applied sciences

MDPI

Article

Efficient Data Delivery Scheme for Large-Scale Microservices in Distributed Cloud Environment

Van-Nam Pham [1], Md. Delowar Hossain [2,3], Ga-Won Lee [2] and Eui-Nam Huh [2,*]

[1] Faculty of Information Technology, Nha Trang University, Nha Trang 650000, Khanh Hoa, Vietnam
[2] Department of Computer Science and Engineering, Kyung Hee University, Yongin-si 17104, Republic of Korea
[3] Department of Computer Science and Engineering, Hajee Mohammad Danesh Science & Technology University, Dinajpur 5200, Bangladesh
* Correspondence: johnhuh@khu.ac.kr; Tel.: +82-031-201-3778

Abstract: The edge computing paradigm has emerged as a new scope within the domain of the Internet of Things (IoT) by bringing cloud services to the network edge in order to construct distributed architectures. To efficiently deploy latency-sensitive and bandwidth-hungry IoT application services, edge computing paradigms make use of devices on the network periphery that are distributed and resource-constrained. On the other hand, microservice architectures are becoming increasingly popular for developing IoT applications owing to their maintainability and scalability advantages. Providing an efficient communication medium for large-scale microservice-based IoT applications constructed from small and independent services to cooperate to deliver value-added services remains a challenge. This paper introduces an event-driven communication medium that takes advantage of Edge–Cloud publish/subscribe brokers for microservice-based IoT applications at scale. Using the interaction model, the involved microservices can collaborate and exchange data through triggered events flexibly and efficiently without changing their underlying business logic. In the proposed model, edge brokers are grouped according to their similarities in event channels and the proximity of their geolocations, reducing the data delivery latency. Moreover, in the proposed system a technique is designed to construct a broker-based utility matrix with constraints in order to strike a balance between delay, relay traffic, and scalability while arranging brokers into proper clusters for efficient data delivery. Rigorous simulation results prove that the proposed publish/subscribe model can provide an efficient interaction medium for microservice-based IoT applications to collaborate and exchange data with low latency, modest relay traffic, and high scalability at scale.

Keywords: microservices; distributed publish/subscribe broker; implicit collaborative filtering; geolocation awareness; microservices-based IoT applications

check for updates

Citation: Pham, V.-N.; Hossain, M.D.; Lee, G.-W.; Huh, E.-N. Efficient Data Delivery Scheme for Large-Scale Microservices in Distributed Cloud Environment. *Appl. Sci.* **2023**, *13*, 886. https://doi.org/10.3390/app13020886

Academic Editor: Panagiotis G. Asteris

Received: 7 December 2022
Revised: 28 December 2022
Accepted: 6 January 2023
Published: 9 January 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Internet of Things (IoT) technologies have been used in various fields, including smart homes, industrial automation, healthcare, and agriculture, bringing about convenience and enormous successes in improving the quality of life [1]. In IoT systems, people, processes, and IoT devices (e.g., home thermostats, security doorbells, wearables, and smart cars) connect to the internet and to each other. A forecast by Cisco estimates that the number of Machine-To-Machine (M2M) connections will be 14.7 billion by 2023, representing 2.4-fold growth from 6.1 billion in 2018 [2]. In addition, a survey from Business Insider predicted that there would be 41 billion IoT devices in operation by 2027. This estimate suggests that the number of IoT devices will continue to rise in the coming years [3]. As a result, the volume of data produced by these devices is enormous. The International Data Corporation (IDC) forecasts that IoT devices will produce 79.4 zettabytes (ZB) of data in the year 2025 [4]. This raises a huge challenges involving data transmission for large-scale IoT systems with higher scalability and lower latency.

The term "microservices", known as microservice architecture, was coined along with the ideas that arose from best practices across software engineering, Service Oriented Architecture (SOA), and enterprise architecture. In this architectural style, large applications are developed from a suite of small autonomous services [5]. Microservices are loosely coupled, can be deployed independently, and communicate with each other via asynchronous messaging protocols or Remote Procedure Invocation patterns [6,7]. To support highly scalable and ever-changing business models, microservice architectures include several essential technologies, such as virtualization platforms, continuous delivery, M2M communication, port and adapter patterns, and domain-driven design [8]. In addition, microservices share a number of similar goals with IoT, such as lightweight communication, independently deployable software, and independent development techniques and technologies [5]. Therefore, IoT applications might adopt certain design decisions of microservices in order to produce value-added services from different IoT vendors at scale.

The pub/sub paradigm (PSP) is an effective communication solution for large-scale real-time applications. People who participate throughout this communication paradigm can take the roles of either subscribers (consumers), who indicate interest in specific events by obtaining relevant alerts from the publishers through subscriptions, or publishers (producers), who announce and publish incidents in publications [9]. Considering that notices are delivered to an event manager of publishing and requests for subscriptions, the broker immediately locates any relevant subscriptions and contacts any interested parties. Using this mechanism, communicating entities are decoupled in terms of time (there is no need to be active at the same time), space (they do not need to know each other), and synchronization (they do not need to be blocked or waiting while sending/receiving). Therefore, anonymity, many-to-many communication, and asynchronous operations are among the important and adaptable features that this paradigm in distributed systems is able to support [10]. As a result, PSP has been used widely as a communication pattern in event-based middleware for IoT or among microservices in IoT systems [7,11]. With this approach, microservices that construct IoT applications can communicate, collaborate, and exchange data via an event-based middleware provided by a pub/sub system. On the other hand, organizing a distributed pub/sub brokers system to support efficient collaboration and data exchange among microservice-based IoT applications at scale remains challenging.

To deal with these issues, this study designed an Edge–Cloud publish/subscribe broker model to provide an event-based communication tool for microservice-based IoT applications to collaborate and exchange data efficiently. The core contribution of this paper can be summarized as follows:

- A scalable and low-delay data delivery scheme is investigated for microservices-based IoT applications.
- An Edge–Cloud Pub/Sub brokers model is proposed to support collaboration and data exchange among microservices on large-scale IoT applications.
- A method is devised to construct a broker-based utility matrix with constraints (for example, event channel similarities, nearby geolocations, and upper limit node degrees) to cluster the pub/sub brokers efficiently. Moreover, in the proposed model the edge brokers are grouped together according to the similarities in their event channels in order to reduce data delivery latency.
- Finally, extensive experiments were conducted to verify the performance of the proposed scheme with respect to latency, relay traffic, and scalability.

The remainder of this paper is structured as follows. Section 2 discusses previous research on pub/sub systems to facilitate microservice-based IoT applications. The proposed system model and the edge broker clustering procedure are described in Section 3. Section 4 presents the comprehensive simulation results, and the article is summarized in Section 5.

2. Related Works

The three primary variations in this field are topic-based, type-based, and content-based publish/subscribe methods [10]. Because of its minimal run-time overhead [12], the topic-based pub/sub (TBPS) scheme is qwll suited to real-time IoT service applications and organizing events into topics. This section reviews several IoT platforms/systems with microservice architectures and pub/sub schemes to support microservice-based IoT applications.

Event-based middleware is one of the existing solutions for developing IoT applications. In this approach, participants, applications, and components interact through events. The events are typically propagated from the producer to the consumer using a publish/subscribe system [11]. For IoT applications, a microservice-based framework [13] focuses on reusable and loosely coupled services. In their design, one core service coordinates all other microservices provided by the framework, such as geolocation, security, tenanting, and automation. In [14], Khazaei et al. reported a programmable and autonomic IoT platform built on microservices. The platform uses container virtualization and hypervisor-based approaches to deploy IoT applications for various use cases. In addition, there is a trend towards separating services and data in the reported microservice-based IoT architectures [15,16]. Villari et al. introduced the term MicroElements (MELs) [15], which is comprised of MicroData (a piece of information transferred between IoT devices) and MicroServices (implementing particular functionalities). This concept can be applied to dynamically manage services and microservices across Cloud, Edge, and IoT devices in the OSMOSIS architecture.

