2022 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress



# Distributed MQTT Brokers at Network Edges: A Study on Message Dissemination

Luoyao Hao, Xiao Yu, Tingrui Zhang, Henning Schulzrinne

Department of Computer Science, Columbia University, New York, NY, USA
Email: {lyhao, hgs}@cs.columbia.edu, {xy2437, tz2477}@columbia.edu

***Abstract*—Edge computing attempts to deliver low-latency services by offloading data storage and processing from remote data centers to distributed edge servers near end users, whereas network protocols, designed for centralized management, do not internally scale to distributed edge scenarios. In this paper, we establish the message dissemination support of MQTT, a de facto protocol for Internet of Things, for fully distributed edge networks. We summarize and formulate existing mechanisms, namely publication flooding and subscription flooding, and propose a topic-centric solution called selective subscription forwarding, which forwards subscriptions only when necessary by leveraging the topic containment of MQTT messages and therefore reduces inter-broker traffics. Evaluation results demonstrate that compared with existing solutions, more than 40% subscription traffic can be reduced with the proposed mechanism.**

## I. Introduction

As we step into the era of edge computing, computing resources are no longer solely located on centralized clouds. Instead, a great deal of computation and processing for Internet of Things (IoT) are being offloaded to distributed nodes [1], [2]. Meanwhile, edge computing brings additional service infrastructures, storage resources, and reduced latency [3].

Although network resources and data processing are relatively plausible to be restructured or migrated from clouds to edges, network protocols were not designed with edge computing in mind. For IoT, widely adopted application-level protocols, such as Message Queuing Telemetry Transport (MQTT) [4] and Constrained Application Protocol (CoAP) [5], have been standardized earlier than the concepts of edge computing became popular. MQTT works in the publish/subscribe pattern by applying a broker as an intermediate between publishers and subscribers. The broker forwards published messages to interested subscribers by matching the topics associated with publications and subscriptions. As IoT data are mostly generated and consumed at network edges [6], scaling out MQTT network by deploying brokers on edge computational nodes is a favorable solution to facilitate large-scale IoT systems, reduce propagation delay, and save network bandwidth [3], [7].

For large-scale publish/subscribe systems, one of the fundamental problems is the reliable and efficient message dissemination over distributed brokers. Messages must always be accurately delivered to all interested clients, regardless of which brokers the publishers or subscribers connect to. For example, in the Internet of Vehicle (IoV) network, each vehicle might need to receive alerts on road safety information no matter which roadside infrastructure it is able to communicate with [8]–[10]. In contrast to rich resources for centralized clouds, the bottleneck of scalability is caused by the limited link capacity and computation capability at the edges of the network. However, we are at a very early stage of investigating the distributed MQTT model. Existing message dissemination solutions rely on unconditional flooding and inevitably lose some efficiency [6], [11], since they process messages independently, while the topics of messages, as a matter of fact, are probably mutually contained with each other.

Recent efforts explore the opportunities of scaling MQTT protocol to distributed scenarios by seeking the inspiration from traditional models. Longo et al. [11] present a spanning tree protocol to create a loop-free architecture of distributed brokers. Banno et al. [6], [12] design two message dissemination algorithms and construct an interworking layer to incorporate heterogeneous brokers. Hasenburg et al. [9] introduce a similar flooding-based approach to broadcast messages with geo-distributed brokers. To reduce inter-broker traffic, Detti et al. [13] apply a greedy strategy for load balancing. However, all of them neglect the topic-centric nature of MQTT and pay little attention to the topic hierarchy of messages.

Our goal is to figure out an efficient message dissemination mechanism for distributed MQTT scenarios. We analyze and compare the existing solutions, namely Publication Flooding (PF) and Subscription Flooding (SF). In order to further alleviate traffic congestion caused by unconditional flooding of PF or SF, we leverage the hierarchy of topic-based subscriptions and propose Selective Subscription Forwarding (SSF) that only floods subscriptions when necessary. The key enabler behind this improvement is the quick determination of topic containment relations between incoming and existing subscriptions, which ties to the MQTT specification of topic hierarchy and wildcards. Thus, SSF is a light-weight and topic-centric approach that efficiently forwards messages while saving valuable bandwidth at network edges. Through an experiment involving seven MQTT brokers and simulations of a thousand brokers, we show the characteristics of the three mechanisms in various conditions and demonstrate that SSF can reduce more than 40% subscription traffic and 20% service latency, compared with SF.

We make the following contributions:

978-1-6654-5417-9/22/$31.00 ©2022 IEEE

DOI 10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics55523.2022.00044

- We formulate the message dissemination problem for distributed MQTT networks and review existing approaches.
- We draw on topic containment problem and propose an efficient approach to reduce inter-broker traffics.
- We evaluate the three approaches through emulations and simulations, under a range of network conditions and performance metrics.

The rest of this paper is organized as follows. Section II introduces MQTT protocol and the distributed model. Section III proposes the message dissemination problem and the topic containment problem. Section IV presents existing solutions and proposes our solution based on the topic containment problem. Section V analyzes and evaluates the mechanisms through emulations and simulations for large-scale setups. Section VI discusses relevant concerns.

## II. Background and System Model

We overview MQTT protocol and introduce the distributed network model formed by MQTT brokers.

### A. MQTT Protocol and System

Message Queuing Telemetry Transport (MQTT) [4] works in a topic-based and event-driven publish/subscribe pattern that provides both temporal and spatial decoupling. MQTT is designed as a light-weight application-layer protocol that utilizes a small but variable size of message header. The light-weight and flexible designs entail MQTT a feasible protocol fitting for many resource-constraint scenarios. Fig. 1 shows a typical MQTT system consisting of a broker between multiple publishers and subscribers. The terminologies widely adopted in such systems are listed below.

