Using geo-context information for efficient rendezvous-based routing in publish/subscribe systems

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

***Abstract*—A promising communication paradigm that enables communication between IoT devices is broker-based publish/subscribe. When the brokers are distributed across the fog, events and subscriptions of clients connected to different broker instances must be routed across the router network. State-of-the art solutions, however, do not take into account that the relevance of the IoT data depends on its origin and purpose. Instead, they assume a uniform data distribution when determining where to match events and subscriptions which degrades system performance.**

In this paper, we propose a routing solution that builds upon geo-context information attached to published events and subscriptions. This way, we can match events close to either the publishers or subscribers of an event, thus, minimizing communication latency while not affecting scalability.

***Index Terms*—Geo-Context, IoT Data, Geo-Distributed Pub/Sub**

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. Hence, rather than interconnecting devices directly, communication is usually handled by some kind of distributed middleware operating in the fog. A promising communication paradigm for this purpose is broker-based publish/subscribe (pub/sub) because it allows client devices to asynchronously communicate without having to know each other [1]: they create subscriptions (subscribers) and send events (publishers) to any of the brokers, brokers match incoming events with created subscriptions and deliver them accordingly.

Because a large chunk of IoT data is only relevant to a relatively small amount of subscribers [2], which are often operating in physical proximity to each other, communication can often be handled by a local fog broker in the same region. Other use cases, however, require inter-region communication, so brokers must be connected to exchange events and/or subscriptions they received from client devices. How this can be achieved for IoT data continues to be an open research question as existing approaches do not take into account that the relevance of data depends on the data origin (i.e., the current location of the sensor) and purpose (e.g., collecting temperature values to control heaters in proximity or to collecting weather information for nation wide forecasts).

In general, existing pub/sub routing strategies can be classified into the three categories *flooding*, *gossipping*, and *selective* [3], [4]. Flooding does not scale as here every broker has to process all events or subscriptions from every other broker. Gossiping sacrifices latency in favor of tolerance of very dynamic environments by distributing messages between brokers randomly. While the 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) fog regions which do not face a high churn rate. Selective approaches are either filter-based or build upon rendezvous points (RP). For the former, filters are distributed across brokers and used to build dynamic multicast trees for each event. Traversing the multicast trees, however, increases end-to-end latency. RPs are are effective in reducing excess data by being a “meeting point” for subscriptions and events for the matching to occur; however, state of the art solutions expect a uniform distribution of data, i.e., they do not take into account that IoT data is often only relevant in a very specific area.

In our previous research [5], [6], we showed that enriching IoT events and subscriptions with geo-context information can reduce excess data and enable new application scenarios. In this paper, we propose to make additional use of this geo-context information to select RPs to ensure data is matched in proximity to where it is relevant which improves system performance. For that sake, we:

- describe how to enrich events and subscriptions with geo-context information (Section II),
- introduce our novel approach on how to use this geo-context information to select appropriate RPs (Section III).

Finally, we draw a conclusion and outline next steps (Section IV).

II. Enriching events and subscriptions with geo-contexts

Before we explain how to use geo-context information to select RPs, we repeat our previous definition of geo-contexts presented in [5], [6]. Note, that we use a slightly updated terminology, as indicated below, that better fits the intended purpose of this paper.

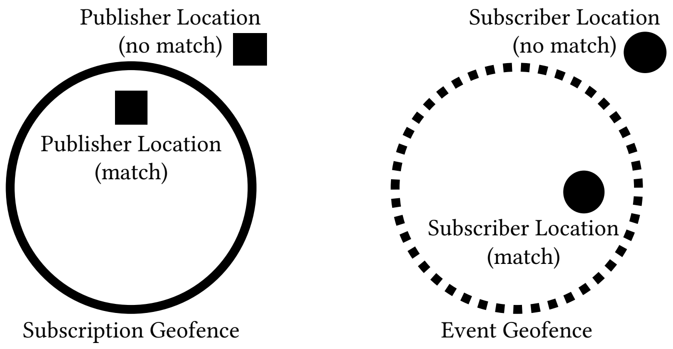

Figure 1. Subscription GeoCheck (left) and event GeoCheck (right) [5], [6].

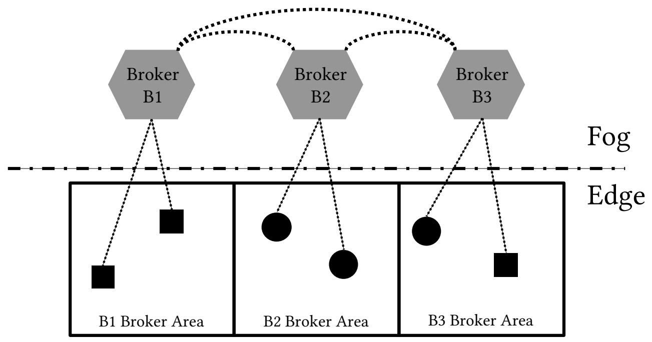

Figure 2. Setup with three brokers deployed in the fog that facilitate communication between publishers (squares) and subscribers (circles).