A microservice-based data-centric End-to-End IoT architecture was suggested in [16]. These microservices are said to be loosely connected, quickly expandable, and deployable in Edge–Cloud virtual environments. On the other hand, scalability and delay difficulties that arise when microservices collaborate and communicate data in large-scale IoT applications, which has not been specifically addressed in previous research. Alam et al. [17] proposed a lightweight virtualization microservices-based architecture utilizing Docker to support IoT applications. By leveraging Docker's orchestration, the given modularity simplifies management and enables dispersed deployments, resulting in a highly dynamic system. In addition, their model offers both fault tolerance and greater modularity. In [18], the authors proposed a microservice deployment problem (MMDP) in edge computing to reduce the communication overhead while simultaneously establishing load balancing among edge nodes. The Deep Q-learning based model was employed in order to solve the deployment problem and acquire the ideal deployment strategy. To create a real-time environmental sensors system with highly scalable applications in a cloud environment, [19] used an architecture based on microservices. Loosely connected and independently deployable services characterize this proposed system architecture. In order to improve hazardous transportation network monitoring, the suggested system is implemented with a number of dispersed sensors and visualization capabilities along with a smart data collector. Moreover, a microservice-based edge server architecture was presented by Alanezi et al. [20] that allowed for sharing of microservices among many Internet of Things applications running simultaneously.

Protocols and techniques for constructing scalable peer-to-peer TBPS systems have been developed to facilitate message delivery in large-scale applications [21–23]. A hybrid TBPS overlay system, as developed by Rahimian et al. [21], provides minimal relay messages between peers and scales effectively with an increasing number of peers and subjects. In this setup, a gossip-based unstructured sampling mechanism is used to group the peers with comparable subscriptions. A structured rendezvous routing method is used to connect these groups. Using topic subscription correlations across peers and dynamically aggregating them on a skewed distributed hash table, Girdzijauskas et al. [23] established a TBPS system that is efficient and scalable in message distribution. A routing approach that considers the underlying routing structure while constructing the multicast trees for locality preservation was presented to reduce the time required to send messages.

Chockler et al. [22] invented SpiderCast, a distributed technology that dynamically organizes peers with similarly associated workloads into an overlay network that can handle TBPS traffic. SpiderCast is the perfect solution because of the number of nodes, with only a few associated subscriptions per node. These techniques can be used to oversee broker-based TBPS systems in which dispersed broker groups mediate the communications between various parties (microservices on IoT devices). In microservice-based IoT systems, however, it is impractical to implement such approaches directly on IoT devices owing to device limitations. In addition, broker/peer locality awareness is lacking, making it difficult to minimize latencies for delay-sensitive applications at scale efficiently when using these approaches.

Establishing scalable broker-based TBPS systems, e.g., deploying pub/sub brokers in the cloud, such as Dynamoth [24], is another interesting method for delivering data in large-scale microservice-based IoT systems. Dynamoth is a scalable pub/sub middleware that can be placed on the cloud, and serves as a data delivery service for both data producers and consumers. Dynamoth is a cloud-based service that allows external customers to use a scalable and load-balanced topic routing service across brokers. On the other hand, long delays in data flow from clients to cloud brokers may cause this solution to not work for delay-sensitive IoT applications. In another approach, distributed brokers may be placed closer to the customers and coordinated with brokers in the cloud to combine the best features of both edge computing and cloud computing. PubSubCoord [25] places brokers in the network periphery while routing servers are placed in the cloud to reduce latency. Despite this, the authors in [25] did not take advantage of topic correlations across neighboring edge brokers to minimize latency. To further enhance system efficiency, it is crucial to offer practical means for Edge/Fog servers to connect and collaborate [26]. Many IoT users will benefit from combining Edge and Cloud IoT services. Therefore, this study aims to supply a coordinated method for distributed pub/sub brokers and techniques for constructing broker overlay networks in order to facilitate massive-scale cooperation and data sharing among microservices in IoT systems.

3. Communication Model for Microservices-Based IoT Applications

This section describes the proposed Publish/Subscribe communication model for microservices-based IoT applications. In the model, IoT applications are constructed from microservices that provide small and autonomous services. These microservices use the publish/subscribe system as an event-based communication tool to collaborate and exchange data between each other efficiently.

3.1. Edge–Cloud Pub/Sub Brokers Model for Microservices-Based IoT Applications

Figure 1 shows the hierarchical Edge–Cloud publish/subscribe system to support collaboration and data exchange among microservices in IoT applications at scale. In the model, the IoT applications are composed of microservices that can be deployed at the IoT devices, IoT gateways, Edge Cloud, and Core Cloud to provide specific IoT functionalities of business logic. The pub/sub system is the common tool for the microservices to interact and collaborate through events. The cloud-based microservices and the cloud pub/sub brokers can be deployed on major cloud provider frameworks such as AWS, Azure, and Google Cloud to take advantage of their high scalability, high availability, and various flexible options.

The model plays as an event-based middleware, which provides a powerful interprocess communication (IPC) tool for microservices-based IoT applications. With pub/sub client modules provided, microservices in IoT devices can collaborate and exchange data with each other with flexibility and efficiency. Communication channels or data that need to be exchanged or transferred among the services are structured as a set of topic channels in the model. In addition, the microservices can be either producers/publishers or consumers/subscribers of related events generated in the topic channels of the pub/sub system. Multiple services can publish events for a topic/event channel, and a service

can subscribe to multiple topic/event channels of interest to it. These are referred to as pub/sub clients in this system. Strategically placed pub/sub brokers in the Edge/Cloud levels establish communication between the pub/sub clients. Each topic/event channel publisher gathers pertinent information or the producer states and transmits them to their pub/sub brokers. The subscribers communicate their interests to the responsible brokers by subscribing to relevant topic/event channels to obtain relevant notifications. The event notifications are then propagated asynchronously from the sending microservices (producers) to the receiving microservices (consumers).

