Ensuring Low-Latency and Scalable Data Dissemination for Smart-City Applications

Shweta Khare*, Hongyang Sun*, Kaiwen Zhang†, Julien Gascon-Samson‡ and Aniruddha Gokhale*
*Vanderbilt University, Nashville, TN 37235, USA; Email: {shweta.p.khare,hongyang.sun,a.gokhale}@vanderbilt.edu
†École de technologie supérieure, Montreal, Canada; Email: kaiwen.zhang@etsmtl.ca
‡University of British Columbia, Vancouver, Canada; Email: julien.gascon-samson@ece.ubc.ca

***Abstract*—Low latency and scalable data dissemination is a critical requirement for many IoT applications, e.g., smart city applications, which are often built over a publish/subscribe communication paradigm. Ensuring low latency requires effective load balancing of the publish/subscribe topics across the different publishers and subscribers. To that end, we present ongoing work on a data-driven approach to learning a latency-aware model of IoT broker loads, and in turn using it to determine broker replication, and balancing topics across them.**

***Index Terms*—topic-based publish/subscribe, IoT, latency, scalability.**

I. Introduction

Applications from IoT domains, such as smart cities, are large-scale, continuously online and highly latency-sensitive in nature. They collect large amounts of monitoring data from a diverse range of sensors/information sources and process it in a streaming fashion to provide a variety of real-time services, e.g., real-time alerts of traffic congestion to drivers. For instance, at rush hour, there can be thousands of commuting vehicles in a medium sized city, thereby driving the need for an efficient, low latency and scalable data dissemination method.

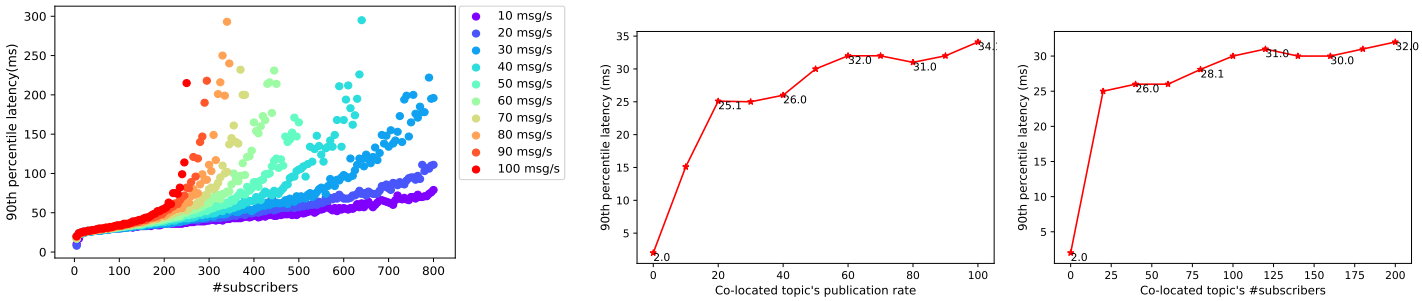
The publish/subscribe (pub/sub) [1] communication paradigm is highly suitable for the data dissemination needs of smart-city applications as it offers scalable, anonymous and decoupled interactions between data producers (publishers) and data consumers (subscribers). Subscribers simply specify their interest in the form of subscriptions (i.e., a topic name in topic-based pub/sub or a predicate in content-based pub/sub). The pub/sub system then forwards the publication messages that match the subscriber's interest/subscription. Such loose coupling allows the system to scale seamlessly.

Although pub/sub is highly amenable to scalable data dissemination, a shared, city-wide pub/sub service that meets the data dissemination needs of all smart-city applications must be able to (i) dynamically adapt to changing load while making efficient use of resources, and (ii) support non-functional QoS properties, such as bounded latency of data delivery. Very few pub/sub systems provide these two properties holistically. For example, IndiQoS [2] ensures bounded latency by reserving network level resources over a distributed hash table (DHT)-based peer-to-peer overlay. However, the capability to make network-level resource reservation is not always available. Harmony [3] continuously monitors link quality and adapts routing paths for QoS management over an unstructured peer-to-peer overlay network. If the underlying transport supports priority-based scheduling, Harmony can make use of it for QoS-aware scheduling of messages at the brokers. Instead of using fixed routing paths, DCRD [4] dynamically switches among next-hop downstream nodes to bypass failures and to meet QoS requirements. For each subscriber, brokers maintain a sorted list of next-hop nodes to choose from. The nodes are sorted on the basis of two metrics – the expected delay and the reliability of message delivery – which are computed for each next-hop node in a distributed manner. On the other hand, FogMq [5] migrates entire brokers from one fog site to another to ensure bounded tail latency using a distributed flocking algorithm.

While these solutions do provide QoS guarantees, they have been designed for peer-to-peer (structured/unstructured), multi-hop overlay networks and are focused largely on re-routing paths for message delivery in a network-aware fashion. These solutions are not directly applicable to single-hop, single broker layer, cloud-based pub/sub systems like MQTT, Kafka, Redis, ActiveMQ, etc, which are increasingly being used in real-world IoT deployments; e.g., SmartSantander [6].

