Mobile Cloud Computing

EINSTEIN CENTER Digital Future

Technische Universität Berlin

Fog Computing

Bermbach | Part 2: Data Distribution

Agenda

Lectures

From Cloud to Fog Computing	**Data Distribution**	**Data Management**	**Platforms & Applications**	**Testing & Benchmarking**	**LEO Edge Computing**

Assignments

Prototyping Assignment	Reading Assignment

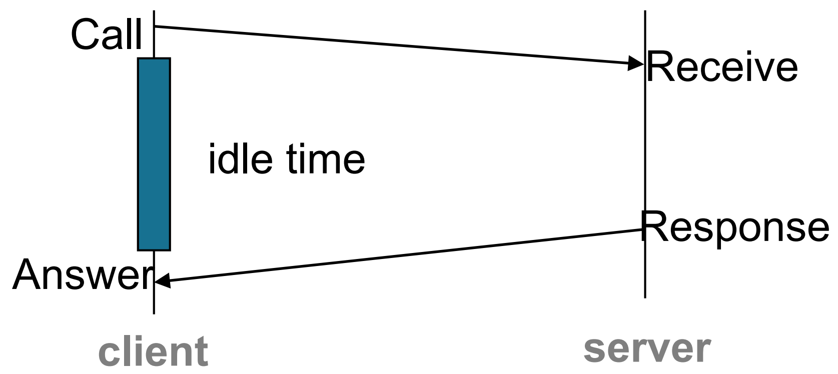
Wrap-up

Q&A	Final Test

Data Distribution

SYNCHRONOUS & ASYNCHRONOUS COMMUNICATION

Synchronous Communication



Traditionally, information systems use blocking, synchronous calls: the client sends a request to a service and waits for a response of the service to come back before continuing doing its work

Examples: phone call, method call in Java

Source: Gustavo Alonso

Blocking or synchronous interaction

Synchronous interaction requires both parties to be “on-line”: the caller makes a request, the receiver gets the request, processes the request, sends a response, the caller receives the response.

The caller must wait until the response comes back. The receiver does not need to exist at the time of the call but the interaction requires both client and server to be “alive” at the same time.

Source: Gustavo Alonso

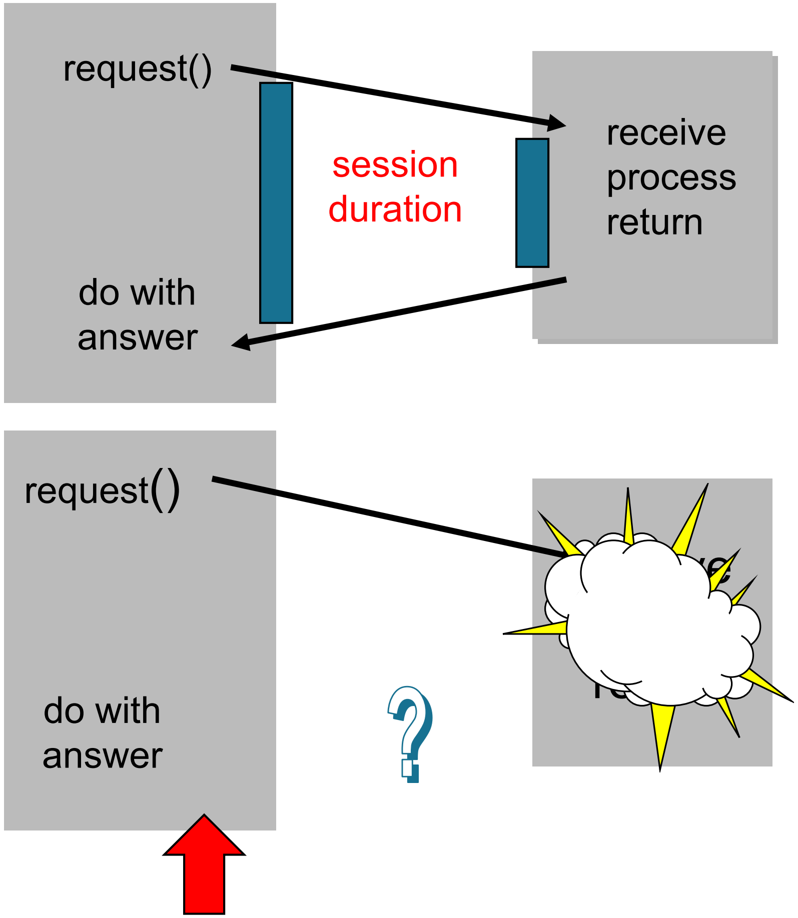
Disadvantages of blocking interactions

In addition to connection overhead, the following disadvantages also apply:

- Higher probability of failures
- Difficult to identify and react to failures
- It is a one-to-one system; it is not really practical for nested calls and complex interactions (the problems become even more acute)

Source: Gustavo Alonso

Failures happen!



Context is lost
Needs to be restarted!!

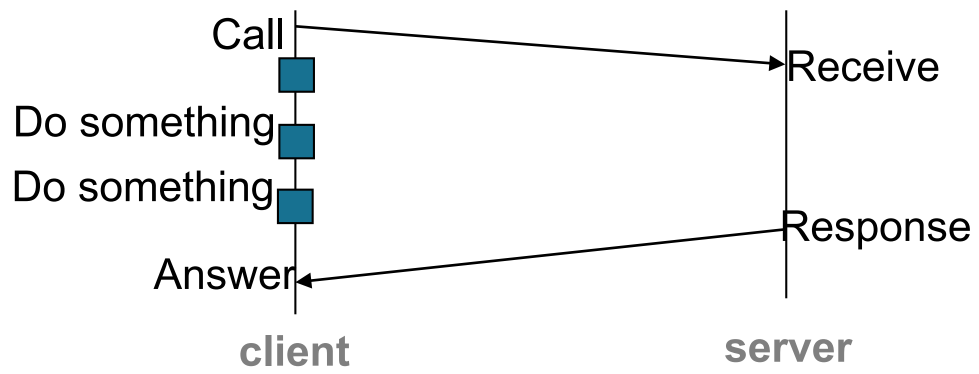
Who is responsible for finding out what happened?

Finding out *when* the failure took place is not easy!

Worse still, if there is a chain of invocations (e.g., a client calls a server that calls another server) the failure can occur anywhere along the chain.

Source: Gustavo Alonso

Asynchronous Communication



An alternative are asynchronous calls: the client sends a request to a service. While it is waiting for a response, it can do other things.

Often implemented with message queues

Examples: Email, Javascript callbacks, NodeJS event loop

Source: Gustavo Alonso

TYPES OF DECOUPLING

Messaging services might offer

Technische Universität Berlin

Space

- Clients do not need to be at the same location
- They do not need to know where the other party is located

Time

- Clients do not need to be online at the same time
- Sometimes: when clients reconnect, prior messages are delivered

Technology

- Clients do not need to be created with the same programming language, tool or framework

Data Format

- Messaging system uses a standardized data format such as JSON
- It may offer interfaces for other formats