**Client** can be either a publisher who publishes messages or a subscriber who subscribes to and receives messages.

**Publication** is associated with a specific topic and contains a payload under the topic.

**Subscription** expresses a subscriber's interests. Each subscription is also associated with a specific topic.

**Broker** receives publications and forwards them to the subscribers who previously subscribed to the same topics.

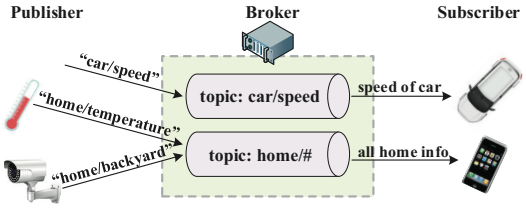



Fig. 1. Topic-based MQTT publish/subscribe system.

MQTT matches interests between publishers and subscribers based on hierarchical topics included in the messages. It uses the forward slash (i.e., "/") as the topic level separator. Each publication and subscription must be associated with an explicit topic. Each broker dynamically maintains a single topic tree, a data structure to store hierarchical topics and corresponding interests by clients. Fig. 2 depicts an example topic tree, where we omit the root of the tree, which is a logical node connected with all first-level nodes globally.

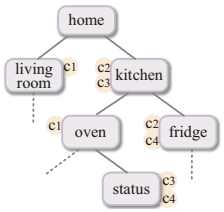



Fig. 2. Example topic tree.

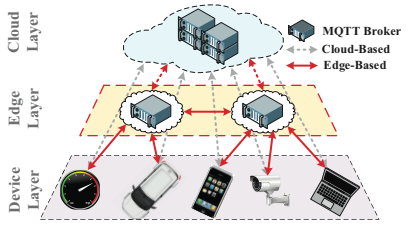



Fig. 3. Three-layer MQTT system architecture.

A client can subscribe to multiple topics in a single request, using single-level wildcard "+" (e.g., "home/kitchen/+/status" indicates status of any kitchen appliance) or multi-level wildcard "#" (e.g., "home/kitchen/oven/#" indicates any information of the oven).

### B. Edge MQTT Brokers

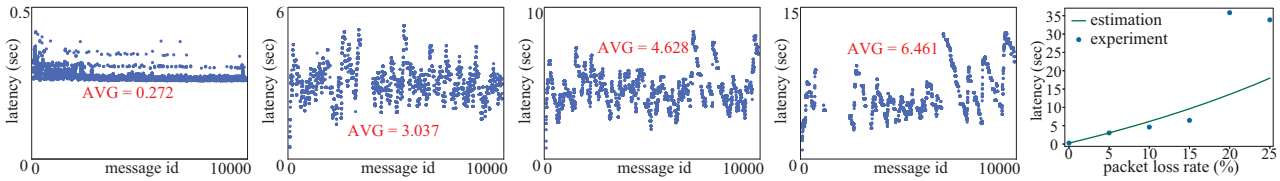
Deploying MQTT brokers to distributed edges is a natural and promising idea, since many subscription interests for IoT are not across an extremely large physical distance and workloads for brokers are moderate enough to be processed at edges [3], [6], [11], [12]. For this purpose, an edge layer encompassing intermediate brokers is introduced between the cloud broker and MQTT clients, as shown in Fig. 3. The major processing and forwarding are managed at the edge layer, and the obligation of the cloud broker is nothing more than basic coordination and forwarding, unless there is a need for centralized performance monitoring or data aggregation. Thus, the MQTT functionalities of the edge layer can be decoupled from the cloud layer [3], [9]. For the rest of this paper, we only focus on the edge layer, where edge brokers are distributed and supposed to work functionally without the cloud. Applications and systems benefited from such deployment appear in road safety [8]–[10], where road infrastructures or Internet of Vehicles are supposed to pass on safety-critical information even the remote server is down, UAV-enabled network consisting of one or more UAVs communicating with a number of mobile sensors and actuators (e.g., for geological prospecting or disaster recovery) [14], messaging applications or geographical applications on edges [3], [15], [16].

Existing implementations on scaling out MQTT, such as Mosquitto Bridge [17] and HiveMQ Cluster [18], are designed for synchronization and replication between multiple brokers on the cloud. Those brokers appear as a single cloud broker to outside clients. Compared with their work, we consider fully distributed edge brokers, and our focus is on message dissemination instead of data replication.

### C. Performance Metrics

*Network traffic.* There are two indicators of network traffic, namely total traffic and maximum traffic of all the links. Involving more brokers for message dissemination introduces more inter-broker traffic. As bandwidth at network edges is usually valuable and limited, reducing network traffic is critical for efficient message dissemination.

*Broker overhead.* The workload of a broker comes from the number of connected brokers and clients, as well as the number of messages in processing. It can be indirectly




(a) no packet loss (b) 5% packet loss (c) 10% packet loss (d) 15% packet loss (e) non-linear growth

Fig. 4. End-to-end delay of MQTT message dissemination with possible packet loss and re-transmission on the overloaded links.

measured by the volume of messages under processing, its CPU utilization or power consumption.

*Service latency.* Unlike a request/response model where the delay is clearly perceivable by users, when a publication is generated is usually unknown to subscribers. However, service latency is significantly important for time-critical applications. We define service latency as the time span from a publisher sending a message to a subscriber receiving the message.

Note that these metrics are correlated. Fig. 4 shows the service latency of message flooding in an experiment of seven MQTT brokers connected one after another. A hundred publishers are connected to one of the end brokers, and each publisher generates one publication per second. As the network congestion increases, more packets need to be re-transmitted, and the service latency increases non-linearly, which fails the prediction of an exponential increase. We observe that a lot of packets do not reach intended recipients over a large duration of time. This result also implies that a flooding process might crash the system when the network traffic is too heavy or any network element is overloaded, which motivates us to design a more efficient approach.

## III. Problem Formulation

We introduce and formulate the message dissemination problem as well as the topic containment problem.

### A. Message Dissemination

To collaborate distributed brokers and deliver services at network edges, the primary need is to correctly disseminate messages. In other words, each publication should be correctly delivered to interested subscribers, no matter which brokers the subscribers are connected with. This is also in accord with the loose coupling and the flexibility features of MQTT protocol, as a client does not need to be aware of where contents come from, neither should they bother to actively choose brokers based on contents.

