Macrothink Institute™

Network Protocols and Algorithms
ISSN 1943-3581
2013, Vol. 5, No. 1

A Hot-topic based Distribution and Notification of Events in Pub/Sub Mobile Brokers

Augusto Morales Domínguez, Tomas Robles, Ramon Alcarria, Edwin Cedeño

Department of Telematics Engineering, ETSIT,

Technical University of Madrid

Avenida Complutense 30, Madrid (Spain)

Tel: 34-915495700 EXT 3035
E-mail: {amorales, trobles, ralcarria, edwinc}@dit.upm.es

Received: February 27, 2013 Accepted: March 15, 2013 Published: March 31, 2013

DOI: 10.5296/npa.v5i1.3326 URL: http://dx.doi.org/10.5296/ npa.v5i1.3326

Abstract

Publish/Subscribe-based networks have emerged as suitable candidates for supporting new sort of devices in the Internet of Things. Despite this, there are few mobile broker models that integrate existing subscription and notification models while meet the requirements of these devices. The challenge of creating an uniform and loosely coupled content dissemination network of fixed and mobile devices is gaining importance; so, mobile brokers have to provide not only protocol compatibility but also event coordination and popularity.

In this paper, we introduce mechanisms that allow mobile brokers to distribute their subscribers and coordinate the notification of events between fixed and mobile brokers. We propose a hot-topic algorithm that marks the event popularity and depending on this, delegates or recovers subscriptions. Furthermore, we also propose extensions for the MQTT protocol, in order to support our mechanisms. Finally, we show the results of our simulations and confirm that our mechanisms offer advantages for our considered IoT scenario.

Keywords: Publish/Subscribe, Topic-based language, MQTT, Rendezvous Mobile Broker, Subscription Delegation, Event-based system, Mobile broker

1. Introduction

The future Internet of Things (IoT) demands a closer breach between fixed and mobile environments in the content plane; so, even if devices are loose coupled the information they produce should be available from end-to-end. In this context, the Publish/Subscribe (Pub/Sub) paradigm [1] has emerged as an attractive communication model for exchanging content because of its asynchronous, time and process decoupled style. On the current Internet, the Pub/Sub systems have been successfully extended to content dissemination services such as: PubSubHubbub [2] as well as they have being considered as a promising content-centric communication model for Future Internet Architectures [3][4]. Despite this, there are still challenges concerning how to deal with the heterogeneity [5]across entities that produce and consume content. Some of these entities can be mobile devices with low performance, non-stable connectivity and tight energy constrains; thus, their communication capabilities largely differ from fixed devices (e.g. a server in a datacenter).

Publish/Subscribe systems, which are also knows as event-based systems, are basically composed of three main components [6] publishers which are the content producers, subscribers that express their willingness to consume specific content; and finally brokers that put in contact publishers and subscribers by storing and forwarding this information. Depending on the scenario, wireless devices can perform as publishers, subscribers or brokers; so their capability of consuming, publishing and matching content (in the form of events) depends on the expressiveness [5] of the subscription language of the Pub/Sub network.

In future IoT scenarios, it could be the case that even if a mobile device is capable of performing as a broker (e.g. a mobile phone, laptop, or medical wireless handhelds), it could still lack of network and processing capabilities in comparison with fixed devices. In addition, as these devices could appear and disappear at any moment they should perform as modular and pluggable components. In this paper we propose solutions that target these scenarios in the form of subscription distribution, notification models and hot-topic classification.

The paper structure is as follows: Section 2 describes related works. Section 3 explains the mobile broker features, the motivation scenario and requirements for the subscription language. Section 4, describes considerations related event notification in distributed Pub/Sub systems. Section 5 presents or solutions for distribution and notification of events. Section 6 explains our validation and results. Finally, we end with our conclusions and future works.

2. Related Works

Pub/Sub systems are quite common in large-scale networks and infrastructures [6], where brokers have enough processing power to put in contact content producers and subscribers while implement complex subscription languages. Pub/Sub brokers are also capable of distributing subscriptions' states, event routing and matching. In this context, there are many algorithms [7] and mechanisms [8] for optimizing these distributed Pub/Sub systems. Pub/Sub systems have also proven their advantages [9] regarding flexibility and integration with other layers. Despite this, these scenarios are less restricted than mobile computing scenarios [10] where resources are constrained and content dissemination strategies are not applicable. Hence, there are still challenges for integrating mobile devices as fully-functional

elements capable of carrying out energy-efficient with multiple brokers. This integration is one of the targets of future ubiquitous computing scenarios [11], where multiple participants might be consuming/publishing information, through standardized interfaces (e.g. the OMA interfaces[12] and the FI-WARE generic interfaces[13]). In the same way, mobile participants will be able to extend their nearby environments (e.g. PAN and WSN networks) to the same content dissemination network.

Some works have tackled [14][7] models and architectures for mobile environments. Other researches are focused on content dissemination strategies and event mobility. Works such as [15] propose theoretical Pub/Sub models, event distribution strategies and describe some high level constrains of mobile brokers. Other models [16] integrate mobile brokers in Pub/Sub systems and target the delegation of subscribers. Concerning the integration of Pub/Sub mobile brokers with nearby environments, other works [17] proposed middleware solutions for unifying the way mobile brokers publish content. Other researches [18] clarify the advantages of integrating portable Pub/Sub systems with existing human networks, sensors and users' content. Some considerations of having a Pub/Sub communication, with unstable connectivity, have been previously discussed in [10]. Other studies [9] describe how Pub/Sub models allow systems to be resilient in unknown environments. Concerning the mobility of subscribers and publishers there are several generic and non-generic solutions [19]; and some of them employ well spread protocols in combination with advance handoff mechanisms [20].

3. From Fixed to Mobile Pub/Sub brokers

In previous works [21] we proposed a Rendezvous Mobile Broker (RMB), which is an extended mobile broker. We defined it as a low-capability and pluggable broker that runs over mobile devices. As any other Pub/Sub broker the RMB supports event matching, routing and notification.

As it is represented in Figure 1, the RMB supports a content dissemination network between the mobile and infrastructure domains. The infrastructure domain represents a set of publishers/subscribers and fixed brokers. Fixed brokers behave as high capability nodes capable of matching complex subscriptions, storing and forwarding high amount of events to clients. On the other hand, RMBs store a lower amount of subscriptions and support fewer ratios of events since their processing constrains. The RMB acts as the link of the content dissemination network that is built with all the Pub/Sub clients in the mobile and fixed plane; so, it performs as the rendezvous point whenever a set of subscriptions and events are distributed to fixed brokers back and forth.

