# Publish/Subscribe Mechanism for IoT: A Survey of Event Matching Algorithms and Open Research Challenges

**Satvik Patel, Sunil Jardosh, Ashwin Makwana and Amit Thakkar**

**Abstract** The number of sensors getting deployed around the world is increasing due to emergence of Internet of Things. It provides advanced connectivity and communication between devices which goes beyond machine-to-machine communication. Huge amount of data is expected to be generated from different locations that will be aggregated, processed and forwarded very quickly. Publish/Subscribe mechanism is powerful way to allow IoT devices to connect and communicate with each other. One of the major bottlenecks in using Publish/Subscribe systems is the efficiency of filtering incoming message. This is a very challenging problem because in a Publish/Subscribe system the number of subscriptions can be very large. There are quite a few event matching algorithms proposed in the literature to improve its efficiency. The aim of this research paper is to study and analyze how existing approaches ensure fundamental event matching requirements and discuss the open challenges and future work in the area.

**Keywords** Internet of things · Event matching · Event filtering · Publish/subscribe · Communication paradigm · Distributed system

S. Patel (✉)
P.I.E.T., Parul University, Vadodara, Gujarat, India
e-mail: satvik.patel@paruluniversity.ac.in

S. Jardosh
Progress Software Development Pvt. Ltd., iLab's Centre, Madhapur, Hyderabad, India
e-mail: sjardosh@progress.com

A. Makwana · A. Thakkar
C.S.P.I.T., CHARUSAT University, Changa, Gujarat, India
e-mail: ashwinmakwana.ce@charusat.ac.in

A. Thakkar
e-mail: amitthakkar.it@charusat.ac.in

N. Modi et al. (eds.), *Proceedings of International Conference on Communication and Networks*, Advances in Intelligent Systems and Computing 508,
DOI 10.1007/978-981-10-2750-5_30

## 1 Introduction

The IoT (Internet of Things) is network of physical objects with advanced connectivity and communication which helps achieve greater value and service [1]. Numbers of devices getting connected in such distributed systems are increasing day by day. Publish/Subscribe mechanism is powerful way to allow IoT devices to connect and communicate with each other [2]. Ability of Publish/Subscribe to supports dynamic, anonymous, many-to-many and asynchronous communication plays very important role in providing high scalability in distributed environment. The loose coupling can be classified into following three ways:

1. Space Decoupling: Publisher can share information with subscriber without knowing each other.
2. Time Decoupling: Publishers and Subscribers are not required to be actively participating in the interaction at the same time.
3. Synchronization Decoupling: Publishers are not blocked while producing the events. Occurrence of the events can be asynchronously notified to the Subscribers.

Each participant in a Publish/Subscribe based communication system can either act as a publisher or a subscriber of the information. Publisher produces information which will be consumed by subscribers. This information is denoted by the term "Event" and the act of delivering it by the term "Notification". Publisher does not directly send information to the corresponding subscribers, but it is sent indirectly according to the content of the notification. A subscriber expresses its interest for specific events by issuing subscriptions. Later on subscriber will be asynchronously notified for all events, produced by any publisher, that match their subscription [3]. Subscribers may also unsubscribe their interest for particular event (Fig. 1).

Efficient filtering of incoming messages is one of the major concerns in using Publish/Subscribe systems for large scale distributed systems [4]. This is a very challenging problem because in a Publish/Subscribe system the number of

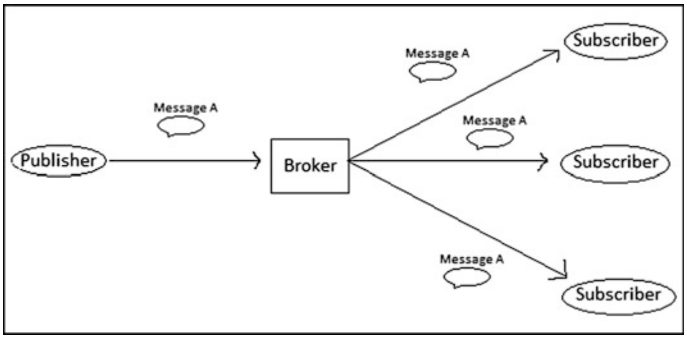

**Fig. 1** Publish/subscribe model

subscriptions can grow very large. Considering high event arrival rates and large volume of subscriptions an efficient event matching algorithm is highly desirable. There are quite a few event matching algorithms proposed in the literature to improve the performance [5–10].