Mobile Cloud Computing

EINSTEIN CENTER Digital Future

Technische Universität Berlin

Data Distribution

MESSAGING PATTERNS

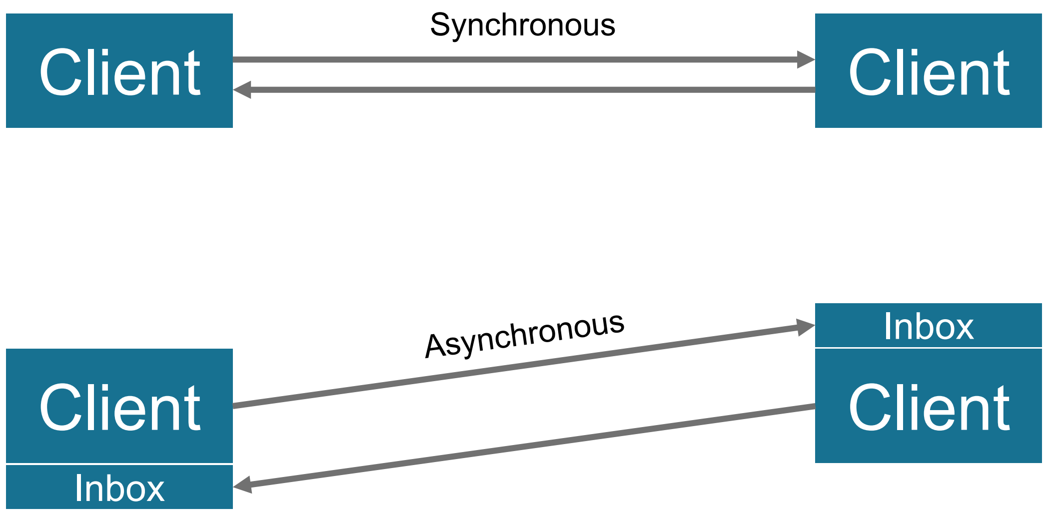
Mobile Cloud Computing

EINSTEIN CENTER Digital Future

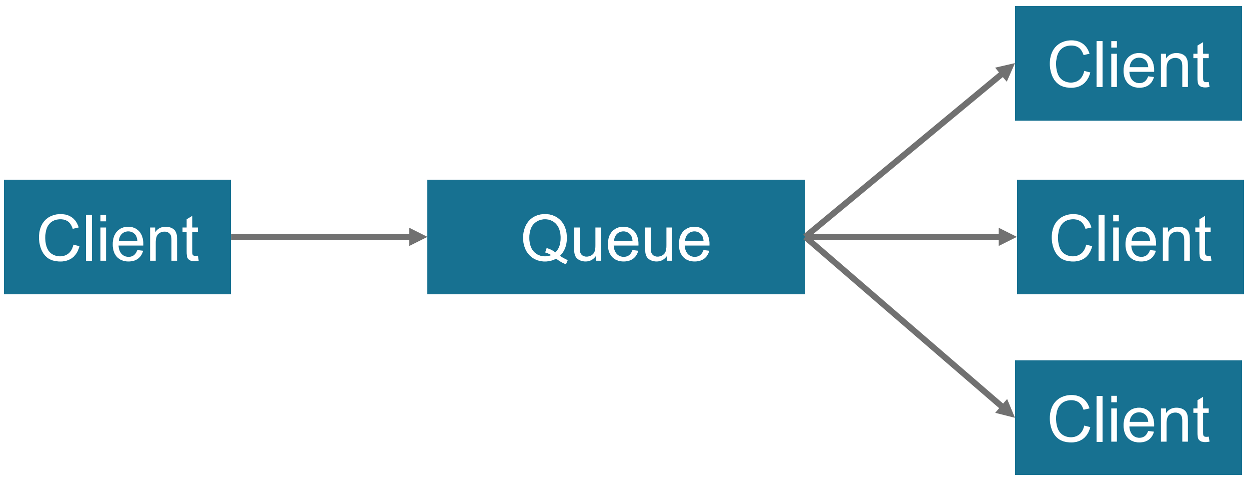
Overview

- Request / Response (1 to 1)
- Load Balancing (1 to many)
- Fan-out / Fan-in (1 to many / many to 1)
- Broadcasting (many to many)
- Pub/Sub (many to many, but structured)

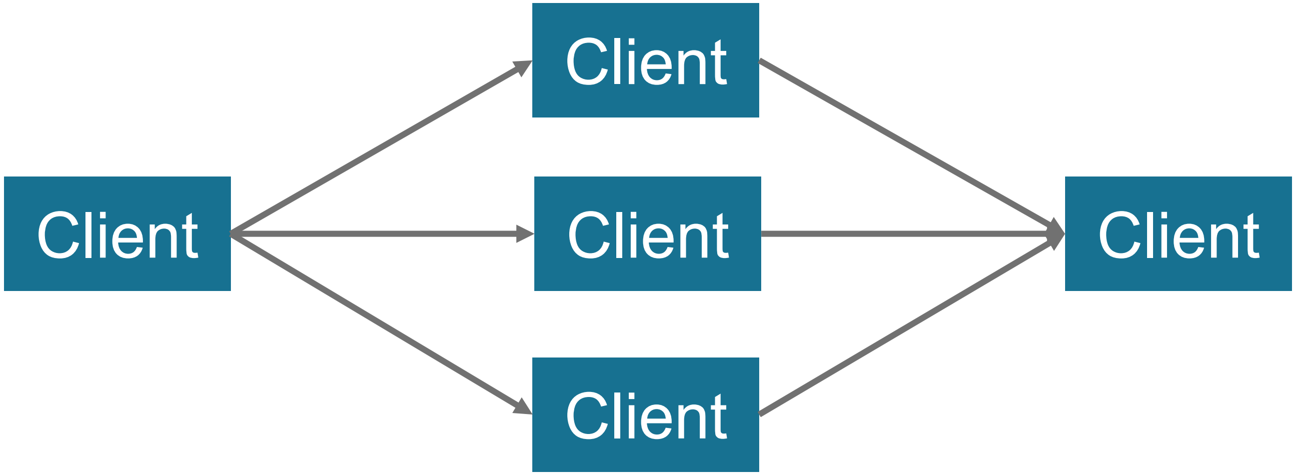
Request / Response



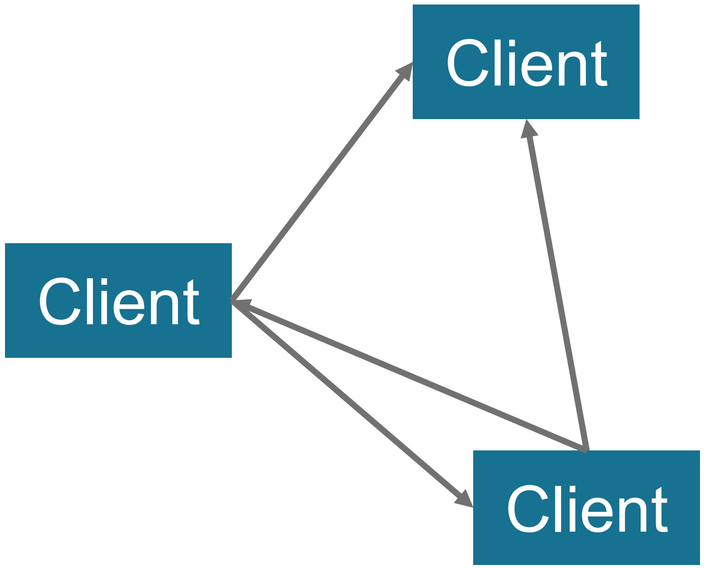
Load Balancing



Fan-out / Fan-in



Broadcasting



Data Distribution

PUB/SUB - BASICS

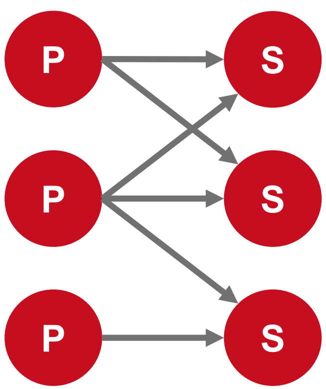
Publish/Subscribe Paradigm

Clients can act as

- Publisher: creates content by publishing events
- Subscriber: consumes content by creating subscriptions that match certain events

By acting as publisher and subscriber simultaneously, clients can send and receive data

