A multiscale approach for a distributed event-based Internet of Things

Denis Conan, Léon Lim, Chantal Taconet, Sophie Chabridon, Claire Lecocq

A multiscale approach for a distributed event-based Internet of Things

Denis Conan, Léon Lim, Chantal Taconet, Sophie Chabridon, and Claire Lecocq
SAMOVAR, Télécom SudParis, CNRS, Université Paris-Saclay
Télécom SudParis, 9 rue Charles Fourier, 91011 Évry Cedex, France
firstname.lastname@telecom-sudparis.eu

***Abstract*—The Internet of Things paradigm calls for exchanging data among dynamic and heterogeneous producer and consumer entities at unprecedented scales. The approach used in this paper to address IoT heterogeneity and scalability is through the modelling of multiple heterogeneous scales along dimensions of the application domain. Then, the concepts of scale and dimension of multiscalability are mapped to the concepts of scope and graph of scopes of distributed event-based systems, which have long been recognised as enabling scalable and flexible communication in a space-, time- and synchronisation- decoupled way. This multiscale approach for a distributed event-based Internet of Things enables the modelling of the decentralisation with human-centred edge computing solutions placing control at the edges of the IoT by leveraging localised scalability. Our implementation and experimentations with the MUDEBS framework show that multiscoping helps to drastically diminish the number of exchanged messages for both subscriptions and notifications.**

***Index Terms*—Internet of Things, Multiscale distributed system, Distributed event-based system, Content-based routing, Localised scalability, Scoping, Multiscoping.**

I. Introduction

The Internet of Things (IoT) paradigm calls for exchanging data among dynamic and heterogeneous producer and consumer entities at unprecedented scales [2], [5]. The publish/subscribe communication model [13], supported by Distributed Event-Based Systems (DEBS) [21], [31], has long been recognised as enabling scalable and flexible communication in highly distributed systems in a space-, time- and synchronisation-decoupled way. However, the dynamicity and heterogeneity characterising the IoT requires additional mechanisms. This paper addresses IoT heterogeneity and scalability through the modelling of multiple heterogeneous scales along dimensions of the application domain. This multiscale approach also enables full decentralisation with human-centred edge computing solutions [19] placing control at the edges of the IoT leveraging localised scalability [35]. We discuss in the following our design choices for a content-based DEBS solution deployed on an overlay of brokers.

In a DEBS, a producer publishes notifications, possibly following an advertisement message describing a set of notifications it is willing to publish, whereas a consumer subscribes to a set of messages it is interested in. Although producers initiate the communication, they do not know any consumer. If a notification matches a subscription, it is delivered to the consumer. Considering the dynamicity and diversity of producers and consumers, IoT systems are open and it is questionable to rely on solutions based on subject-based filtering (also called topic-based filtering) in which subscribers specify explicitly the topic of interest (via a few words with meta-characters that match for instance several characters or words) in their subscriptions [9], [31]. Instead, we adopt the full expressivity of content-based filters —i.e. they are constraints expressed on the whole content of notifications [8], [15], [30]. In addition, in order to stay as open as possible, we prefer using semi-structured data models *à la* XML [1], [10], [22] rather than structured data models that organise notifications as records of pairs (attribute name,value). The main reason is to interoperate with approaches such as sensors as a service and with information processing approaches such as reasoning with ontologies where data may be structured with OWL[1] (e.g. SSN[2]).

DEBS solutions are originally typically implemented as overlay networks of brokers [31]. The access broker of the consumer is responsible for installing the subscription filter on brokers of the overlay network so that notifications that match the subscription are routed towards the consumer. Next, broker-less epidemic solutions have been used to disseminate information with some filtering for decreasing the probability that they receive information of no interest [14]. The main issue with this approach is that, even when a peer knows that its neighbours are not interested in a given notification, it is difficult to state whether a given neighbour is critical to reach a consumer. To the best of our knowledge, filtering in broker-less solutions is topic-based (e.g. [12]).

