Enabling Location-aware Operation in Decentralized IoT Communications

Matteo Visotto[+] and Luca Mottola[+*†]
[+]Politecnico di Milano (Italy), [*]RI.SE Computer Science (Sweden), [†]Uppsala University (Sweden)

ABSTRACT

We present an efficient design to enable location-aware operation in decentralized IoT communications. Large-scale IoT systems represent the backbone of a smart city functioning, allowing pervasive environmental sensing across devices and networks. However, existing IoT communication systems are largely driven by data types and miss out on embracing *data location*, which is fundamental in environment sensing. To address this issue, we demonstrate it is possible to efficiently embed a notion of location within the Zenoh protocol. We make it possible to steer message routing based on both data type and location, yet without altering the existing routing core and message forwarding, unlike most existing solutions. We also present three encoding techniques for location data, each of them representing a different trade-off between expressiveness and performance overhead. Our evaluation uses a virtualized environment and real-world packet traces of heterogeneous networks. We show, for example, that our design decreases the average message latency by more than 50% when routing data also based on location, while increasing throughput, compared to two different baselines.

CCS CONCEPTS

• **Networks** → **Network protocol design**; **Location based services**.

KEYWORDS

Location-awareness, IoT, Pub/Sub, Req/Resp, Protocol, Zenoh

ACM Reference Format:

Matteo Visotto[+] and Luca Mottola[+*†], [+]Politecnico di Milano (Italy), [*]RI.SE Computer Science (Sweden), [†]Uppsala University (Sweden). 2024. Enabling Location-aware Operation in Decentralized IoT Communications. In *The 2nd Workshop on Advances in Environmental Sensing Systems for Smart Cities Workshop Chairs (EnvSys '24), June 3–7, 2024, Minato-ku, Tokyo, Japan.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3661813.3661817

1 INTRODUCTION

Large-scale Internet of Things (IoT) systems form the backbone of a smart city operation. Embedded sensors and edge devices enable fine-grained sensing of the city environment and are instrumental to improve the use of resources and assets, to investigate the impact of human activities, and to understand climate changes [6].

Problem. Compared to the complexity of environment sensing in smart cities, the underlying IoT network support falls short of expectations. Most IoT protocols, indeed, only support quite straightforward communication paradigms, which may often result in suboptimal performance, as we further articulate in Sec. 2.

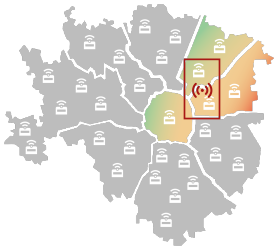


Figure 1: Air quality sensing in Milano (Italy). *Existing IoT protocols incur in large network overhead when the locations of interest arbitrarily intersect the existing logical slicing, as in the red rectangle.*

Consider for instance an air quality sensing application for the city of Milano (Italy): a setting we have first-hand experience with [4]. An example air quality map is shown in Fig. 1. Milano is split into different municipalities. Air quality sensors throughout the city are characterized by their GPS coordinates and the municipality they belong to. Existing protocols, for instance, those supporting the Publish/Subscribe (Pub/Sub) paradigm, suffice as long as data is pulled from *air quality sensors* belonging to a *specified subset of municipalities*, as is the case for the three colored municipalities in Fig. 1. Similar considerations equally apply to protocols providing Request/Response (Req/Resp) forms of interaction.

Most existing protocols, however, offer no native support to reason on the actual *location of data*, besides some coarse-grained logical location. Pulling data *from a programmer-defined area* that intersects the logical slicing in arbitrary ways, such as the red rectangle of Fig. 1, is extremely complex. Existing solutions force developers to obtain data from the three colored municipalities anyway and discard messages originating outside of the red rectangle, but within the three considered municipalities. This incurs additional programming effort and, most importantly, potentially generates unnecessary network load.