We assume that the network is loop-free in this paper, as constructing a loop-free topology from a distributed broker network can be addressed using in-band MQTT messages [11].

### B. Subscription Containment Problem

Containment problems, in the complicated forms, can be NP-Complete or co-NP-Complete [19]. For MQTT protocol, fortunately, the complexity is reduced to polynomial time, benefited from the restrictions on the topic hierarchy and conventions. This lays us an opportunity to design an efficient mechanism. More details are given in Section IV-B.

As there is no clear formulation we can directly take, we define the topic containment relation in Definition 1 where the expression is consistent with [19]. Then, we extend Definition 1 and define the MQTT Subscription Containment Problem (MSCP), as formulated in Definition 2, which is to determine whether one subscription contains the other.

**Definition 1** (Topic Containment Relation)**.** *Given two hierarchical MQTT topics $x$ and $y$ (each topic is partitioned by "/", and wildcards are allowed), $x \sqsubseteq y$ denotes the relation that any possible semantic expressed by $x$ is included in $y$.*

**Definition 2** (MQTT Subscription Containment Problem (MSCP))**.** *Given two MQTT subscriptions $s_1$ and $s_2$ that are associated with the topic $t_1$ and $t_2$, respectively, the objective is to figure out whether or not $t_1 \sqsubseteq t_2$, or in reverse, $t_2 \sqsubseteq t_1$.*

As an example shown in Fig. 5, any topic with the same hierarchy containing specific routes to the destination or road information to be alerted is contained in both of the example topics, and additionally, any topic beginning with "NY/World Trade Center" is contained in the lower topic. Topic containment relations yield subscription containment relations associated with those topics.

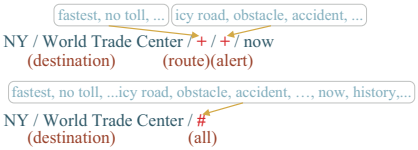



Fig. 5. Example of MQTT topic containment with wildcards.

## IV. Message Dissemination Mechanism

We review existing message dissemination mechanisms for edge MQTT brokers and propose a topic-centric approach.

### A. Publication Flooding and Subscription Flooding

To disseminate messages precisely, existing solutions flood either publications or subscriptions [6], [9], [11], [13], we call them Publication Flooding (PF) and Subscription Flooding (SF), respectively.

Publication flooding is the most straightforward approach. As the name suggests, it floods published messages over the network to make sure that all of the brokers receive every single publication. Subscriptions are processed as usual, as they do not go beyond the responding broker that subscribers directly connect to. As a result, no matter which broker a subscriber connects to, the interested publication will reach the broker anyway and then be forwarded to the subscriber. Fig. 6 shows the basic message flow in a simple topology.

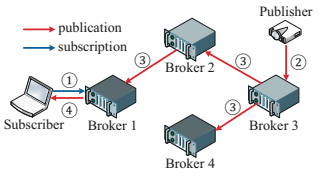



Fig. 6. PF example.

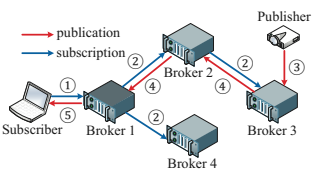
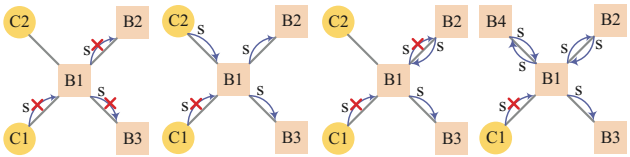



Fig. 7. SF example.

Subscription flooding, in contrast with publication flooding, floods every subscription. A subscription carries interested topics of a subscriber. By flooding the subscription, each broker actually relays the interests to its neighbor brokers, informing them to send messages back when they receive publications associated with the same topics. Each broker now needs to maintain a mapping from those topics to interested neighbors including both clients and connected brokers. Afterwards, if any broker receives a publication, it has a clear understanding on which neighbors need the publication, so the publication is eventually sent along the precise paths to the interested clients. Fig. 7 shows the basic message flow in the same topology.

For subscription flooding, one apparent optimization can be made by not flooding repeated subscriptions which is also implied in [6]. As such optimization can be considered as a special case of topic containment, we use SF in its plain version for the rest of this paper.

Handling unsubscription is a bit tricky. The state of the broker cannot simply remain the same upon receiving an unsubscription, as long as there is a neighbor still subscribes to the topic. As an example in Fig. 8, in fact, when there is a client or at least two neighbor brokers still subscribe to the topic being unsubscribed, the subscription relations between the broker and its neighbor brokers remain unchanged. Otherwise, if no client but only one neighbor broker $b'$ still needs the topic, the subscription from the original broker $b$ to $b'$ can be left out if it exists, since the subscription relation between $b$ and $b'$ becomes unidirectional (i.e., $b$ no longer needs the messages from $b'$ under the topic).




(a) one client (b) two clients (c) one broker (d) two brokers

Fig. 8. Four different unsubscription situations.

### B. Selective Subscription Forwarding

MQTT systems are topic-centric in nature, while existing mechanisms focus on pure publication-based flooding or subscription-based flooding, which turn out to be more generalized but less specialized or efficient for MQTT.

The problem of PF or SF is that they do not pay attention to MQTT hierarchical topics. As a result, a huge number of redundant subscriptions are likely flooded over the entire network, introducing unnecessary traffic congestion and overheads of brokers. However, such deficiency can be avoided by keeping the records of subscriptions and avoiding useless flooded messages, if MSCP is properly addressed. We thereby propose Selective Subscription Forwarding (SSF) where subscribed topics between brokers are never repeated or overlapped. SSF gets rid of useless subscriptions by examining the containment relations of subscriptions before flooding.

