GeeLytics: Geo-distributed Edge Analytics for Large Scale IoT Systems Based on Dynamic Topology

Bin Cheng, Apostolos Papageorgiou, Flavio Cirillo, Ernoe Kovacs NEC Laboratories Europe, Heidelberg, Germany

2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)

Introduction

- IoT analytics
  - Find relevant pieces of information in the flood of IoT data

- IoT system characteristics
  - IoT data are usually unstructured stream data and constantly generated from geo-distributed sensors over time
  - Mobility and colocation of sensors and actuators
  - Actuators often expect actionable low latency results
  - Raw data and derived results are expected to be shared

Introduction

- Move analytics to the edge
  - Process or compress data before transmitting the data to the Cloud, or transmit only selected data or derived results
  - Reduce the bandwidth cost between the network edges and the Cloud

Contributions

- A set of use faces
- An gap analysis between existing stream processing platforms and the edge analytics platform
- A high-level approach for enhancing the dynamically of the actual execution of the topology tasks
- The preliminary design of an edge analytics platform (GeeLytics)

Motivating Use Cases and Requirements

- Use case 1: smart traffic light system with connected cars
  - IoT data and the driver must be immediately informed about an accident or danger probabilities
- Use case 2: multi-modal data fusion for crowd prediction
  - It is important to keep the results up-to-date in real time, as well as avoid sending/storing irrelevant data to the Cloud
- Use case 3: distributed object tracking for public safety
  - Police officers can ask the platform to track specific objects and receive fast notifications
- Use case 4: globalized smart city platform
  - Support real-time applications in the cross-continent city despite platform sharing

Motivating Use Cases and Requirements

- The latency is expected to be as low as possible
- The impact of data transmission from the edges to the Cloud can be substantial
- The workload introduced by geo-distributed actuators is dynamic
- Location-awareness and mobility of actuators must be considered