Communication is many-to-many



Matching of Events and Subscriptions

There are various event/subscription matching strategies

- Channel-based
  - Subscriptions target individual channels
- Topic-based
  - Subscriptions are based on types/subjects/headers
- Content-based
  - Subscriptions are based on content of the event itself

Low level of expressiveness

↓

High level of expressiveness

Example: MQTT Pub/Sub Protocol

Lightweight and open Pub/Sub protocol

Designed for devices that run in constrained environments

Topic-based, topics can have multiple levels

	Event Topic				
Subscription Topic	a/b	a/c	a/b/b	a/b/c	b/b/b
a/b	✓	✗	✗	✗	✗
a/b/c	✗	✗	✗	✓	✗
+/b/+	✗	✗	✓	✓	✓
a/#	✓	✓	✓	✓	✗

Broker-based vs. PTP-based Pub/Sub

Broker-based

- Publisher and subscriber do not interact directly
- A broker handles client communication
- MQTT protocol assumes broker-based setup => relieve client devices

PTP-based

- Publisher and subscriber have to route messages themselves
- For this, they need to match messages and be aware of other clients
- Good for smaller setups with strong machines

=> Broker-based setups are (often) a good fit for the fog

Data Distribution

PUB/SUB - MESSAGE DISSEMINATION

Setup

Various strategies for message dissemination with broker-based setup

There are multiple geo-distributed brokers

Clients connect to the physically closest broker

Inter-Broker Routing Strategies

Event Flooding or Subscription Flooding

- Events or subscriptions are broadcast to all other brokers
- Minimizes end-to-end latency
- Possibly a lot of excess data

Gossiping

- Messages are distributed based on probability distribution
- High tolerance for very dynamic environments
- Messages might not arrive at all or with high delay

Inter-Broker Routing Strategies

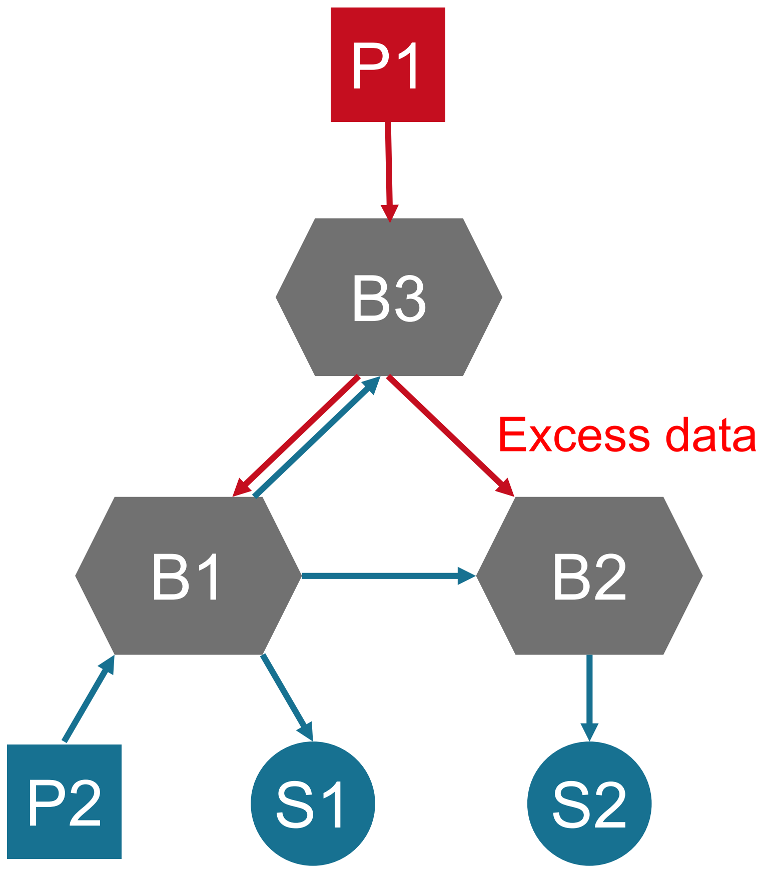
Selective - Filtering

- Good for network configurations where not all brokers are interconnected
- Subscription information are exchanged with neighbors
- Events are only forwarded towards brokers that lie on a path to a subscriber

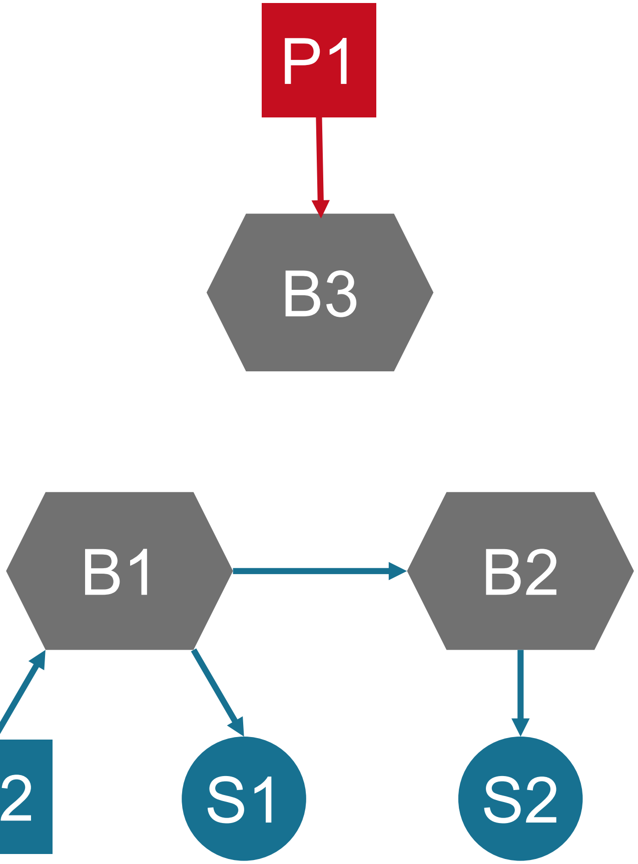
Selective - Rendezvous Points (RP)

- RPs are a meeting point for events and subscription
- Must be close to clients -> otherwise high end-to-end latency