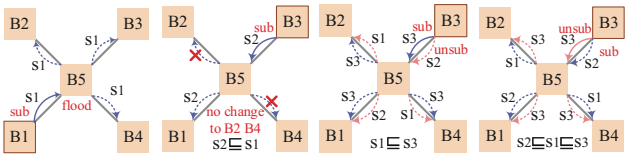
In SSF, when receiving a subscription, the broker traces back to existing subscriptions, and only if the new subscription is not contained in existing ones will the broker floods this subscription. Similarly, when receiving an unsubscription, the broker also checks if it is currently subscribed. If not, it can be simply removed from the records. Otherwise, there might be some subscriptions contained by this one that were not flooded before. Those subscriptions should now substitute for the unsubscription, while taking good care of their containment relations, to make sure interests among brokers are still correctly communicated. Algorithm 1 presents the underlying mechanism of SSF, from the perspective of a broker $b$, where $t_x$ denotes the topic of the message $x$ (publication, subscription, or unsubscription), and $b$ maintains three mappings: $TI_b$ maps each topic to a set of interested neighbor brokers or clients, $HT_b$ maps each of $b$'s next-hop brokers to a set of topics interested by $b$, and $HS_b$ maps each of $b$'s next-hop brokers to a set of topics subscribed by $b$. $HS_b \subseteq HT_b$.

**Algorithm 1:** Selective Subscription Forwarding (SSF)

**if** *receive subscription* $s$ *from broker* $b_i$ *or client* $c_j$ **then**
  store $\{t_s : b_i \text{ or } c_j\}$ to $TI_b$;
  **foreach** *neighbor broker* $b'$ *except* $b_i$ **do**
    store $\{b' : t_s\}$ to $HT_b$ if not exists;
    **if** $\exists t_{s'} \in HS_b(b')$ *s.t.* $t_s \sqsubseteq t_{s'}$ **then** *pass*;
    **foreach** $t_{s'} \in HS_b(b')$ **do**
      **if** $t_{s'} \sqsubseteq t_s$ **then** unsubscribe $s'$;
    add $b'$ to a flooding group $F$, update $HS_b(b')$;
  flood $s$ to brokers in $F$; `/* selective flood */`
**if** *receive publication* $p$ *from broker* $b_i$ *or client* $c_j$ **then**
  **if** $TI_b(t_p) = B' \cup C'$ *and* $B' \cup C' \neq \varnothing$ **then**
    forward $p$ to the brokers in $B'$ and the clients in $C'$;
**if** *receive unsubscription* $us$ *from broker* $b_i$ *or client* $c_j$ **then**
  remove $b_i$ *or* $c_j$ from $TI_b(t_{us})$;
  **if** $TI_b(t_{us}) = \varnothing$ **then**
    **foreach** *neighbor broker* $b'$ *except* $b_i$ **do**
      **if** $t_{us} \in HS_b(b')$ **then**
        find minimum set $T = \{t_k\}$ and $t_k \sqsubseteq t_{us}$ from $HT_b(b')$, *s.t.* $\forall t_m \in HT_b(b')$ && $t_m \sqsubseteq t_{us}$, either $t_m \in T$ or $\exists t_n \in T, t_m \sqsubseteq t_n$;
        subscribe topics in $T$, update $HS_b(b')$;
        add $b'$ to a flooding group $F$;
      remove $t_{us}$ from $HT_b(b')$;
    flood $us$ to brokers in $F$;

An example with five brokers is shown in Fig. 9. In sequence, $B1$ subscribes $s1$, $B3$ subscribes $s2$, $B3$ subscribes $s3$, and finally $B3$ unsubscribes $s3$. In Fig. 9(b), as $B5$ has flooded the subscription $s1$ that contains $s2$, $B5$ only needs to send $s2$ to $B1$, instead of flooding $s2$ to all neighbors. In Fig. 9(c) and Fig. 9(d), as $s2 \sqsubseteq s1 \sqsubseteq s3$, when $s3$ is

subscribed or unsubscribed, $s1$ or $s2$ has to be unsubscribed or subscribed in order to make sure that there is neither redundant nor missing subscriptions between brokers[1].




(a) simple flood (b) $B3$ sub $s2$ (c) $B3$ sub $s3$ (d) $B3$ unsub $s3$

Fig. 9. Example of SSF for a case of five brokers. An arrow with solid line represents the initiative action, and an arrow with dashed line represents the affiliated actions that have to be done to maintain correct subscriptions.

MQTT protocol enforces two constrains on the use of wildcards: (1) only the single-level wildcard "+" and the multi-level wildcard "#" are allowed; (2) the multi-level wildcard "#" must be the last character of a topic, if present. Taking advantages of this simplification, we can solve MSCP in linear time using dynamic programming. In Algorithm 2, we denote four kinds of relations between two topics $t_1$ and $t_2$ as: (1) exactly the same ($t_1 = t_2$); (2) $t_1$ contains $t_2$ ($t_2 \sqsubseteq t_1$); (3) $t_2$ contains $t_1$ ($t_1 \sqsubseteq t_2$); (4) no containment relation ($t_1 \neq t_2$). Obviously, the process runs in $O(\min\{|l_1|, |l_2|\})$, linear time. This result justifies that addressing MSCP would not burden the system, given that we do not usually have a large number of hierarchical levels for the topics. Indeed, it is the efficiency of determining the subscription containment relation that makes SSF a feasible solution.