Motivating Use Cases and Requirements

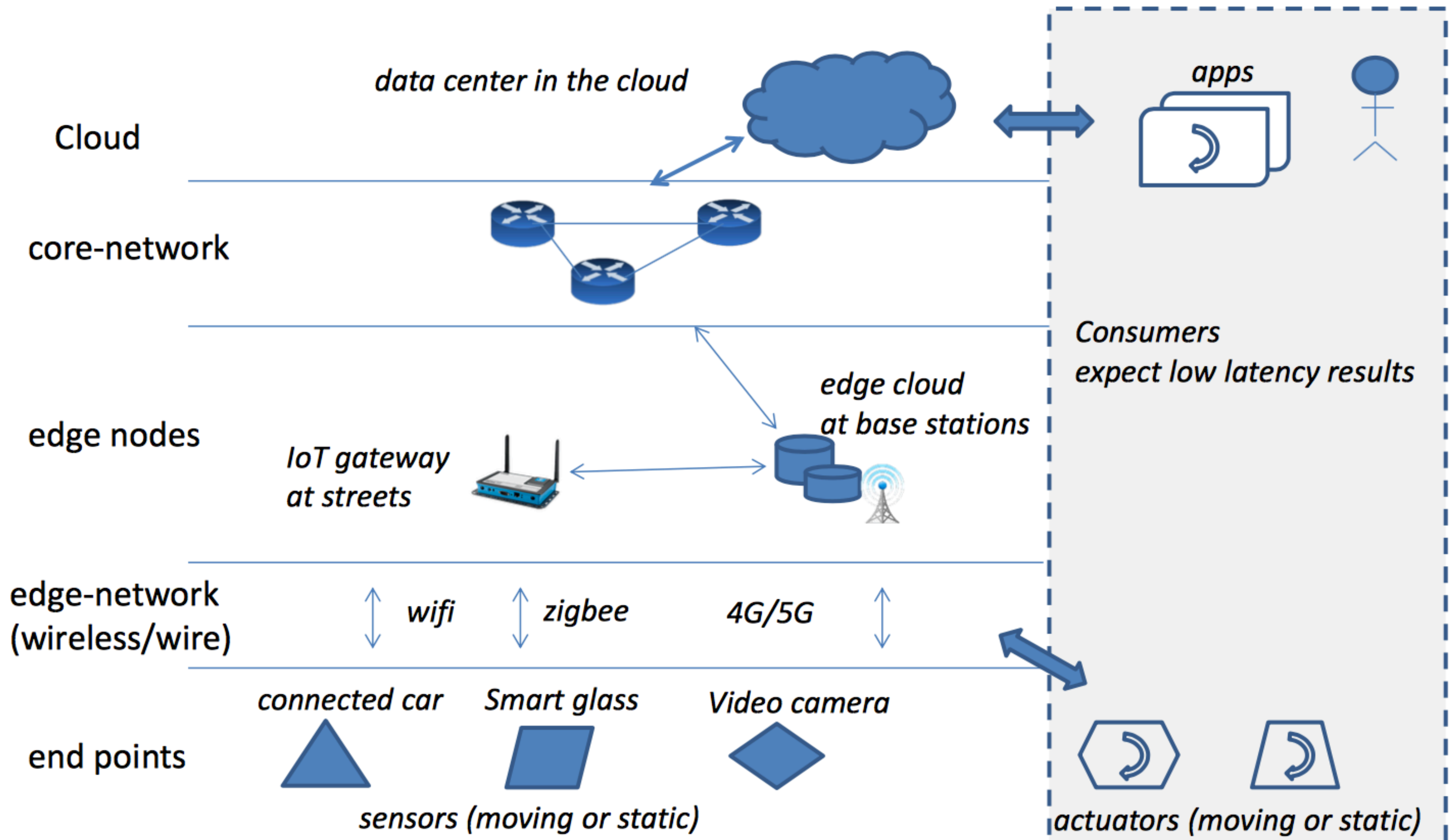

Fig. 1: System setup for GeeLytics

Gap Analysis - Related Work

- Fog computing
  - Extend cloud computing to the edge by allowing data processing to happen at the network edge
  - Lacks sophisticated data analytics platforms that allow us to efficiently utilize the power of the edges and the Cloud together

Gap Analysis - Related Work

- Stream processing platforms
  - Enable real-time stream processing in the Cloud
  - Only support static task topologies and have poor support for multi-tenancy
  - Reducing the internal network traffic and dealing with the heterogeneity of nodes are not their major focus

Gap Analysis - Related Work

- Stream processing platforms
  - Only consider the case that analytics results are derived from the entire dataset and consumed from the Cloud
  - The number of edge nodes is beyond what any current stream processing platforms can handle

Gap Analysis - Related Work

- Edge analytics
  - Leverage the power of fog computing and cloud computing to support real time stream processing
  - Does not consider how to define topologies to do customized stream processing on top of the edges and the Cloud
  - None of them seems to support multi-tenancy and dynamic topology execution.

Gap Analysis - Target of GeeLytics

- Support flexible and efficient edge analytics for large scale IoT systems based on dynamic topology execution
- It allows application developers to run customized stream processing tasks over network edges and the Cloud
- Dynamically instantiate involved tasks according to the current workload and schedule them to the right place