More recently, DEBS solutions built over peer-to-peer (P2P) overlay networks have been proposed. As highlighted by [24], P2P overlay infrastructures represent a promising infrastructure to implement large-scale content-based DEBS since they are characterised by self-organisation and are highly flexible with respect to communication topology changes. Basically, P2P-based DEBS rely on the assumption that peers may act both as clients and servers, solving intrinsically the scalability requirement, as the number of potential servers increases linearly with the size of the system. However, to the best of our knowledge, existing P2P-based DEBS solutions assume either subject-based filtering or content-based filtering with

[1]https://www.w3.org/OWL
[2]http://www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628/

structured data models. In addition, in the context of the IoT with edge computing, hosts do not necessarily have the same capability and should not play the same role: for instance, some devices cannot communicate with each other due to NAT. Finally, by considering the P2P overlay infrastructure, the principle of locality may be violated since data are routed between peers regardless of their locality. For all of these reasons, we focus on DEBS solutions based on overlay networks of brokers and with a semi-structured data model.

DEBS for broad IoT face unprecedented scales in terms of the volume of exchanged data, number of participants and communication distance. As many brokers may be involved, a high quantity of messages may be exchanged when installing subscription filters and most importantly when routing numerous events from producers to consumers. In this work, we take benefit from the inherently heterogeneous nature of broad IoT systems to control and limit the amount of exchanged data. For example, when a consumer declares an interest in the transportation information concerning members of the group of UK supporters going to athletic competitions in Stade de France in the north of Paris, the brokers may deduce that some parts of the overlay network are not concerned. In other words, some sources of heterogeneity (e.g. geographical heterogeneity, group membership heterogeneity) may delimit visibility scopes [16] for data distribution, with notifications being visible only in certain scopes. We therefore combine the concept of multiscale distributed systems [33] with visibility scopes to bring into play the concept of multiscoping in a distributed event-based IoT, and we extend the requirements of distributed routing accordingly.

The remaining sections of this paper are structured as follows. Section II describes the multiscale approach we propose, illustrated on a Smart city scenario. The corresponding multiscale characterisation (viewpoints, dimensions, and scales) is defined, the graph of scopes are then derived with the associated visibility filters. Section III describes the concept of multiscoping allowing to specify scopes and visibility filters along multiple dimensions. Section IV summarises the principles of distributed routing with multiscoping that we implemented in our open source MUDEBS framework. Section V puts multiscoping in action for purposes of experimenting its impact on message traffic. Section VI then discusses related works focusing on scoping approaches enforced by an overlay of brokers. Section VII concludes the paper and gives some further research directions.

II. Multiscale approach

In order to master the complexity of routing and filtering notifications in the IoT, our approach builds on the concept of multiscale distributed systems [33] and implements it using multiscoping to control notification dissemination in a DEBS.

The word "multiscale" qualifies extremely diverse systems [7], [18], [25], [33]. Following [33], we consider the heterogeneity aspect according to several viewpoints and for each viewpoint several dimensions. For instance, in the geographical viewpoint, one may consider the administrative area or the distance dimensions; in the user viewpoint, one may consider the membership dimension to gather users into groups. Thereupon, a distributed system is qualified as being multiscale when the projection of its entities are associated with different scales in at least one dimension.

We match the system concept of scale to the DEBS concept of scope. Scopes are used for structuring publish/subscribe systems by putting the concept of visibility of notifications forward. Notification visibility limits the set of consumers that may get access to this notification. We take the definition presented in [16]: "A scope is an abstraction that bundles a set of clients (producers and consumers) in that the visibility of notifications published by a producer is confined to the consumers belonging to the same scope as the producer; a scope can recursively be a member of other scopes".

Thereafter, we abstract the customisability of DEBS with multiscoping —i.e. filtering is impacted by the visibility of notifications that are analysed according to several dimensions of scopes. A client advertises or subscribes providing a filter that is tagged with a set of scopes, with at most one scope per dimension. A notification is visible to a client if it is visible in all the dimensions (cf. Section III).

In Section II-A, we motivate the approach with an illustrative scenario. In Section II-B, we detail the multiscale characterisation obtained for this scenario. Then, we present in Sections II-C and II-D the benefits of this characterisation to manage the graph of scopes and to control the routing of events.

A. Smart city illustrative scenario

In preparation of the possible Olympic games in Paris in a near future, the city of *Paris* and the region *Île-de-France* are setting up smart services for the million of users of the public transportation. The services are able to provide customised applications to each of its users based on privacy requirements, location, group membership, and public transportation events (incidents, opportunities, etc.).

The system should provide information sharing functionalities for groups of users. For instance, supporters of the United Kingdom team may wish to meet at *Stade de France* for the athletics competitions. On the way to *Stade de France*, the service proposes to share information such as the current location of each member of a group, their current means of transport (e.g., bus line, taxis, subway car, suburban trains), and forecast arrival time. The group may be joined on-demand and at any time.