Solution. We present a design that enables *location-aware operation* in an existing IoT protocols *without changing the routing core and message forwarding*, as described in Sec. 3. Such a feature is crucial because it lets us benefit from optimizations and testing of the existing protocol implementation. We achieve this by modifying the protocol architecture in a way that completely decouples the location-specific functionality from the rest of the protocol. As a by-product of this, we gain the freedom of deciding how to encode location information. We present three example encodings with different trade-offs between expressivity and overhead. We can also make (non-)location-aware messages co-exist and retain backward compatibility with the original protocol implementation.

We implement a prototype on top of Zenoh [19], one of the few decentralized IoT protocols supporting both Req/Resp and Pub/Sub

EnvSys '24, June 3–7, 2024, Minato-ku, Tokyo, Japan

ACM ISBN 979-8-4007-0659-2/24/06
https://doi.org/10.1145/3661813.3661817

interactions. We make both paradigms operate in a location-aware manner. We use the prototype running in a virtualized environment and real-world packet traces to emulate different network conditions and measure key performance metrics, as reported in Sec. 4. We demonstrate, for example, that our design decreases the average message latency by more than 50% in scenarios akin to Fig. 1, while increasing throughput.

2 BACKGROUND AND RELATED WORK

Existing efforts tackle the problem of IoT location-aware communications *separately* for Req/Resp or Pub/Sub systems.

Several designs exist that extend the Pub/Sub paradigm with location-aware functionality when using the *content-based message model* [7]. Examples include systems to provide customized services, mainly for advertisement, based on user location [3, 5, 8, 10, 13, 16]. In smart city applications, on the other hand, data flows are usually specified based on data types. Pub/Sub systems using the *topic-based message model* [7] provide efficient network support for this, with the MQTT protocol being the most representative example. Several works extend MQTT with location-aware functionality.

MQTT-G [2] extends the original MQTT design by adding location data between the message header and the payload. Unlike what we do, this requires modifying the core functionality of the MQTT broker, which must retain location information for subscribers, a new message type, and new APIs that replace the original MQTT interface. Backward compatibility is achieved by switching between the two implementations based on a flag in the message header. Similarly, MQTT5 [11] requires new message types to manage location information, transmitted in addition to the regular MQTT messages, placing additional strain on the network.

In other works, GeoMQTT [9] presents an MQTT extension that adds both spatial and time information. The design includes a new indexing structure at broker side to store and retrieve client information. Location information are embedded within the message payload, which therefore requires parsing each and every single message at the broker, adding processing overhead. LA-MQTT [14] relies on an external software component to manage location information, leaving the original MQTT implementation unchanged. The external component is deployed on a backend and coordinates with the original MQTT broker by subscribing to dedicated control topics that allows it to receive and dispatch messages with location information. This solution adds a new single point of failure in addition to the existing MQTT broker.

IoT communication systems based on Req/Resp interactions rarely offer location-aware functionality. Existing works focus on frameworks to handle location information at the servers. For example, LAISYC [1] presents a design using UDP to manage location-based information for HTTP applications, mainly applicable to real-time GPS-based mobile applications.

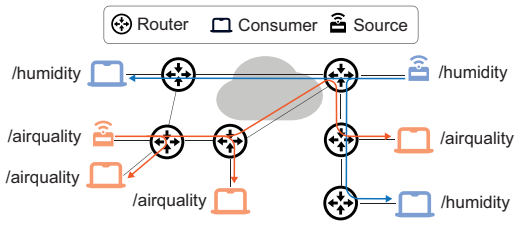


Figure 2: Zenoh architecture.

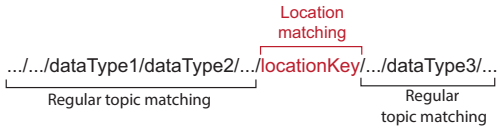


Figure 3: Topic with location key.