**Algorithm 2:** MQTT Topic Containment

**Input :** Two topics $t_1$ and $t_2$ with wildcards allowed
**Output:** Integer $r$ indicating the relation between $t_1$ and $t_2$ ($r = 0 \sim 3$ indicates the four relations respectively)

split $t_1$ and $t_2$ to lists $l_1$ and $l_2$ by "/", set $r = 0$;
**while** $i \leq \min\{|l_1|, |l_2|\}$ **do**
  **if** $l_1[i] = l_2[i]$ **then** $i++$;
  **else if** $l_1[i] =$"#" **then** if $r = 2$, return 3; else $r = 1$;
  **else if** $l_2[i] =$"#" **then** if $r = 1$, return 3; else $r = 2$;
  **else if** $l_1[i] =$"+" **then** if $r = 2$, return 3; else $r = 1$;
  **else if** $l_2[i] =$"+" **then** if $r = 1$, return 3; else $r = 2$;
  **else** return 3 /* omit $i++$ above */;

## V. Performance Evaluation

To evaluate SSF and existing solutions, PF [9], [11] and SF [6], we first analyze the situation where no topic containment relation exists. Then, we set up a small-scale testbed consisting of seven MQTT brokers and multiple clients to conduct emulations for small-scale scenarios. Furthermore, we run extensive simulations to evaluate the large-scale network with a thousand brokers.

### A. Disparate Subscription Topics

If no subscription is contained by one another, the behaviors of SF and SSF are the same[2]. To model the client and broker behaviors, we consider two parameters: Interested Broker Ratio (IBR) and Subscription Publication Ratio (SPR). IBR denotes the percentage of brokers in the system which share the same interest. In other words, IBR represents the ratio of brokers that need messages of certain topics. SPR represents the ratio of total size of subscriptions over total size of publications generated by clients. We deduce these two factors stage by stage along with following analysis.

For PF, as brokers only forward publications, the total network traffic over the distributed brokers is:

$$T_{PF} = (m-1)\sum_i (PB_i \times h_i) = (m-1)\sum_i P_i \quad (1)$$

where $m$ is the number of brokers, $h_i$ is the number of published messages under a topic $i$, $PB_i$ is the total size of packets to deliver a published message between two brokers (e.g., for QoS=1, this means the total size of a Publish, a Pubback, and the ACK packets) under the topic $i$. $\sum_i P_i$ represents the total size of transferring all published messages between any two connected brokers in the network.

For SF, brokers first flood subscriptions and then forward publications along the specific paths to the interested subscribers. Similarly, we can get the total network traffic:

$$T_{SF} = \sum_{k=1}^{m}\sum_i P_{ik} + (m-1)\sum_j (SB_j \times h_j) \quad (2)$$

where $h_j$ is the number of subscriptions under a topic $j$, $SB_j$ is the total size of packets to deliver a subscription between two brokers (e.g., for QoS=1, this means the total size of a Subscribe, a Subback, and the ACK packets) under the topic $j$. $P_{ik}$ denotes the total size of transferring all published messages under a topic $i$ through the k-th edge in the topology. $\sum_{k=1}^{m}\sum_i P_{ik}$ adds up published traffics for all possible edges. Derived from Equation 2, we have:

$$T_{SF} = \mathcal{N}\mathcal{P}' + (m-1)\sum_j S_j \quad (3)$$

where each element $n_i$ in matrix $\mathcal{N} = [n_0, n_1, n_2, ...]$ indicates the total number of edges in the network that forward publication under the topic $i$, and $\mathcal{P} = [P_0, P_1, P_2, ...]$. $(m-1)\sum_j S_j$ indicates the total traffic for flooding subscriptions. This equation can be considered topology independent, as all it needs is to count the number of edges.

To conduct a fair evaluation, we adopt the uniform distribution for all publications and all subscriptions, so that $\forall i$, $P = P_i$ and $\forall j$, $S = S_j$, which ensures that the data volume for all messages under any published topic is the same, as well as for subscriptions. In addition, we simplify the case by connecting all subscribers to an end broker, and denote the number of distinct published and subscribed topics as $p_t$ and $s_t$ respectively. Then,

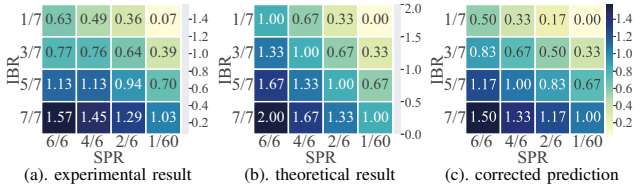
$$T_{SF} = (IBR \times m - 1)(p_t \times P) + (m-1)(s_t \times S) \quad (4)$$

$$\frac{T_{SF}}{T_{PF}} = \frac{IBR \times m - 1}{m-1} + \frac{S}{P}\frac{s_t}{p_t} = \frac{IBR \times m - 1}{m-1} + SPR \quad (5)$$

By far, we get IBR and SPR. IBR $\in [0, 1]$, and usually SPR $\in [0, 1]$, since a subscriber likely receives a series of publications under a single subscribed topic.

[1] we use $\sqsubseteq$ between two subscriptions or two topics interchangeably in this example for easier expression.

[2] Therefore, SF also represents SSF, in this subsection.




(a). experimental result (b). theoretical result (c). corrected prediction

Fig. 10. Comparison of total network traffic between SF and PF ($T_{SF}/T_{PF}$).

To validate our analysis, we set up a testbed consisting of seven MQTT brokers, hosted in Docker contrainers, to emulate a small-scale network. The broker is implemented based on Eclipse Mosquitto [17]. Two more containers are created as publishers and subscribers to deploy client-side programs. We analyze the network packets captured by Wireshark [20].

The experimental result with different settings of IBR and SPR is shown in Fig. 10(a), which exhibits a similar trend as the theoretical estimation generated in Fig. 10(b) by Equation 5. On the top right of Fig. 10(a), SF saves superfluous network traffic, since the total traffic of flooded subscriptions is much less compared to that of publications and only a small fraction of brokers are interested in the messages. On the other hand, PF is advantageous if the total size of subscriptions is large and the publications are interested by a majority of brokers. Note that many scenarios actually fall in relatively low SPR in the long run, as it is rare for a client to subscribe to a topic without receiving further publications.

The theoretical and experimental results slightly differ, because of the message grouping feature of the Mosquitto implementation. Mosquitto brokers opportunistically merge some subscriptions arrived in a short interval into a single packet and forward them altogether. This optimization shifts the dividing line from the diagonal in Fig. 10(a) towards bottom left in Fig. 10(b). Fig. 10(c) is a semi-corrected predicted result considering 50% possibility of message grouping.