Gap Analysis - Target of GeeLytics

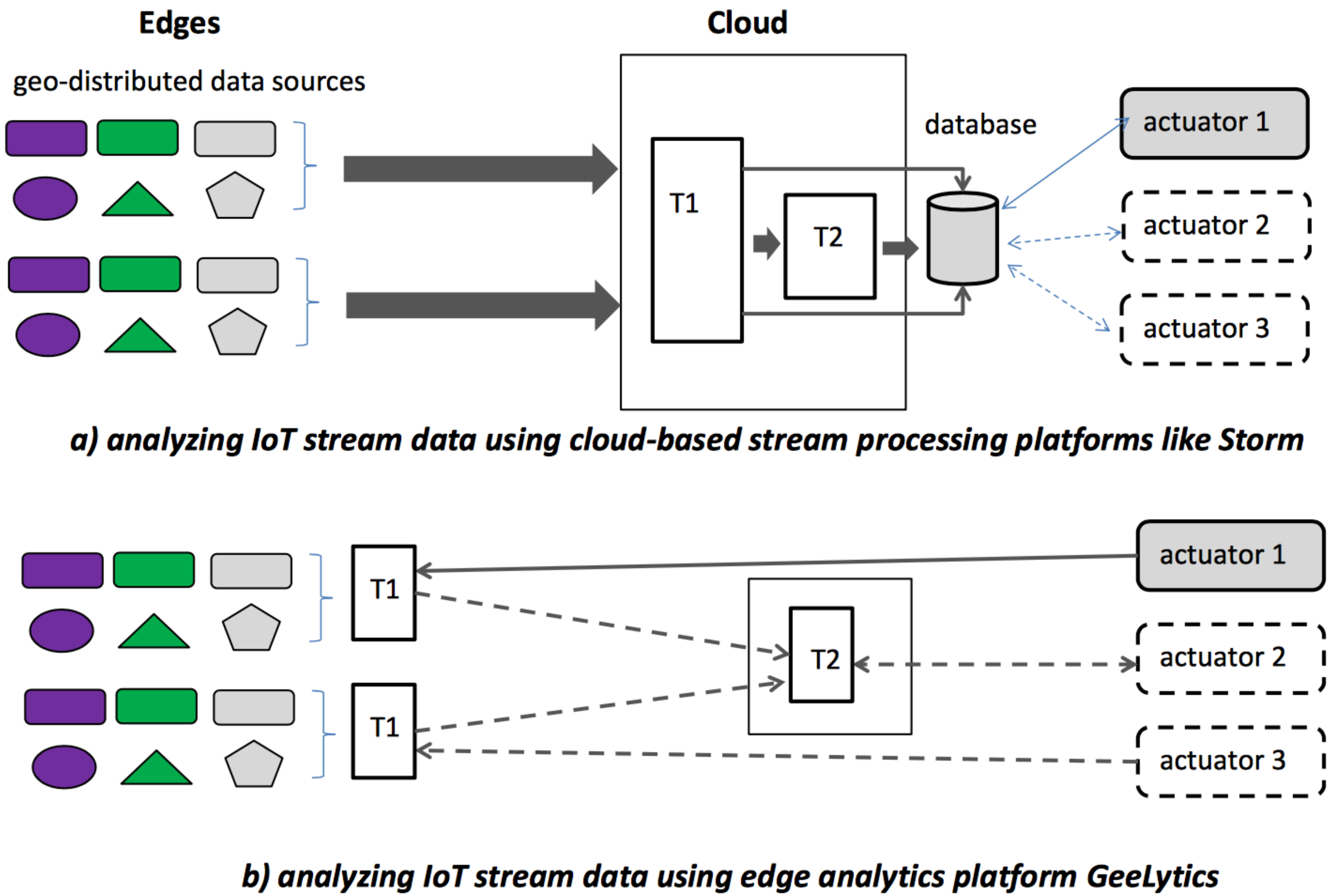

Fig. 2: Illustration of the key difference between GeeLytics and existing stream processing platforms

Gap Analysis - Target of GeeLytics

- GeeLytics is designed to meet the following design goals
  - Scalability
  - Flexible Application Interfaces
  - Multi-tenancy Support
  - Openness and Security

GeeLytics - System Architecture

- IoT Agent:
  - The worker that is capable of performing stream processing tasks
- Task Container
  - Every task is wrapped up as an application container
- Topology Master
  - Manage all involved stream processing task
- Controller
  - Manage all system resources and core components

GeeLytics - System Architecture

- Front-end Server
  - Application interfaces are supported by the front-end server via HTTP REST
- Broker
  - Distributed message exchange system
- Global State Storage System
  - Save some of the intermediate states to tolerate unexpected failures