The system should control the dissemination of notifications crossing administrative areas according to system performance measurements. In particular, if the monitoring system detects a sudden deterioration of latencies due to a burst of traffic in some parts of the overlay network of brokers, the system should be able to isolate the brokers of some scopes to limit the quantity of notifications that can span the entire network. This (visibility) filtering can take into account the execution context at border crossing brokers and classifying notifications based on additional information such as priority or provenance.

The system of public cars proposes a service to indicate all the parking places located at a walking distance from a user, but this information is of no interest out of a given quarter. For this purpose the system limits the broadcast of these events to the concerned quarters.

B. Multiscale characterisation

In order to obtain the relevant scopes for this scenario, we characterise the multiscale nature of the IoT system in construction using the MuSCa[3] software framework [33], which follows a model-driven engineering approach [36]. The multiscale concepts, mainly viewpoints, dimensions and scales are defined in a meta-model. A specific editor guides the system designer to produce a multiscale characterisation and leads to a system-specific multiscale vocabulary —i.e. a model.

The left part of Figure 1 outlines the result of the multiscale characterisation for the illustrative scenario. The multiscale framework proposes pre-defined commonly-used viewpoints and dimensions and allows system designers to define new ones. We select the pre-existing *user* and *geography* viewpoints, and we add the *public transportation* viewpoint. Then, for each viewpoint, we choose the following dimensions of interest to characterise the system entities. For the *geography* viewpoint, we select the *geographical area* dimension, with the *region*, *department*, *city*, *quarter*, *street*, *building* scales. For the public transportation viewpoint, we define the *transport organisation* dimension that shows the organisation of the public transports into network companies organised into *intercity*, *suburban*, *area*, *city* and *dedicated destination* scales. The *city* scale of the *transport organisation* dimension exemplifies a situation in which a node can have several "parents", then leading to hierarchies of the form of trees, but also forests. Finally, for the *user* viewpoint, we design the *membership* dimension, with its hierarchical *group* scales.

C. From the multiscale characterisation to the graph of scopes

In its centre, Figure 1 shows graphs of scopes for the chosen dimensions. The scopes are instances of the scales defined in the model. For instance, the Versailles scope is an instance of the *City* scale (model level) for the *geographical area* dimension. For each dimension, we translate the hierarchy of scale instances into a hierarchy of scopes: Each scope has superscopes (parent scopes) and subscopes (children scopes).

The right part of Figure 1 presents the distributed architecture of an overlay network of brokers settled by an administrator for a given infrastructure deployment. In each dimension, a broker "belongs" to zero, one or several scope areas (cf. Sections IV and V for the construction of the scope areas with the JoinScope action). For instance, in dimension d_2, the Roissy area scope is managed by the brokers B_{10}, B_{11}, and B_{13}. These scope areas are either built statically at deployment time or dynamically on demand when a client wants to produce or consume notifications in a scope that is not already known by its access broker. Let us illustrate the dynamic extension of a scope area. When initially deployed, and since there is no client, the brokers are not aware of the groups of the *membership* dimension. And this is so even if the graph of scopes is fixed (i.e., the decomposition into subgroups is stable). Then, the overlay network of brokers managing the uk scope evolves: when a member of the group moves or connects to a broker that does not belong to the uk scope, the given broker joins the scope and the JoinScope action may lead to other brokers joining the scope to extend the scope area of the uk scope.

[3]https://fusionforge.int-evry.fr/www/musca/

D. From scopes to filters

The scopes are used to limit the dissemination of notifications inside one or a restricted number of scope areas. This limitation is performed through the means of visibility filters associated with the edges of the graphs of scopes. Each edge of the graphs of scopes is associated with a content-based visibility filter to filter out outgoing notifications from the subscope to the superscope, plus another visibility filter for filtering out the incoming notifications from the superscope to the subscope. These filters are installed at border crossing brokers, precisely at those knowing both of the involved scopes. The visibility filters may be installed by 1) the administrator of the infrastructure for all the system services, or 2) system services from the point of view of the producers.

Firstly, visibility filters may be installed by system administrators. For instance, the administrator may install visibility filters at brokers B_9 and B_{13} to control the traffic burst due to too numerous events produced during competitions at Stade de France and confine their dissemination to the Seine et marne scope. The filter may state by whatever means which ones come from official sources of information (or using many more resources, which notifications are potential duplicates), and filter out unofficial (and duplicate) notifications before forwarding them to the brokers of the Ile de france scope.