The outline of the contributions of this paper is as following.

1. We provide an overview of some of the key event matching techniques presented in the literature and provide a summary of related research work, open challenges and future work.
2. We present need for new and/or improved event matching mechanism for Publish/Subscribe system.

The rest of the paper is organized as follows. Section 2 depicts overview about event matching process. In Sect. 3 we discuss about existing work done in the area of event matching. Section 4 finally concludes the paper providing insight on the future work in the area.

## 2 Overview of Event Matching

As seen in above section Publish/Subscribe is an important communication paradigm used in IoT. Due to its ability to provide dynamic, anonymous, many-to-many and asynchronous communication between the sender and the receiver it is gaining increasing attention in large scale distributed system [11].

Event matching is one of the key aspects of Publish/Subscribe system [12]. Event matching is a process of finding set of subscriptions from large volume of subscriptions that successfully match against the occurring events. Event matching in Publish/Subscribe system is a tough challenge as these volumes of subscriptions and event arrival rate could be very high for the applications [13].

In Publish/Subscribe system the subscriptions are represented in the form of boolean expressions [14]. This offers greater flexibility to user to represent their interests in the events. Key terms involved in event matching are as follows:

1. Predicate: It is the primary unit in the boolean expression. It is a triplet consisting of an attribute A, an operator OP and an operand OD. It takes an input value X and outputs a boolean value indicative of whether or not the operator constraint is satisfied or not.

   $P^{(A,\ OP,\ OD)}(X) \rightarrow \{\text{True, False}\}$, where P is a predicate and X is a input value.

2. Boolean Expression/Subscription: A boolean expression is combination of predicates formed by using conjunction (AND) and/or disjunction (OR). A subscription S (boolean expression) can be defined over n predicates are follows:

**Table 1** Subscription list

| Subscription ID | Query |
|---|---|
| S1 | (A = True and B = True) |
| S2 | (A = True and B = True) or (C = True) |
| S3 | (A = True and B = True and C = False) |
| S4 | (A = True or D = True) |
| S5 | (A = False or B = False) and D = False |

S = P1(A, OP, OD)(X) AND P2(A, OP, OD)(X) AND P3(A, OP, OD)(X) AND…AND Pn(A, OP, OD)(X), where S is a subscription and P1…. Pn are predicates. Table 1 shows the list of five subscriptions (S1 to S5) where A, B, C and D are the attributes.

3. Event: An event is the attribute-value pair published by the publishers. E: (A = v), where E is an event, A is an attribute and v is value for an attribute A.

   Example: E1: {T = 50}, where E1 is the event occurred with attribute T with a value of 50.
4. Boolean Expression/Subscription Match/Event Match: We can say successful subscription match when predicates of a subscription have corresponding attribute-value pair occurred in the form of events.

   Example: Suppose event E1: D = True occurs then, with the help of event matching algorithm we can come to a conclusion that subscription S4 is successfully matched and its corresponding subscriber needs to be notified for this event.

With this basic overview about Publish/Subscribe system and Event Matching process, we discuss its related work and open challenges in the next section.

## 3 Related Work and Analysis

As discussed above one of the major bottlenecks in using Publish/Subscribe systems for large scale distributed systems is the efficiency of filtering incoming messages. Also this problem is very challenging in a Publish/Subscribe system as the number of subscriptions can be very large and event arrival rate is also very high. There are quite a few event matching algorithms proposed in the literature to improve its efficiency.

Lot of work has also been carried out in the area of indexing to efficiently identify matching subscriptions [5–10]. The subscription database is divided into subsets of predicates using any of the existing hashing technique and each predicate subset is organized using inverted list data structure. For every attribute-value pair p in the incoming event, appropriate inverted index lists are searched to identify predicates of subscription that match p, and a counting mechanism is used to determine matching subscriptions for occurring events.

It is a tough challenge to efficiently indexing Disjunctive Normal Form (DNF) and Conjunctive Normal Form (CNF) of Boolean expressions over a high-dimensional multi-valued attribute space. The objective is to quickly find the set of complex boolean expressions (including NOTs) that evaluate to true for a given assignment. Inverted list data structure can be used to efficiently match complex boolean expressions [9]. Proposed design talks about working with DNF and CNF boolean expressions with different set of algorithms. Working with combination of DNF and CNF boolean expression could be tough here and can be taken up as an extension to this work.

