DisGB: Using Geo-Context Information for Efficient Routing in Geo-Distributed Pub/Sub Systems

Jonathan Hasenburg and David Bermbach
Mobile Cloud Computing Research Group
TU Berlin & Einstein Center Digital Future
Berlin, Germany
Email: {jh,db}@mcc.tu-berlin.de

***Abstract*—IoT data are usually exchanged via pub/sub, e.g., based on the MQTT protocol. Especially in the IoT, however, the relevance of data often depends on the geo-context, e.g., the location of data source and sink. In this paper, we propose two inter-broker routing strategies that use this characteristic for the selection of rendezvous points. We evaluate analytically and through experiments with a distributed pub/sub prototype which strategy is best suited in three IoT scenarios. Based on simulation, we compare the performance and efficiency of our approach to the state of the art: Our strategies reduce the event delivery latency by up to 22 times compared to the only alternative that sends slightly fewer messages. Our strategies also require significantly less inter-broker messages than all other approaches while achieving at least the same performance.**

***Keywords*-Geo-Distribution, Geo-Context, Pub/Sub, Rendezvous Points**

I. Introduction

The vision of the Internet of Things (IoT) is to connect billions of devices. These devices usually operate at the edge of the network; thus, they might be battery powered or have a slow and unstable connection to the wide area network. Rather than interconnecting devices directly, communication in the IoT is usually done asynchronously via broker-based pub/sub (publish/subscribe) [1]: client devices create subscriptions (subscribers) and send events (publishers) to any of the brokers which then match incoming events with created subscriptions and deliver them accordingly. Many mature solutions for inter-broker event and subscription routing exist that have already proven their effectiveness, e.g., [2]–[6].

There are special cases, however, in which additional information can be used to further optimize inter-broker routing: In IoT scenarios, the relevance of data for individual devices often depends on the geo-context of the data, e.g., the current location of the sensor. In our previous research [7], [8], we showed that this can be leveraged when matching events and subscriptions at a single broker by enriching messages with geo-context information which significantly reduces excess data transmission and enables new application scenarios.

In this paper, we propose to use geo-context information to improve inter-broker routing[1]. The goal is to match data close to either the publishers or subscribers of an event. Therefore, we make the following contributions:

- We propose two novel strategies which use geo-context information to improve inter-broker routing (Section III).
- We evaluate analytically (Section IV) and through experiments (Section V) which strategy is best suited in three IoT scenarios.
- Using simulation, we show that our proposed strategies reduce the event delivery latency by up to 22 times compared to the only state-of-the-art alternative that sends slightly fewer messages (DHT). We also show that our strategies require significantly less inter-broker messages than all other approaches while achieving at least the same performance (Section VI).

With these contributions, we show that using geo-context information can significantly improve inter-broker routing.

II. Background

Distributed pub/sub is a very mature research domain. In the following, we briefly describe the three categories in which most pub/sub routing strategies can be categorized (Section II-A). We also discuss how well these strategies are suited for geo-distributed routing. Furthermore, we summarize our previous definition of geo-contexts presented in [7], [8] (Section II-B).

A. Inter-Broker Routing Strategies

In general, existing pub/sub inter-broker routing strategies can be classified into the three categories *flooding*, *gossiping*, and *selective* [10, p. 145]. With flooding, events or subscriptions are broadcast to all other brokers. While this ensures minimum end-to-end latency, it does not scale well as every broker has to process all events or subscriptions from every other broker which also leads to a high network load. Gossiping focuses on tolerating very dynamic environments

[1]We have pre-published a technical report [9] that describes the abstract concept proposed in this paper.

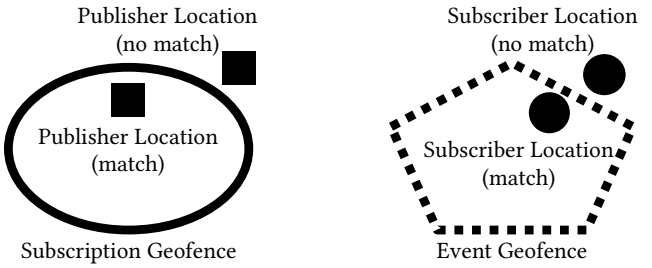

Figure 1: Subscription GeoCheck (left) and event GeoCheck (right), adapted from [7], [8].

by distributing messages between brokers randomly, it thus sacrifices latency. While IoT devices might operate in such an environment, the brokers, to which the routing approach is applied, do not. Instead, it is more likely that the brokers are deployed in a limited number of (cloud) regions with low churn rates. Selective approaches are either filter-based or build upon rendezvous points (RP). For the former, filters are distributed across brokers and form dynamic multicast trees for each event. Traversing the multicast trees, however, results in high end-to-end latency. RPs reduce communication cost by being a "meeting point" for subscriptions and events: the matching occurs at the RP brokers [10, p. 166]. Hence, they limit propagation of events and/or subscriptions to a small subset of nodes which improves system efficiency. If the RPs are close to the publishers or subscribers of an event, the end-to-end latency is comparable to the one of flooding solutions. The approach we propose in this paper builds upon RP-based routing and specifically focuses on selecting RPs that are close to either the publisher or the subscribers of an event.

B. Enriching Events and Subscriptions with Geo-Context

In [7], [8], we have proposed to use geo-context information for event matching. We update our previous terminology slightly to better fit the purpose and scope of this paper:

There are four geo-context dimensions. Clients have a geographic location, which consists of a latitude and a longitude value. To the location of publishers, we refer as **publisher location**, for subscribers as **subscriber location**. Beyond this, each event and subscription has an area it relates to; these can be described with geofences: The **event geofence** ensures that only subscribers located in the specified area receive the event, i.e., subscriber locations must be inside the event geofence. The **subscription geofence** ensures that only the events of publishers located in the specified area may be delivered to the subscriber, i.e., publisher locations must be inside the subscription geofence[2]. Geofences can be arbitrarily complex polygons.

[2]In previous work, we referred to these four dimensions as producer location, consumer location, producer geofence and consumer geofence.

For using both geofences and locations in event matching, two checks are necessary to decide whether data should be delivered to a specific subscriber (Figure 1) – first, from the subscribers's perspective with the help of the subscription geofence and publisher location (subscription GeoCheck) and, second, from the publishers's perspective with the help of the event geofence and subscriber location (event GeoCheck).

For a more detailed discussion and explanation of the geo-context model, we refer to our previous work [7]. Furthermore, it usually makes sense to combine the two GeoChecks with an additional ContentCheck, e.g., based on topics. For a more detailed discussion on how this can be used to build a (single-node) data distribution service leveraging geo-contexts, we refer to our previous work [8]. In this work, we describe how such single-node system instances ("brokers") can communicate via RPs which are selected based on geo-context information.