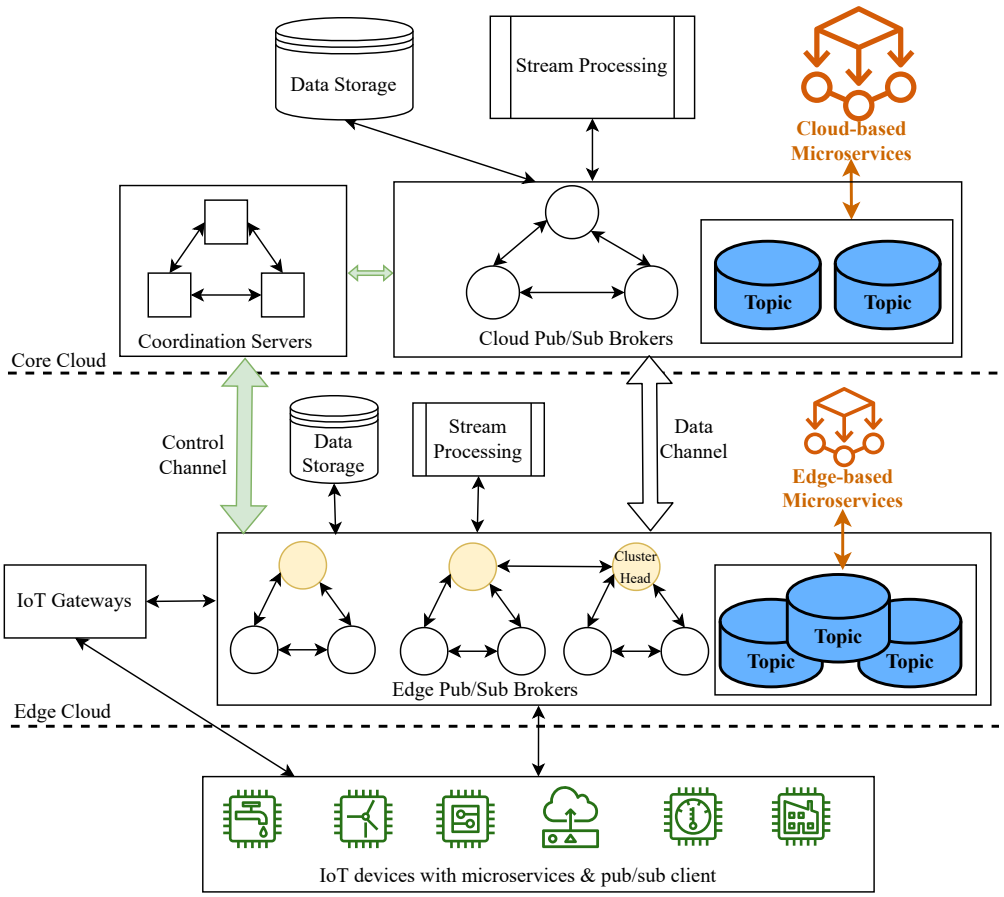


Figure 1. Hierarchical Pub/Sub system for microservices-based IoT applications.

Services, including IoT data aggregation and validation, real-time data processing, and local data storage, are supported by the pub/sub brokers in the Edge–Cloud layer of the model. Data collected or generated by specialized services from IoT devices are gathered and aggregated at IoT gateways or Edge brokers for preprocessing and appropriately instructing the devices to react accordingly. Hence, computation, message, and storage services can be offered to IoT applications with low latency and modest bandwidth consumption. A new technique was devised to organize the edge brokers into proper clusters to reduce data delivery delay further. Furthermore, a cloud-based pub/sub broker infrastructure was implemented to facilitate communication between the various edge broker nodes. In addition to providing services for aggregating data filtered by edge brokers, cloud brokers collect data for deep analytics or long-term storage in the Core Cloud, from which additional valuable insights can be extracted. The coordination part of the TBPS system keeps tabs on each broker topic channel data, helping to ensure that all pub/sub brokers are properly linked and delivering data as quickly as possible.

3.2. Processes for Edge Broker Clustering and Topic Channel Routing

In this subsection, the main parts of the scheme for providing data delivery for microservices-based IoT applications are described. In addition, this subsection reveals how the coordinator clusters edge brokers into proper groups, as shown in Figure 2, and coordinates all pub/sub brokers in the system to perform end-to-end event notifications for the related microservices.

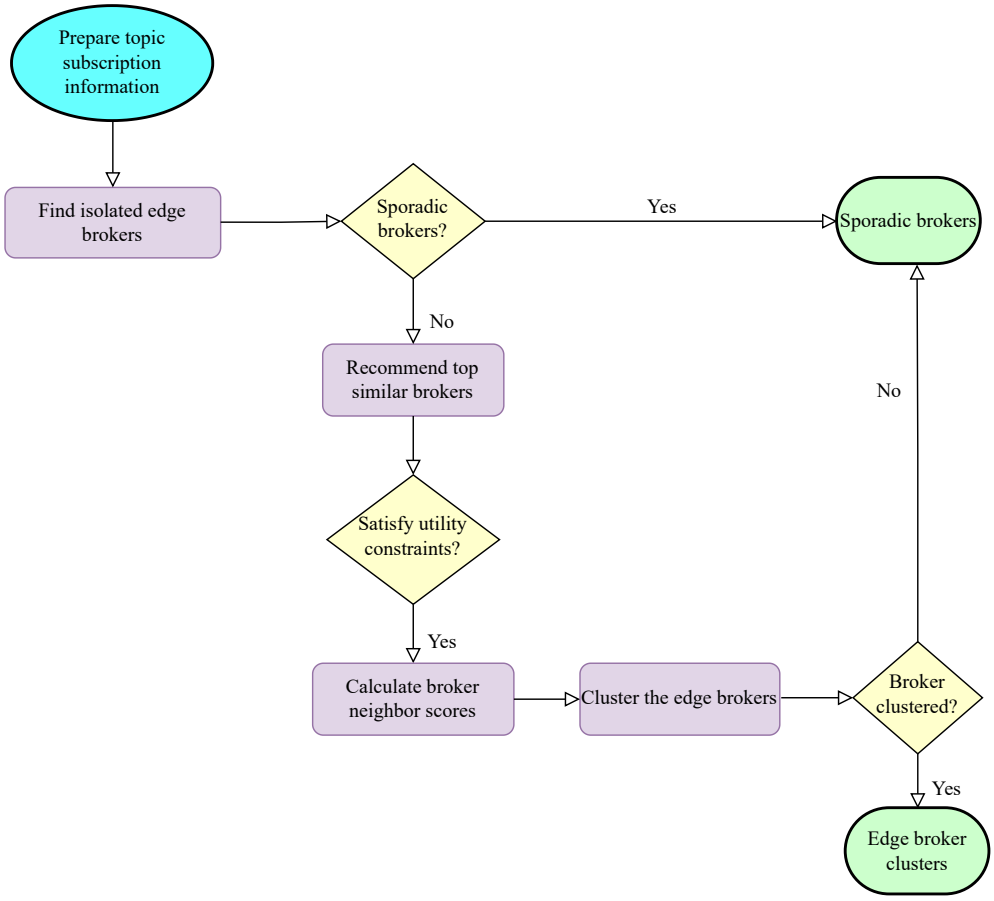


Figure 2. Process for edge brokers clustering.

Consider that at time T the system needs to enhance the performance of event notifications by clustering pub/sub brokers of the edge to proper groups. First, the coordinator entails the topic subscription information of the edge brokers in the system so as to exclude the isolated edge brokers from the clustering process. This is accomplished by applying the HDBSCAN algorithm [27] on the coordinates (latitude, longitude) of the edge brokers to find and exclude the isolated points. These brokers, called sporadic brokers, use the routing service provided by relay cloud brokers for data delivery with other brokers. Second, the model recommends top similar neighbor brokers for each member of the above-selected brokers based on their event channel similarities by applying the implicit feedback collaborative filtering (CF) techniques presented in [28].