GeeLytics - System Architecture

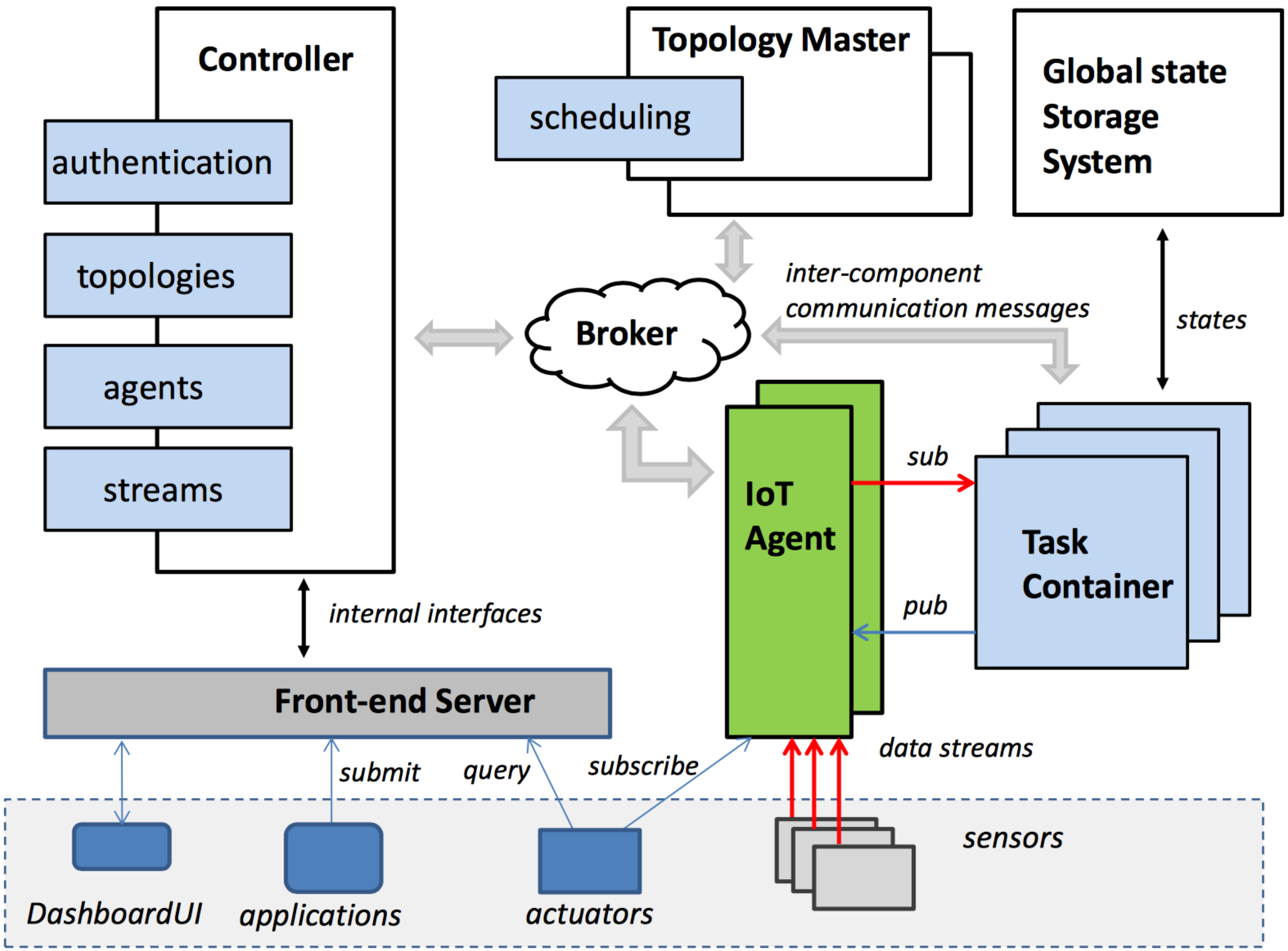

Fig. 3: System architecture of GeeLytics

GeeLytics - Task and Topology

- Task topology is used to define the relationship between different stream processing tasks
- All data streams generated by each task in the task topology are accessible to actuators
- The processing topology is constructed and changed as actuators join and leave
- A task just needs to follow a pub/sub communication interface