Zenoh [19] provides both Req/Resp and Pub/Sub interaction paradigms using a topic-based approach. It supports fully decentralized architectures, as exemplified in Fig. 2, enabling device interconnections also over the public internet. Zenoh uses three different device configurations. *Peer*s are devices running the Zenoh protocol and capable of sending or receiving messages; they can dynamically discover and connect with other peers as needed. *Router*s manage data flows across different endpoints; they can dynamically adjust routes to ensure reliable and efficient message forwarding. *Client*s are end devices that cannot route messages.

Zenoh has no built-in location-aware functionality. Because of the unified support to both interaction paradigms and modular design, we use it as a basis for our work, as described next.

3 DESIGN

We set two primary goals when enabling location-aware functionality in Zenoh: *i)* to leave the routing core and message forwarding unchanged, and *ii)* to ensure co-existence and maintain backward compatibility with the original version of the protocol.

We achieve both goals by introducing location information within a message topic, formatted in a way that *allows us to interject when this information does appear*, as intuitively shown in Fig. 3. Upon inspecting a topic and recognizing location information, we delegate matching *of that part of the topic* to an additional component we develop, shown within the complete architecture in Fig. 4. This effectively decouples the location matching process from the original topic matching functionality. The latter applies unchanged to the remaining part of the topic. The outcomes of the two matching processes are taken together to determine overall matching.

Our design brings several benefits. Matching of location information is orthogonal to the routing functionality and becomes an additional input to decide message forwarding. This happens *without altering* the existing routing core and message forwarding, as matching of location information is embedded within the overall topic matching. Changes to the original protocol implementation are minimal, only requiring modifications to 6 code lines. This allows us to benefit from the optimizations and testing of the existing protocol implementation. If location information does *not* appear in a topic, the original topic matching applies, effectively ensuring the co-existence of (non-)location-aware messages. Finally, making an external component responsible for location matching provides freedom in deciding how to encode this information. We describe

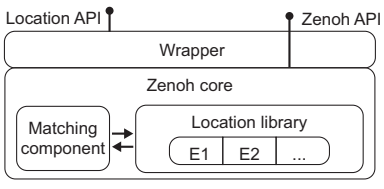


Figure 4: Architecture design.

next three example encodings that strike different trade-offs between expressivity and overhead.

We target large distributed systems, akin to the air quality monitoring example presented in the Introduction. As we explain next, consumer selection at the routers based on both topic and location generally decreases the network traffic, which is advantageous also in constrained and/or congested networks. However, applying our design to other brokered architetures, like MQTT, would increase the computational load on the broker, which still represents a single point of failure, making our techniques less effective.

3.1 Location Matching Component

We call *location key* the specific topic level where location information appears, if any, as shown in Fig. 3. Regardless of the encoding of location information, the location key includes three parts: a prefix, the actual encoded location information, and an optional flag space to specify additional information. While parsing a topic name, upon recognizing the prefix, we extract and forward the location key to the location matching component.

The location matching component exposes a single operation that takes as input two location keys and returns a Boolean value representing the possible match. From the prospective of the original Zenoh implementation, the location matching component operates as a black box with a well-defined Boolean interface.

Upon receiving two location keys through its interface, the location matching component first recognizes the specific encoding technique employed among the supported ones; then it accordingly decodes the location information. Matching of location information is then performed based on the specific semantics, as dictated by the encoding at hand, and the result is returned to Zenoh as part of the complete topic matching process.

3.2 Encoding Location Information

We design three encoding techniques, each providing a different tradeoff between expressiveness and processing overhead.

JSON+Base64. We employ a JSON object, then encoded in Base64, to describe location information. This offers the highest expressiveness among the encodings we experiment with. One can describe complex shapes, including both user-defined shapes and ego-centric definitions [15]. Information sources, such as publishers, are associated to single points in space, whereas information consumers, such as subscribers, are associated to areas of interest. A match occurs if the point associated to the source falls within the area associated to the consumer.