III. Optimizing Routing with Geo-Contexts

In Section II-A, we already mentioned that our proposed approach builds upon RP-based routing as this is a good choice when selected RPs are close to either the publishers or subscribers of an event. Current state of the art solutions, e.g., [4], [11], distribute RPs uniformly over available brokers which works well if the message traffic is also uniformly distributed. IoT data traffic, however, is often non-uniformly distributed: published events are most relevant to devices located in a particular geographical area. This relevance can be expressed with geo-contexts as discussed in Section II-B. Thus, our key idea is to use geo-contexts to identify RPs that are close to either publishers and/or subscribers of an event.

In this section, we first discuss some assumptions (Section III-A) before we describe how geo-context information can be used to select RPs close to subscribers (Section III-B) or close to publishers (Section III-C). Both strategies come with their own advantages and disadvantages. Which one is better depends on the application scenario; we discuss this in Section IV.

A. Assumptions

For our approach, we assume a setup that comprises multiple geo-distributed brokers and clients, i.e., IoT devices and services. Even though brokers are geo-distributed, they are aware of each other, typically have a good inter-connection, and are well equipped in terms of computing power. Clients, on the other hand, might operate in a constrained environment and only communicate with the physically closest broker: their *local broker* (LB). Consequently, a broker is responsible for communication with all clients located in the region surrounding its physical location as this asserts low

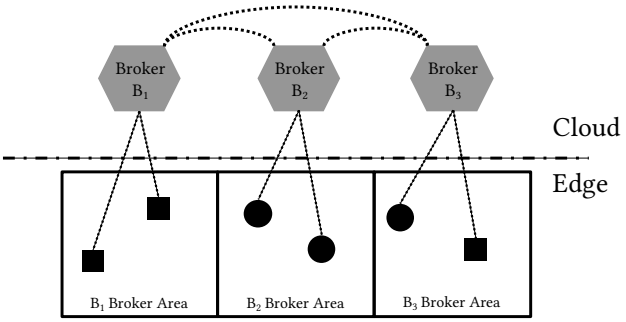

Figure 2: Setup with three brokers that support communication between publishers (squares) and subscribers (circles).

latency communication[3]. We refer to this region as *broker area*, see also Figure 2.

Subscriptions and published events comprise a payload, some kind of content filter (e.g., a topic), and geo-context information. When a client creates a subscription, it creates the subscription at its LB. Similarly, when a client publishes an event, it sends the event to its LB. Depending on the strategy (Sections III-B and III-C), as soon as the LB has received an event or subscription, it distributes them to the RP where the matching occurs.

B. Selecting RPs Close to the Subscribers

With this strategy, 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. Thus, the RPs are the LBs of these subscribers. Hence, subscriptions are not distributed to other brokers as subscribers create subscriptions at their LB. The event, on the other hand, is distributed to all brokers which might manage a matching subscription. Fortunately, the event geofence can be used to select these RPs because only broker areas intersecting with the event geofence can contain subscribers that pass the event GeoCheck (subscriber location inside event geofence).

Figure 3 shows an example with one publisher (P) that is located in the broker area of broker B_1 and publishes three events—each has a different event geofence (EG):

- EG_1 does not intersect with the broker area of broker B_2 (on the right) so the event does not need to be forwarded for matching to B_2.
- EG_2 intersects with the broker areas of B_1 and B_2 so the event needs to be matched at B_1 and must be forwarded for matching to B_2.
- EG_3 only intersects with the broker area of B_2 so the event needs to be forwarded for matching to B_2. Note, that matching at B_1 can be omitted, as no subscription created by the subscribers located in the broker area of B1 can pass the event GeoCheck.

[3]Using the network distance instead of physical distance to determine local brokers might be more accurate, but is also more complicated in an environment with changing network conditions.

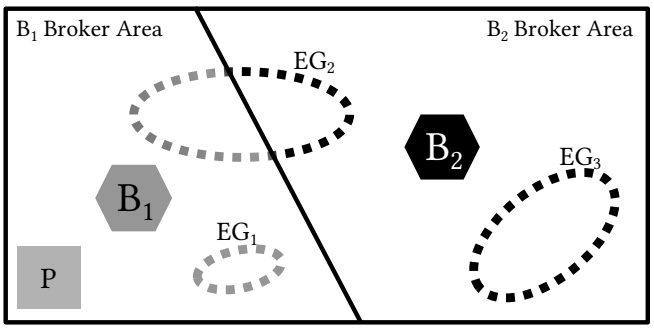

Figure 3: An event only needs to be sent to brokers with a broker area that intersects with the event geofence.

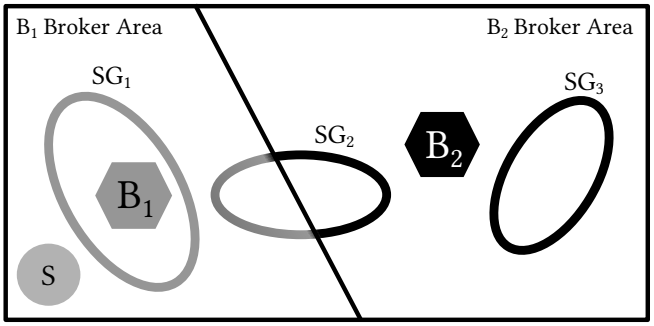

Figure 4: A subscription only needs to be sent to brokers with a broker area that intersects with the subscription geofence.

The key benefit of this strategy over state-of-the-art strategies is that by using geo-context information events only need to be sent to a small subset of brokers.

C. Selecting RPs Close to the Publishers

With this strategy, the RP for an event is the broker closest to the publisher of that event, i.e., the RP is the publisher's LB. Thus, matching only occurs at a single broker. In exchange, all subscriptions must be distributed to all brokers to which a matching event might be published; subscription updates must also be propagated in a similar fashion. Fortunately, the subscription geofence can be used to select these RPs because only broker areas intersecting with the subscription geofence might contain publishers that pass the subscription GeoCheck (publisher location inside subscription geofence).

Figure 4 shows an example with one subscriber (S) that is located in the broker area of B_1 and creates three subscriptions with different subscription geofences (SG) each:

- SG_1 and the broker area of B_2 do not intersect, so the subscription does not need to be forwarded to B_2.
- SG_2 intersects with the broker areas of B_1 and B_2 so the subscription needs to be maintained at B_1 and updates must be forwarded to B_2.

- SG_3 only intersects with the broker area of B_2 so the subscription needs to be forwarded to B_2. Note, that the subscription can be discarded at B_1, as none of the publishers managed by B_1 can publish an event that passes the subscription GeoCheck.

After matching the event, it still needs to be distributed to the LBs of subscribers with matching subscriptions as these brokers are the ones communicating with the subscribers. Still, in contrast to selecting RPs close to the publishers, events are only distributed based on actual matches rather than on potential matches. Hence, the key benefit of this strategy over state-of-the-art strategies is that by using geo-context information subscriptions only need to be forwarded to a small subset of brokers and events are only forwarded to brokers that are confirmed to be the LB of a matching subscriber.