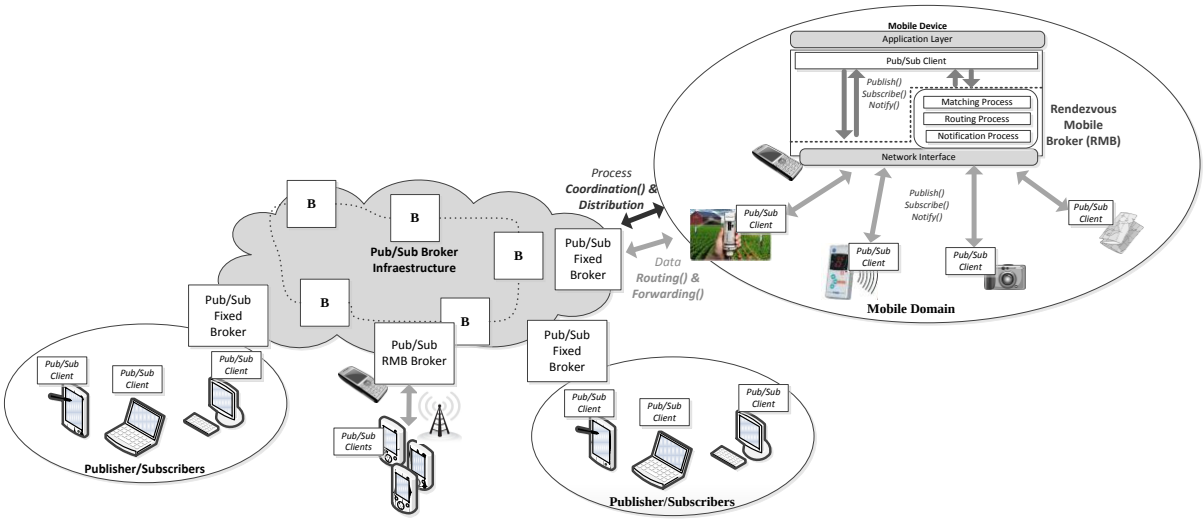

Figure 1. Overall Pub/Sub content dissemination network

3.1 Motivation scenario

The RMB extends a content dissemination network to scenarios where brokers have to be pluggable, mobile and performance-limited. Following these premises, we analyze our scenario which is based on the IoT Smart Farming [22] user cases. In this, a farmer takes makes use of the Pub/Sub infrastructure to monitor their crops in real time with moderate investment. Crop monitoring is done by providing a set of sensors capable of measuring crop and land properties such as temperature, humidity, chemical reactions, or level of pesticides. Some actuators can also receive this information and act accordingly. Sensors communicate with handheld devices [23] which are to disposition to the farmer for monitoring and data gathering purposes. As some sensors, placed and distributed in the land, have short communication range for exchanging information with other devices and actuators, a mobile broker, running over a handheld device, is proposed to act as data distributor and notifier. The mobile nature of this broker enables multiple locations to be covered in short term.

Figure 2 shows the main elements of the scenario. A fixed broker (1) is located in the vicinity of a field to enable communication between sensors and actuators. Since some sensors, by the nature of the supporting communication technology, are not covered by the fixed broker, there is a mobile broker (2) that moves through the sensorized space. The fixed broker acts as a supporting broker for the communication since it compasses more storage and processing capabilities than the mobile broker. Sensors communicate (3) through the mobile broker when it is in coverage and stop to communicate their data (4) when the mobile broker is moving out of range. Other sensors that are located close to the fixed broker can directly communicate with it (5).

In this scenario the fixed broker behave as a supporting broker for mobile brokers through a broadband wireless network (e.g. WiMAX or LTE). As an example, in the case a sensor subscribes to information that has been published by (5), supposing than the mobile broker (2) moves to a no-coverage area (of the sensor), the supporting broker can store the subscription

until other mobile broker (6) approaches to the coverage area of the same sensor; so, it can then push the non-received events. Depending on its capabilities (e.g. limited processing power) and state (e.g. interest to move), the mobile broker (6) can decide that it is only interested in supporting only one data type (e.g. the most popular or priority sensors). Therefore, it can negotiate with supporting broker of the farm, which interests it will process and delegate. In the case external subscribers are connected to the supporting broker (e.g. subscribers from the Internet), the mobile broker can also act as a network gateway for publishing information generated by sensors. If the farmer decides to move some sensors, under the control of the mobile broker, to other farm plot, the sensors' interests will remain the same and no additional subscriptions will be needed; so then mobile broker can decide, depending on the capabilities of the new fixed broker, to maintain all the interests of the delegated sensors.

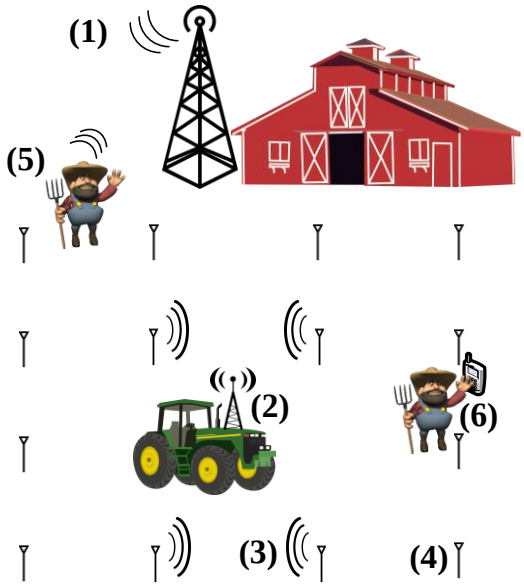

Figure 2. Smart Farming application scenario

3.2 Pub/Sub language for mobile brokers

According to Eugster et al. [1], Pub/Sub languages can be classified in three categories based on their expressiveness: topic-based, content-based and type-based. Expressiveness has impact on the performance of a broker, since the more expressiveness has a language the more resources it requires for matching events. In the context of mobile devices running a RMB, challenges such as variable connectivity, protocol support and content organization can also affect the way it serves to clients.

The RMB supports a topic-based language as we take into consideration the following statements. Topic-based languages basically use a formatted string for identifying and interest on a resource (e.g. ATOM [24] uses it). Hence, if a resource is plainly identified (e.g. a soil moisture sensor in the WSN of the farm) the RMB can establish a relationship of this identifier and the topic that will represent its data in the Pub/Sub network. In the case subscribers have non-complex and stable interests, a topic-based RMB can fit their needs. In addition, a

topic-language can speed up a simple and efficient implementation [25] of RMB, which can later be pluggable with fixed brokers and new protocols. Concerning content organization a topic-based language allows a hierarchical classification of resources using existing domain name systems; so, it will be feasible for RMBs to organize their subscribers, delegate them to other brokers and finally perform as gateways with the Pub/Sub infrastructure. Regarding protocol support, currently, there are M2M protocols (e.g. MQTT-S [26]) which already link resources using multi-level identifiers and topic-based languages. Hence, The RMB can implement the same components and support future protocol enhancements. In the same way, some of these protocols let nodes to publish content while maintain in a packet header the corresponding identification (e.g. a topic) of this content. The cost of decoding only the header can lead to a lightweight RMB implementation and a better performance of the underlying OS. Table 1 summarizes the characteristics of the three Pub/Sub languages from the point of view of our scenario.

Table 1. Comparison of Pub/Sub languages

Characteristic	**Type of Pub/Sub Language**		
	Topic-based	***Content-based***	***Type-based***
Content Addressing level	Multi-level and hierarchical	Depending on attributes	Hierarchical or depending on objects types
Content Extendibility level	Limited by topics' hierarchy	Practically Unlimited	Practically unlimited
Abstraction level of matched events	Topic == event	Attribute && operator == event	type (optional: Attribute && operator) == event
Implementation level examples	IoT scenarios[27], large scale real time notifications[2].	Large-scale data dissemination [28].	Large-scale data dissemination[1]

4. Considerations for distributed matching and notification

The RMBs supports Pub/Sub clients in the mobile domain. In our model the matching and notification processes can be distributed. The RMB and the fixed broker implements mechanisms for managing these processes; so this section clarifies them. In the RMB the routing process refers to the mechanisms which can be used for disseminating events among brokers [29]; however these mechanisms are out of the scope of this article.