For example, an information consumer defines a circular shape centered on itself as

```
{c: {x: 10.10, y: 20.54}, r: 10}
```

where `c` indicates the circular shape centered on the consumer position `(x,y)`, with radius `r`. This JSON object requires 44 bytes for its Base64 representation. Overall, this encoding incurs higher network and processing overhead compared to the others, primarily due to its high expressiveness.

MGRS. The Military Grid Reference System (MGRS) is a geocoordinate system used to specify locations on the Earth's surface, originated in military applications. It is derived from the Universal Transverse Mercator (UTM) coordinate system [12] and features

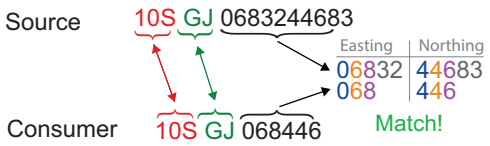


Figure 5: MGRS match example.

an easy to parse, compact representation. The MGRS is arranged as a grid with 100000-meter squares, specified with Easting and Northing values. It can represent areas from $6° \times 8°$ grid zone polygons down to $1m^2$ squares, progressing by orders of magnitude. The matching semantics is similar to JSON+Base64. However, sources can only be associated with the smallest location MGRS can represent, that is, a $1m^2$ square area, rather than single points.

MGRS does not allow for the definition of intersections. Matching is therefore as simple as recursively checking the containment of one coordinate into another. Fig. 5 shows an example where we represent consumers with $100m^2$ precision. The match is positive because the two coordinates share both the grid zone (10S), the 100000-meter square (GJ), and Easting and Northing values. As for the latter, the match occurs because the two coordinates share the values of their shortest representation, indicating containment of the source coordinates within the consumer coordinate.

MGRS evidently provides lower expressiveness compared to JSON+Base64, because of the rigid representation of location information. This comes in exchange of a much more compact encoding and a simplified matching process, which come handy in constrained networks, as we demonstrate in Sec. 4.

Bloom filters. A Bloom filter is a *probabilistic* data structure to efficiently check the membership of an element in a set [18]. To use Bloom filters to encode location information, we first partition the space into a grid of given granularity. We incorporate each element of the grid that falls within the consumer's area of interest into a filter and insert the resulting bit array in the location key. At the source, we replicate the same process by only incorporating the grid element corresponding to its location. Matching is performed in a *bitwise* manner at the routers, as in Fig. 6. The 1 bits in the source filter represent the outcome of the hash functions computed for the single grid element of the source. In the consumer's filter, the 1 bits represent the grid elements covering the area of interest.

By design, Bloom filters may allow false positives to occur but never produce false negatives. This makes them appropriate to encode location information in a producer-consumer communication system. Because of the lack of false negatives, we ensure that messages are always delivered to the target consumers; in the worst case, because of false positives, a consumer with a non-matching area of interest may however receive spurious messages.

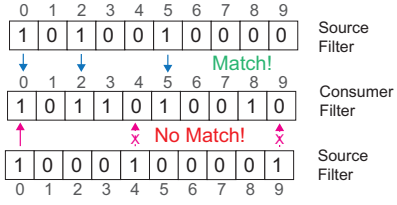


Figure 6: Bloom filters match example.

3.3 Developer APIs

Our design is available to programmers through an additional API exposed by a custom wrapper, as shown in Fig. 4.

Key to our design is for a device to provide its own location. Programmers decide how often to update this information. We offer an operation to indicate the current device position, together with a corresponding *lifetime* [17]. If set to infinite, the device is considered stationary. Otherwise, the behavior is different at sources or consumers. Sources check a change in device location whenever a message is generated, ensuring that the location information attached to the message is most recent. Consumers check for changes in the device location upon expiration, according to the programmer-provided lifetime. If the location does change, any current subscription that uses the device location, for example, the ego-centric example for JSON+Base64, is updated accordingly.