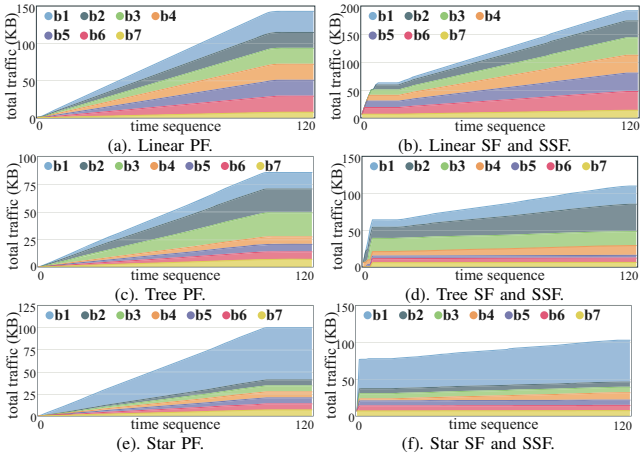
Furthermore, we explore the network behaviour in three basic topologies: linear, binary tree, and star, as depicted in Fig. 11, and the cumulative traffic of each mechanism is shown in Fig. 12. As expected, SF generates much excessive flooding in the liner topology but reduces traffic for the publication period. Theses results actually present the worst case for SF, as we set $SPR = 1$, which is inclined to PF. This result indicates that when publications are more concentrated and intensive, SF can be a wise choice as it offloads the peak traffic during the publication process, and vice versa.


(a). Linear topology (b). Tree topology (c). Star topology

Fig. 11. Three basic topologies for small-scale tests.

### B. Overlapped Subscription Topics

To test the performance in larger-scale networks, we conduct extensive simulations using SimPy [21] framework. We suspect hundreds of brokers should be a fairly large size, therefore to test the scalability in the extreme case, we generate 1000 brokers randomly connected as an acyclic graph, attached by 8000 subscribers and 2000 publishers. Each publisher or




(a). Linear PF. (b). Linear SF and SSF. (c). Tree PF. (d). Tree SF and SSF. (e). Star PF. (f). Star SF and SSF.

Fig. 12. Cumulative traffic of seven brokers connected as different structures. For PF, the publication process is from the beginning. For SF, the publication process starts at time 15, shortly after the subscription process is completed.

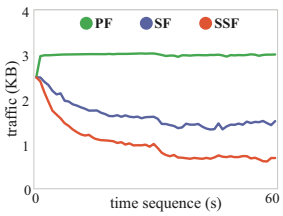



Fig. 13. Output traffic, $\alpha = 0.3$.

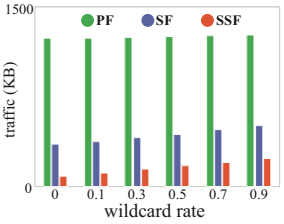
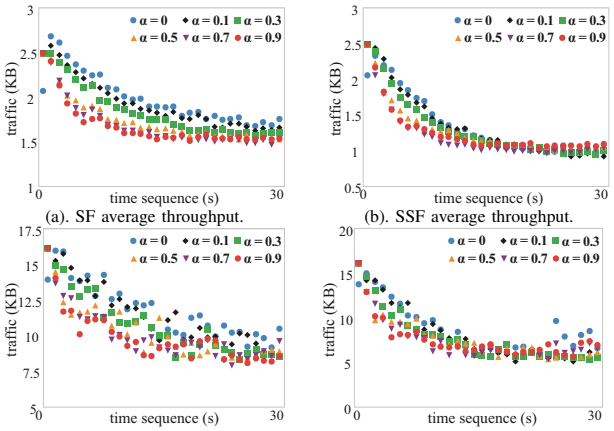



Fig. 14. Total traffic per broker.

subscriber generates a message per second and specifies one of 100 topics randomly selected from a pool of hierarchical topics. Each topic has up to 5 hierarchical levels with 5 unique expressions per level (i.e., $5^5 = 3125$ different topics).

Given that a multi-level wildcard would have more impact on the containment relations, we assume that it appears less frequently than the single-level wildcard. To quantify the probability, we denote the probabilities that a single-level wildcard ("+") and a multi-level wildcard ("#") exist in a topic as $\alpha$ and $\beta$, respectively, varying from no wildcards to high wildcards probabilities, and $\beta = 1/10\alpha$ for testing purpose. Due to space constraints in the graphs, we may only mention $\alpha$ in most of the following figures. The wildcard rates have little impact on PF, as publications are flooded anyway regardless of what topics are. For SF, when wildcards are more likely to appear, more publications are delivered, since subscribers have broader interests indicated by wildcards. For SSF, increasing wildcard rates is expected to have evident effects, since more wildcards should introduce more overlapped topics.

In Fig. 13, we show the average output traffic of brokers. The output traffic for SF and SSF decreases over time, since they have less traffic to flood and subscriptions are more likely contained by previous ones with time elapsing. As the same trend applies to other combinations of $\alpha$ and $\beta$, we only take $\alpha = 0.3$ and $\beta = 0.03$ as an example here. In Fig. 14, we show the mean traffic per broker during the entire message dissemination, where the clients actively send messages during the first 30 seconds and wait until all messages are delivered. As we expected, PF remains constant, while the total traffic




(a). SF average throughput. (b). SSF average throughput. (c). SF highest throughput. (d). SSF highest throughput.

Fig. 15. Average and highest throughput of all brokers.

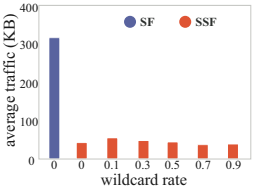



Fig. 16. Subscription per broker.