The matching process evaluates an event with a set of active subscriptions in the broker. Subscriptions can be distributed between a RMB and a fixed broker so notifications that are the result of this match could be again re-evaluated by the RMB if coordination between brokers is missing. This action can be counterproductive for the RMB, because it will have to re-evaluate events which should have been filtered only once on its fixed broker. Even if there are mechanisms for delegating the matching to a fixed broker in the process plane the RMB still needs to recognize already-matched events in the content plane. So, we establish

the following criteria (1): *for every event that arrives to a set of brokers, even if there are several matching process there must be only one notification process for a given subscriber.*

The notification process delivers matched events to subscribers. In our scenarios there are three possible cases: events that are produced and consumed in the mobile domain, events that are produced outside the mobile domain but consumed inside it, and events that are produced inside the mobile domain but consumed outside. In the last case, assuming than the matching process resulted negative, the RMB directly routes the produced events to the fixed broker; so there is no need of any adaptation in the notification process. In a similar case, if the matching process resulted positive, it means than the fixed broker supports subscribers which were already interested in these events, so the notification process directly uses the routing process for disseminating events (e.g. using a reverse delivery path or a gossip routing). In the first and the second case the RMB has to coordinate the notification processes because the following statements.

In the first case the notification process resides in the RMB. Events are matched and *locally delivered* to subscribers' "callback" addresses which could be reachable by the underlying network interface or middleware [30]. As is shown in Figure 3 as the RMB contains all the subscriptions of Subscriber B, after the Publisher sends a message to the broker, the RMB can immediately match and deliver this message to Subscriber B. On the other hand, if the notification process, for a set of subscriptions of Subscriber A, has been previously distributed (d1) and is currently supported by a fixed broker, this broker should implement mechanisms (d2) to identify that these events (generated within the mobile domain) must be sent back to the RMB. Hence, even if the matching process (e1) is enforced locally (in the RMB), or externally (in the fixed broker) these events will be externally delivered from the mobile domain perspective. This scenario comes out with communication challenges as these events could leave the mobile domain and return; so, brokers must coordinate the notification process in order to prevent the non-duplicity notification we established in criteria (1). In the case of subscribers served by the fixed broker, no special modifications are needed because the fixed broker can forward these events to other brokers in the network.

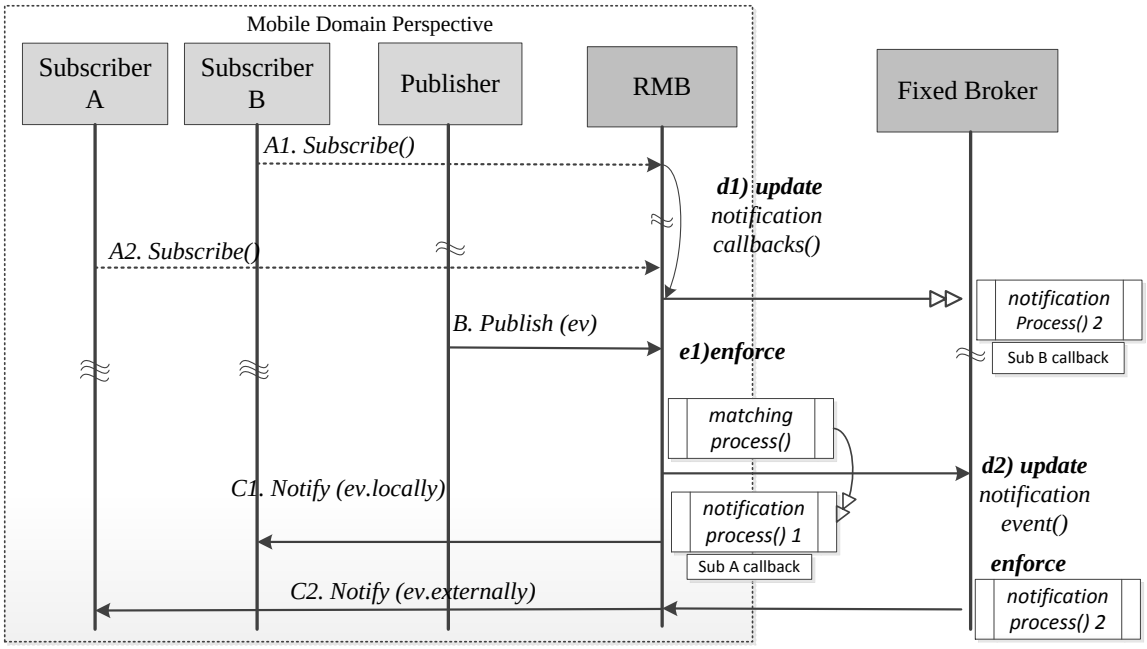

Figure 3. Communication diagram for mobile and fixed domains

In the second case, publishers outside of the mobile domain generate events that are later consumed by mobile subscribers; so, even if the callback addresses of subscribers are distributed or not, the RMB has to implement mechanisms for discriminating which of these events should be matched or not. Hence, this case also confirms that brokers must coordinate the distributed notification processes.

5. Distribution and Notification of Events

In our model the RMB can delegate and recover subscriptions to/from fixed brokers. Subscriptions are composed as: $S_T = \{T_P, C_B\}$; where T_P represents the topic that identifies a resource and C_B the callback network address of the subscriber. In our subscription model the RMB maintains a single registry per every topic it has to match, so it stores the clients' subscription that match a single topic in this registry. This method has more advantages in comparison with storing one single registry per client, because most of the current topic-basic subscriptions include the notion of recursive subscription (e.g. the MQTT protocol), so topics can be hierarchically grouped. A broker stores two different subscribers: active and delegated. Active subscribers consist on subscribers that are being used in runtime, by the broker, for the matching process. Delegated subscribers represent clients which subscriptions are currently deactivated for matching (in the broker). In the case a fixed broker receives an event that matches an active subscriber, which is delegated from the point of view of the RMB, it marks the messages to comply with criteria (1) (this is further explained in section 5). Brokers negotiate active and delegated subscribers on-the-fly using a different Pub/Sub channels (Process Plane) rather than using the same Content Plane for the message notification. Hence, this offer some sort of redundancy when multiple Fixed Brokers server RMBs. For ensuring the subscriptions consistency the RMB maintains the ownership of its subscribers and interests; but

subscriptions can still be stored in other brokers. Figure 4 depicts the differences between process and content plane, as well as the organization of the subscriptions.