Many Publish/Subscribe systems are capable of handling large volume of subscriptions and high event arrival rate. In many cases these system does not prove to be effective with high dimensional and sparse database (e.g. E-Commerce Database). An efficient in-memory index (OpIndex) is scalable to the volume and updating of the subscriptions. It is also scalable to the arrival rate of events and the variety of attributes that can be subscribed. OpIndex uses a two-level partitioning scheme to organize the subscription predicates into disjoint subsets. Each of these subsets is independently and efficiently indexed to minimize the number of corresponding subscriptions accessed for event matching. In this way, OpIndex design is a highly efficient and extensible approach for subscription matching which can support complex predicate matching operators [5]. One of the drawbacks here is in its inability to handle temporal events (events of our interest occurring at different time stamp). One could extend the current work to efficiently handle the temporal events.

Sometimes users are interested in receiving up-to-date geo-textual objects according to their requested location and interest. The continuous queries issued by user could be very large, which can post challenge in efficiently matching them against geo-textual objects. A system called SOPS (Spatial-Keyword Publish/ Subscribe System) is capable of efficiently processing spatial keyword continuous queries. It uses IQ-tree (Inverted File Quad Tree) data structure to efficiently match events [8]. Queries are organized in IQ-tree and geo-textual objects are matched using this index. For each new object the IQ-tree is traversed to find the queries that have object as the result. One of the drawbacks here is that the spatial information is always prioritized while index construction not considering the distribution pattern of the queries. One can extend the current work to consider distribution pattern of the queries while index construction and traversal.

The amount of geo-spatial data being generated is increasing day by day at an unprecedented scale. Users want them to be notified of interesting geo-textual objects during a period of time. Matching stream of incoming Boolean Range Continuous (BRC) queries over a stream of incoming geo-textual objects in real time can post challenge as the number of continuous queries issued by users could be very large. An index structure called IQ-tree (Inverted File Quad-tree) can be used for matching the queries with the incoming geo-textual objects. The IQ-tree integrates the Quad-tree for organizing the spatial information and the inverted file for organizing the keyword expression of the BRC queries. The IQ-tree is nothing but a Quad-tree extended with inverted files. Each node in the IQ-tree is associated with an inverted file which organizes the keyword expression of the BRC queries that are associated with the node [7]. One of the drawbacks here is that the spatial information is always prioritized while index construction not considering the distribution pattern of the queries. One can extend the current work to consider distribution pattern of the queries while index construction and traversal.

Efficiently processing continuous spatial-keyword queries over geo-textual streams is a tough challenge. Simple variant of inverted index data structure does not perform well in such cases. R-tree and IQ-tree are two indexing structure available to handle geo-textual information. They suffer from fundamental drawback like "the spatial factor is always prioritized during the index construction not considering the keyword distribution of the query set". An index structure called Adaptive spatial-textual Partition Tree (AP-Tree) can be used to effectively organizes continuous spatial-keyword queries and overcome the drawback of spatial factor always first in IQ-tree and R-tree [10]. This method will adaptively select either keyword partition or spatial partition depending on the evaluation of the cost model.

Location aware Publish/Subscribe systems have become widely available on mobile devices. Since the message and the subscription contains both the location and textual information, a high performance location aware systems are required to deliver publisher's message to relevant subscribers. We can use R-tree based index structure by integrating textual description into R-tree nodes to achieve the same [6].

Solutions proposed in the literature involve usage of the following techniques Inverted List, IQ-tree, R-tree, OpIndex and AP-tree. Below Table 2 enlists the benefits and limitation of the techniques used in the proposed solution of the literature.

Solutions discussed above suffer from one common problem that is assumption of assignment (group of events) generation. But practically assignment generation does not seem possible due to following reasons.

1. Group of events cannot be done at gateway node as the very basic feature named loose-coupling of publish/subscribe will be disturbed. Also not all data required by an application gets generated under single gateway node.
2. Group of events cannot be done at central node as well, as in real life the events occur asynchronous and independent from each other. Also events need not come together due to communication delay as well.

So in practical world generation of assignment is not possible. Hence subscribers (applications) subscribing with more than two predicates will not get successfully matched and notified. So considering events as an assignment is not a correct way to handle them.

In order to overcome the above mentioned problem one could think of designing the solution which is based on temporal (events of our interest occurring at different time stamp) and causal (Occurrence of the event influences the end result) property of the events. This area could be explored further, to successfully match subscriptions with multiple predicates against the events occurring at different time stamp.