All other operations in the additional API use the same operation names of the original APIs and extend them with location parameters. The wrapper consists of a new class that replicates the original Zenoh `Session` class, where we extend the parameter set of each operation requiring a topic with the needed location information. The wrapper generates the location key and calls the equivalent Zenoh's operation with the extended topic structure.

4 EVALUATION

Sec. 4.1 introduces our experimental setup, the metrics we consider, and the baselines we compare with. We discuss the results we obtain in Sec. 4.2, leading to three key conclusions:

(1) in a setting with logical partitions, our design decreases message latency up to a 50% improvement compared to the baselines, while increasing throughput;
(2) in a setting with no logical partitions, our design still reduces the load on intermediate routers, yielding a 55% (20%) lower latency compared to the Topic (Payload) baseline;
(3) MGRS encoding reduces message latency up to 40% of the other two encoding techniques, due to a more compact representation and more efficient matching.

4.1 Settings

The performance of a decentralized IoT communication protocol depends on multiple factors, including network links and device processing. Using a network simulator would only capture a few of these, for example, lacking a model of local processing times, which is however crucial to test different encoding schemes.

Setup. We opt to create a virtualized environment to realistically measure performance. We create a variable number of virtual machines (VM) in a Proxmox cluster, each running Ubuntu Server 22.04.1 and equipped with 4 cores plus 4GiB of RAM. The number of cores and their performance match an average IoT edge device, such as a RaspberryPI. Each VM acts as a Zenoh client or router.

We meticulously select paradigmatic network topologies to understand the specific trade-offs at stake. To realistically model network conditions and ensure repeatability across experiments, each Zenoh client runs MahiMahi v0.98, a toolset for emulating dynamic links, such as the one in cellular networks. We use the MahiMahi `TMobile-LTE-driving.down` trace file emulating a 100Mbps LTE connection. To place an additional stress factor, each Zenoh router uses Wondersharper v1.4.1 to limit the bandwidth available along the router backbones to only 5Mbps, representing a case where Zenoh must co-exist with significant network traffic.

We carefully choose system configurations that may emulate the behavior of wider networks. For instance, we replace a higher number of sources with fewer sources with higher message rates, which is effectively immaterial as long as the load on the links is the same. To ensure fine-grained control of the experiments and equal conditions against the baselines, we configure Zenoh clients not to simultaneously act as routers. We also fix network links forming the topology a priori, essentially skipping Zenoh's discovery phase, which may lead to different network topologies being used in different experiments. Bloom filters are generated with a 25% probability of false positives. We discuss this specific parameter choice at end of the next section.

This setup does provide the greatest realism, as it is closest to a real deployment. We also acknowledge that it comes at the cost of making experiments more time consuming and less scalable: the experiments run in real time and available hardware resources allow us to virtualize only a limited number of nodes.

Metrics. We measure message *throughput* and system *latency*, which are staple networking metrics and directly impact the end-user's perceived quality of service. In our design, they also help identify how different location encoding techniques influence the number of transmitted messages and their transmission times.

Clocks across VMs are synchronized as they are all virtualized over the same Proxmox physical node, and hence virtual their clocks are all attached to the same physical clock. We measure message latency as $L = t_r - t_p$, where t_p is when the message is sent and t_r is when the message is received. We measure throughput as the number of messages *matching a consumer's interest* are received within a given time frame Δt. We run each experiment 12 times for a duration of 3 minutes each.

Baselines. We compare our design against two baselines, called Topic and Payload, and representative of many of the existing solutions discussed in Sec. 2. We implement both baselines on top of the original Zenoh protocol v0.7.2-rc.

Topic incorporates location details right into the first two levels of the topic structure. For example, it uses a topic

```
<latitude>/<longitude>/airquality
```

to generate air quality data from a sensor located at (`<latitude>`; `<longitude>`). Message consumers use a wildcard in the first two topic levels, such as

```
*/*/airquality
```

and therefore receive every message carrying air quality data. Upon receiving the message, the consumer extracts the coordinates from the first two topic levels of the message and determines whether to continue processing the message or to drop it, depending on whether it falls within the geographical area of interest.