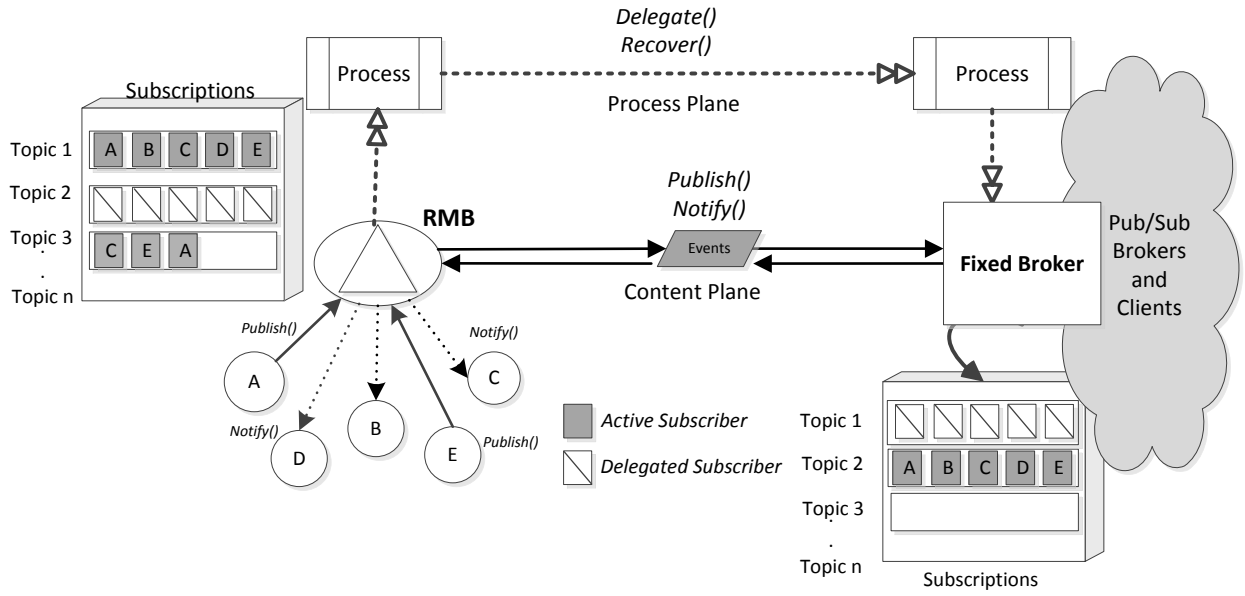

Figure 4. Content and process plane

5.1 Topic Matching

The topic matching, especially in RMB, demands an efficient organization of the topic-to-subscription relationships and the notification of events. In addition, this process must be cost-effective from the network perspective as subscribers and RMB are constantly moving. In our proposal, before any delegation we detect the popularity of the events that match subscribers in order to improve these relationships. Hence, we proposed a hot-topic matching algorithm which organizes topics and sets which of them will be delegated or not. This algorithm employs the rate of events for classifying which are the most and less popular topics, based on a sliding window. Once a RMB implements this algorithm, it can delegate less popular topics and recover popular topics, fact that improves the network delay as we explain in our results. Hence, in Figure 5, we show the first part of the algorithm: the publication monitoring procedure, which organizes the events in registries.

```
1: procedure Publication_monitoring (ev)
2:       for each Ts  ϵ Registry.Calc do
3:            if (Ts.topic  matches ev.topic)
4:                 Ts.count++;
5:                 Ts.time← getTimeStamp()
6:                 add Ts.time & Ts.topic to Registry.Time
7:                 return
8:            endif
9:       endfor
10:      Ts ← create ()
11:      Ts.count++
12:      add Ts.count to Registry.Calc
13:      Ts.time← getTimeStamp()
14:      add Ts.time & Ts.topic to Registry.Time
```

Figure 5. First part of Hot-Topic algorithm

In the first step a matched event that will be sent to a subscriber enters in the procedure. The second step of the algorithm, obtains the list of all the topics (*Registry.*T_{TS}) which were previously matched, and then compares (3) if the incoming event's topics exists *(ev.topic).* In a true case, the algorithm increments (4) the counter of the topic *(Ts.count)*, gets (5) the current time of the broker *(Ts.time)* and then adds (6) this time to a Registry of Times *(Registry.Time).* This registry will contain all the topics and the timestamp of every event. In the case the ev.topic is missing, the algorithm creates its registry and increments its counter. Next, these new values are added to the *Registry.*T_{TS} and the algorithm executes the same procedure of getting timestamp and save it. At the end of the procedure, the algorithm maintains two registries one for topics and timestamps and another for topics and counter.

In the second phase, Figure 6, the algorithm employs two registries. The *Out.Registry* contains all the topics that will be delegated to the fixed brokers as their popularity level is low. The *In.Registry* contains topics with high popularity; so, events that match this topic will always be locally delivered to subscribers in the mobile domain.

```
1: procedure Hot_topic_calculation (X.time, Y.time, Registry.Time, Registry.Calc)
2:     Start.Thread (X.time)
3:     C.Time = getCurrentTime()
4:     for each (Ts.time ϵ Registry.Time do )
5:             if    C.Time.-Y.time    < Ts.time      then
6:                     add Ts.topic to Registry.Calc
7:             endif
8:     endfor
9:     Tp.count.finals ← copyCounts(Ts.counts)
10:    for each (Ts.topic ϵ Registry.Calc do )
11:            Ts.Tp.count.final    ++
12:    sortTopicsWeights(Registry.Calc)
13:    for each (Ts.topic ϵ Registry.Calc until counter < Registry.Calc /2 do )
14:            add Ts.Tp.topic to In.Registry & remove from Out.Registry
15:            counter++
16:    endfor
17:    add Registry.Calc to Out.Registry & remove from In.Registry
18:    sendToBroker (Out.Registry)
19:    requestToBroker (In.Registry)
```

Figure 6. Second part of Hot-Topic algorithm

The second part of the algorithm calculates the hot-topics taking as an input the previous procedure. RMB calculate the hot-topics each *X.Time*. Depending on the scenario a RMB could have to deal with bursty traffic at any time so marking popular topics using a fixed time could barely reflect, in some cases, real network status. As an example, a topic has a very low rate of events at the beginning but this rate suddenly increases (e.g. an RMB approaches to an area with multiple Pub/Sub clients). If the popularity of a topic (which is more demanded in this moment) remains lower than other topics which were more popular in the past, this topic (and its bursty-traffic) will be categorized with a lower score; so, the topic will be delegated and its events will go through unnecessary hops. In order to be flexible to this kind of situation, the algorithm employs a sliding time (*Y.Time*). This parameter allows the broker to set, depending on the conditions (e.g. connectivity, rate of events), which of the latest topics will be catalogued as Hot-Topics. Nevertheless, more sophisticated mechanisms are applicable such as using the expected event distribution [31] and connectivity patterns [30] but they require more complex and demanding implementations.

Step 4 recovers each Ts.Time in the *Registry.Time* and verifies if its value is higher than the current time of the broker minus the sliding time. In a true case the broker includes this registry in the *Registry.Calc*. Step 9 copies all the *Ts.counts* to a similar registry that exists in the *Registry.Calc* under the *Ts.topic* field. Next, the algorithm increments the value of the *Tp.count.final*, and then it sorts (12) all the topics using the value of *Tp.count.finals*. At this point the *Tp.count.final* of a *Ts.topic* can be considered as a score which reflects the

Hot-Topic level for *Y.time*. Next, the algorithm recovers (13) the 50% of the highest topics and puts them (14) in the *In.Registry* one by one and removes it from the *Out.Registry.* The remaining topics are located in the *Out.Registry* and removed from the *In.Registry*. Finally, the algorithm sends the *Out.Registry* to the fixed broker and requests the *In.Registry.*

5.2 Subscriptions Coordination Model

After a mobile broker detects the popularity of the topics, it coordinates with the fixed broker these topics and their subscribers using a Pub/Sub channel. For this task we define two basic primitives: *delegation and recovery.* Both share the following topic-based structure: *<domain>/<broker_id>/<action>*. Domains enclose the domain name where the broker belongs, thus, it can be a DNS-based domain (e.g. *dit.upm.es/*) or any tailored domain (e.g. *farm_x/sector_y*). Broker_id is the broker identifier in the current domain. Actions represent the two primitives that brokers carry out for the coordination (delegation and recovery), so they are reserved words.