IV. Evaluation: Scenario Analysis

Depending on the application scenario, each RP selection strategy results in a different number of RPs; the lower the number of well chosen RPs, the lower the total number of messages. In this section, we first calculate how the two selection strategies affect the number of RPs (Section IV-A). Then, based on these calculations, we discuss which RP strategy is the best choice for three example scenarios (Section IV-B).

A. Calculating the Number of RPs

There are four client actions that require RPs:

i) a subscriber updates its location,
ii) a subscriber creates/updates[4]/deletes a subscription,
iii) a publisher updates its location,
iv) and a publisher publishes an event.

For this analysis, we define the following: When an event is published, E is the set of all broker areas that intersect with the event's geofence. Every subscriber has a set of active subscriptions $\{s_i \mid i \in \{1, \ldots, n\}\}$, with n being the subscriber's total number of subscriptions. S_i is the set of all broker areas that intersect with the subscription geofence of s_i. Finally, b is the total number of brokers.

i) Subscriber Location Update: An updated subscriber location must be distributed to all brokers that require it for the matching of events. When selecting RPs close to the subscribers, events are only matched at the LB of the subscriber, so the number of RPs is 1. When selecting RPs close to the publishers, events might be matched at any broker whose broker area intersects with any of the subscribers subscription geofences, so the number of RPs is $|\bigcup_{i=1}^{n} S_i|$.

ii) Subscription Update: An update of subscription s_i must be distributed to all brokers that require it for the matching of events. When selecting RPs close to the subscribers, events are only matched at the LB of the respective subscriber, so the number of RPs is 1. When selecting RPs close to the publishers, events might be matched at any broker whose broker area intersects with the geofence of s_i, so the number of RPs is $|S_i|$.

iii) Publisher Location Update: An updated publisher location must be distributed to all brokers that require it for the matching of events. When selecting RPs close to the subscribers, events might be matched everywhere, so the number of RPs is b. However, an LB could also piggyback the current publisher location on each event of the same publisher it has to distribute (see iv) below). In this case, the number of RPs is 1 for publisher location updates as they are not distributed separately. When selecting RPs close to the publishers, events are only matched at the LB of the publisher, so the number of RPs is 1.

iv) Event Publishing: A published event must be distributed to all brokers that match the event with managed subscriptions. When selecting RPs close to the subscribers, events are matched at any broker whose broker area intersects with the geofence of the given event, so the number of RPs is $|E|$. When selecting RPs close to the publishers, events are only matched at the LB of the publisher, so the number of RPs is 1.

Table I: Number of RPs for each type of client action and RP selection strategy when geofences exist.

Client Action Type	RPs at subscriber	RPs at publisher
Subcriber Location Update	1	$\lvert\bigcup_{i=1}^{n} S_i\rvert$
Subscription Update	1	$\lvert S_i\rvert$
Publisher Location Update	1	1
Event Publishing	$\lvert E\rvert$	1

In summary (Table I), the number of RPs, and thus the overhead of inter-broker communication, depends on the scenario specific workload. Selecting RPs close to the subscribers is better for workloads that involve

- many subscriber location updates,
- many subscription updates,
- and large subscription geofences that intersect with many broker areas

as subscription information does not need to be distributed by the LB. Selecting RPs close to the publishers, on the other hand, is better for workloads that involve

- a high volume of published events
- as well as large event geofences that intersect with many broker areas

as the events can be matched at the LB of each publisher.

B. Discussion based on Example Scenarios

In [7], we presented three (IoT) scenarios that use geo-context information. In the following, we use these three scenarios to discuss which RP selection strategy is better

[4]For example, to use another subscription geofence.

suited in what situation. For more information and examples, we refer to the original paper.

1) Open Environmental Data: In this scenario, IoT sensors provide data access to all clients that subscribe to related topics such as temperature, humidity, or barometric pressure. Clients subscribe to topics based on their individual content interests. Furthermore, by using a subscription geofence, they only receive data of sensors that are located in the specified geofence. For example, a smart blinds control system located in Delft could subscribe to the temperature topic and use a subscription geofence that only contains the Netherlands.

The most important geo-context related characteristics in this scenario are:

- Event geofences do not exist, so no event GeoCheck is needed.
- Subscription geofences have arbitrary size, as subscribers can be interested in very small, but also very large regions.
- Subscriptions are updated rarely as subscribers do not have to update their subscription geofences multiple times a day.
- Events (sensor readings) are published frequently, but in many cases no matching subscriber exists.

As event geofences do not exist, $|E|$ equals the number of all available brokers. Subscription geofences can have arbitrary sizes, so $|S_i|$ is somewhere between 1 and the number of all available brokers. This is an unfavorable combination as subscription updates as well as published events require inter-broker communication. The subscription update frequency, however, is lower than the event publishing frequency. Thus, selecting RPs close to the publishers is the better strategy. Another benefit of this strategy is that events are not distributed to brokers without a matching subscriber.

2) Local Messaging and Information Sharing (Hiking): In this scenario, clients consume and share data of other clients in proximity while being mobile. For this, each client creates subscriptions to topics of interest and a subscription geofence that covers the immediate surrounding area. Furthermore, clients also publish events to fitting topics with an event geofence that covers the immediate surrounding area.

The most important geo-context related characteristics in this scenario are:

- Geofences are small and unlikely to intersect with multiple broker areas.
- Publisher and subscriber locations are updated frequently as both are mobile.
- Subscriptions are updated frequently as subscription geofences depend on the respective subscriber location.
- Events are published frequently.

As both geofences only intersect with a very small number of broker areas (often only with a single one), $|E|$ and $|S_i|$ usually equal 1. In addition, it is very likely that the publisher and all matching subscribers for a given event are connected to the same LB, so only a small amount of data has to be distributed to other brokers. Therefore, none of the two RP selection strategies has a clear advantage over the other one.

3) Context-based Data Distribution: In this scenario, events are delivered based on the content interests of subscribers and the domain knowledge of publishers. This way, all subscribers must only specify once what kind of data they want to receive while publishers define in which geographic area their events are relevant. For example, citizens could subscribe to events that carry emergency alerts while public authorities can accurately define in which area their alerts should be received. Then, without having to update their subscriptions again, citizens could travel between districts or cities and still get all relevant alerts.

The most important geo-context related characteristics in this scenario are:

- Event geofences have arbitrary size but are considered to be relatively small and intersect with only a few broker areas.
- Subscription geofences do not exist, so no subscription GeoCheck is needed.
- Subscriber locations are updated frequently as subscribers are mobile.
- Subscriptions are updated rarely as subscribers have to subscribe to their desired topics only once.
- Events can be published at any frequency as this depends on the kind of data (e.g., advertisements vs. emergency alerts).

As event geofences are considered to be small, $|E|$ usually equals 1. Subscription geofences do not exist, so $|S_i|$ equals the number of all available brokers. Subscriber location updates must be forwarded to $|\bigcup_{i=1}^{n} S_i|$ brokers, so due to the high update frequency, selecting RPs close to the publishers is an unfavorable approach for this scenario. Instead, selecting RPs close to the subscribers is the better approach as publisher locations do not need to be forwarded (no subscription GeoCheck) and $|E|$ is considerable smaller than $|\bigcup_{i=1}^{n} S_i|$ and $|S_i|$.