Payload embeds location information within the message payload. The topic only includes information on the data type, as in `/airquality`, and is used the same at sources and consumers. Upon receiving a message, the same process as in the Topic baseline applies, with location data extracted from the payload.

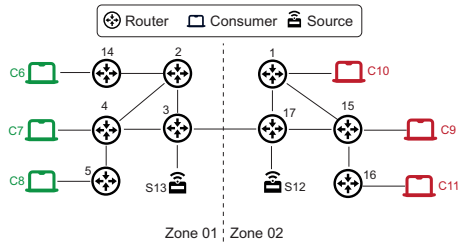
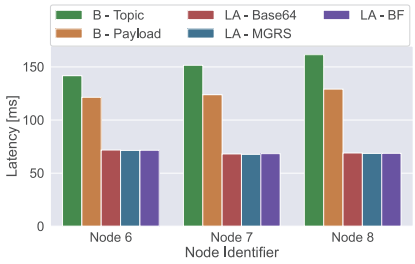
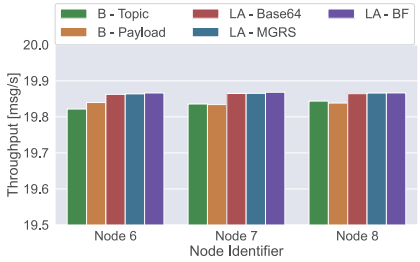


Figure 7: Logical partition network topology.



(a) Logical partition: latency.



(b) Logical partition: throughput.

Figure 8: Logical partition: results.

4.2 Results

We consider different scenarios of interest.

Logical partitions. We partition the network into two logical zones, as depicted in Fig. 7, modeling the air quality scenario of the Introduction. Each zone includes one source and three consumers. Consumers in green match messages generated in the area of S13. This is the area of interest for the application, that is, the colored rectangle in Fig. 1. S13 injects 20 msg/sec. Consumers in red match messages generated in the area of S12. S12 generates 3200 msg/sec, mimicking the network traffic generated in the rest of the city.

Fig. 8a shows the average latency at each consumer. Without location-aware operation, messages generated at S13 or S12 are routed all the way to the opposite network partition only to be discarded at the consumer. Messages generated by S12, in particular, greatly impact the baselines' latency as they end up where they should not be routed at all. In contrast, *without altering Zenoh's routing core*, our design prevents the unnecessary routing, reducing the average latency of about 50% compared to the baselines.

Fig. 8b shows the average throughput we measure. The baselines show lower throughput compared to our design, regardless of location encoding. This arises as the consumers in the baselines can only process a smaller number of messages from S13 within the same interval because, with the same processing resources, they must also handle (and discard) messages from S12 simultaneously. Note how the plot zooms in along the y-axis for better clarity.

This phenomenon is also apparent when looking at the standard deviation computed across all latency values for each consumer. The higher number of messages that consumers must handle when using either baseline yields longer processing queues, producing a ≈53% higher variability in measured message latency.

