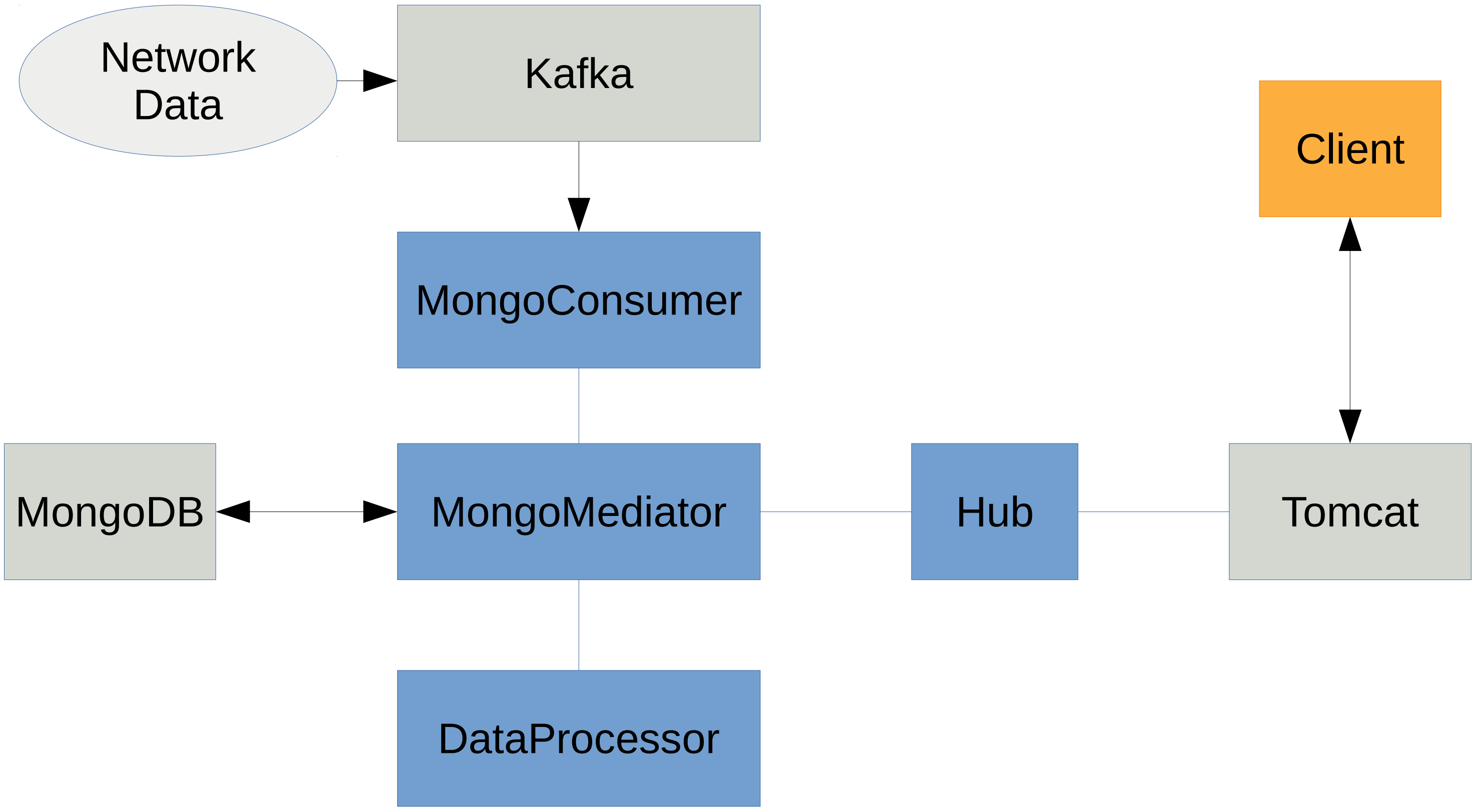
Architecture & Design

- Client-Server Architecture

- Back-End:
  - Mediator pattern
  - Strategy pattern

- Front-End:
  - Model-View-Controller
  - Observer

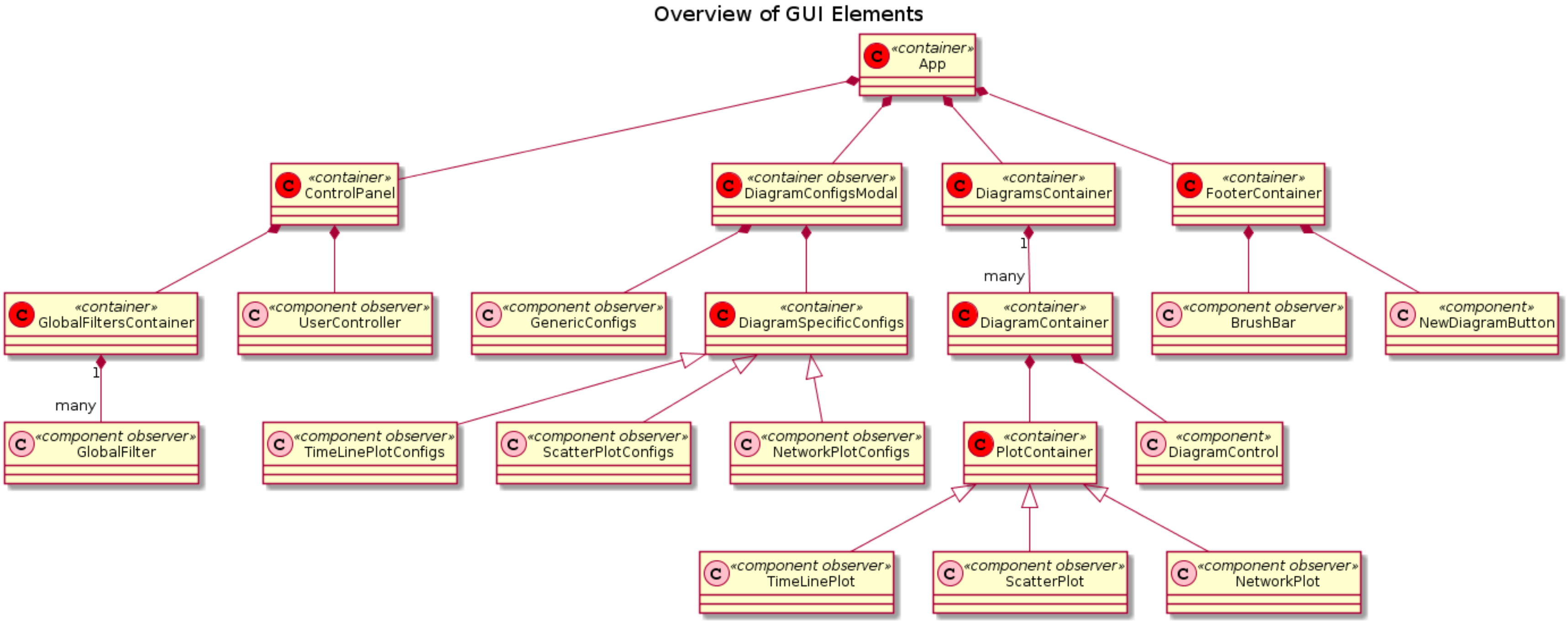
Back-End Design



Back-End Components

- Written in Java and using open source components
- Kafka streaming server
- MongoDB for both archived and real-time data
- Consumer moves data to MongoDB
- Network hub realized as servlet running in Apache Tomcat
- A Mediator component for the database accesses

Front-End Design



Overview of GUI Elements

«container» App
«container» ControlPanel
«container observer» DiagramConfigsModal
«container» DiagramsContainer
«container» FooterContainer
«container» GlobalFiltersContainer
«component observer» UserController
«component observer» GenericConfigs
«container» DiagramSpecificConfigs
«container» DiagramContainer
«component observer» BrushBar
«component» NewDiagramButton
«component observer» GlobalFilter
«component observer» TimeLinePlotConfigs
«component observer» ScatterPlotConfigs
«component observer» NetworkPlotConfigs
«container» PlotContainer
«component» DiagramControl
«component observer» TimeLinePlot
«component observer» ScatterPlot
«component observer» NetworkPlot

Front-End Components

- Written in Javascript
- React library
- D3 graphics library
- nivo diagram components