Event Flooding

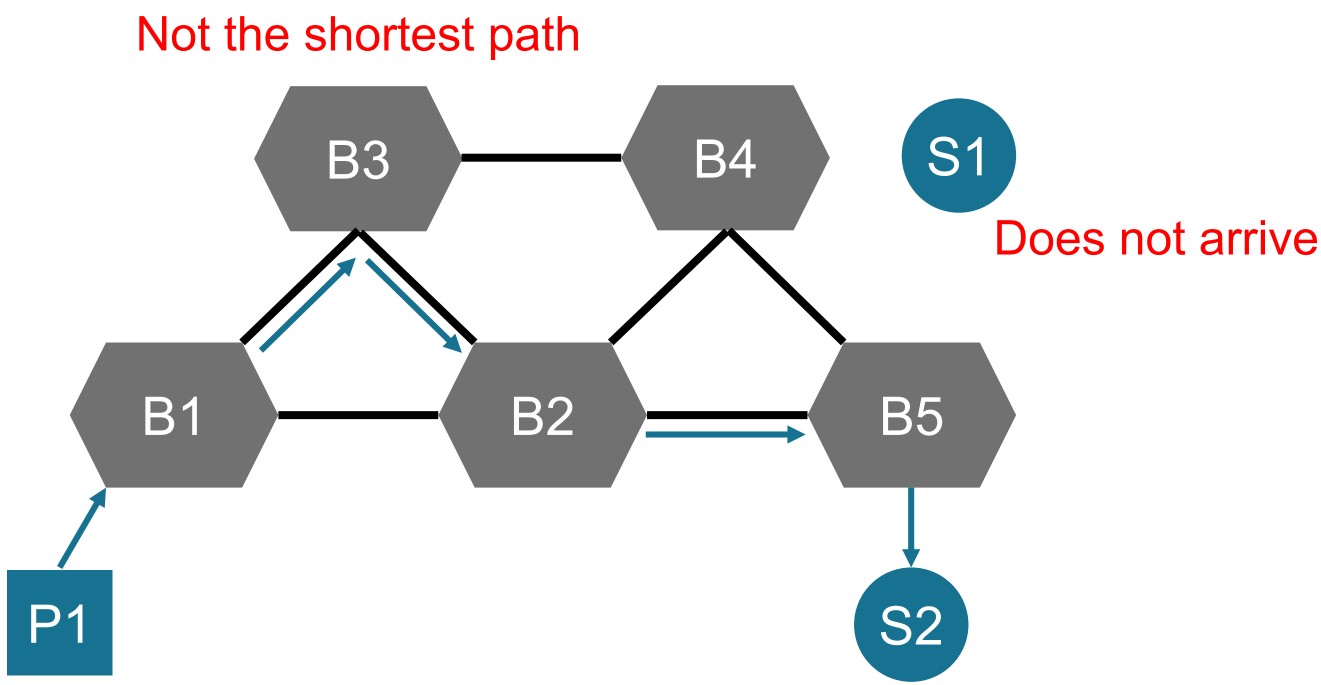


Subscription Flooding

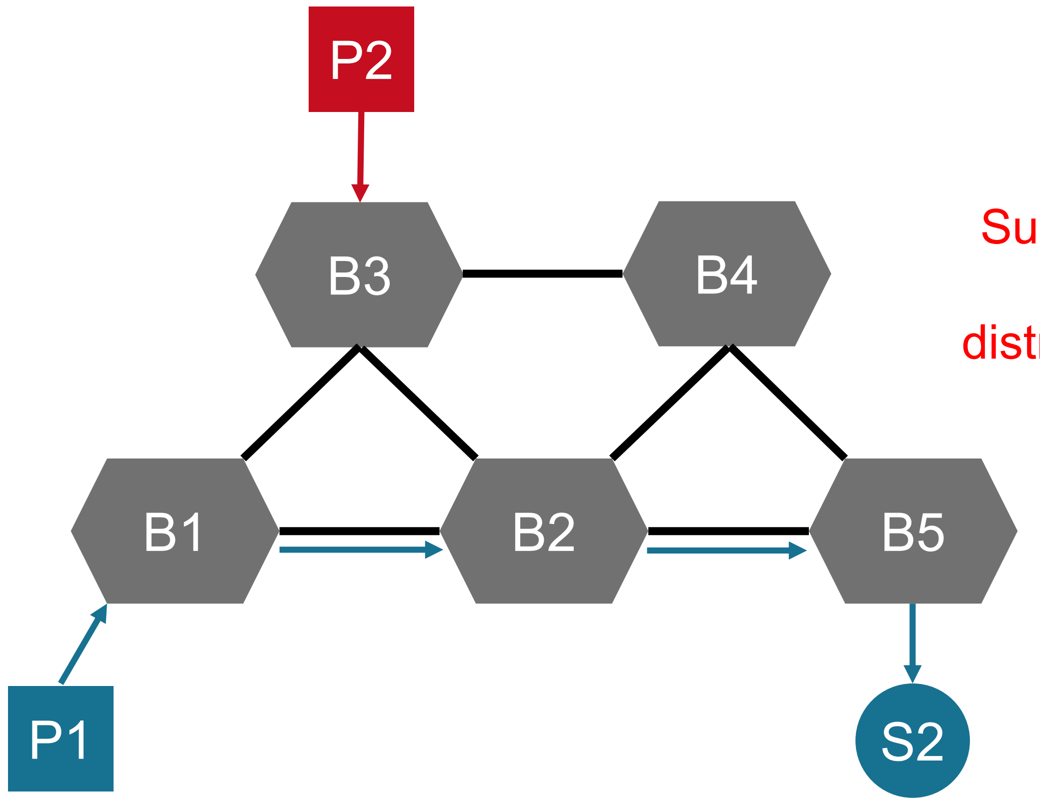


Subscription flooding might lead to excess data too!

Gossiping

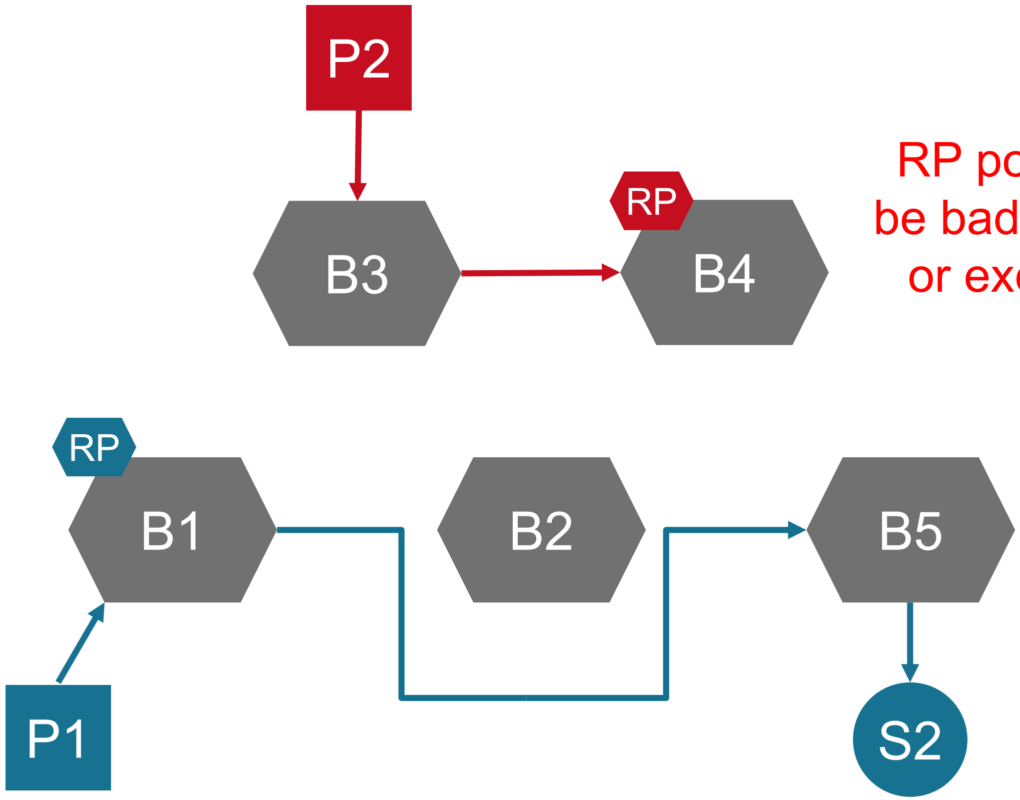


Selective – Filters



Subscriptions must be distributed first!