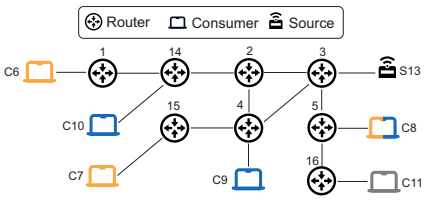


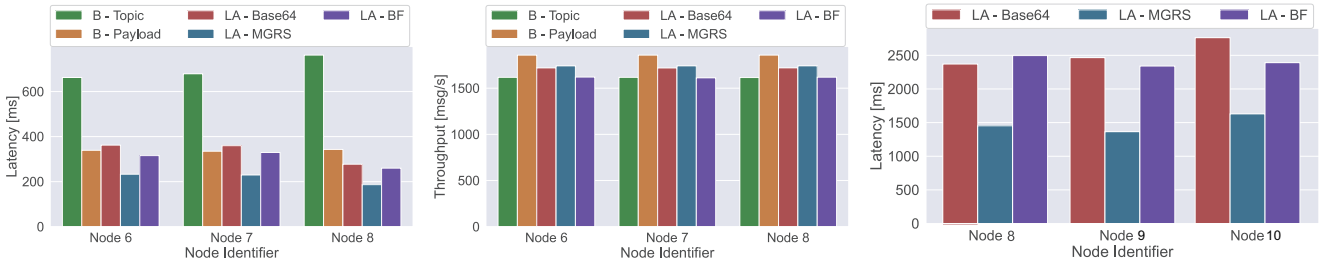
Figure 9: Mixed network topology.

Mixed network layouts. We evaluate the performance differences between our design and the baselines in the absence of logical partitioning. We consider the topology in Fig. 9: the three consumers shown in orange match both data types and location of the single source. The other three consumers match the data type but belong to a different geographical zone, where no source generates matching messages. The source sends 3200 msg/sec to stress the system.

Fig. 10a depicts message latency at the three matching subscribers. Our design provides better performance than the baselines also with no logical partitioning. The latter pay a penalty due to the unnecessary traffic generated to reach consumers where messages are eventually dropped. The `Payload` baseline shows a good performance like the location-aware techniques. The 2-level shorter topic require less time to process, compensating the time needed to filter messages upon reception. This solution, however, does not scale. With larger networks, the number of unnecessary messages increases, reapproaching the results of Fig. 8. Unlike Fig. 8b, however, the throughput does not suffer as much in the baselines, as shown in Fig. 10b. There is indeed a "pile up" effect of sort at the queues of intermediate routers, yet the additional latency this generates is roughly the same for all messages, eventually leading to a similar number of messages received within the same time window.

We use the same network layout to evaluate the latency difference among location encoding techniques, shown in Fig. 10c, swapping the role of the two groups of consumers. MGRS encoding is the best performing, owing to its concise representation that leads to a reduced message size and is amenable to an efficient recursive matching process that approaches $O(k)$, where k is a constant. As expected, the worst-performing technique is the JSON+Base64 one, which is verbose and complex to parse, yet provides the highest expressiveness. The performance of Bloom filters depends on the constituent parameters, such as domain definition and desired probability of false positives. Since we configure our experiments to obtain an encoding length comparable to the Base64 one, the Bloom filter performance shows how matching using bitwise operations abates processing time, reducing latency.

Bloom filters: probability of false positives. The 25% probability of false positives we use is not a random value. We rather determine this parameter to ensure that no false positives occur in our specific setting, based on a dedicated experiment.



(a) Mixed network layout: latency.

(b) Mixed network layout: throughput.

(c) Latency comparison for location encoding.

Figure 10: Mixed network: results.

Subscriber ID	Expectation
S6	6
S7	6
S8	4
S9	4
S10	1
S11	3

Table 1: Expected number of messages.

False positive percentage	Result
1%	No false positive
10%	No false positive
20%	No false positive
25%	No false positive
30%	1 false positive
40%	4 false positive

Table 2: False positive experimental outcome.

We define a 5×5 grid domain with one message source and six consumers covering a portion of the domain. The source sends one message for each domain element. Each consumer records every message it receives, including information of the cell the message originates from. Tab. 1 reports the number of messages each consumer *expects to receive*, depending on how each of them matches a given slice of the domain. We regenerate the Bloom filters with increasing probability of false positives, starting from 1%. Tab. 2 shows that we record the first false positive with a 30% false positive probability. The 25% setting we use is the greatest setting that produces no false positives.

