

Real-time visualization of analyzed industrial communication network traffic

Implementation Report

PSE Group

Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB

Advisor: M.Sc. Ankush Meshram

Version 1.0.0

Contents

1 Design

1.1 Introduction

XXX

1.2 Changes in the Design

XXX

from tips.pdf 7.2, page 15: "Dokumentation "uber "Anderungen am Entwurf, beispielsweise entfernte oder neu hinzugef"ugte Klassen und Methoden. Gruppirt (und zusammengefasst) werden sollte nach dem Grund f"ur die "Anderung und nicht nach der ge"anderten Klasse."

1.2.1 Refactoring for cleaner code and changes for convenience reasons

- Add parameter
Added parameter **DBname** to **MongoConsumer(user, pass, dbName)** for creating a reference to pass onto the **MongoClientMediator**
- Refactoring
Add attribute **private KafkaConsumer<String, String> consumer** because other functions need to use the consumer
- Refactor: extract instance attribute
Add attribute **private MongoDBDatabase db** as a reference to the database all methods need to access.
- Convenience functions for different data types
Added variations of **addRecordToCollection(Record record, String collection)** that take a document or an list of documents or an array of record instead of a Record.
- Add convenience function
Added **getCollectionAsRecordsArrayList()** to **DataProcessor**.
- Refactor passing the current mediator object
Add parameter **MongoClientMediator** to **public static void ProcessData:processData(String collectionName, MongoClientMediator clientMediator)** so that **processData** can use it to write the processed data to the database. Remove attribute **ProcessData:MongoClientMediator client** which was used for this before.
- Add convenience function
Add method **public static void processData(ArrayList<String> collectionNames, MongoClientMediator clientMediator)** to process a list of collections (instead of calling **processData** for each collection.

- Add convenience function
Added method **public Document getNewAggregatorDocument(Date tstamp)** for easier handling of date values.
- Add convenience attributes
Add the variables **private ArrayList<Map<String, Object>> connectionsMapList** and **private Document currentDocument** to the classes **FlowRatePerSecond** and **NumberOfConnectionsPerNode** to keep track of which document is being processed now and which connections happened within this second.
- Refactoring for cleaner code in protocol handling
Change the protocol parsing in class **ClientProtocolHandler** from a switch construct to using a private enum.

1.2.2 Changes because of clarified requirements

- Differing input formats for Date/Timestamp
Split class **PacketRecord** into **PacketRecordDesFromMongo** and **PacketRecordDesFromKafka** to handle different formats.

1.2.3 Changes because of oversights

- added dbName to MongoClientMediator since we need to know from which DB we want to read/write collections.
- Unspecified return type
The return type of **public ArrayList<Document> processData(ArrayList<Record> records)** in **IAggregator** was unspecified in the Design document.
- Session handling
To handle session state, **Hub:login()**, **Hub:loginWithToken()**, and **Hub:logout()** were added. To keep track of client session state, the private attributes **Hub:sessions** and **Hub:loginTokens** were added.

1.2.4 Changes because of unexpected complexity

- Workaround for Kafka's API
Change **getAllTopics()** to **getAllTopicsPartitions()**: return a Collection of topic partitions essentially to force kafka to send all records from the start. It was complex to make kafka read all the topics from the beginning. Secondary aspect: convenient because it relegates topic creation to another method.
- Workaround for Kafka's API
Add method **ArrayList<String> getTopicsForProcessing()** because there are some topics in kafka which are for internal use, e.g. `__consumeroffsets`. This returns the topics we need to process.

- Exception handling
The constructor for class **MongoClientMediator** now throws a `LoginFailureException` instead of forwarding an unchecked exception.
- Converting between different APIs
Add method **mongoliteratorToStringArray(Mongoliterable)** because the hub expects an array but the mongodb returns a `Mongoliterable`.
- Handling the login happening in another websocket session than the main app
To deal with a restart of the websocket connection when changing from the login page to the main page, session handling was changed. Added the **LOGIN_TOKEN** request to the protocol and **Hub:loginWithToken**.
- Adapt to React and MobX
To adapt to the observer-driven architecture of React and MobX, store data from the server in datastructures **dataStore.rawData** and **dataStore.alarms** instead of returning it as return values of **getAvailableCollections()**, **getCollection()**, **getCollectionSize()**, **getRecordsInRange()** and **getRecordsInRangeSize()** in **wsutils.js**.

1.3 List of implemented must- and should-criteria

1.3.1 List of implemented must-criteria

FR100, FR110, FR200, FR300, FR400, FR700, FR710, FR720, FR1310

in progress: FR500, FR1300

not yet: FR800, FR900, FR910

1.3.2 List of implemented should-criteria

- FR1332 filter to compute flow rate
 - this has instead been implemented in the backend which provides this as a new data stream
- FR1400

1.3.3 List of not implemented must-criteria

- FR600 dynamically change the selected/displayed components
- FR1000 auto scroll
- FR1100 pick data points, hover
- FR1110 node-link diagram: picking both nodes and links
- FR1200 selecting data points
- FR1210 create new diagram from selected data
- FR1330

1.3.4 List of not implemented should-criteria

- FR1320 per-diagram filters

1.4 Delays

XXX

Welche Verzögerungen gab es im Implementierungsplan? Kann beispielsweise als zweites GANTT Diagramm am Ende dargestellt werden.

1.5 Overview of unit tests

XXX