GeeLytics - Task and Topology

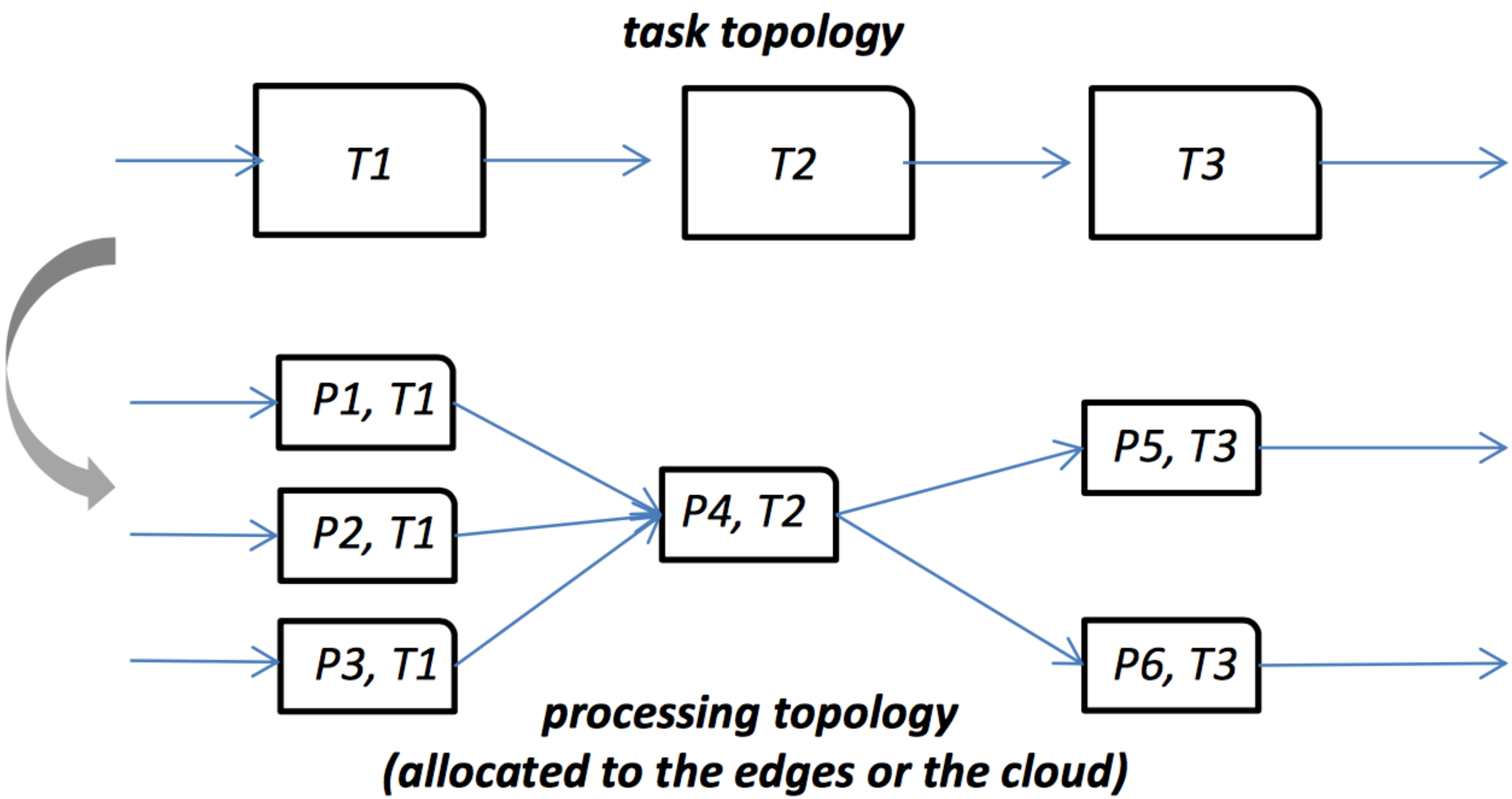

Fig. 4: task topology and processing topology

Open Research Issues

- Resource Orchestration
- Task Scheduling
- Security of System and Data

Conclusion

- GeeLytics is designed as a geo-distributed edge analytics platform
- They only proposed the preliminary design and didn't mention the details