**Table 2** Benefits and limitations of event matching techniques

| S. No. | Technique | Benefits | Limitation |
|---|---|---|---|
| 1 | Inverted Index | • Less index construction time Scalable | • Suitable for datasets with less number of attributes<br>• Memory intensive |
| 2 | IQ-Tree (Quad Tree + Inverted File) | • Update friendly<br>• Supports different indexing granularities for different queries<br>• Good Performance in two-dimensional space (if tree is balanced) | • Not useful in high dimensional space<br>• Only spatial feature is preferred during index construction<br>• Inefficient if data resides in external storage |
| 3 | OpIndex | • Less index construction and query processing time<br>• Less memory consumption<br>• Scalable | • Does not support geo-spatial data |
| 4 | AP-Tree | • Supports adaptive spatial and textual partitioning<br>• Handles large scale streaming data | • Not well-suited to the change of query workload, if AP-Tree structure built on a small proportion of the query set |
| 5 | R-Tree | • Good performance in low-dimensional spaces | • Spatial feature is preferred during index construction<br>• Basic operations like insert, update, delete are expensive |

## 4 Conclusion and Future Work

The Publish/Subscribe paradigm is well suited for large scale distributed systems due to its ability to provide dynamic, anonymous, many-to-many, asynchronous communication between publisher and subscriber. We can say event matching is one of the key feature of Publish/Subscribe system, where there is significant amount research scope in terms of providing better efficiency, scalability and high performance. We have discussed few of the state-of-the-art research that have been carried out in the area of event matching for Publish/Subscribe system and their related challenges. Challenges could be summarized as designing a Publish/Subscribe system based on temporal and causal property of the events, Publish/Subscribe system to support efficient event matching for geo-spatial data and Publish/Subscribe system providing high flexibility to user in registering their interest in the events.

Any enrichment to existing or new algorithm which provide better and/or flexible trade-off between scalability, expressiveness and quality of service are highly desirable and could be taken up as future work for any of the existing research. We believe this survey may prove to be an important contribution to the research community, by discussing the current state and open challenges of this important and dynamic area of the research. This would help readers interested in developing new solutions or enriching existing solutions to address challenges in event matching for Publish/Subscribe system in IoT.

## References

1. Internet of Things, https://en.wikipedia.org/wiki/Internet_of_Things.
2. Banavar, Guruduth, et al. "An efficient multicast protocol for content-based publish-subscribe systems." Distributed Computing Systems, Proceedings. 19th IEEE International Conference on. IEEE, (1999).
3. Eugster, Patrick Th, et al. "The many faces of publish/subscribe." ACM Computing Surveys (CSUR) 35.2, (2003).
4. Campailla, Alexis, et al. "Efficient filtering in publish-subscribe systems using binary decision diagrams." Proceedings of the 23rd International Conference on Software Engineering. IEEE Computer Society, (2001).
5. Zhang, Dongxiang, Chee-Yong Chan, and Kian-Lee Tan. "An efficient publish/subscribe index for e-commerce databases." Proceedings of the VLDB Endowment 7.8, (2014)
6. Li, Guoliang, et al. "Location-aware publish/subscribe." Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, (2013).
7. Chen, Lisi, Gao Cong, and Xin Cao. "An efficient query indexing mechanism for filtering geo-textual data." Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. ACM, (2013).
8. Chen, Lisi, et al. "Sops: A system for efficient processing of spatial-keyword publish/subscribe." Proceedings of the VLDB Endowment 7.13, (2014).
9. Whang, Steven Euijong, et al. "Indexing boolean expressions." Proceedings of the VLDB Endowment 2.1, (2009).
10. Wang, Xiang, et al. "Ap-tree: Efficiently support continuous spatial-keyword queries over stream." Data Engineering (ICDE), IEEE 31st International Conference on. IEEE, (2015).
11. Tarkoma, Sasu. Publish/subscribe systems: design and principles. John Wiley & Sons, (2012).
12. Kale, Satyen, et al. "Analysis and algorithms for content-based event matching." Distributed Computing Systems Workshops, 25th IEEE International Conference on. IEEE, (2005).
13. Shraer, Alexander, et al. "Top-k publish-subscribe for social annotation of news." Proceedings of the VLDB Endowment 6.6, (2013).
14. Mishra, Tania Banerjee, and ShashankSahni. "PUBSUB: An efficient publish/subscribe system." Computers and Communications (ISCC), IEEE Symposium on. IEEE, (2013).