Selective – Rendezvous Points



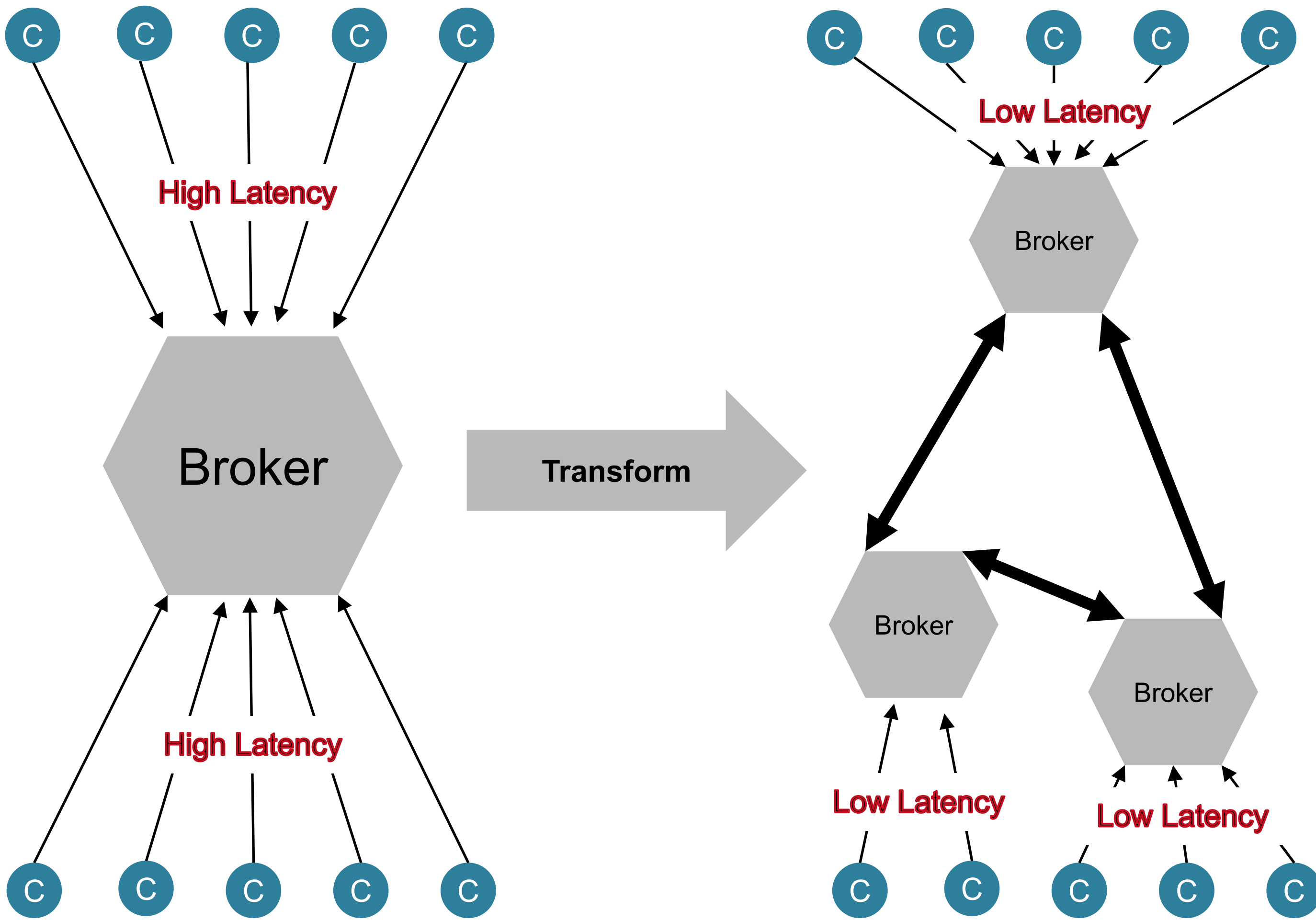
RP position can be bad for latency or excess data

Data Distribution

CASE STUDY: BROADCAST GROUPS

Hasenburg, Jonathan, Florian Stanek, Florian Tschorsch, and David Bermbach. "**Managing Latency and Excess Data Dissemination in Fog-Based Publish/Subscribe Systems**." In *2020 IEEE International Conference on Fog Computing*. IEEE, 2020.