- MobX state management
- All of the above are open source

Development Tools Used

- Tools for Back-end development:
  - Eclipse
  - Maven
  - Junit

- Tools for Front-end development:
  - Visual Studio Code
  - Parcel.js
  - Netlify (CD)

- Git, GitHub
- Slack
- Latex

Implementation Summary

Implemented:

- User access control
- Accessing archived and live data
- Different diagram types
- Brushing
- Modular Structure
- 19 of 24 functional requirements

Implementation Summary

Failed to implement:

- Node-link diagram (partially)
- Filtering (partially working)
- Live data display
- Selecting data points
- Keeping lag under 2 sec

Unexpected Difficulties

- Javascript and its libraries are more functional oriented than OO
- Complexity of D3
- Nivo components have inconsistent features
- MongoDB idiosyncracies

Challenges

- Only four team members

- Many different technologies:
  - Java, Javascript
  - Kafka
  - MongoDB
  - React framework
  - D3 graphics library
  - Nivo graphics library

- Larger scope than expected

Lessons Learned

- Design more thoroughly
  - Especially data structures
- Plan and schedule more strictly
- Evaluate third party components more thoroughly
- Waterfall model didn‘t work well

Conclusion

- We produced a mostly working system
- Usable as a good and extensible base for future work
- Gained experience with teamwork