V. Evaluation: Experiments

The goal of our experiments is twofold. First, we want to show that attaching the geo-context to events and subscriptions for the selection of RPs is feasible in practice. Second, we want to experimentally validate our scenario discussions from the previous section. Note, that comparing our RP selection approach to the state of the art is not the goal of this section. We do this in depth in Section VI through simulation, because experiments with multi-threaded distributed systems can never be fully deterministic. Furthermore, we already showed in [8] that the overhead of processing geo-contexts is negligible.

Table II: The characteristics of each workload depend on the underlying scenario.

Number of ...	Open Env.	Hiking	Ctx.-based
Location updates	7,200	193,709	69,236
Subscription updates	75,168	128,008	5,544
Subscription geofence overlaps	1,225	502	–
Event publishings	216,029	111,811	62,675
Event geofence overlaps	–	623	222

A. Prototype

We have extended the GeoBroker prototype from [8] with the capability to use RPs for inter-broker communication. The source code of this distributed GeoBroker (DisGB) is available as open source on GitHub[5]. When starting DisGB, we can select one of three modes: single, RP_subscriber, and RP_publisher. At single, DisGB behaves similar to GeoBroker, i.e., there is only one broker node to which all clients connect and thus no inter-broker communication. In the two other modes, DisGB supports a distributed broker deployment in which RPs are either selected close to the subscribers (RP_subscriber) or the publishers (RP_publisher). For the current prototype, broker addresses and non-overlapping broker areas must be provided as configuration data; more advanced solutions based on, e.g., dynamic discovery, could be added if needed.

To assert the correctness of our implementation and setup, we ran a very simple workload prior to our scenario experiments with DisGB configured in single mode, RP_subscriber mode, and RP_publisher mode. We then verified that in each run all messages had been sent and that each client had received the same set of messages. We also verified that event matching is done correctly through another small experiment. Here, we determined for each published event the correct subscribers manually to then verify that the event matching carried out by our prototype delivers the same result. It should be noted that geofences can be arbitrarily complex polygons which can, for example, specified in the `Well-Known Text` [12] format.

B. Experiments

For the experiments, we setup three brokers in different AWS regions. Two brokers are near to each other (Paris and Frankfurt), and one is located further away (Ohio). In each region, 1200 clients running on separate machines connect to their local broker and continuously send messages that resemble the scenario workloads from Section IV-B.

Each scenario workload comprises a set of client actions such as location updates, subscription updates, and event publishings. See Table II which shows a high level overview of the individual workload characteristics resulting from running each workload for 15 minutes. Geofence overlaps describe how many geofences overlap with more than one broker area; when no geofence exists ("–" in table), an event or subscription must be distributed to all brokers.

[5]https://github.com/MoeweX/geobroker

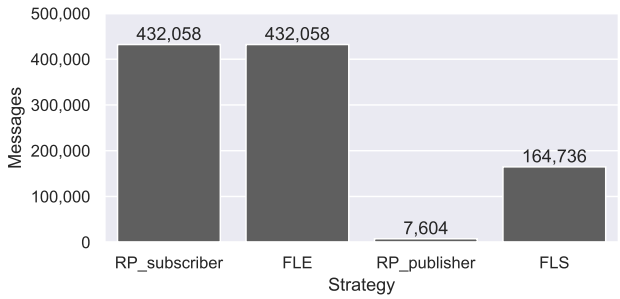

Figure 5: Open environmental data: selecting RPs close to the publisher is the most efficient strategy.

C. Results

A key characteristic when comparing pub/sub routing strategies is the amount of inter-broker messages, i.e., location updates, subscription updates, and published events, which are distributed by an LB of a client to RPs. Fewer inter-broker messages indicate a more efficient routing strategy. Figures 5, 6, and 7 show the number of inter-broker messages for each strategy and scenario. The DisGB values are derived directly from the experiments. As a baseline, we could not use DisGB in single mode, as then all messages are processed by a centralized broker; here, no routing is necessary. Instead, we calculated how flooding events (FLE) or flooding subscriptions (FLS) would perform [10, p. 151]. We chose these strategies as a baseline, as flooding events is very similar to selecting RPs close to the subscriber, and flooding subscriptions is very similar to selecting RPs close to the publishers. Both also result in the lowest possible latency as events/subscriptions are sent directly to all brokers that might need them for matching. The difference is that our strategies use the event or subscription geofence to limit the number of RPs which reduces the number of inter-broker messages. We present a more extensive comparison with approaches from related work in Section VI.

When selecting RPs close to the subscribers, events must be distributed to $|E|$ brokers, i.e., all brokers whose broker area intersects with the event's geofence. Thus, the numbers are as expected from our scenario analysis (see Section IV-A) because the number of inter-broker messages equals the number of event geofence overlaps (there are no overlaps between Columbus + Paris or Columbus + Frankfurt).

When flooding events, every published event must be distributed to all other brokers. For example, this leads to 223,622 (=2∗111,811) inter-broker messages for the Hiking scenario. Note, that the real inter-broker messages from the DisGB experiment match the calculated FLE messages in

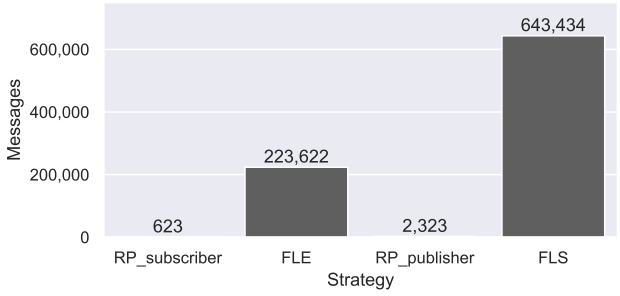

Figure 6: Hiking: both RP selection strategies are more efficient than the flooding strategies.

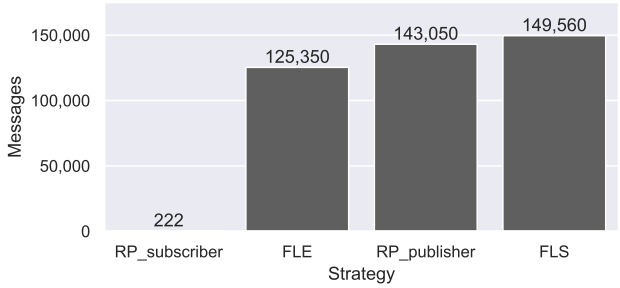

Figure 7: Context-based data distribution: selecting RPs close to the subscribers is the most efficient strategy.

the Open Environmental Data scenario. This makes sense, as there are no event geofences and DisGB can, thus, not limit the number of target brokers.