Goal: Bringing pub/sub to the fog

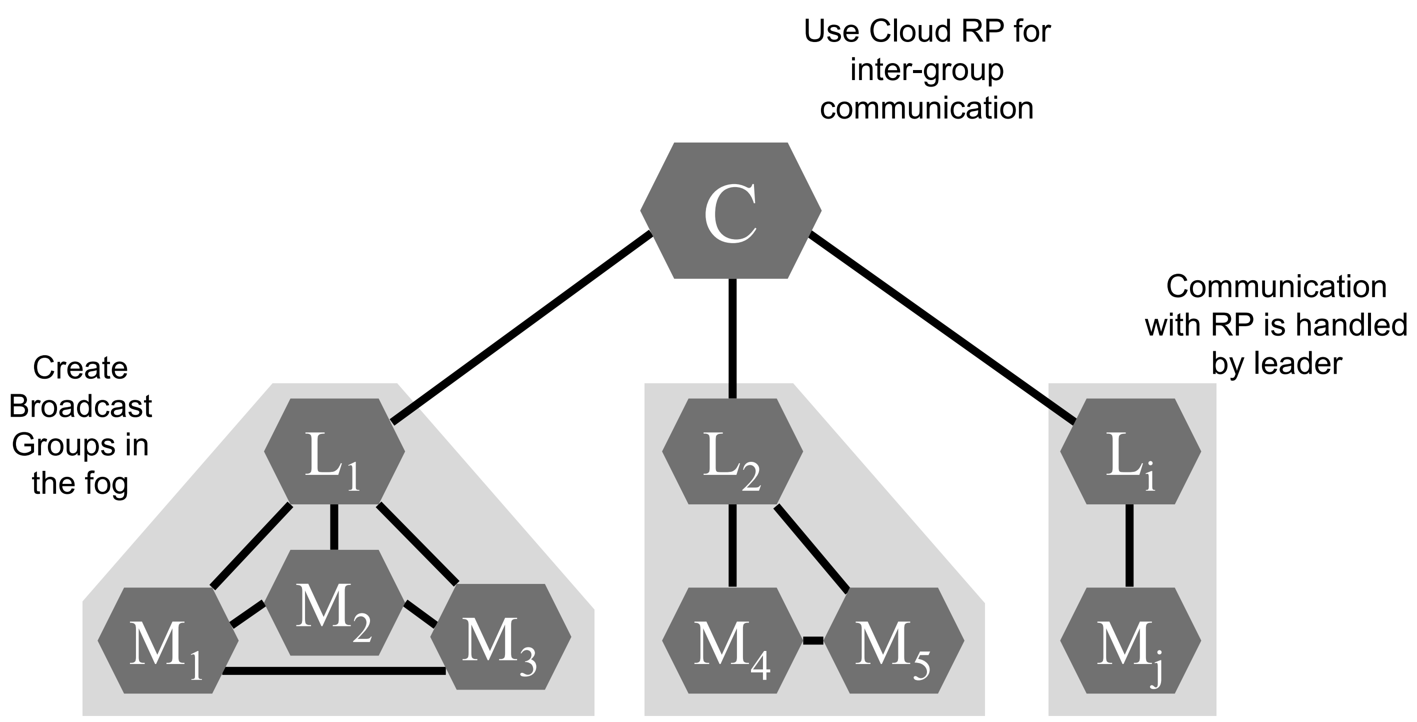


Combining Flooding and Rendezvous Points

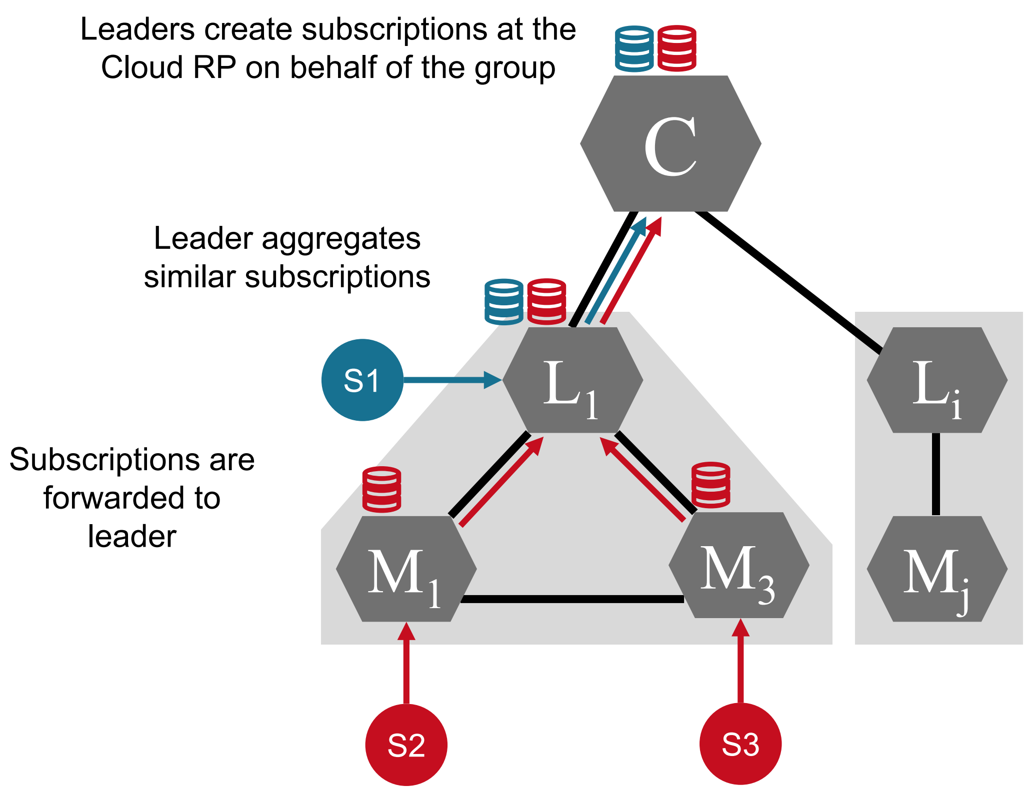
- Global flooding
  - Broadcast messages to all brokers
  - Communication latency is optimal, but a lot of excess data
- Rendezvous Point in the Cloud
  - Fog brokers forward events to a central cloud broker -> cloud decides which other fog brokers need events
  - Minimizes excess data, but increases latency

➢ Tradeoff between latency and excess data dissemination
➢ Idea is to combine both and manage the tradeoff

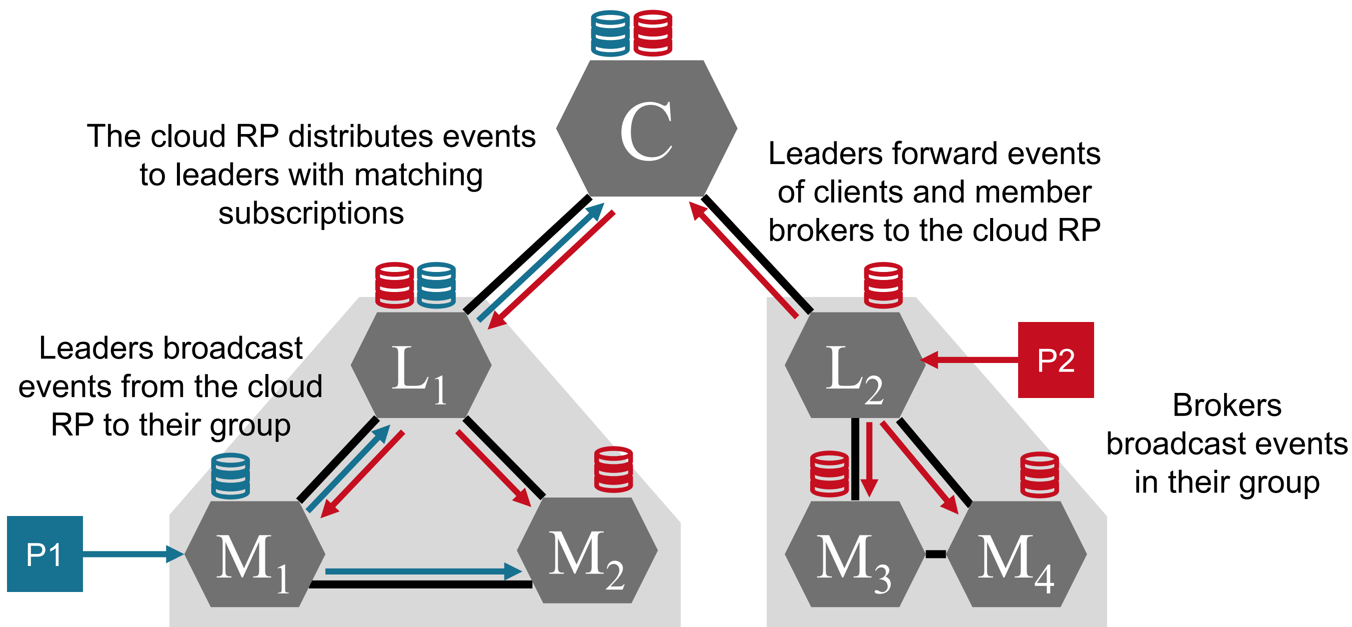
Balancing the tradeoff with broadcast groups



Dissemination of subscriptions



Dissemination of events



Broadcast group formation

Initially, each broker takes the role of a leader

Leaders subscribe to a dedicated topic at the cloud RP to detect other leaders

Leaders measure latency to other leaders
=> if below given latency threshold, do a merge

For merge:
- Determine new group leader, e.g., based on compute resources
- Migrate members to new leader

If latency to a leader is above given latency threshold, leave group

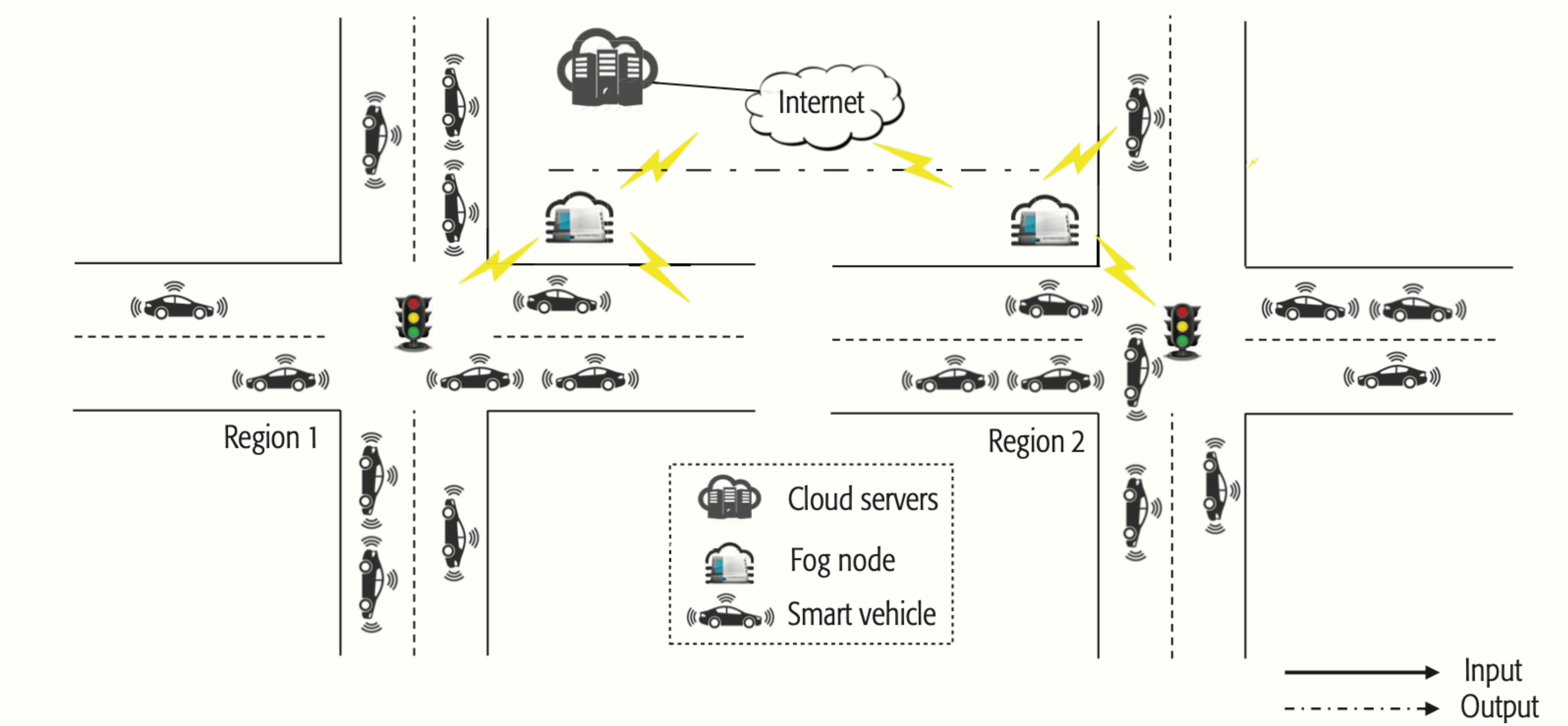
⇒ Latency threshold controls group size
⇒ Can be used to manage the latency vs excess data tradeoff