Assume that the pub/sub system has a list of m pub/sub brokers and a set of n event channels. The amount of microservice clients of a broker b subscribed to an event channel e is represented by the element s_{be} of the event channel subscription matrix, denoted by $\mathbf{S} \in \mathbb{R}^{m \times n}$, where s_{be} is zero if no microservice client of b broker subscribes to the event channel e. Each broker b is linked to a broker-factors vector $\mathbf{x}_b \in \mathbb{R}^f$, and each event channel e is linked to an event-factors vector $\mathbf{y}_e \in \mathbb{R}^f$. This is done to approximate the matrix $\mathbf{S}$ into two latent factor matrices, $\mathbf{B}^{m \times f}$ (a broker factors matrix) and $\mathbf{E}^{n \times f}$ (an event channel factors matrix), as closely as possible, with the following form: $\mathbf{S} \approx \mathbf{B}\mathbf{E}^T$, where the

approximate value of s_{be} is computed via an inner product, i.e., $\widehat{s_{be}} = \mathbf{x}_b^T \mathbf{y}_e$. The variable p_{be} with the formula $c_{be} = 1 + \alpha p_{be}$ is introduced to signify the likelihood of microservice clients of b broker subscribing to event channel e with confidence level c_{be}. These techniques are adapted from the implicit feedback CF methods provided elsewhere [28]. The following expressions are used to calculate the p_{be} values for a given s_{be}:

$$p_{be} = \begin{cases} 1, & \text{if } s_{be} > 1, \\ 0, & \text{if } s_{be} = 0. \end{cases} \tag{1}$$

The broker and event channel factors are then computed by minimizing the following loss function:

$$\min_{\mathbf{x}*,\mathbf{y}*} \sum_{b,e} c_{be} \left(p_{be} - \mathbf{x}_b^T \mathbf{y}_e \right)^2 + \lambda \left(\sum_b \|\mathbf{x}_b\|^2 + \sum_e \|\mathbf{y}_e\|^2 \right). \tag{2}$$

The term $\lambda \left(\sum_b \|\mathbf{x}_b\|^2 + \sum_e \|\mathbf{y}_e\|^2 \right)$ is used to regularize the model to reduce overfitting. An alternating least squares (ALS) method was used to tackle the optimization problem in (2), in which the factors of the broker are first fixed to optimize the factors of the event channel, then the opposite. As reported previously [28], in order to minimize the broker factors the event channel factors are set as constants and the derivative of (2) is taken to calculate $\mathbf{x}_b$:

$$\mathbf{x}_b = \left(\mathbf{Y}^T \mathbf{Y} + \mathbf{Y}^T (\mathbf{C}^b - \mathbf{I}) \mathbf{Y} + \lambda \mathbf{I} \right)^{-1} \mathbf{Y}^T \mathbf{C}^b \mathbf{p}(b). \tag{3}$$

Alternately, to minimize the event channel factors, the broker factor vectors are made constant and the derivative of (2) is taken to calculate $\mathbf{y}_e$:

$$\mathbf{y}_e = \left(\mathbf{X}^T \mathbf{X} + \mathbf{X}^T (\mathbf{C}^e - \mathbf{I}) \mathbf{X} + \lambda \mathbf{I} \right)^{-1} \mathbf{X}^T \mathbf{C}^e \mathbf{p}(e). \tag{4}$$

The loss function (2) can be minimized by computing (3) and (4) repeatedly in alternating fashion until reaching convergence. This yields two matrices, **B** and **E**, which are referred to as the "broker factors matrix" and "event channel factors matrix", respectively. With these matrices, similar top brokers for each pub/sub broker b can be recommended based on their similarity scores. The similarity between brokers is calculated by computing the dot-product between the broker-factors vectors and their transpose:

$$broker_similarity_scores = \left(\mathbf{B} \cdot \mathbf{B}_b^T \right). \tag{5}$$

Third, with the recommended broker information provided, a broker utility value is calculated for each pair of brokers to build a broker-based utility matrix for further clustering. In this step, further constraints can be applied while considering potential neighbors for each broker, such as nearby geolocations (i.e., $MAX_DISTANCE = 200$ km) and upper limit node degrees (i.e., $UPPER_LIMIT = 30$). A proposed utility function from [29] was adapted to calculate the broker utility value for brokers b_i, b_j:

$$broker_utility(b_i, b_j) = \frac{1}{distance(b_i, b_j)} * \frac{\sum_{e \in evcn(b_i) \cap evcn(b_j)} evrate(e)}{\sum_{e \in evcn(b_i) \cup evcn(b_j)} evrate(e)} \tag{6}$$

In (6), $distance(b_i, b_j)$ is the geographical distance between the two brokers (coordinates) in the sphere, $evcn(b_i)$ denotes the list of event channels of broker b_i, and $evrate(e)$ indicates the transmission rate of event channel e. Finally, the edge brokers are arranged into proper clusters based on the constructed utility matrix by applying the normalized spectral clustering algorithm using a normalized Laplacian (a symmetric matrix), as presented

elsewhere [30,31]. Subsequently, a previous work [32] was used to perform intra-cluster and inter-cluster routing for the broker clusters to link all event channels in the system.

4. Simulation Scenarios and Results

This section delineates the evaluation scenarios of the proposed model and validates the results to support data delivery for microservice-based IoT applications. A discrete event simulation framework, SimPy [33], based on standard Python, was used to employ the pub/sub system. For the pub/sub broker locations, approximately 3000 coordinates of the Starbucks store (longitude and latitude) from [34] were used as a coordinate pool in the evaluation scenarios during the experimental analysis. Using these coordinates for broker clustering it was assumed that, depending on how far apart the brokers were located, the RTT (Round-trip time) delay between them was a random variable with a mean proportional to their distance. A uniform distribution of RTTs between pub/sub clients and cloud brokers was assumed, with the mean delay being the average delay determined from the broker's coordinate pool. Table 1 lists the other common parameter settings for our simulation scenarios. The run time for each simulation was 15 time units (TU), of which five TUs were saved for the bootstrap operations at the beginning and only tallied messages for the remaining 10 TUs.

Table 1. Common parameter settings for evaluation scenarios.

Parameter	Value
RTT between edge-edge, edge-cloud brokers	Proportional to their distance
RTT between cloud brokers	Distributed uniformly between U[1, 2] ms
RTT between clients and edge brokers	Distributed uniformly between U[1, 4] ms
Number of clients per event channel per broker	1 publisher, 10 subscribers
Edge Broker's port rate	10^8 bit per second
Cloud Broker's port rate	10^9 bit per second
Packet size	64 bytes
Subscription distribution	Zipf
Simulation duration	15 time units per setting

Other related schemes were implemented and run with similar simulation setups to obtain performance results and compare with the proposed system, called ECBC_Proposal. Specifically, a cloud-centric publish/subscribe model, known as PSCoreCloud, to which pub/sub brokers are situated in the Core Cloud, was implemented to directly serve message delivery for pub/sub clients [24]. An Edge–Cloud pub/sub broker system inspired by An et al. [25], called PubSubCoord-like, was designed as well. In this system, edge brokers handle data delivery for nearby pub/sub customers. These brokers use the cloud broker routing service to connect all the topic channels. Furthermore, similar setups run in a previous work [35], called PSECTO, were compared with the proposed one. For comparison, three crucial metrics are discussed to assess the mentioned schemes.

- Average Delivery Latency (ADL): This value is determined according to the end-to-end message delivery delay for each scenario on all tested subject channels from producers/publishers to consumers/subscribers.
- Average Forwarding Traffic (AFT): The data packets of the channels must be forwarded to one another by pub/sub brokers to connect joint topic channels. The AFT is calculated as the fraction of forwarded packets relative to the total number of packets delivered by brokers. This value aids in determining how effectively brokers in the system are able to bridge topics.
- Node Degree (ND): The number of connections between brokers required for data sharing on shared subject channels is known as a broker's node degree. This value reflects the effort put in by brokers to maintain topic overlays for delivering the data.

4.1. Assessment of Delays and Forwarding Traffic While Changing the Number of Topics

In this simulation scenario, the number of topics that every pub/sub broker is responsible for was varied in order to observe the performance of the four schemes through the mentioned metrics where appropriate for processing/forwarding. A pool of 1000 event channels or topics was formed and 100 brokers were deployed, with random coordinates chosen from the pool of coordinates. Each broker dealt with 100, 150, and 200 topics for each simulation run period. These topics were chosen in a distribution that followed Zipf's law from a pool of 1000 topics.