Latency is affected by both the processing delay at the broker and the network link characteristics. Therefore, in single-hop pub/sub systems, managing load at the pub/sub brokers becomes important for ensuring acceptable end-to-end data dissemination latencies. In topic-based pub/sub systems, the load is typically balanced by placing topics on different brokers, and partitioning their connected endpoints (i.e., publishers and subscribers) among these brokers. MultiPub [7] finds optimal placement of topics across cloud data-centers to ensure per-topic 90th percentile latency of data delivery for geographically distributed endpoints. However, MultiPub assumes some local load balancing logic exists at each datacenter site. Kafka supports topic replication to balance load among brokers, but it has to be done manually. Dynamoth [8] balances the load dynamically among brokers through topic partitioning when manually set network thresholds at a broker are exceeded. Moreover, Dynamoth's objective is only to balance the load among brokers, and not with the objective of ensuring QoS of data delivery.

To overcome the limitations in prior work and to support a low-latency and scalable pub/sub data dissemination mechanism, we present a dynamic data-driven approach to load balancing at the IoT pub/sub broker. Our approach uses ma-



(a) Subscription size and publication rate (b) Co-located topic's publication rate (c) Co-located topic's subscription size

Fig. 1: Sensitivity Analysis for Broker Load Modeling

chine learning of a model of the broker load and its impact on system latency. The learned broker load model then provides latency-aware thresholds which we use to dynamically balance the load among broker replicas.

II. Proposed Solution

In order to learn a load model for the brokers, we performed some sensitivity analysis experiments to understand the impact of various pub/sub features on a topic's 90th percentile latency. We have conducted these experiments on our pub/sub broker [9] implemented using the ZMQ[1] sockets library. Our broker implementation is similar to Kafka, where topics are hosted on a flat layer of brokers managed by the Zookeeper[2] coordination service. All experiments were conducted on our private cluster comprising 40 heterogeneous servers running Ubuntu 16.04. Given the limited scale of our test-bed, we have restricted the broker to run on one core to create sufficient load.

As shown in Figure 1a, we observe that the tail latency for a topic increases as the number of connected subscribers increases, for a given publication rate. Tail latency is also impacted by other co-located topics at the broker. Figure 1b shows how the tail latency for a topic is impacted with increasing the rate of publication for another co-located topic. Similarly, Figure 1c shows how the tail latency is affected by increasing the number of connected subscribers on another co-located topic. By using input features such as: i) number of publishers, ii) publication rate, iii) number of subscribers, iv) number of co-located topics, v) total number of publishers on all co-located topics, vi) total number of subscribers on all co-located topics, vii) total publication rate on all co-located topics, viii) cpu and ix) network utilization of the broker, we learn a regression model for a topic's experienced 90th percentile latency under different broker load configurations.

A periodically executing load balancer then uses this learned model to identify which topic's QoS is expected to get violated and take corrective actions either by replicating that topic and partitioning its load on another/new broker or by migrating that topic entirely to another/new broker.

[1]http://zeromq.org/

[2]https://zookeeper.apache.org/

III. Conclusion

To meet data dissemination needs of smart-city applications, we need a dynamically scalable pub/sub system which ensures QoS of data delivery. To this end, we have presented our proposed data-driven solution to learn performance-aware thresholds for balancing the topic load across brokers. We plan to compare our solution to a load-balancing solution [8] based on empirically set network utilization based threshold.

Acknowledgments

This work is supported in part by NSF US Ignite 1531079. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSF.

References

[1] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM Comput. Surv.*, vol. 35, no. 2, pp. 114–131, Jun. 2003. [Online]. Available: http://doi.acm.org/10.1145/857076.857078

[2] N. Carvalho, F. Araujo, and L. Rodrigues, "Scalable qos-based event routing in publish-subscribe systems," in *Network Computing and Applications, Fourth IEEE International Symposium on*. IEEE, 2005, pp. 101–108.

[3] H. Yang, M. Kim, K. Karenos, F. Ye, and H. Lei, "Message-oriented middleware with qos awareness," in *ICSOC/ServiceWave*, 2009.

[4] S. Guo, K. Karenos, M. Kim, H. Lei, and J. Reason, "Delay-cognizant reliable delivery for publish/subscribe overlay networks," in *2011 31st International Conference on Distributed Computing Systems*, June 2011, pp. 403–412.

[5] S. Abdelwahab and B. Hamdaoui, "Fogmq: A message broker system for enabling distributed, internet-scale iot applications over heterogeneous cloud platforms," *arXiv preprint arXiv:1610.00620*, 2016.

[6] L. Sanchez, L. Muñoz, J. A. Galache, P. Sotres, J. R. Santana, V. Gutierrez, R. Ramdhany, A. Gluhak, S. Krco, E. Theodoridis, and D. Pfisterer, "Smartsantander: Iot experimentation over a smart city testbed," *Comput. Netw.*, vol. 61, pp. 217–238, Mar. 2014. [Online]. Available: http://dx.doi.org/10.1016/j.bjp.2013.12.020

[7] J. Gascon-Samson, J. Kienzle, and B. Kemme, "Multipub: Latency and cost-aware global-scale cloud publish/subscribe," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, June 2017, pp. 2075–2082.

[8] J. Gascon-Samson, F. P. Garcia, B. Kemme, and J. Kienzle, "Dynamoth: A scalable pub/sub middleware for latency-constrained applications in the cloud," in *2015 IEEE 35th International Conference on Distributed Computing Systems*, June 2015, pp. 486–496.

[9] K. An, S. Khare, A. Gokhale, and A. Hakiri, "An autonomous and dynamic coordination and discovery service for wide-area peer-to-peer publish/subscribe: Experience paper," in *Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems*, ser. DEBS '17. New York, NY, USA: ACM, 2017, pp. 239–248. [Online]. Available: http://doi.acm.org/10.1145/3093742.3093910