Data Distribution

CASE STUDY: VEHICULAR FOG COMPUTING

Huang, Cheng, Rongxing Lu, and Kim-Kwang Raymond Choo. "**Vehicular Fog Computing: Architecture, Use Case, and Security and Forensic Challenges**." *IEEE Communications Magazine* 55, no. 11 (November 2017)

System overview



Scenario: traffic control data flows (1)

Vehicles

- Collect data
- Use it for vehicle-level decisions
- Transmit data to closest fog nodes
  - ➢ Asynchronous Request/(Reply) or Fan-Out

Fog nodes

- Process data of multiple cars for area-level decisions
- Send instructions to traffic lights
  - ➢ Synchronous Fan-In / Fan-Out
- Sent aggregated status reports to cloud
  - ➢ Synchronous Fan-In

Scenario: traffic control data flows (2)

Technische Universität Berlin

Traffic lights

- Operate as defined by instructions

Cloud

- Processes data from fog nodes for city-level decision
- There might be an internal load balancer
- Publish traffic information to subscribed vehicles
    - ➢ Publish/Subscribe

Mobile Cloud Computing

EINSTEIN CENTER Digital Future

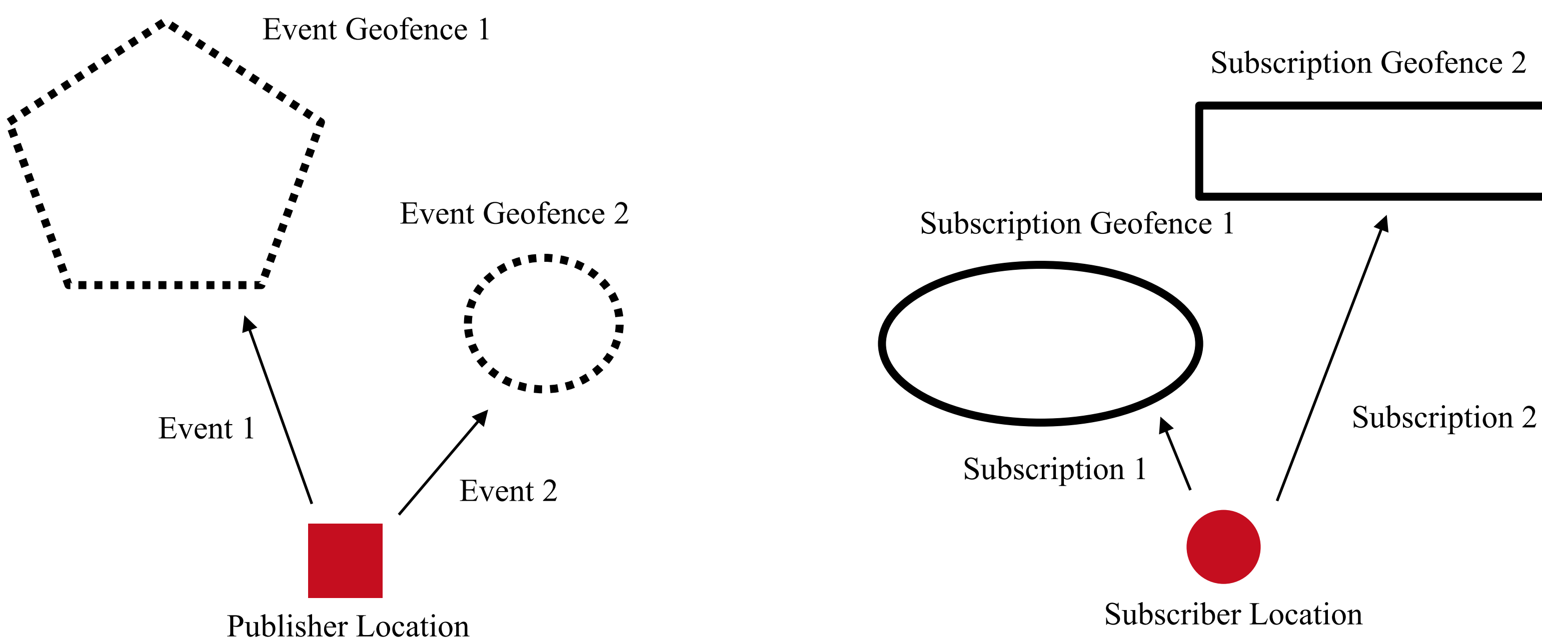
Data Distribution

CASE STUDY: DISGB

Hasenburg, Jonathan and David Bermbach. "**DisGB: Using Geo-Context Information for Efficient Routing in Geo-Distributed Pub/Sub Systems**" In *2020 IEEE/ACM International Conference on Utility and Cloud Computing*. ACM, 2020.