Figures 3 and 4 display the respective ADL and AFT values of the comparative pub/sub schemes with varying numbers of topics. The proposed scheme was the best in AFT, and better than PSCoreCloud and PubSubCoord-like in ADL. As expected, the PSCoreCloud scheme experienced the highest (worst) ADL in the experiment, approximately 75.09 ms, because of the long delay when messages were transferred from clients to cloud brokers and returned to the interested clients. The PubSubCoord-like approach experienced high delay as well, approximately 73.36 ms on average, because the edge brokers forwarded joint topic channel data to cloud brokers for the relaying service for other interested brokers. In contrast, the ADL of the proposed scheme increased slightly from 44.54 ms to 45.68 ms and to 49.49 ms, while the brokers' workload was increased from 100 to 150 and 200 topics, respectively. In addition, the AFT of the proposed scheme experienced only a small increment, from 6.26% to 6.67% of the average forwarding traffic among the pub/sub brokers, in the evaluation scenario. These performance metrics demonstrate that the suggested strategy can effectively handle data delivery while altering the workloads of the brokers. Note that the PSCoreCloud AFT results were excluded because all the relay brokers are positioned in the Core Cloud. Even if the AFT of this scheme is good, it does not preserve internetwork traffic significantly because of its inefficient relaying mechanism, as shown above. Thus, the PSCoreCloud approach was omitted from the following comparison scenarios.

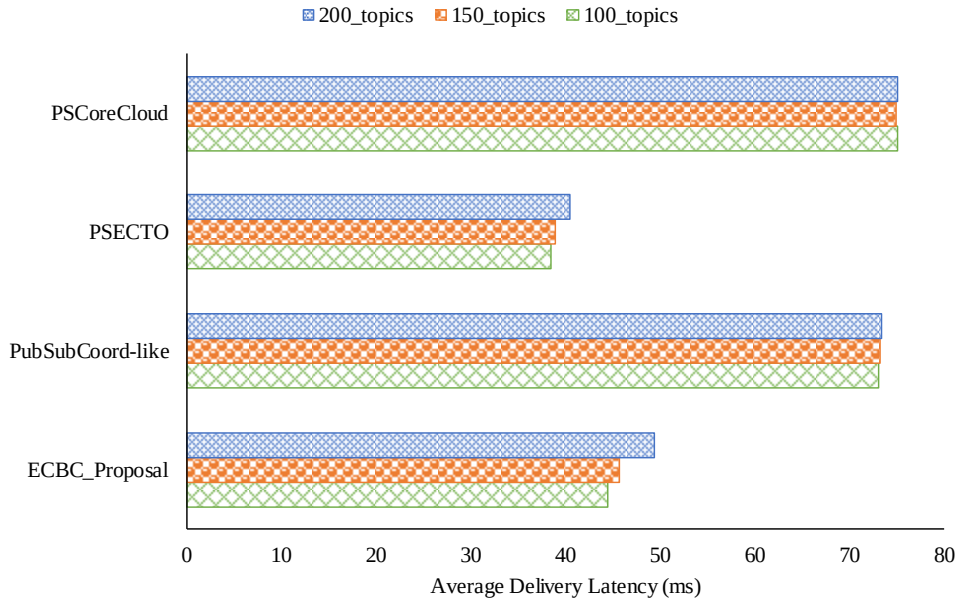


Figure 3. Effect of varying the number of topics on the average delay.

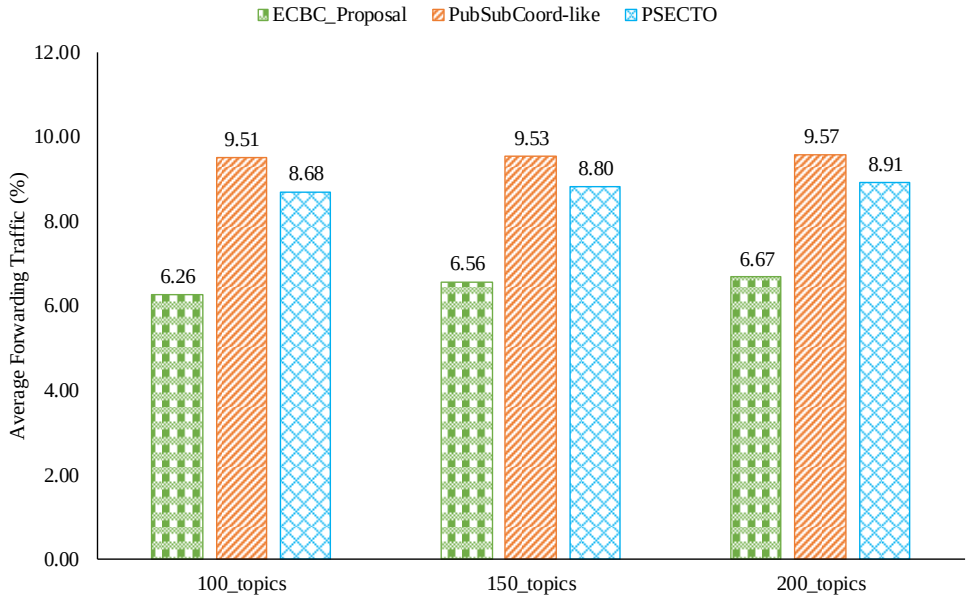


Figure 4. Effect of differing number of topics on forwarding traffic.

4.2. Assessment of the Changing Data Locality Rates

This simulation scenario was implemented with the following settings to evaluate the impact of microservice event channel subscription locality on the data delivery performance of the compared approaches. Two hundred brokers were instantiated, with their coordinates chosen randomly from the coordinate pool. Every broker was in charge of traffic forwarding for 200 event channel topics, with microservice clients subscribing to the channels from a pool of 1000 topics in a distribution that followed Zipf's law. Furthermore, for each broker the local subscription rate (LSR) was changed from 0.2 to 0.8 in steps of 0.2. A higher LSR means that more microservice clients of that broker share local event channels.

Figures 5 and 6 show the simulation results of the scenario in terms of the ADL and AFT, respectively. Although the PSECTO scheme was better than the proposed model in ADL, the scheme achieved the best AFT. When the LSR was increased from 0.2 to 0.8 (indicating that microservice clients are more interested in local events), the ADL of the proposed scheme decreased from 57.04 ms to 43.43 ms, a 23.87% improvement in terms of ADL. In addition, the AFT of the proposed scheme was on average approximately 86.73% that of PubSubCoord-like (8.26% versus 9.52%). The reason for this is that the proposed method can take advantage of common event channels among the microservice clients of involved brokers to reduce the relayed traffic of the corresponding brokers by clustering them into proper groups. In contrast, the PubSubCoord-like scheme cannot take full advantage of the high data locality rates, as it does not cluster similar edge brokers together. This explains why it experienced the worst performance in terms of ADL and AFT in this scenario.