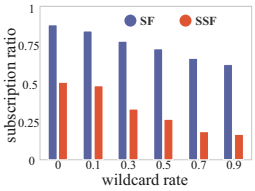
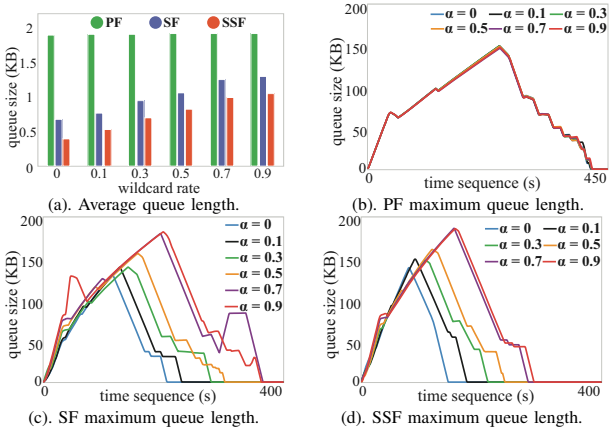
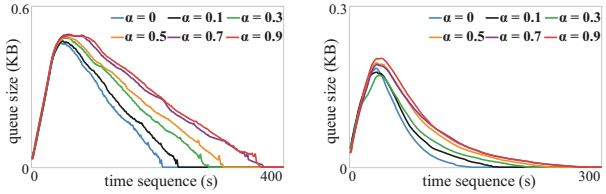



Fig. 17. Proportion of subscriptions.




(a). Average queue length. (b). PF maximum queue length. (c). SF maximum queue length. (d). SSF maximum queue length.

Fig. 18. Average and maximum queue length of brokers.




(a). SF queue size. (b). SSF queue size.

Fig. 19. Average subscriptions inside the broker queues.

for both SF and SSF increase due to the broader interests from subscribers indicated by wildcards. Thus, the increasing number of publications needed to be delivered dominates the increase in network traffic. For the worst case, where $\alpha = 0.9$ and $\beta = 0.09$, SF yields a 60% network traffic reduction, and SSF yields a 81% traffic reduction, compared with PF.

Fig. 15 shows the average and highest throughput in the network. We leave out the results for PF, as PF constantly consumes network bandwidth same as that in Fig. 13. The results for both SF and SSF first drop and then tend to flatten out. This is because some brokers in the network are congested at the very beginning, which become the bottlenecks of the network. For both SF and SSF, plenty of publications should be dropped by brokers during the first period of time, since no client has subscribed to them. With the increase of the wildcard rates, more publications need to be delivered and make the bottlenecks congested earlier, but eventually those publications will be delivered to the subscribers. Compared with SF, SSF converges faster and utilizes lower bandwidth.

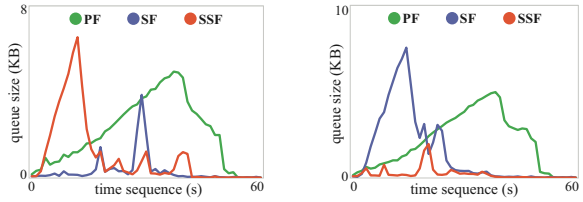
To eliminate the impact of excessive publications brought by the growth of wildcards, we monitor the output subscriptions in the network to compare SF and SSF. Fig. 16 shows that only 12% to 18% subscriptions are necessary to be flooded by SSF. Note that the subscription traffic for SF is constant regardless of the wildcard rate. Fig. 17 presents the ratio of subscription traffic over the total traffic in the network. The ratio decreases as more publications are involved. Compared with SF, SSF achieves 42% to 73% reduction as the wildcard rate increases from 0 to 0.9.

Fig. 18 shows the changes of the queue length. We configure an infinite queue for each broker. The real-time data volume that a broker is processing indicates the broker workload. Fig. 18(a) shows the average queue size when clients actively send messages. As expected, the queue size for PF is not affected by the wildcard rates. In Fig. 18(c) and Fig. 18(d), the queue sizes become larger as the wildcard rate increases. For $\alpha = 0.7$ and $\alpha = 0.9$, the lines are closer to each other, since it is more likely that new subscriptions have been processed in the past, as the wildcard rate increases.

Similarly, to eliminate the impact of excessive publications, we take subscriptions in the queues and show the results in Fig. 19. It turns out that SSF not only reaches lower peak values, but also drops much faster, while SF drops slowly and linearly from the peak values. This is because when a subscription is propagated broker by broker, it cumulates the chance that a broker has already flooded a subscription containing the current one and would not forward further in SSF. Thus, addressing the topic containment problem brings cumulative benefits as the network grows larger.

Although most results favor SF and SSF, we try to explore the potential vulnerability of them during the simulations. We construct the situations where subscribers are located more concentrated, making flooded subscriptions congest part of the network in SF or SSF, even severer than the congestion caused by flooding publications in PF. To better display this situation, we reduce the network to one-tenth of the original scale, so the results in Fig. 20 only reflect the problem but are not numerically comparable with previous values. We find that SF and SSF might cause hotspots exceeding the peak values of PF. These hotspots depend on the distribution of subscriptions in time and space and therefore appear largely at random.

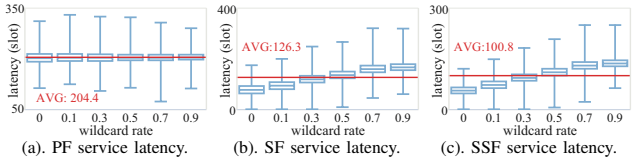
In addition, we mark down the timestamps of sending and receiving each publication, to show the service latency in Fig. 21. We notice that the service latency for each single




(a). $\alpha = 0.3$, $\beta = 0.03$. (b). $\alpha = 0.5$, $\beta = 0.05$.

Fig. 20. Network hotspots indicated by the maximum queue size.

case is highly concentrated around the median, while the highest and lowest values vary. 51% and 20% reductions in service latency are achievable by SSF, compared with PF and SF, respectively. Such reductions benefit from the efficient determination of topic containment relations and also because the network is relatively crowded making many messages queued. If the network is much idle and the processing delay dominates, PF would achieve lower latency.




(a). PF service latency. (b). SF service latency. (c). SSF service latency.