When selecting RPs close to the publishers, subscriber location updates need to be distributed to $|\bigcup_{i=1}^{n} S_i|$ brokers; it is not trivial to calculate this number. The only exception is the context-based data distribution scenario, for which no subscription geofence exists. Here, every subscription update and location update is sent to the two other brokers: 149,560 (=2∗69,236 + 2∗5,544). However, DisGB does not distribute the location updates of a subscriber if the subscriber has not created a subscription yet; thus, the actual number is lower (143,050). Furthermore, events that have been successfully matched against a subscription created by a client connected to another broker must be sent to the LB of this subscriber as well.

When flooding subscriptions, every subscription or subscriber location update must be distributed to all other brokers. In the Hiking scenario, this leads to 643,434 (=2∗193,709 + 2∗128,008) inter-broker messages. In addition, successfully matched events of subscribers connected to another broker must also be distributed. Successful matches can only be determined through simulation or an experiment, so the calculated number of inter-broker messages reported in the figures is slightly lower than FLS would show in practice.

D. Summary

With our experiments, we demonstrate that using geo-context information to select RPs is feasible in practice. We also validated our scenario analysis from Section IV-B: For scenarios without or with very large event geofences (Open Environmental Data scenario), selecting RPs close to the publisher is the better strategy, as this reduced inter-broker messages by 98.2% (7,604 inter-broker messages compared to 432,058). For scenarios with similar event and subscription geofences (Hiking scenario), none of the two strategies has a clear advantage over the other one (2,323 inter-broker messages compared to 623), while both performed significantly better than their respective baselines. For scenarios without or with very large subscription geofences (Context-based Data Distribution scenario), selecting RPs close to the subscribers is the better strategy, as this reduced inter-broker messages by 99.8% (222 inter-broker messages compared to 143,050).

VI. Evaluation: Simulation

In this section, we describe the results of comparing our proposed RP selection strategies to related work. For this, we use simulation (Section VI-A) with a newly developed simulation tool (Section VI-B) that allows us to compare different routing strategies (Section VI-C) in a fully deterministic way. Our results (Section VI-D) indicate that our strategies reduce the event delivery latency by up to 22 times while still requiring less inter-broker messages than most other strategies.

A. Simulation Design

Our simulation model follows a realistic setup: brokers are distributed across the globe and clients connect to their respectively nearest broker. For each simulation run, based on a given total number of brokers and clients, we determine broker and client locations as follows:

1) Place brokers at the locations of randomly chosen cities with a population of more than 1 million citizens based on the world cities dataset[6].
2) Assign clients in proportion to the population of a broker's city and the total population of all chosen cities to each broker.
3) Generate random locations for each client of a broker in the area that is closest to this broker[7]; this area is also the broker area needed for the DisGB strategies.

We developed a tool that visualizes simulation setups, a live demo for our setup is available on GitHub[8]. In total, we

[6]https://simplemaps.com/data/world-cities

[7]We determine this area by dividing the earth surface in non-overlapping squares and then assigning each square to the broker whose location is closest to the square's center. Other methods such as Voronoi diagrams could also be used.

[8]https://moewex.github.io/DisGB-Simulation/

compare seven RP selection strategies (Section VI-C). Our simulation workload is based on the Hiking scenario from Section V as for this neither of our two strategies has a clear advantage. While the distribution of subscriptions and events between brokers is different for each strategy, the result from the client perspective does not change: each client receives the same events based on its individual subscriptions regardless of the RP selection strategy used. For this, all brokers have to do a ContentCheck as well as the two GeoChecks when matching messages (see Section II-B). Otherwise, for example, the other strategies would also deliver events to subscribers that are not present in the event's geofence. While each RP selection strategy has its own way of distributing events, subscriptions, or both to the corresponding RPs, not all have the capability to distribute client locations appropriately. Still, client locations are needed for the two GeoChecks. Enhancing the strategies to distribute location updates is out of scope for this paper. Thus, for all simulation runs, we assume that each broker floods client location updates to all other brokers, even though our proposed RP selection strategies are more efficient than this. Due to this decision, we disregard effects of location updates for the analysis of simulation results (Section VI-D).

We decided to have 1000 distinct topics, each client subscribed and published to five topics. Furthermore, we used circular geofences (arbitrary shapes are possible in practice); all event geofences had a radius of 5°(about 555km at the equator) and all subscription geofences had a radius of 10°(about 1110km at the equator). We decided to use such large values for the geofences to increase the amount of inter-broker traffic; smaller geofences particularly benefit DisGB. For each RP selection strategy, we ran one simulation round with 100,000 clients and 25 brokers[9], one with 1,000,000 clients and 25 brokers, and one with 100,000 clients and 256 brokers[10].

With the simulation, we can discuss how each RP selection strategy affects three aspects: First, the number of inter-broker event and subscribe messages; fewer messages means that the strategy is more efficient. Second, the event delivery latency: how much time does it take for a client to receive an event after it has been published by any other client connected to any of the brokers. Third, the subscription update delay: how much time does it take for a broker to receive a subscription update distributed by another broker. In general, having a lower event delivery latency and subscription update delay means that a strategy offers a superior performance compared to a strategy with higher values.

[9]We chose this number based on the active AWS Cloud regions (24).

[10]The GQPS strategy (Section VI-C) works best if the square root of the number of brokers is an integer.

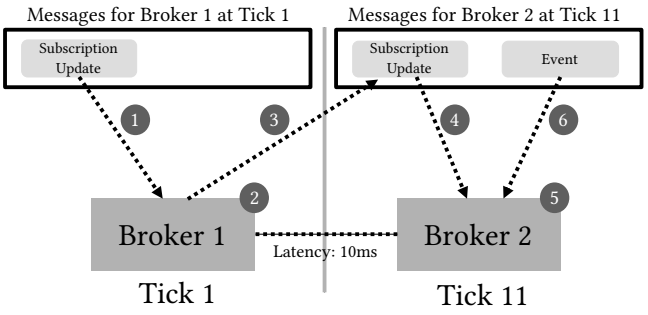

Figure 8: Broker 1 receives a subscription update at Tick 1 (1). Besides updating its state (2), it must distribute the update to Broker 2 as specified by its RP selection strategy (3). As the latency is 10ms, Broker 2 receives the subscription update at Tick 11 (4). It does not need to be further distributed so it is only used to update the broker's state (5). After all brokers have finished processing subscription updates, Broker 2 can continue to process the next type of message (6).

B. Simulation Tool

The simulation tool can be customized with a number of parameters. The six most important parameters are: the chosen RP selection strategy, the number of brokers, clients, and distinct topics, the size of event geofences, and the size of subscription geofences. The simulation tool is fully deterministic, i.e., repeating a simulation run with the same parameters (and random seed) leads to the same result. Our tool implements a parallel discrete-event simulation [13]. The simulation time granularity is 1ms, we refer to one time unit as a *tick*.