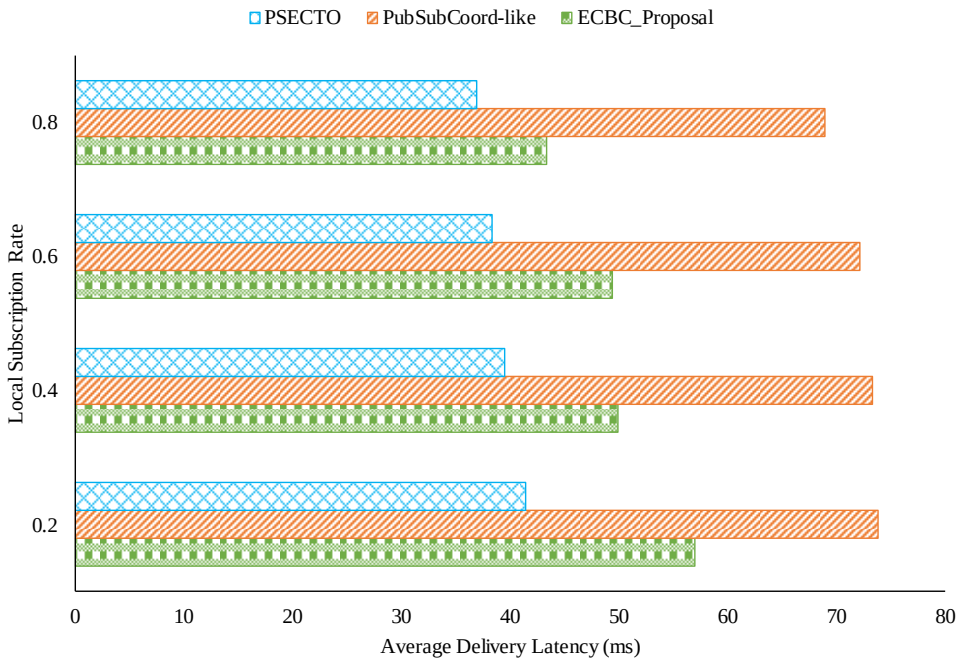


Figure 5. Effect of varying data locality rates on the average delay.

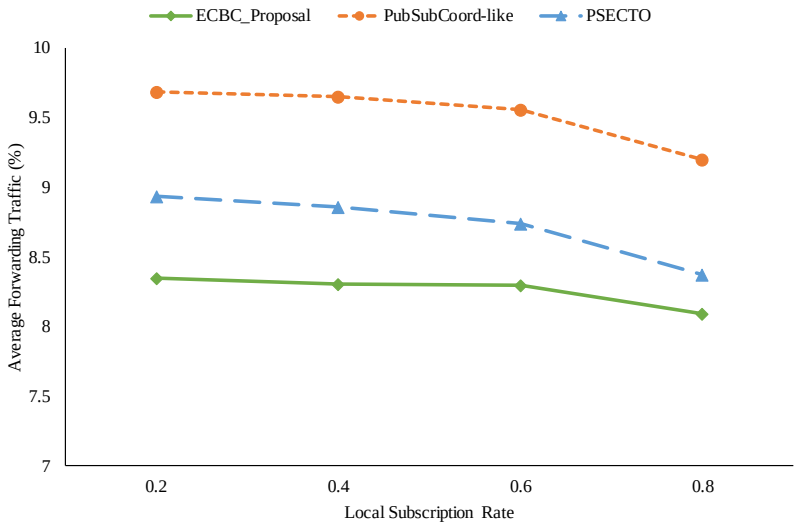


Figure 6. Effect of differing data locality rates on forwarding traffic.

4.3. Assessment of Varying Event Channel Subscription Correlations Using a Multi-Modal Model

Inspired by Wong et al. [36], a multi-modal event channel subscription model was used to examine how the correlations in event subscriptions among brokers affected the efficiency with which data was delivered. From the coordinate pool, 100 brokers were instantiated in this simulation scenario, with their locations chosen at random. Furthermore, a 1000-topic pool was developed to allocate 100 event channel topics for each broker's traffic-forwarding workloads. Specifically, edge brokers' event channel subscription correlations were developed using the multi-modal model to measure their event notification performances as follows:

- There were n categories of the event channel space
- Each broker randomly and uniformly chose p categories from n

- There were fixed values for the ratio of p/n, which were 0.1 and 0.2. Various values of p and n were used to change the correlation levels of the event channel; in particular, with $p = 1$, $n = [10, 5]$ and with $p = 5$, $n = [50, 25]$
- From these p categories, one hundred topics of the event channel were selected for each broker, which were distributed by following Zipf's law

Table 2 and Figure 7 show the average forwarding traffic among brokers and average delivery latencies of the compared schemes in the multi-modal model scenario, respectively. In general, both ADL and AFT values of the three algorithms decreased in the experiment when event channel correlations among brokers increased (from the case of $[p = 5; n = 25]$ to $[p = 1; n = 10]$). The PubSubCoord-like scheme experienced the highest (worst) ADL and AFT values in this evaluation setting, as this approach does not consider clustering edge brokers with high event channel subscription correlations and proximity geolocations to reduce the delivery latency and forward traffic, as in the proposed approach. Although the proposed scheme achieved the best AFT on average, it was only slightly better than the PSECTO scheme, at 8.12% versus 8.39%, respectively. As expected, the PSECTO approach achieved the lowest ADL values in the simulation setup, as it takes advantage of brokers' event channel subscription correlations and proximity locations and uses direct connections between brokers for transferring data. On the other hand, overlay networks for data distribution require considerable work from the brokers, which could lead to scalability issues.

Table 2. AFT (%) in the multi-modal model scenario.

Algorithms / Modal	ECBC_Proposal	PubSubCoord-like	PSECTO
$p = 5; n = 25$	9.08	10.87	8.48
$p = 5; n = 50$	8.88	10.81	8.40
$p = 1; n = 5$	7.98	10.78	8.34
$p = 1; n = 10$	6.52	10.48	8.33

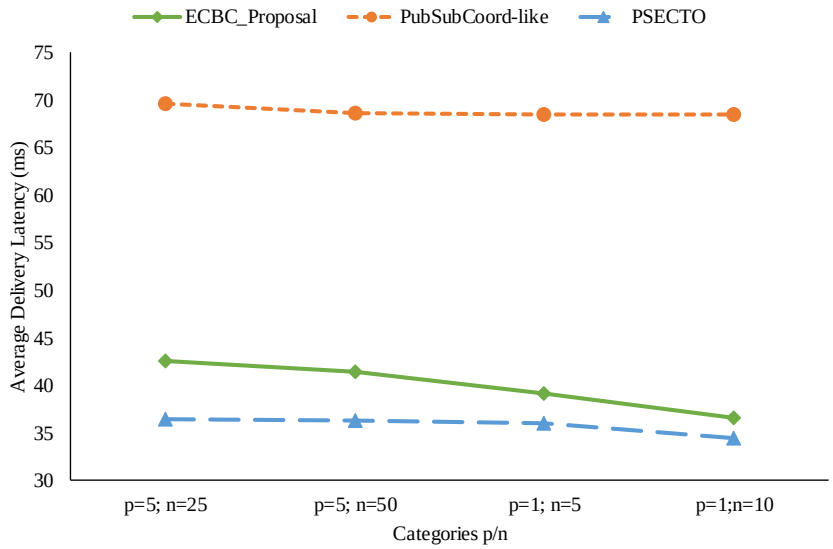


Figure 7. ADL in the multi-modal model scenario.

4.4. Assessment of Scaling Number of Brokers

This subsection evaluated the scalability metric of our proposal in comparison to other similar schemes. In this scenario, the number of brokers was increased from 200 to 500 in steps of 100 for each of the four simulations conducted with each scheme. The capabilities of the comparable pub/sub systems were investigated to support many-to-many communication patterns among the brokers' microservice clients. Each broker was specifically in

charge of 100 event channels, of which microservice clients published or subscribed to 70% on the local level and selected the remaining 30% from the outside brokers. Accordingly, multi-producer and multi-consumer scenarios of event channels were generated in the experiments to measure the ADL, AFT, and ND results of data delivery with the compared schemes. The results are shown in Table 3 and Figures 8 and 9.