There are four geo-context dimensions. Clients have a geographic location, which consists of a latitude and a longitude value. For publishers, the corresponding dimension is called **publisher location**; for subscribers, the corresponding dimension is called **subscriber location**. Beyond this, each event and subscription has an area it relates to; we propose to use geofences to describe these areas. 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[1].

For bringing geofences and locations together, two checks are necessary to decide whether data from a given publisher should be delivered to a given subscriber (Figure 1) – first, from the subscribers's perspective with the help of the subscription geofence and publisher locations (subscription GeoCheck) and, second, from the publishers's perspective with the help of the event geofence and subscriber locations (event GeoCheck). For a more detailed discussion and explanation of the geo-context model, we refer to our previous work [5]. Furthermore, it usually makes sense to combine the two GeoChecks with an additional ContentCheck, i.e., 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 [6]. In this work, we describe how such single-node service instances (here called brokers) can communicate via rendezvous-based routing.

III. Rendezvous node selection

In this section, we present our RP selection approach that builds upon geo-context information. RPs reduce communication cost by being a "meeting point" for subscriptions and events: the matching occurs at the RP brokers [4, p. 166]. Hence, they constrain propagation of events and/or subscriptions to a small subset of nodes which improves system efficiency. A major challenge with rendezvous-based routing in wide-area deployments is to select an RP that is close to the subscribers or publishers of an event. Many state of the art solutions distribute RPs uniformly over available brokers [7], [8]; this is problematic for IoT data traffic which is non-uniform. Our key idea is to use the IoT data itself to identify RPs physically close to publishers and/or subscribers; this is only possible if the necessary geo-context information is attached to the events and subscriptions.

In the following, we first describe some assumptions (Section III-A) before we present 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. Finally, we discuss how both strategies compare against the baseline, flooding of either events or subscriptions to all brokers, and other rendezvous-based routing approaches found in the literature (Section III-D).

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 a single broker, i.e., their local broker (LB). A broker is responsible for communication with all clients located in its *broker area*; usually, a broker area covers the region surrounding the physical location of the corresponding broker as this asserts low latency communication between the broker and clients located in the area[2], see also Figure 2. Subscriptions and published events comprise the 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 (Section III-B and III-C), as soon as the LB has

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

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

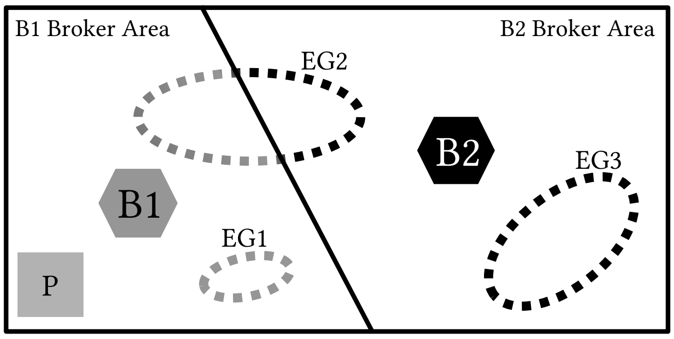

Figure 3. An event only needs to be sent to brokers with a broker area that intersects with the event geofence.

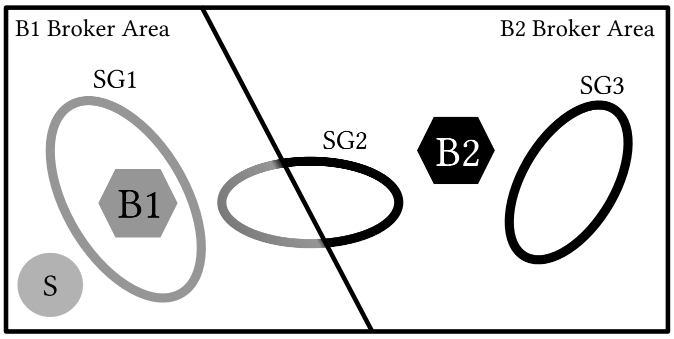

Figure 4. A subscription only needs to be sent to brokers with a broker area that intersects with the subscription geofence.