Since content discovery is out of the scope of this article, we start from the fact that brokers have each other's domain, identifiers and the protocol stack each other support. At the beginning of the process the fixed broker subscribes to the *delegation* action of the RMB (e.g. *dit.upm.es/fbroker1/delegate*), and then the RMB subscribers to the *recovery* action of the fixed broker (e.g. *dit.upm.es/rmb1/recover*). Now, the RMB is ready for delegating a set of topics and all their corresponding subscribers, so whenever it starts to publish data into the network the fixed broker will receive it. The RMB stores subscribers' data using the Subscription Allocation Algorithm (SAA) [21] in the form of Counting Bloom Filters (CBF). In the following steps, the RMB publishes the list of all the compatible callback address of the subscribers it manages. Hereinafter, every time the RMB calculates, through the Hot-Topic algorithm, the *Out.Registry* it will just have the corresponding CBF for every topic in this registry. At this point the fixed broker will be ready to notify subscribers in the RMB domain. In the case a subscriber, in the mobile domain, issues a un-subscription message that targets a delegated topic, the RMB adds an */unsubscription/single* string to the delegation topic, includes in the event payload the callback address of the subscribers, and finally publishes the message to network. There could be the case that the RMB moves to a different fixed brokers; in this case, it must inform its serving fixed broker (because it has to stop matching and notifying for the delegated topics) so the RMB adds the */unsubscription/all* string and sends an empty event. Depending on the RMB's capability of supporting topics (and their associated event rate), RMB can optionally store (e.g. in a non-volatile memory) all the CBFs and later send them to the fixed broker. In the case storing is unfeasible the RMB can subscribe to the *recovery* topic, so whenever it *publishes()* a message under the topic */delegate/deactivate/all*, it will asynchronously receive all the topics and will be ready to move to other domain without losing the clients' subscriptions.

5.3 Event Notification Model

As there could be topics and subscribers distributed over RMB and fixed brokers, the latter

can also notify subscribers in the mobile domain. If a subscriber supports a network-layer technology that is unavailable in the fixed broker, the RMB performs as a gateway; so notifications go through it. In this scenario there are two cases: the RMB acting as a resource mapper, in the protocol or application layer, or as a transparent device. As an example of the first scenario, the MQTT-S specification [26] defines gateway mechanisms to interoperate with MQTT brokers, in this, topics that identify content are maintained; so, the RMB performs as a resource mapper in the protocol layer. On the other hand, the RMB associates content generated by devices with different link technologies (e.g. Wi-fi, Bluetooth and Zigbee) and acts as a mapper in the application layer since it decouples content and its identification from the underlying technology.

In another case, fixed brokers and mobile subscribers support TCP/IP layers. However, if a common link layer is missing (e.g. one uses Ethernet and the other Wi-fi), in this case the RMB perform as a transparent device. In the case of a RMB acting as resource mapper in the application layer, if an event arrives from the fixed broker under the *notify()* primitive, the RMB detects that this event was previously matched. Therefore, as the fixed broker already recognizes reachable callback addresses, it appends the list of non-reachable callback addresses to the event. Once the RMB recognizes this event, it skips the matching process and directly re-directs the message to each one of the subscribers and their real callback address.

6. Evaluation

The evaluation is divided in two parts. In the first part, we simulate the motivation scenario presented in Section 3.1. We based or evaluation in MQTT which is M2M/IoT protocol [27] based on Pub/Sub and a good candidate for covering our needs it. Nevertheless, as it misses some of the functionalities for our models, such as: support for distributed notification and subscriptions, we propose extensions for version 3.1. Then, we implement these new extensions and verify the advantages of our Hot-Topic algorithm for the Smart Farming scenario.

6.2 MQTT Protocol Extension

In case A of Figure 7, the RMB broker executes the Hot-Topic algorithm, so after a given time, it is capable of marking popular and non-popular topics. Brokers subscribe to delegate and recovery actions using MQTT SUBSCRIBE messages (1.A, 1.B). In step 2, since clients must connect first to brokers before any publication or subscription we have replaced, in the CONNECT* (2) command, the protocol version number (byte 9) from version 0x03, to version 0xFF. We employ the 0xFF value to prevent overlapping with upcoming MQTT versions. This modification maintains the compatibility with standard brokers, because brokers that are unaware of this version will send back an "*unacceptable protocol version*" message; so clients can re-issue a connect message by changing version back to 3. Then, the RMB acknowledges a new MQTT CONNACK*(3) message with a code in the variable header in the field 6-255 (reserved for future use). This message is a "*Connection Renew"* message with field *Enumeration-6, HEX-0x06*. The RMB includes in the payload, the network address of the fixed broker in the UTF-8 string: *ip_address: port.* A broker classifies callbacks as

reachable (TCP/IP address) or non-reachable (remaining); nevertheless, most sophisticated negotiation mechanism can apply. In step 5, the RMB delegates topics/subscribers and publishes this information (CBF of topics and callbacks) using a standard MQTT PUBLISH message, so no specific protocol modifications are needed. In this step, the Subscriber has to disconnect to the RMB and then attach to the fixed broker. Thus, the RMB sends a (7) a new CONNACK* message with a code in the variable header in the field 6-255 (reserved for future use). This message is a "*Connection Renew: Active Topics"* message with field *Enumeration-7*, *HEX-0x07*. Finally, the subscriber connects to the Fixed Broker using the CONNECT* and is ready to receive notifications.

In case B, the same client is still attached to the Fixed Broker. Then, the RMB calculates and publishes (10) a new list of hot-topics it needs to recover, using the topic schema defined in section 5.2, so later it receives in the PUBLISH message payload (11) the list of the CBF/callbacks. Next, the Fixed broker sends a CONNACK* message (12) to each one of the subscribers it currently servers and matches these topics. Hence, subscribers can CONNECT again to the RMB. In the case a mobile broker, with no subscription delegation characteristics, starts to serve subscribers, first, it must issue a PUBLISH message with an empty payload and attaches to the topic the string *delegate/deactivate/no*. Then, the Fixed Broker sends to all its serving subscribers a new CONNACK* message with a code in the variable header in the field 6-255 (reserved for future use). This message is a "*Connection Renew: Renew Topics"* message with field *Enumeration-8*, *HEX-0x08;* and a payload that include the *ip_address:port* of the new broker coded in UTF-8. Therefore, Subscribers resend a standard CONNECT and then a new SUBSCRIBE (topics) message to the mobile broker. This new message let RMB to be pluggable, even if clients do not support the proposed MQTT extensions.

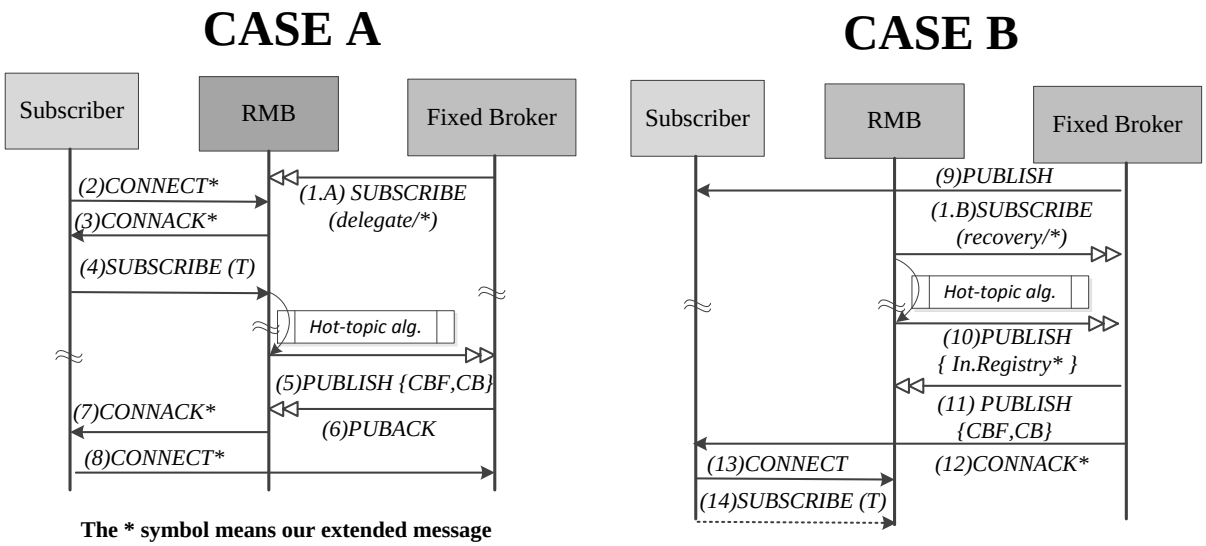

Figure 7. MQTT message extensions.