Table 3. AFT (%) with scaling number of brokers.

# Brokers \ Algorithms	ECBC_Proposal	PubSubCoord-like	PSECTO
200	8.07	9.44	8.60
300	8.28	9.53	8.75
400	8.39	9.59	8.88
500	8.50	9.61	8.94

As shown in Table 3, our proposal achieved the best outcome in terms of AFT. A small increase in average traffic exchanged among brokers in the system, from 8.07% to 8.50%, was observed when the number of brokers was increased from 200 to 500, respectively. The PubSubCoord-like scheme experienced the highest (worst) percentages of average exchanged traffic, because the edge brokers in this approach have to send data through relay cloud brokers for joint event channels among them.

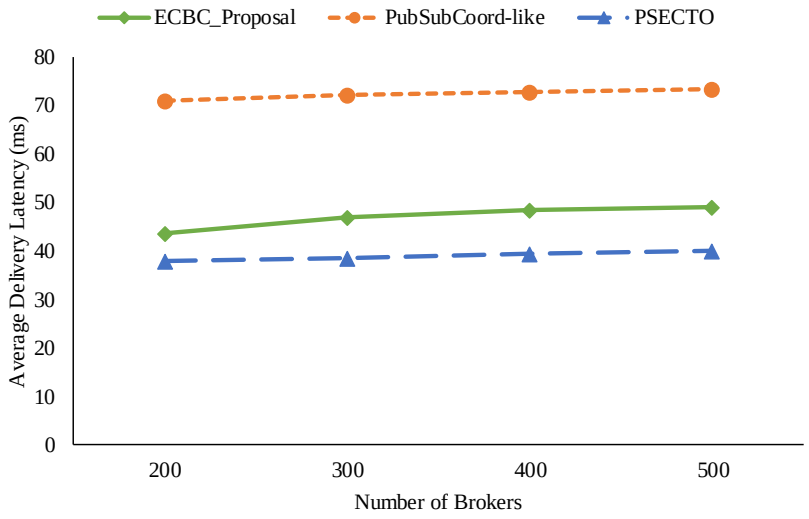


Figure 8. Effect of scaling the number of brokers on average delays.

Figure 8 plots the ADL results of the three approaches measured from the performed experiments. Our proposal has a much lower ADL than that of the PubSubCoord-like scheme. For example, in the simulated example with 500 brokers the ADL value ouf the proposed scheme was 49.1 ms, while that of PubSubCoord-like was 73.41 ms. On average, the ADL of the proposed method was approximately 65% that of PubSubCoord-like. On the other hand, the PSECTO scheme achieved the best (lowest) ADL values in all of the tested cases. The PSECTO technique uses numerous direct connections between edge brokers to convey data for joint event channels among these brokers, which accounts for this remarkable result. Nevertheless, scaling issues may arise due to the high resources required by the pub/sub brokers for keeping the topic overlay networks operational in this approach. Therefore, the average ND values in the experiments were calculated in order to scrutinize the issue further. Figure 9 shows the brokers' average node degrees in the three schemes. The PSECTO scheme experienced the highest average ND values in all the test cases of the evaluation setting, as this approach promotes direct connections between brokers that have joint event channels for data delivery. Specifically, the average broker

node level increased significantly from 147 to 223 as the number of brokers increased from 200 to 500, respectively. Note that the higher average number of connections maintained by each broker to other brokers means that more of the broker's resources are consumed to maintain the connection states. While the proposal achieved much lower average node degree results than the PSECTO model, the PubSubCoord-like scheme achieved the best ND results in all the cases tested. This is because in the PubSubCoord-like scheme the edge brokers need to connect to only a small number of cloud brokers for joint events routing in the whole system. On the other hand, this mechanism causes a much higher delay in end-to-end data delivery for their clients, as reported in the above ADL results. Overall, this concept achieves an optimal harmony between ADL, AFT, and ND metrics, allowing for high scalability to facilitate data delivery in microservices-based IoT systems.

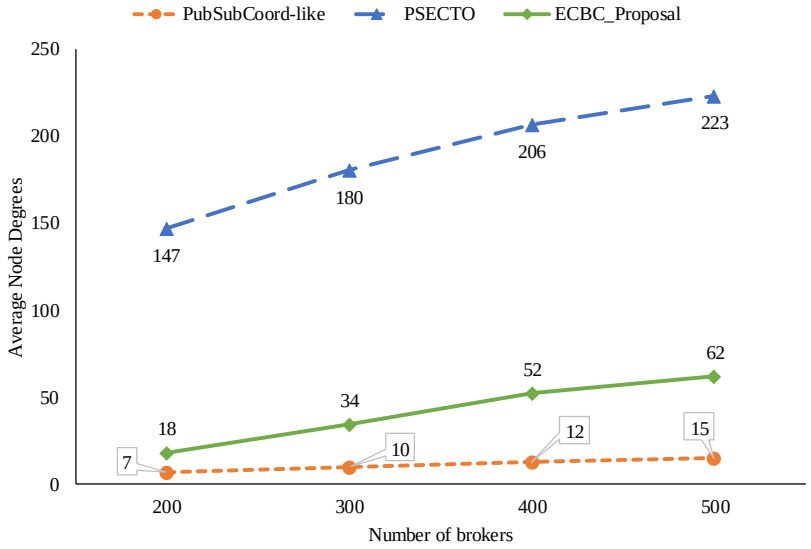


Figure 9. Average node degree with varying number of brokers.

5. Conclusions

This paper has described an effective data delivery model for IoT applications using microservices. In our proposed model, microservices that construct IoT applications collaborate and exchange data with each other via an event-based communication tool provided by the proposed pub/sub system with flexibility and efficiency. As corroborated by experiments, this approach can handle data delivery for large-scale microservice-based IoT applications with low delay, modest relay traffic, and high scalability while maintaining an excellent balance among ADL, AFT, and ND.

Author Contributions: Conceptualization, V.-N.P.; Project administration, E.-N.H.; Software, V.-N.P.; Supervision, E.-N.H.; Writing—original draft, V.-N.P and M.D.H.; Writing—review and editing, V.-N.P., M.D.H., G.-W.L. and E.-N.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by an Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korean Government (MSIT) (No.2202-0-00047, Development of Microservices Development/Operation Platform Technology that Supports Application Service Operation Intelligence).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: [https://data.world/data-hut/starbucks-store-location-data accessed on 8 November 2022].

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

IoT	Internet of Things
M2M	Machine-To-Machine
IDC	International Data Corporation
SOA	Service Oriented Architecture
PSP	Publish/Subscribe Paradigm
TBPS	Topic-based Publish/Subscribe
MEL	MicroELement
IPC	Interprocess Communication
CF	Collaborative Filtering
ALS	Alternating Least Squares
RTT	Round-Trip Time
TU	Time Unit
LSR	Local Subscription Rate