Fig. 21. Service latency of PF, SF, and SSF for different wildcard rates.

## VI. Discussion

*Message dissemination:* Message dissemination for traditional large-scale publish/subscribe networks were mostly on efficient routing and multicast schemes [22], [23]. These efforts mainly focus on the subscriber side only or different architectures that can be hardly generalized to distributed edge MQTT networks [11]. With the increasing interests in IoT protocols, interconnecting and collaborating distributed MQTT brokers becomes an emerging focus [3], [6], [11], [13]. In addition to flooding-based mechanisms, alternative efforts attempt to build UDP extensions or using SDN to support multicast for MQTT [24]–[26].

*Co-existence and unanimous switching:* An interesting idea that deserves further investigation is to enable PF, SF, and SSF at the same time and switch among them. In this case, to deliver end-to-end messages reliably, all of the brokers along the path should behave in the same mode for each topic. Thus, switching between flooding mechanisms might be a gradual process rather than an instantaneous transit. It is possible that some of the brokers have to work in a hybrid mode for a period of time, e.g., run PF for some topics and SF for others. There is no standard way to establish consensus for MQTT scenarios yet, but existing solutions for other scenarios are likely generalized [11].

## VII. Conclusion

To scale MQTT to distributed edges, we study a fundamental problem, message dissemination. We review PF and SF, and propose SSF to selectively forward subscriptions that are not contained by previous ones, enabled by the quick determination of MQTT topic containment problem, which makes SSF feasible to reduce network traffic without introducing much computational overhead. We conduct emulations and simulations to explore the characteristics of those mechanisms under small and large networks. We demonstrate that SSF can achieve more than 40% reduction on subscription traffic and 20% reduction on service latency at the same time.

## Acknowledgement

This work is supported by the National Science Foundation under grant CNS 19-32418.

## References

[1] A. Nußbaum, J. Schütte, L. Hao, H. Schulzrinne, and F. Alt, "Tremble: Transparent emission monitoring with blockchain endorsement," in *IEEE iThings*, 2021, pp. 59–64.
[2] L. Hao and H. Schulzrinne, "Goldie: Harmonization and orchestration towards a global directory for IoT," in *IEEE INFOCOM*, 2021.
[3] M. Veeramanikandan and S. Sankaranarayanan, "Publish/subscribe based multi-tier edge computational model in Internet of Things for latency reduction," *Elsevier JPDC*, vol. 127, pp. 18–27, 2019.
[4] 2019. [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html
[5] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," IETF, RFC 7252, Jun. 2014. [Online]. Available: http://tools.ietf.org/rfc/rfc7252.txt
[6] R. Banno, J. Sun, M. Fujita, S. Takeuchi, and K. Shudo, "Dissemination of edge-heavy data on heterogeneous MQTT brokers," in *IEEE CloudNet*, 2017.
[7] A. E. Redondi, A. Arcia-Moret, and P. Manzoni, "Towards a scaled IoT pub/sub architecture for 5G networks: the case of multiaccess edge computing," in *IEEE WF-IoT*, 2019, pp. 436–441.
[8] https://5gsafeplus.fmi.fi/, 2020.
[9] J. Hasenburg, F. Stanek, F. Tschorsch, and D. Bermbach, "Managing latency and excess data dissemination in fog-based publish/subscribe systems," in *IEEE ICFC*, 2020, pp. 9–16.
[10] T. Limbasiya, D. Das, and S. K. Das, "MComIoV: Secure and energy-efficient message communication protocols for Internet of vehicles," *IEEE/ACM TON*, vol. 29, no. 3, pp. 1349–1361, 2021.
[11] E. Longo, A. E. Redondi, M. Cesana, A. Arcia-Moret, and P. Manzoni, "MQTT-ST: A spanning tree protocol for distributed MQTT brokers," in *IEEE ICC*, 2020.
[12] R. Banno and K. Shudo, "Adaptive topology for scalability and immediacy in distributed publish/subscribe messaging," in *IEEE COMPSAC*, 2020, pp. 575–583.
[13] A. Detti, L. Funari, and N. Blefari-Melazzi, "Sub-linear scalability of MQTT clusters in topic-based publish-subscribe applications," *IEEE TNSM*, vol. 17, no. 3, pp. 1954–1968, 2020.
[14] M. Chen, W. Liang, and S. K. Das, "Data collection utility maximization in wireless sensor networks via efficient determination of UAV hovering locations," in *IEEE PerCom*, 2021.
[15] R. Kawaguchi and M. Bandai, "Edge based MQTT broker architecture for geographical IoT applications," in *IEEE ICOIN*, 2020, pp. 232–235.
[16] L. Hao, V. Naik, and H. Schulzrinne, "Dbac: Directory-based access control for geographically distributed iot systems," in *IEEE INFOCOM*, 2022, pp. 360–369.
[17] https://mosquitto.org/.
[18] https://www.hivemq.com/.
[19] P. Ramanan, "Efficient algorithms for minimizing tree pattern queries," in *ACM SIGMOD*, 2002, pp. 299–309.
[20] https://www.wireshark.org/.
[21] https://simpy.readthedocs.io/en/latest/, 2020.
[22] F. Cao and J. P. Singh, "Efficient event routing in content-based publish-subscribe service networks," in *IEEE INFOCOM*, 2004, pp. 929–940.
[23] A. Majumder, N. Shrivastava, R. Rastogi, and A. Srinivasan, "Scalable content-based routing in pub/sub systems," in *IEEE INFOCOM*, 2009, pp. 567–575.
[24] J.-H. Park, H.-S. Kim, and W.-T. Kim, "DM-MQTT: An efficient MQTT based on SDN multicast for massive IoT communications," *Sensors*, vol. 18, no. 9, p. 3071, 2018.
[25] https://www.oasis-open.org/committees/download.php/66091/MQTT-SN_spec_v1.2.pdf, 2013.
[26] https://mqtt-udp.readthedocs.io/, 2019.