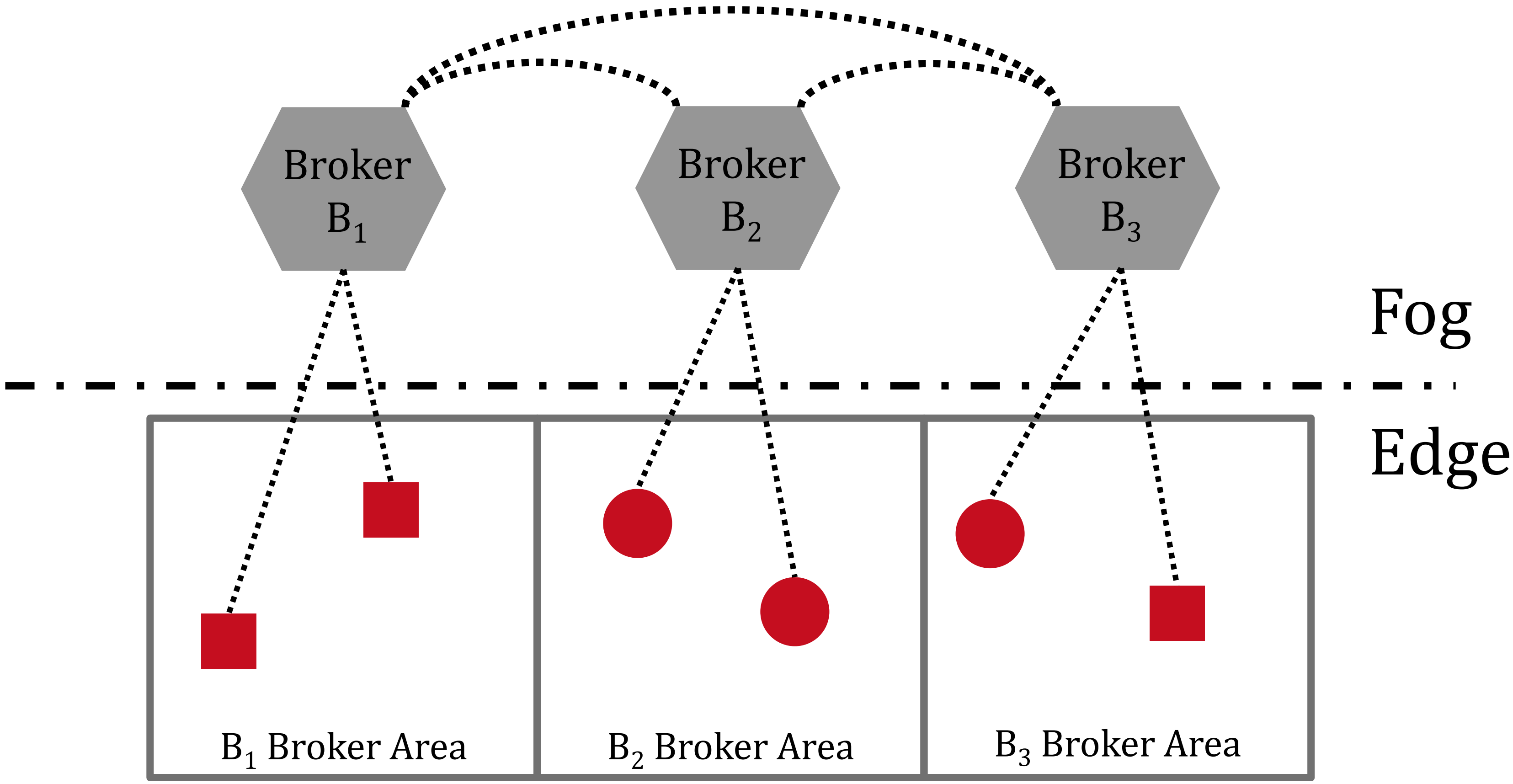
Geo-context aware data distribution

- IoT data distribution is often non-uniform
- ➢ It depends on where events are relevant / where relevant events can come from
- ➢ Can be expressed with geo-contexts



How can this information be used in geo-distributed systems?

Technische Universität Berlin



Mobile Cloud Computing

EINSTEIN CENTER Digital Future

Idea: use geo-contexts to identify RPs

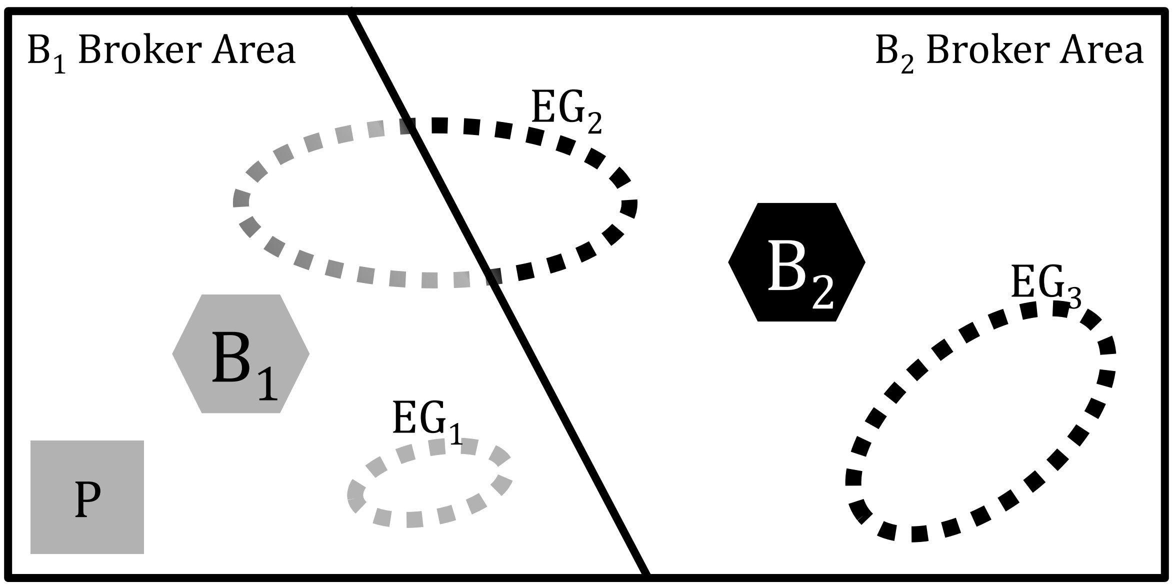
- The event geofence can be used to identify RPs that are close to the subscribers of an event
- The subscription geofence can be used to identify RPs that are close to the publishers of events

=> There are two strategies for selecting RPs based on geo-context

Selecting RPs close to the subscribers

The RPs for an event are all brokers that are the respectively closest broker to each of the subscribers that have created a matching subscription.

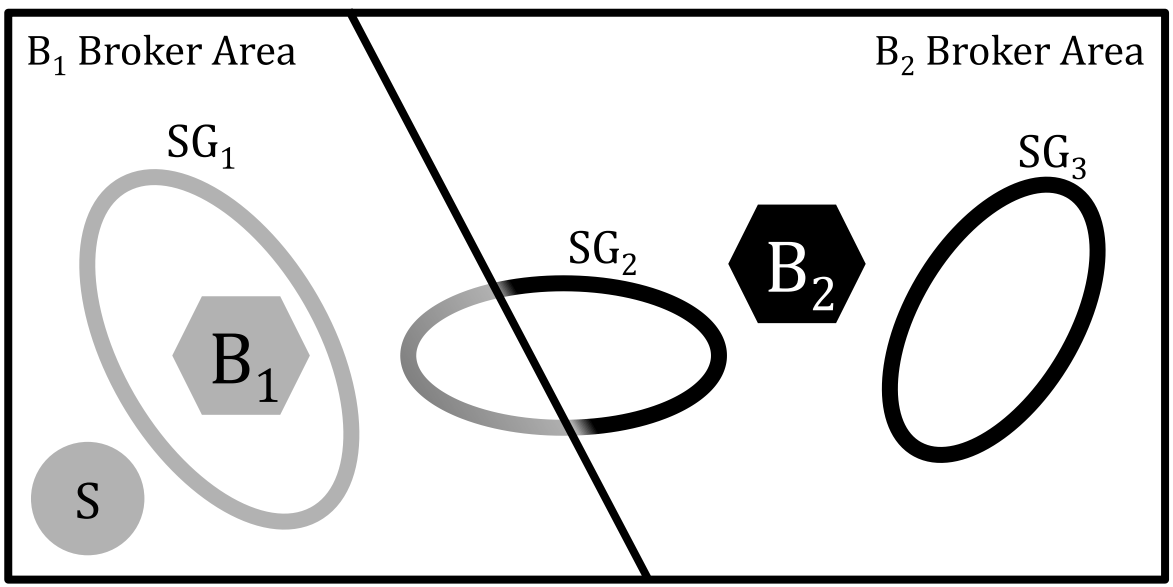
- Subscriptions are not distributed
- Similar to *flooding events*, events are distributed
- Event geofence is used to select the RPs => send only to brokers whose broker area might contain a client that is allowed to receive the event



Selecting RPs close to the Publishers

The RP for an event is the broker closest to the publisher of that event.

- Events are not distributed (for the matching)
- Similar to *flooding subscriptions,* subscriptions are distributed
- The subscription geofence is used to select the RPs => only brokers where an event within the desired geofence might originate can become RPs



Questions?