References

1. Al-Fuqaha, A.; Guizani, M.; Mohammadi, M.; Aledhari, M.; Ayyash, M. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 2347–2376. [CrossRef]
2. Cisco Annual Internet Report, 2018–2023. Available online: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html (accessed on 25 February 2022).
3. The Internet of Things 2020. Available online: https://www.businessinsider.com/internet-of-things-report (accessed on 25 February 2022).
4. How You Contribute to Today's Growing DataSphere and Its Enterprise Impact. Available online: https://blogs.idc.com/2019/11/04/how-you-contribute-to-todays-growing-datasphere-and-its-enterprise-impact/ (accessed on 25 February 2022).
5. Butzin, B.; Golatowski, F.; Timmermann, D. Microservices approach for the internet of things. In Proceedings of the 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), Berlin, Germany, 1–6 September 2016; IEEE: Piscataway, NJ, USA; pp. 1–6.
6. Pattern: Microservice Architecture. Available online: https://microservices.io/patterns/microservices.html (accessed on 2 March 2022).
7. Kul, S; Sayar, A. A Survey of Publish/Subscribe Middleware Systems for Microservice Communication. In Proceedings of the 2021 5th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), Ankara, Turkey, 21–23 October 2021; IEEE: Piscataway, NJ, USA; pp. 781–785.
8. Newman, S. *Building Microservices*; O'Reilly Media, Inc.: Neewtown, MA, USA, 2021.
9. Baldoni, R.; Contenti, M.; Virgillito, A. The evolution of publish/subscribe communication systems. In *Future Directions in Distributed Computing*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 137–141.
10. Eugster, P.T.; Felber, P.A.; Guerraoui, R.; Kermarrec, A.M. The many faces of publish/subscribe. *ACM Comput. Surv. CSUR* **2003**, *35*, 114–131. [CrossRef]
11. Razzaque, M.A.; Milojevic-Jevric, M.; Palade, A.; Clarke, S. Middleware for internet of things: A survey. *IEEE Int. Things J.* **2015**, *3*, 70–95. [CrossRef]
12. Shi, Y.; Zhang, Y.; Jacobsen, H.-A.; Tang, L.; Elliott, G.; Zhang, G.; Chen, X.; Chen, J. Using Machine Learning to Provide Reliable Differentiated Services for IoT in SDN-Like Publish/Subscribe Middleware. *Sensors* **2019**, *19*, 1449. [CrossRef] [PubMed]
13. Sun, L.; Li, Y; Memon, R.A. An open IoT framework based on microservices architecture. *China Commun.* **2017**, *14*, 154–162. [CrossRef]
14. Khazaei, H.; Bannazadeh, H; Leon-Garcia, A. End-to-end management of IoT applications. In Proceedings of the 2017 IEEE Conference on Network Softwarization (NetSoft), Bologna, Italy, 3–7 July 2017; IEEE: Piscataway, NJ, USA; pp. 1–3.
15. Villari, M.; Fazio, M.; Dustdar, S.; Rana, O.; Chen, L; Ranjan, R. Software defined membrane: Policy-driven edge and internet of things security. *IEEE Cloud Comput.* **2017**, *4*, 92–99. [CrossRef]
16. Datta, S.K; Bonnet, C. Next-generation, data centric and end-to-end iot architecture based on microservices. In Proceedings of the 2018 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), Jeju, Republic of Korea, 24–26 June 2018; IEEE: Piscataway, NJ, USA; pp. 206–212.
17. Alam, M.; Rufino, J.; Ferreira, J.; Ahmed, S. H.; Shah, N.; Chen, Y. Orchestration of microservices for IoT using Docker and edge computing. *IEEE Commun. Mag.* **2018**, *56*, 118–123.
18. Lv, W.; Wang, Q.; Yang, P.; Ding, Y.; Yi, B.; Wang, Z.; Lin, C. Microservice Deployment in Edge Computing Based on Deep Q Learning. *IEEE Trans. Parallel Distrib. Syst.* **2022**, *33*, 2968–2978.
19. Cherradi, G.; Bouziri, A.E.; Boulmakoul, A.; Zeitouni, K. Real-time microservices based environmental sensors system for Hazmat transportation networks monitoring. *Transp. Res. Procedia* **2017**, *27*, 873–880.

20. Alanezi, K.; Mishra, S. Utilizing Microservices Architecture for Enhanced Service Sharing in IoT Edge Environments. *IEEE Access* **2022**, *10*, 90034–90044.
21. Rahimian, F.; Girdzijauskas, S.; Payberah, A.H.; Haridi, S. Vitis: A Gossip-based Hybrid Overlay for Internet-scale Publish/Subscribe Enabling Rendezvous Routing in Unstructured Overlay Networks. In Proceedings of the 2011 IEEE International Parallel & Distributed Processing Symposium, Anchorage, AK, USA, 16–20 May 2011; pp. 746–757. [CrossRef]
22. Chockler, G.; Melamed, R.; Tock ,Y.; Vitenberg, R. Spidercast: A scalable interest-aware overlay for topic-based pub/sub communication. In Proceedings of the 2007 Inaugural International Conference on Distributed Event-Based Systems, Toronto, ON, Canada, 20–22 June 2007; ACM: New York, NY, USA, 2007; pp. 14–25. [CrossRef]
23. Girdzijauskas, S.; Chockler, G.; Vigfusson, Y.; Tock, Y.; Melamed, R. Magnet: Practical subscription clustering for internet-scale publish/subscribe. In Proceedings of the 4th ACM International Conference on Distributed Event-Based Systems (DEBS), Cambridge, UK, 12–15 July 2010. [CrossRef]
24. Gascon-Samson, J.; Garcia, F.; Kemme, B.; Kienzle, J. Dynamoth: A Scalable Pub/Sub Middleware for Latency-Constrained Applications in the Cloud. In Proceedings of the 2015 IEEE 35th International Conference on Distributed Computing Systems, Columbus, OH, USA, 29 June–2 July 2015; pp. 486–496. [CrossRef]
25. An, K.; Khare, S.; Gokhale, A.; Hakiri, A. 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, Barcelona, Spain, 19–23 June 2017; pp. 239–248.
26. Atlam, H.F.; Walters, R.J.; Wills, G.B. Fog computing and the internet of things: A review. *Big Data Cognit. Comput.* **2018**, *2*, 10. [CrossRef]
27. Campello, R.J.; Moulavi, D.; Sander, J. Density-based clustering based on hierarchical density estimates. In Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, Gold Coast, Australia, 14–17 April 2013.
28. Hu, Y.; Koren, Y.; Volinsky, C. Collaborative filtering for implicit feedback datasets. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, Pisa, Italy, 15–19 December 2008; pp. 263–272. [CrossRef]
29. Rahimian, F.; Huu, T.L.N.; Girdzijauskas, S. Locality-awareness in a peer-to-peer publish/subscribe network. In *IFIP International Conference on Distributed Applications and Interoperable Systems*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 45–58.
30. Chung, F.R.; Graham, F.C. Spectral graph theory. *Am. Math. Soc.* **1997**, *92*, 109–121. [CrossRef]
31. Von Luxburg, U. A tutorial on spectral clustering. *Stat. Comput.* **2007**, *17*, 395–416.
32. Pham, V.-N.; Lee, G.-W.; Nguyen, V.; Huh, E.-N. Efficient Solution for Large-Scale IoT Applications with Proactive Edge-Cloud Publish/Subscribe Brokers Clustering. *Sensors* **2021**, *21*, 8232. [CrossRef]
33. SimPy. Available online: https://simpy.readthedocs.io/en/latest/ (accessed on 11 August 2021).
34. Starbucks Store Location Data. Available online: https://data.world/data-hut/starbucks-store-location-data (accessed on 11 August 2021). [CrossRef]
35. Pham, V.-N.; Nguyen, V.; Nguyen, T.D.T.; Huh, E.-N. Efficient Edge-Cloud Publish/Subscribe Broker Overlay Networks to Support Latency-Sensitive Wide-Scale IoT Applications. *Symmetry* **2020**, *12*, 3. [CrossRef]
36. Wong, T.; Katz, R.; Mccanne, S. An evaluation of preference clustering in large-scale multicast applications. In Proceedings of the IEEE INFOCOM, Tel Aviv, Israel, 26–30 March 2000; pp. 451–460.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.