Upon startup, our simulation tool generates a workload based on its input parameters. Every workload comprises a set of brokers that use the specified RP selection strategy, and a set of client messages that should be received by an LB at a certain tick. After the initialization, the simulation tool sequentially processes ticks and the corresponding messages. During each tick, the tick's messages are delivered in a predefined order to brokers which use them to update their state, e.g., managed client locations or subscriptions, accordingly. Besides state updates, brokers might also further distribute messages to other brokers or deliver events to clients. For this, they determine when a message would be received by their target and then add this message to the event queue for the corresponding tick (see Figure 8). The latency between two brokers is calculated by multiplying their physical distance with 0.021 ms/km. We determined this constant by analyzing the 2016 IPlane traceroute dataset[11]. For communication between a client and its LB, we use a fixed latency of 5ms so that we can determine which share

[11]https://web.eecs.umich.edu/~harshavm/iplane/

of the overall latency is caused by inter-broker traffic.

Especially for larger client and broker numbers, it is not feasible to store the full set of sent and received messages for analysis. Thus, we calculated all result values presented in Section VI-D by analyzing the stream of data. We used the P2 algorithm by Jain and Chlamtac [14] to compute percentiles heuristically. The mean squared error of the P2 algorithm "is comparable to that obtained by order statistics and [...] both tend to zero as sample size is increased" [14]. The simulation tool is implemented in Kotlin and available as open source on GitHub[12].

C. RP Selection Strategies

In our simulation tool, we have implemented our two RP selection strategies (Section III) and five additional strategies from related work. In the following, we briefly describe at which brokers the matching occurs for these five additional strategies (i.e., how the RPs are determined). We chose these strategies because they are either well known or because our two strategies do not obviously offer a superior performance when geo-context information is available.

Flooding Events (Flood_E): Every broker is an RP for every event. When an LB of a publisher receives an event, it distributes it to all other brokers. After matching this event, brokers can deliver the event to their local subscribers directly.

Flooding Subscriptions (Flood_S): The LB of a publisher is the only RP for a given event. When an LB of a subscriber receives a subscription update, it distributes it to all other brokers. After matching an incoming event, the broker might have to notify remote brokers about successful matches so that they can deliver the event to their local subscribers. This is necessary, as each client only communicates with its LB.

Consistent Hashing (DHT): This strategy builds on distributed hash tables and is used in pub/sub systems such as Scribe [4] or Hermes [11]. The RP is determined by mapping the event and subscription topics to a particular broker with consistent hashing [15]. Once an RP has matched the event, it notifies the LBs of matching subscribers about successful matches so that they can deliver the event to their local subscribers.

Grid Quorum (GQPS): To determine RPs, an application-level overlay network is created that makes each broker addressable by a position in a grid, i.e., by its row and column. RPs are all brokers in the same row or column as the LB, so this is where events and subscriptions must be sent to [16]. After matching an event, the RP notifies the LBs of matching subscribers that have not been in the same row/column as the LB of the publisher about successful matches.

Broadcast Groups (BG): Physically close brokers organize themselves in broadcast groups in which events are flooded to other group members for matching, i.e., all group members of a publisher's LB are an RP. Furthermore, one broker of each broadcast group (the leader) aggregates and forwards events and subscriptions originating in its group to a centralized cloud broker[13]. The cloud broker matches events with subscriptions created by other leaders, therefore, it is also an RP. If an event is matched successfully at the cloud broker, every member of the corresponding broadcast group also becomes an RP for this event to match and deliver it [17].

In the following, the abbreviation for selecting RPs close to the subscribers is DisGB_E, as with this strategy events are distributed. Similarly, the abbreviation for selecting RPs close to the publishers is DisGB_S, as with this strategy subscriptions are distributed.

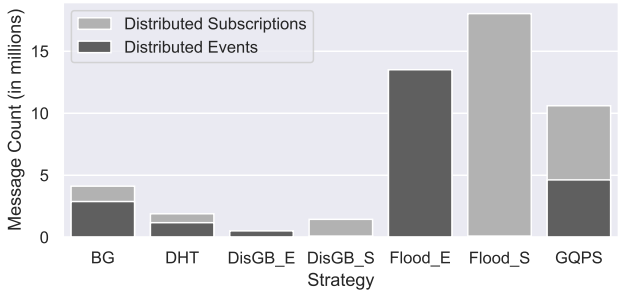

Figure 9: For 100,000 clients and 25 brokers, both DisGB strategies require less inter-broker messages than all other strategies.

D. Simulation Results

In total, we ran 27 simulation rounds that each comprised a fifteen minute workload in simulation time[14].

Figure 9 shows the number of distributed inter-broker event or subscription messages for 100,000 clients and 25 brokers. The first notable observation is that **both DisGB strategies require less inter-broker messages than all other strategies**. When observing the two flooding strategies, one can see that the number of distributed subscriptions is higher than the number of distributed events. Therefore, clients update subscriptions more often than they publish events in the workload used. This, and the fact that the subscription geofence radius is twice the radius of event geofences, also explains why the DisGB_E strategy requires less inter-broker messages than DisGB_S. Using 10 times more clients (1,000,000) only has the effect that each strategy requires about 10 times more inter-broker messages. Using approximately 10 times more brokers (256), however, does reduce the amount of inter-broker messages for most

[12]https://github.com/MoeweX/DisGB-Simulation

[13]AWS operates its biggest data center in West Virginia, USA, so that is where we put the cloud broker for the simulation.

[14]The execution time was up to 16 hours for a single simulation run.

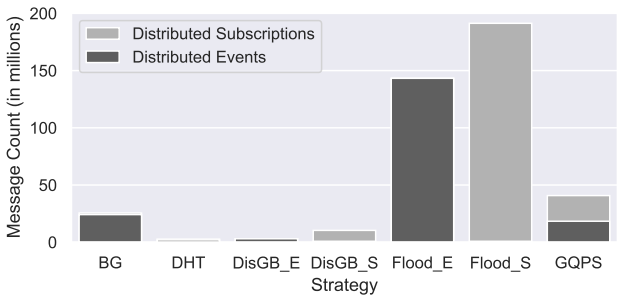

Figure 10: For 100,000 clients and 256 brokers, DHT requires the fewest inter-broker messages.

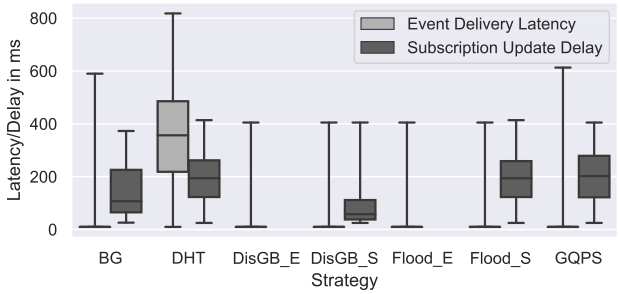

Figure 11: For 100,000 clients and 25 brokers, both DisGB strategies offer a similar performance as their respective flooding counterparts.

strategies compared to the two flooding strategies (see Figure 10). Furthermore, the DHT strategy now requires the lowest amount of messages because exactly one broker is responsible for the matching of events and subscriptions of a given topic.