In the case a MQTT-compliant RMB acts as a resource mapper it has to redirect messages to subscribers' callbacks; so our objective is to allow RMB to notify subscribers without digging into the payload. The MQTT specification establishes that a payload part of a PUBLISH message contains application-specific data only and no application-specific flags are allowed in the fixed or variable headers; so sending callbacks within the payload breaks this principle. Hence, we propose a new MQTT command message with the following characteristics: *Message Type: PUBCALLBACK** and *Enumeration 16.* The variable header has at least 2 bytes that represent the MSB and LSB (string lengths) of a UTF-8 STRING which contains all the subscribers' callbacks. The character data of this STRING are encoded starting from byte 3 of the variable header. Callbacks are separated by a character: 0x22 and start and end with a 0x7B and 0x7D respectively (e.g. {cb1"cb2}). The payload is encoded after the last character.

An alternative way to support compatibility of our notification model is to implement a pure-topic communication; in other words without defining new protocol extension. However, it comes up with disadvantages. Since subscribers must know when they have to move to a new broker, they should subscribe to one additional and tailored topic in order to receive re-connection events of the new broker. In addition, RMB and fixed brokers will have to exchange these topics and maintain the additional reference callback-TO-re-connection/topic because subscribers could be required to move at any time. Having a topic-based/re-connecting mechanism obligates to have a coupled matching-notification layering in the brokers side, which also breaks the decoupled communication model offered by the Pub/Sub paradigm. This model also forces additional session "negotiation" mechanisms whenever a RMB delegates and recover a channel; so, subscribers will be un-aware of delegation and re-subscriptions compatibility until runtime, however, with our solution the subscriber will know in advance the notification model of its serving broker.

6.3 Validation Scenario

We validate our scenario through a combination of real and simulated environments based on the ns-3 [32]. We consider brokers that communicate using a modified version of the java-based Moquette MQTT broker [33] which runs over the application layer over the Linux Containers (LXC). Communication between application and network layer is achieved by binding LCXs with the ns-3 environment through ghost nodes. Figure 8 shows the organization of these layers. The simulation environment runs over a server with CPU Core i7 2.80 GHz, 8GB of RAM and Ubuntu 12.04 64 bits. We employ two different large range wireless technologies LTE and WiMAX for communicating a mobile broker in LXC 1 and the fixed broker in the LXC 2. The distance between the mobile and fixed broker is set to 1 KM and a common bandwidth of at least 10Mbps

The WiMAX network has been configured with a constant position mobility model. It considers three nodes: 2 user equipment (brokers) and one base station. WiMAX physical modulation type is set to QAM 16 and best effort traffic. The power transmission of the nodes is set to 50dB, which is guarantees low packet loss at the used distance in comparison with the

30dB default. The LTE network has been configured with the 2 user equipment (UE) and an enhanced Node Deb(eNB). There are 25 downlink sub-channels and 25 uplink channels for UE and eNB. We connect Pub/Sub clients in both sides of the brokers, using real Wi-fi 802.11g and gigabit Ethernet.

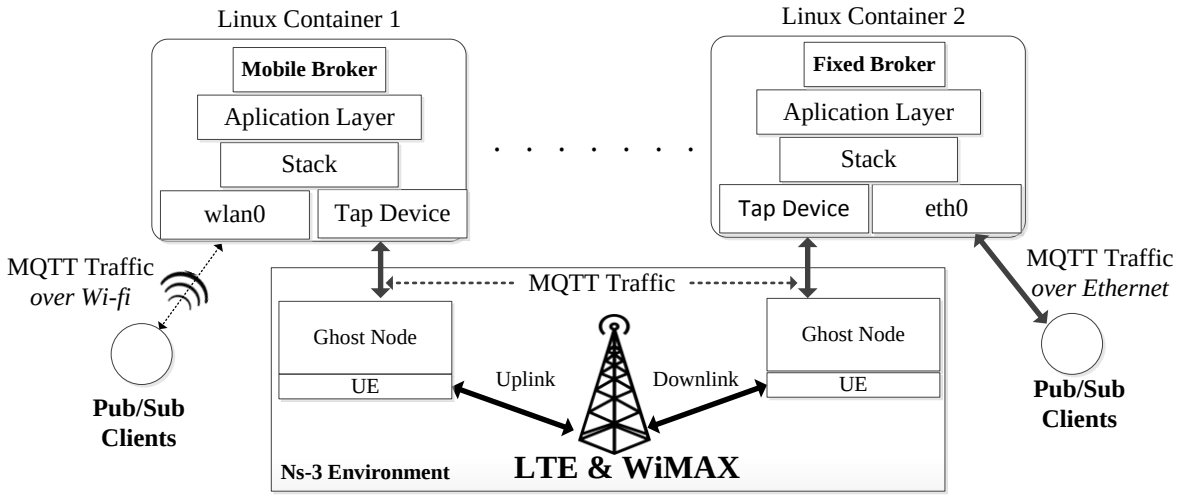

Figure 8. Simulated scenario

6.4 Results

First, we measured the delay of a MQTT *publish* message. The obtained values were 0.473 ms using Wi-fi from a node to the mobile broker, 20.7 ms for the mobile to fixed broker over WiMAX and 15.12 ms over LTE. These values are consistent with the standard values obtained by other studies [34][35].

For comparing the Hot-Topic algorithm we defined a **Naïve algorithm** for topic delegation using a simple approach. This algorithm delegates half of the topics to the fixed broker and lets the other half to be managed by the mobile broker. The delegation decision is independent from the probability of occurrence of a given topic.

We designed an event generation model with different operating modes in order to demonstrate the advantages of using the Hot-Topic algorithm for different types of traffic. *RandomTraffic* mode allows nodes to generate a set of publications on different topics such that the probability distribution is uniform. *BiasedTraffic* mode enables to generate a set of topics around one given, establishing a probability that this topic appears frequently. *BurstTraffic* mode consists of several bursts of traffic that can be random traffic and biased traffic. A random traffic cannot differentiate hot-topics and therefore, the performance of the Hot-Topic algorithm over the Naïve algorithm is similar for a traffic generated by using the *RandomTraffic* mode.

The benefit of detecting hot-topics is clear when one or more topics occur with more probability. This traffic pattern is natural in infrastructures where sensors exchange heterogeneous information such as our Smart Farming scenario. To validate the theoretical contribution we

represent the average delay delivering a publication message in Figure 9 for Naïve (N-*) and Hot-Topic (HT-*) algorithms in LTE and WiMAX escenarios. For this test we increase the percentage of participation of a given topic (e.g. topic 2) over a set of 10 topics (e.g. topic 1 to topic 10) using the traffic generated in *BiasedTraffic* mode. The average delay delivering a publication messages is deceased while increasing the percentage of participation of a particular topic, as the matching and delivery processes are mostly performed in the mobile broker. The generated traffic has a set of 200 publications and the values chosen for the implementation of the algorithm is 400ms (*X.Time*) of frequency and 2000ms of sliding window size.