5 CONCLUSION

We presented an efficient design to enable location-aware operation on top of the Zenoh protocol, supporting both Req/Resp and Pub/Sub interaction paradigms. Our design delegates handling of location information to an external component that is completely decoupled from the rest of Zenoh's implementation, retains the original functioning of Zenoh's routing core and thus benefiting from existing optimizations and testing. This also allows us to separate out the encoding of location information: we indeed present three techniques to encode location information, each of them representing a trade-off between performance and expressiveness. Our evaluation uses a virtualized environment and real-world message traces. We demonstrate, for example, that our design reduces the average message latency by more than 50% in a network with a logical partitioning, while increasing throughput compared to the two baselines we consider. The MGRS location encoding technique, moreover, provides 40% lower latency than the other two encoding techniques, at the cost of reduced expressivity.

The implementation is publicly available [20].

Acknowledgments. This work was partly supported by the National Recovery and Resilience Plan (NRRP), Mission 4 Component 2 Investment 1.3 - Call for tender No. 1561 of 11.10.2022 of Ministero dell'Università e della Ricerca (MUR); funded by the European Union - NextGenerationEU.

REFERENCES

[1] S. J. Barbeau et al. 2011. A Location-aware Framework for Intelligent Real-time Mobile Applications. *IEEE Pervasive Computing* (2011).
[2] R. Bryce et al. 2018. MQTT-G: A Publish/Subscribe Protocol with Geolocation. In *International Conference on Telecommunications and Signal Processing (TSP).*
[3] X. Chen et al. 2003. An Efficient Spatial Publish/Subscribe System for Intelligent Location-Based Services. (2003).
[4] Citcom.AI Consortium. 2024. Citcom.AI. https://citcom.ai/
[5] P. Costa et al. 2007. Reconfigurable Component-based Middleware for Networked Embedded Systems. *International Journal of Wireless Information Networks* (2007).
[6] R. Dameri et al. 2013. Searching for Smart City Definition: a Comprehensive Proposal. *International Journal of Computers & Technology* (2013).
[7] P.Th. Eugster et al. 2003. The Many Faces of Publish/Subscribe. *ACM Comput. Surv.* (2003).
[8] P.Th. Eugster et al. 2005. Location-based Publish/Subscribe. In *Fourth IEEE International Symposium on Network Computing and Applications.*
[9] S. Herle and J. Blankenbach. 2016. GeoPipes Using GeoMQTT. In *Geospatial Data in a Changing World.*
[10] H. Hu et al. 2015. A Location-aware Publish/Subscribe Framework for Parameterized Spatio-textual Subscriptions. In *2015 IEEE 31st International Conference on Data Engineering.*
[11] F. Ihirwe et al. 2021. Towards an MQTT5 Geo-location Extension for Location-aware Applications. In *International Conference on Telecommunications and Signal Processing (TSP).*
[12] R.B. Langley. 1998. The UTM Grid System. *GPS world* (1998).
[13] G. Li et al. 2013. Location-aware Publish/Subscribe. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining.*
[14] F. Montori et al. 2022. LA-MQTT : Location-aware Publish-subscribe Communications for the Internet of Things. *ACM Transactions on Internet of Things* (2022).
[15] L. Mottola et al. 2007. Enabling Scope-based Interactions in Sensor Network Macroprogramming. In *International Conference on Mobile Adhoc and Sensor Systems (MASS).*
[16] L. Mottola et al. 2008. A Self-repairing Tree Topology Enabling Content-based Routing in Mobile Ad Hoc Networks. *IEEE Transactions on Mobile Computing* (2008).
[17] L. Mottola and G. P. Picco. 2007. Programming wireless sensor networks with logical neighborhoods: a road tunnel use case. In *SENSYS.*
[18] S. Tarkoma et al. 2011. Theory and Practice of Bloom Filters for Distributed Systems. *IEEE Communications Surveys & Tutorials* (2011).
[19] Zettascale Technology. 2024. Zenoh. https://zenoh.io/
[20] M. Visotto and L. Mottola. 2024. Location-aware version of Zenoh Source Code. https://lazenoh.neslab.it