Figure 11 shows the event delivery latency and the subscription update delay for 100,000 clients and 25 brokers. As the subscription update delay describes how much time it takes for a broker to receive a subscription update distributed by another broker, Flood_E and DisGB_E do not have such a delay. With all other strategies, a subscriber might still receive events to which it has already unsubscribed at its LB. The minimum event delivery latency is 10ms; this latency can only be achieved if the publisher and subscriber are connected to the same broker. This is the case for more than 75% of the delivered events, so the first three quartiles of the event delivery latency for BG, DisGB_E, DisGB_S, Flood_E, Flood_S, and GQPS are at 10ms. Still, comparing the mean and standard deviation of these strategies (Table III) reveals that **both DisGB strategies offer the same event delivery latency as their respective flooding counterparts**. Furthermore, the event delivery latency of both DisGB strategies is up to 22x lower than the one achievable by strategies found in related work. Using 10 times more clients (1,000,000) does not significantly influence the event delivery latency or subscription update delay. Using 10 times more brokers (256), however, does influence both values (see Figure 12 and Table III) as subscribers are more often connected to a different broker than the publisher of a matching event (the third quartile is not at 10ms anymore). Still, both DisGB strategies continue to offer a similar performance to their respective flooding counterparts.

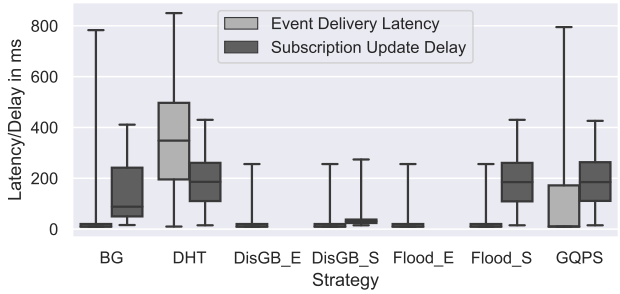

Figure 12: For 100,000 clients and 256 brokers, latency and delay increase slightly compared to Figure 11 as having more brokers increases the likelihood of inter-broker communication.

Table III: Average event delivery latency and standard deviation (in brackets) in ms for different broker (B.) and client (C.) numbers.

Strategy	25 B., 100k C.	25 B., 1M C.	256B. 100k C.
BG	22 (63)	27 (75)	24 (69)
DHT	353 (188)	348 (184)	355 (195)
DisGB_E	16 (26)	19 (31)	16 (8)
DisGB_S	16 (26)	19 (31)	16 (8)
Flood_E	16 (26)	19 (31)	16 (8)
Flood_S	16 (26)	19 (31)	16 (8)
GQPS	31 (85)	39 (98)	105 (145)

VII. Related Work

Our two RP selection strategies use the geo-context information that we proposed in our previous work [8]. There, we have already extensively discussed related work on geo-contexts and (centralized) location-based service such as [18]–[21]. Thus, we refrain from repeating this discussion and focus on distributed pub/sub systems. Furthermore, we have already discussed [4], [11], [16], [17] in Section VI.

Efficient routing of events and subscriptions in geo-distributed pub/sub systems is a mature research domain and many surveys that summarize the state of the art exist, e.g., [1], [22]. There are, however, only a few publications that propose to use geo-context information to improve inter-broker routing.

Frey and Roman [23] propose an event routing strategy that also only selects RPs within a defined *context of relevance* (comparable to our event geofence). They do not,

however, distinguish between geofences and client locations; therefore, their approach does not work well when clients are mobile as they would have to continuously update their context of relevance (and thus all their subscriptions). Furthermore, they only support the distribution of events while we can also distribute subscriptions. Kawaguchi and Bandai [24] propose a distributed broker system in which each event is only routed to a single broker that is identified by a custom topic structure which embeds geographic information. This approach does not work well when some clients are located far away from the broker responsible for one of their topics due to high round-trip latency. Cugola and Munoz de Cote [25] use geo-context information, similar to the one of our two strategies, for the routing of events or subscriptions. Their approach, however, builds upon selective filtering rather than rendezvous points; so events must traverse a multi-cast tree which increases end-to-end latency. Furthermore, it is also not possible to directly connect all brokers because each broker needs to know the current location of all clients connected to neighboring brokers. This prevents scaling to large amounts of mobile clients in fully meshed environments. Chapuis et al. [26], [27] propose a distributed pub/sub architecture that supports matching based on a geo-context subset. With their architecture, they specifically aim to enable horizontal scalability of clustered machines and it cannot be used for geo-distributed deployments.

There are also geo-distributed approaches that build on RPs but do not consider geo-context information, e.g., [28]–[31]. Thus, such approaches do not consider that IoT data traffic is often non-uniform when selecting RPs which leads to worse results. This is also the problem with other geo-distributed pub/sub system approaches that do not consider geo-context information, e.g., [2], [3], [5], [6], [32]–[35]. These systems are highly optimized for a variety of execution environments and purposes; including P2P environments [34] and distributing vast data volumes at Facebook [35]. Still, they do not use geo-context information and can therefore not ensure that data is only distributed to where it is relevant.

VIII. Discussion

The results from the experiments and the simulation show that the two DisGB strategies provide an event delivery latency comparable to flooding. Furthermore, the simulation confirmed that DisGB requires less inter-broker messages than strategies from related work. This, however, is only the case if geo-context information is available. Without such information, the DisGB strategies require as many inter-broker messages as the flooding strategies.

Therefore, when building a communication middleware for the IoT, we recommend to use a hybrid approach: If geo-context information is available, RPs should be selected using either of the two DisGB strategies, in particular depending on whether clients update subscriptions or publish events more often. If not, we recommend to use either broadcast groups or DHTs – depending on whether the focus is on event delivery latency or the total number of messages.

IX. Conclusion

In this paper, we proposed two inter-broker routing strategies that use geo-context information for the selection of rendezvous points. We evaluated analytically and through experiments with a distributed pub/sub system prototype which strategy is best suited for three kinds of IoT scenarios. Based on simulation, we compared the performance and efficiency of our approach to the state of the art: our strategies reduced the event delivery latency by up to 22 times while still requiring less inter-broker messages than most other strategies.

References

[1] P. Bellavista, A. Corradi, and A. Reale, "Quality of service in wide scale publish-subscribe systems," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1591–1616, 2014.

[2] G. Cugola, E. Di Nitto, and A. Fuggetta, "The JEDI event-based infrastructure and its application to the development of the OPSS WFMS," *IEEE Transactions on Software Engineering*, vol. 27, no. 9, pp. 827–850, 2001.

[3] L. Fiege, F. C. Gartner, O. Kasten, and A. Zeidler, "Supporting mobility in content-based publish/subscribe middleware," in *Middleware 2003*. Springer, 2003.

[4] A. Rowstron, A.-M. Kermarrec, M. Castro, and P. Druschel, "Scribe: The design of a large-scale event notification infrastructure," in *Networked Group Communication*, J. Crowcroft and M. Hofmann, Eds. Springer, 2001, vol. 2233, pp. 30–43.