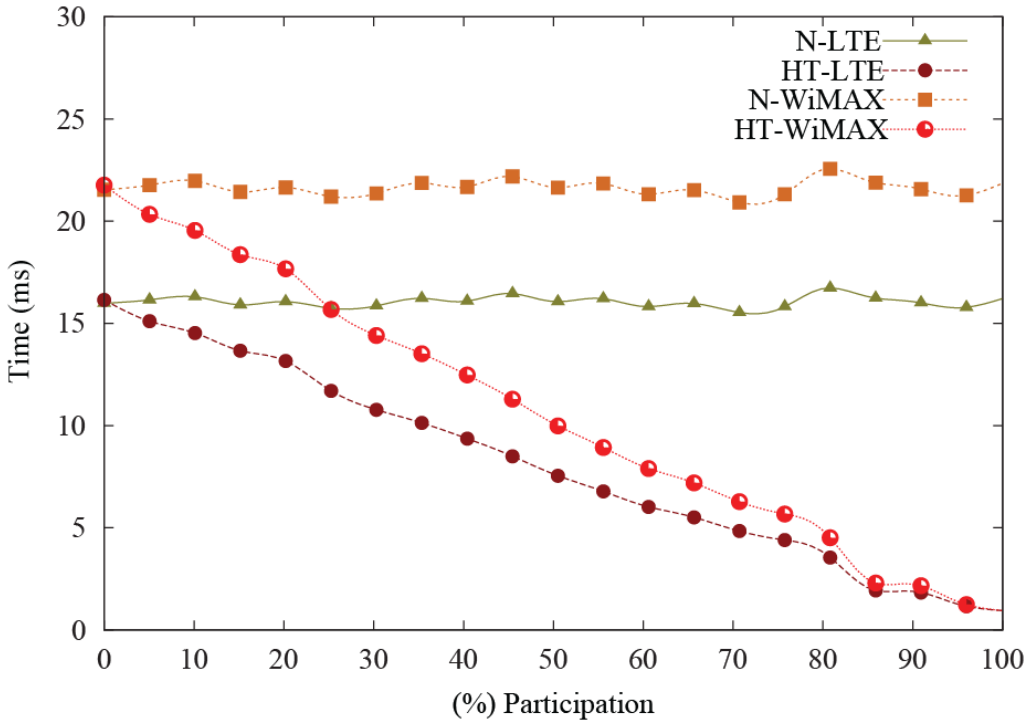

Figure 9. Average end-to-end delay for publication messages

When the most probable topic is changing over time, it is necessary to maintain a small sliding window so that topics occurred more recently are given best scores, as prospective topics are more dependent on these topics that on older ones.

In the second test we generate bursts of biased traffic by changing the probable topics. For this test a variable sliding window has been considered in order to check that the Hot-Topic algorithm manages the impact of recent topics on future topics. The results are shown in Figure 10. As it can be seen, by increasing the sliding window size from 100ms to 20 seconds, the average delay in delivering a *publish* message using the *Hot-Topic* algorithm gets similar to the delay offered by the naïve algorithm.

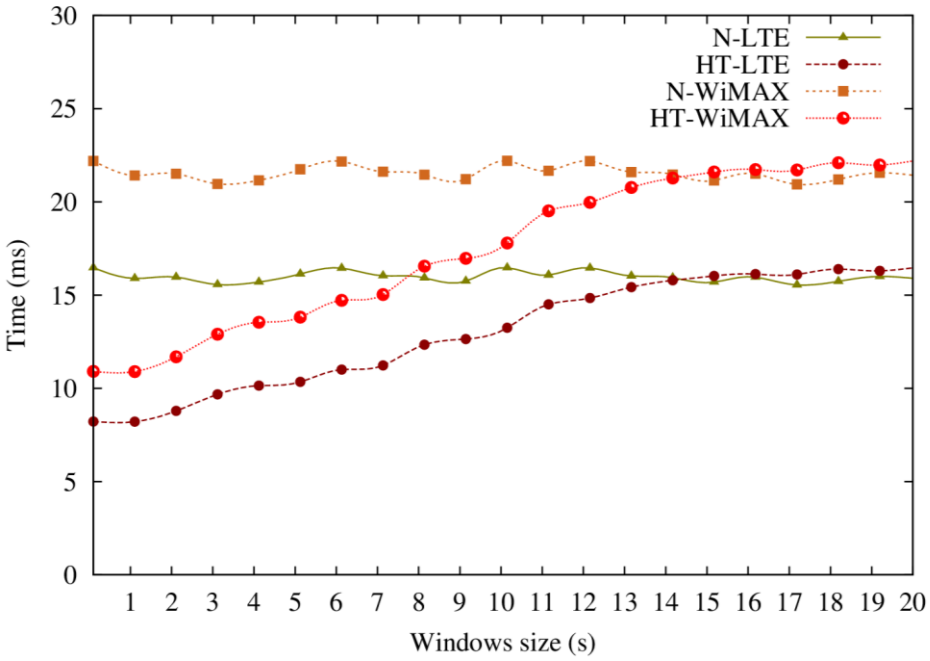

Figure 10. Average end-to-end delay for burst of biased traffic

6. Conclusions and Future Work

This paper proposes a solution for distributing hot-topics, subscribers and the process of notifying events over mobile and fixed brokers. This solution is focused on IoT environments, in which mobile devices act as pluggable component for a Pub/Sub-based network. The requirements imposed by this scenario determine that the way a broker manages topics has an impact on the notification of the matched event. Besides this, as IoT environments are dynamic scenarios, M2M protocols have to be enough flexible for letting mobile broker to appear and disappear, while maintain the same service level to publishers and subscribers. Focusing on this fact, our main contribution is a mechanism for improving the capabilities of mobile brokers in the form of topic-distribution and notification-awareness, and their implementation through extensions we have proposed for the MQTT protocol.

Our results show that our Hot-Topic algorithm improves the publication delay whenever a RMB delegates the notification of events to the fixed broker, in comparison with a naïve approach. We also show that our sliding window can be used for adapting the algorithm to different set of traffic, fact that increases the pluggability of mobile brokers in unknown scenarios. In random traffic the performance of the hot-topic algorithm is similar to the naïve approach, so using the hot-topic algorithm neither improves nor worsens the publication delay.

In the notification and matching layer, we are actively exploring lightweight mechanisms for allowing subscribers to negotiate with mobile brokers the event notification path. We are also working on an inter-broker coordination of the hot-topic algorithm in order to have a complete status of all the topics and their events, in the Pub/Sub network. In the routing layer

we will investigate to adapt gossip-based algorithms to the requirements of mobile networks and some application areas they could target as Collaborative Mobile Services [36].

Acknowledgement

This work was undertaken as part of the project SmartAgriFood funded by the European Community's Seventh Framework program.