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 could possible manage a matching subscription. Fortunately, the event geofence can be used to select these RP because only broker areas intersecting with the event geofence might contain clients 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 B1 and publishes three events—each has a different event geofence (EG):

- EG1 does not intersect with the broker area of broker B2 so the event does not need to be forwarded for matching to B2.
- EG2 intersects with the broker areas of B1 and B2 so the event needs to be matched at B1 and be forwarded for matching to B2.
- EG3 only intersects with the broker area of B2 so the event needs to be forwarded for matching to B2. Note, that matching at B1 can be omitted, as no subscription created by the clients located in the broker area of B1 can pass the event GeoCheck.

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. Thus, the RP is the LB of the publisher. While this means matching only occurs at a single broker, it also implies that 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 clients 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 broker B1 and creates three subscriptions—each has a different subscription geofence (SG):

- SG1 does not intersect with the broker area of broker B2 so the subscription does not need to be forwarded to B2.
- SG2 intersects with the broker areas of B1 and B2 so the subscription needs to be maintained at B1 and be forwarded to B2.
- SG3 only intersects with the broker area of B2 so the subscription needs to be forwarded to B2. Note, that the subscription can be discarded at B1, as none of the clients managed by B1 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.

D. Discussion

It is straightforward to calculate how many inter-region messages can be saved when using our rendezvous node based approaches compared to a flooding solution for a given event/subscription. In both cases, the given event/subscription must only be distributed to the brokers whose broker areas intersect with the corresponding geofence, rather than to all brokers participating. Consider the following example: if one data-distribution broker instance runs in each of the 21 currently available AWS regions, a published event that is only targeted at clients in Europe would have an event geofence that only intersects with the broker areas of 5 AWS regions and thus reduces the amount of inter-node messages by 16 ($>75\%$). In case of more dense deployments or events that are only relevant to an even smaller number of subscribers, benefits could become even higher.

We see the most closely related work in two areas: rendezvous-based pub/sub, e.g., [7]–[9], and geo-distributed, location-based pub/sub, e.g., [2], [10]–[12]. Still, none of these approaches aims to use geo-context information to select RPs close to the publishers or subscribers of an event.

With our current approach, each broker has to know about all other brokers and their broker areas. This is plausible, if the number of brokers is limited to, for example, one very scaleable instance per AWS data center. One interesting aspect about rendezvous-based routing is, however, that matching not always has to occur at the RP when there is a network/hierarchy of nodes. This could potentially be used to further extend our approach to support broker deployments in which not all brokers are connected directly, e.g., because some brokers are also running at the Edge.

IV. Conclusion

In this paper, we proposed to make use of geo-context information attached to published messages and subscriptions to select RPs. By doing so, we can ensure that events and subscriptions are matched close to the publishers or subscribers of an event. This can significantly reduce the amount of data that needs to be distributed between geo-distributed broker instances. In the future, we plan to more thoroughly study the effects of our approach on excess data and quantify effects on system characteristics such as the communication latency between IoT devices. We are currently implementing a broker prototype for use case driven experiments and are also preparing a simulation-based study.

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] 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.

[3] R. Baldoni, L. Querzoni, and A. Virgillito, "Distributed event routing in publish/subscribe communication systems: a survey," p. 27, 2005.

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

[5] J. Hasenburg and D. Bermbach, "Towards geo-context aware IoT data distribution," in *4th Workshop on IoT Systems Provisioning & Management for Context-Aware Smart Cities (ISYCC)*. Springer, 2019.

[6] ——, "GeoBroker: Leveraging geo-context for IoT data distribution," *Computer Communications*, 2020.

[7] 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 Berlin Heidelberg, 2001, vol. 2233, pp. 30–43.

[8] P. Pietzuch and J. Bacon, "Hermes: a distributed event-based middleware architecture," in *Proceedings 22nd International Conference on Distributed Computing Systems Workshops*. IEEE, 2002, pp. 611–618.

[9] A. Gupta, O. D. Sahin, D. Agrawal, and A. El Abbadi, "Meghdoot: Content-based publish/subscribe over p2p networks," in *Middleware 2004*, H.-A. Jacobsen, Ed. Springer Berlin Heidelberg, 2004, vol. 3231, pp. 254–273.

[10] G. Cugola and J. Munoz de Cote, "On introducing location awareness in publish-subscribe middleware," in *25th IEEE International Conference on Distributed Computing Systems Workshops*. IEEE, 2005, pp. 377–382.

[11] Y. Teranishi, R. Banno, and T. Akiyama, "Scalable and locality-aware distributed topic-based pub/sub messaging for IoT," in *2015 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2015, pp. 1–7.

[12] R. Kawaguchi and M. Bandai, "A distributed MQTT broker system for location-based IoT applications," in *2019 IEEE International Conference on Consumer Electronics (ICCE)*. IEEE, 2019, pp. 1–4.