[5] A. Carzaniga, "Design and evaluation of a wide-area event notification service," *ACM Transactions on Computer Systems*, vol. 19, no. 3, pp. 332–383, 2001.

[6] J. Gascon-Samson, J. Kienzle, and B. Kemme, "MultiPub: Latency and cost-aware global-scale cloud publish/subscribe," in *IEEE 37th Int. Conf. on Distributed Computing Systems*. IEEE, 2017.

[7] J. Hasenburg and D. Bermbach, "Towards geo-context aware IoT data distribution," in *Service-Oriented Computing – ICSOC 2019 Workshops*. Springer, 2019.

[8] ——, "GeoBroker: Leveraging geo-contexts for IoT data distribution," *Computer Communications*, vol. 151, pp. 473–484, 2020.

[9] ——, "Using geo-context information for efficient rendezvous-based routing in publish/subscribe systems," in *KuVS-Fachgespräch Fog Computing 2020*. TU Wien, 2020.

[10] Sasu Tarkoma, *Publish/Subscribe Systems - Design and Principles*, ser. Wiley Series in Communications Networking & Distributed Systems. Wiley, 2012.

[11] P. Pietzuch and J. Bacon, "Hermes: a distributed event-based middleware architecture," in *22nd Int. Conf. on Distributed Computing Systems Workshops*. IEEE, 2002.

[12] R. Lott, "Geographic information-well-known text representation of coordinate reference systems." http://docs.opengeospatial.org/is/12-063r5/12-063r5.html, accessed 15/11/2019, 2015.

[13] J. Liu, "Parallel discrete-event simulation," *Wiley Encyclopedia of Operations Research and Management Science*, 2010.

[14] R. Jain and I. Chlamtac, "The p2 algorithm for dynamic calculation of quantiles and histograms without storing observations," *Communications of the ACM*, vol. 28, no. 10, pp. 1076–1085, 1985.

[15] D. Kargerl, E. Lehmanl, T. Leightonl', R. Panigrahyl, and D. Lewinl, "Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web," in *29th ACM Symposium on Theory of Computing*. ACM, 1997.

[16] Y. Sun, X. Qiao, B. Cheng, and J. Chen, "A low-delay, lightweight publish/subscribe architecture for delay-sensitive IOT services," in *2013 IEEE 20th Int. Conf. on Web Services*. IEEE, 2013.

[17] J. Hasenburg, F. Stanek, F. Tschorsch, and D. Bermbach, "Managing latency and excess data dissemination in fog-based publish/subscribe systems," in *2020 IEEE Int. Conf. on Fog Computing*. IEEE, 2020.

[18] X. Chen, Y. Chen, and F. Rao, "An efficient spatial publish/subscribe system for intelligent location-based services," in *2nd Int. Workshop on Distributed Event-based Systems*. ACM, 2003.

[19] G. Li, Y. Wang, T. Wang, and J. Feng, "Location-aware publish/subscribe," in *19th ACM SIGKDD Int. Conf. on Knowledge discovery and data mining*. ACM, 2013.

[20] L. Guo, D. Zhang, G. Li, K.-L. Tan, and Z. Bao, "Location-aware pub/sub system: When continuous moving queries meet dynamic event streams," in *ACM SIGMOD Int. Conf. on Management of Data*. ACM, 2015.

[21] S. Herle and J. Blankenbach, "Enhancing the OGC WPS interface with GeoPipes support for real-time geoprocessing," *Int. Journal of Digital Earth*, vol. 11, no. 1, pp. 48–63, 2018.

[22] R. Baldoni, L. Querzoni, and A. Virgillito, "Distributed event routing in publish/subscribe communication systems: a survey," in *MiNEMA State-of-the-Art*, H. Miranda, L. Rodriguez, and B. Garbinato, Eds. Springer, 2005.

[23] D. Frey and G.-C. Roman, "Context-aware publish subscribe in mobile ad hoc networks," in *Int. Conf. on Coordination Languages and Models*. Springer, 2007.

[24] R. Kawaguchi and M. Bandai, "A distributed MQTT broker system for location-based IoT applications," in *2019 IEEE Int. Conf. on Consumer Electronics*. IEEE, 2019.

[25] G. Cugola and J. Munoz de Cote, "On introducing location awareness in publish-subscribe middleware," in *25th IEEE Int. Conf. on Distributed Computing Systems Workshops*. IEEE, 2005.

[26] B. Chapuis and B. Garbinato, "Scaling and load testing location-based publish and subscribe," in *IEEE 37th Int. Conf. on Distributed Computing Systems*. IEEE, 2017.

[27] B. Chapuis, B. Garbinato, and L. Mourot, "A horizontally scalable and reliable architecture for location-based publish-subscribe," in *IEEE 36th Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2017.

[28] H. Nguyen, M. Y. S. Uddin, and N. Venkatasubramanian, "Multistage adaptive load balancing for big active data publish subscribe systems," in *13th ACM Int. Conf. on Distributed and Event-based Systems*. ACM, 2019.

[29] A. Gupta, O. D. Sahin, D. Agrawal, and A. El Abbadi, "Meghdoot: Content-based publish/subscribe over p2p networks," in *Middleware 2004*. Springer, 2004.

[30] R. Banno, S. Takeuchi, M. Takemoto, T. Kawano, T. Kambayashi, and M. Matsuo, "Designing overlay networks for handling exhaust data in a distributed topic-based pub/sub architecture," *Journal of Information Processing*, vol. 23, no. 2, pp. 105–116, 2015.

[31] Y. Teranishi, R. Banno, and T. Akiyama, "Scalable and locality-aware distributed topic-based pub/sub messaging for IoT," in *2015 IEEE Global Communications Conf.* IEEE, 2015.

[32] G. Cugola, E. Di Nitto, and A. Fuggetta, "Exploiting an event-based infrastructure to develop complex distributed systems," in *20th Int. Conf. on Software Engineering*. IEEE, 1998.

[33] A. Zeidler and L. Fiege, "Mobility support with REBECA," in *23rd Int. Conf. on Distributed Computing Systems Workshops*. IEEE, 2003.

[34] V. Setty, M. van Steen, R. Vitenberg, and S. Voulgaris, "PolderCast: Fast, robust, and scalable architecture for p2p topic-based pub/sub," in *Middleware 2012*. Springer, 2012.

[35] Y. Sharma, P. Ajoux, P. Ang, D. Callies, A. Choudhary, L. Demailly, T. Fersch, L. A. Guz, A. Kotulski, S. Kulkarni, S. Kumar, H. Li, J. Li, E. Makeev, K. Prakasam, R. van Renesse, S. Roy, P. Seth, Y. J. Song, K. Veeraraghavan, B. Wester, and P. Xie, "Wormhole: Reliable pub-sub to support geo-replicated internet services," in *12th USENIX Symposium on Networked Systems Design and Implementation*. USENIX, 2015.