References

[1] Eugster, P.T., Felber, P.A., Guerraoui, R., & Kermarrec. A.-M. (2003). The many faces of publish/subscribe. *ACM Comput. Surv*. 35 (2), 114-131. http://doi.acm.org/10.1145/857076.857078

[2] Pubsubhubbub project homepage. http://code.google.com/p/pubsubhubbub/. (last access February 2013)

[3] Xylomenos, G., Vasilakos, X., Tsilopoulos, C., Siris, V.A., & Polyzos, G.C. (2012). Caching and mobility support in a publish-subscribe internet architecture. *Communications Magazine, IEEE*. 50 (7), 52-58. http://dx.doi.org/10.1109/MCOM.2012.6231279

[4] FI-WARE Publish/Subscribe Generic Enabler. https://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FIWARE.ArchitectureDescription.Data.PubSub (last access February 2013)

[5] Fiege, L., Cilia, M., Muhl, G., & Buchmann, A. (2006). Publish-subscribe grows up: support for management, visibility control, and heterogeneity. *IEEE Internet Computing*, 10 (1), 48- 55. http://doi.ieeecomputersociety.org/10.1109/MIC.2006.17

[6] Fidler, E., Jacobsen, H.-A., Li, G., & Mankovski, S. (2005). The PADRES distributed publish/subscribe system. *ICFI*. http://dx.doi.org/10.4018/978-1-60566-697-6

[7] Salvador, Z., Alzua, A., Larrea, M., & Lafuente, A. (2010). Mobile XSiena: towards mobile publish/subscribe. In Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems, pp. 91-92. http://doi.acm.org/10.1145/1827418.1827434

[8] Kiani, S.L., Knappmeyery, M., Baker, N., & Moltchanov, B. (2010). A Federated Broker Architecture for Large Scale Context Dissemination. In Proceedings of the 10th IEEE International Conference on Computer and Information Technology, pp. 2964-2969. http://dx.doi.org/10.1109/CIT.2010.495.

[9] Hoffert, J., Mack, D., & Schmidt, D. (2010). Integrating machine learning techniques to adapt protocols for QoS-enabled distributed real-time and embedded publish/subscribe Middleware. Network Protocols and Algorithms, 2 (3), 37-69. http://dx.doi.org/10.5296/npa.v2i3.429.

[10] Caporuscio, M., Carzaniga, A., & Wolf, A.L. (2003). Design and evaluation of a support service for mobile, wireless publish/subscribe applications. *IEEE Transactions on Software Engineering*, 29 (12), pp. 1059-1071. http://dx.doi.org/10.1109/TSE.2003.1265521

[11] Kanjo, E. (2012). Tools and Architectural support for Mobile Phones based Crowd Con-

trol Systems. Network Protocols and Algorithms. 4 (3). http://dx.doi.org/10.5296/npa.v4i3.2052

[12] OMA Next Generation Services Interface V1.0. NGSI-10. http://technical.openmobilealliance.org/Technical/release_program/ngsi_v1_0.aspx

[13] FI-WARE Connected Devices Interface (CDI) Generic Enabler. https://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FIWARE.ArchitectureDescription.I2ND.CDI

[14] Rezende, C.G., Rocha, B.P.S., & Loureiro, A.A.F. (2008). Publish/subscribe architecture for mobile ad hoc networks. In Proceedings of the 2008 ACM symposium on Applied computing. 1913-1917. http://doi.acm.org/10.1145/1363686.1364149

[15] Muhl, G., Ulbrich, A., & Herrman, K. (2004). Disseminating information to mobile clients using publish-subscribe. *IEEE Internet Computing*, 8 (3), 46-53. http://doi.ieeecomputersociety.org/10.1109/MIC.2004.1297273

[16] Huang, Y., & Garcia-Molina, H. (2004). Publish/subscribe in a mobile environment. *Journal of Wireless Network*. 10, 6.

[17] Tong, X., & Ngai, E.C.-H. (2012). A Ubiquitous Publish/Subscribe Platform for Wireless Sensor Networks with Mobile Mules. In 8th International Conference on Distributed Computing in Sensor Systems, pp.99-108, 16-18 May 2012. http://doi.ieeecomputersociety.org/10.1109/DCOSS.2012.65

[18] Zhao, Y., & Wu, J. (2011). ZigZag: A Content-Based Publish/Subscribe Architecture for Human Networks. In Proceedings of 20th International Conference on Computer Communications and Networks, pp. 1-6, July 31 -Aug. 4. DOI. http://doi.ieeecomputersociety.org/10.1109/ICCCN.2011.6005931

[19] Tarkoma, S. (2012). Publish/Subscribe Systems: Designs and Principles, (1st ed.). Ed. Wiley, pp. 270-276.

[20] Tarkoma, S., & Kangasharju, J. (2007). On the cost and safety of handoffs in content-based routing systems. *Computer Networks*. 51 (6), 1459-1482. http://dx.doi.org/10.1016/j.comnet.2006.07.016

[21] Morales Dominguez, A.; Robles, T.; Alcarria, R.; Cedeño, E. A Rendezvous Mobile Broker for Pub/Sub Networks. GreeNets 2012 LNICST (pp. 16-27, 2013). http://dx.doi.org/10.1007/978-3-642-37977-2_2

[22] EU-FP7 SmartAgrifood Project. http://www.smartagrifood.eu/SAF (last access on February 2013)

[23] Walton, J. C., Larson, J. A., Roberts, R. K., Lambert, D. M., English, B. C., Larkin, S. L., Marra, M. C., Martin, S. W., Paxton, K. W., & Reeves, J. (2010). Factors Influencing Farmer Adoption of Portable Computers for Site-Specific Management: A Case Study for Cotton Production. *Journal of Agricultural and Applied Economics*, Southern Agricultural Economics Association, 42(2).

[24] IETF. RFC 4287. The Atom Syndication Format.

[25] Eugster., P. (2007). Type-based publish/subscribe: Concepts and experiences. *ACM Trans. Program. Lang. Syst*. 29 (1), 6. http://doi.acm.org/10.1145/1180475.118048.

[26] MQTT-S Specification. http://mqtt.org/MQTT-S_spec_v1.2.pdf (last access February 2013)

[27] MQTT Protocol Website. http://www.mqtt.org . (last access February 2013)

[28] Carzaniga, A., Rosenblum, D.S., & Wolf, A.L. (2001). Design and Evaluation of a Wide-Area Event Notification Service. *ACM Transactions on Computer Systems*. 10(3), 332–383. http://doi.acm.org/10.1145/380749.380767

[29] Morales Domínguez, A., Alcarria, R., Robles Valladares, T., & Cedeño Herrera, E. (2012). Improving Cooperativity in a Workflow Coordination Model over a Pub/Sub Network. Lecture Notes in Computer Science Ubiquitous Computing and Ambient Intelligence, 216-22. http://dx.doi.org/10.1007/978-3-642-35377-2_30

[30] Alcarria, R., Robles, T., Morales, A., López-de-Ipiña, D., & Aguilera, U. (2012). Enabling Flexible and Continuous Capability Invocation in Mobile Prosumer Environments. *Sensors*. 12 (7), 8930-8954. http://dx.doi.org/10.3390/s120708930

[31] Chen, W., Ge, Z., Kurose, J., & Towsley, D. (2005). Optimizing event distribution in publish/subscribe systems in the presence of policy-constraints and composite events. 13th IEEE International Conference on Network Protocols. 10, pp. 6-9. http://dx.doi.org/10.1109/ICNP.2005.31

[32] NS-3 Simulator. http://www.nsnamp.org (Accessed on February 2013)

[33] Moquette MQTT Broker. http://code.google.com/p/moquette-mqtt/ (last access February 2013).

[34] Westall, J.M., & Martin, J.J. (2011). Performance Characteristics of an Operational WiMAX Network. *IEEE Transactions on Mobile Computing*, 10 (7), 941-953. http://dx.doi.org/10.1109/TMC.2010.226

[35] Laner, M., Svoboda, P., Romirer-Maierhofer, P., Nikaein, N., Ricciato, F., & Rupp, M. (2012). A comparison between one-way delays in operating HSPA and LTE networks. 10th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, pp.286-292, 14-18 May 2012.

[36] Alcarria, R., Robles, T., Domínguez, A. M., & Cedeño, E. (2012). Re-solving Coordination Challenges in Cooperative Mobile Services. Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing), http://dx.doi.org/10.1109/IMIS.2012.131

Copyright Disclaimer