Secondly, visibility filters may be installed by system services. For instance, the group communication service can provide an option for producers to indicate whether their notifications should be kept in the scope of their production only (that information is included in every notification). The service can implement a content-based visibility filter that filters out outgoing notifications when the content of the notification indicates that the notification is declared as "producer-scope only". Finally, the service associates the visibility filter with the subscope-superscope relation of edges of the graph of scopes of the *membership* dimension so that they are installed at the border crossing brokers when the client joins a group. Clearly, these visibility filters target only scalability. They can be complemented with privacy management as proposed by MUDEBS with attribute-based access control [29].

In addition, it is noticeable that if the data model of some of the notifications includes the production scope, clients can use this information in advertisement and subscription filters for that kind of notifications. For instance, assume that

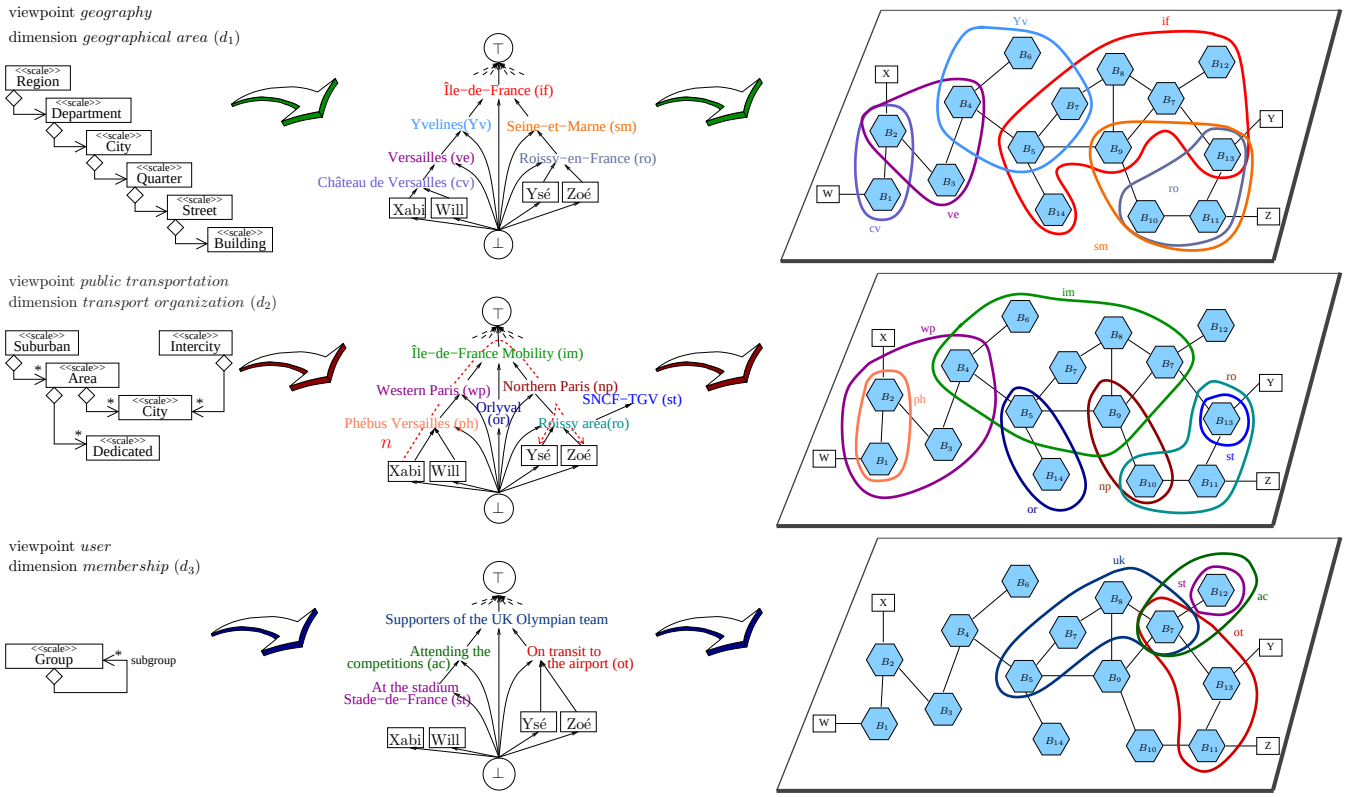

Fig. 1: Multiscale characterisation and DEBS with multiscoping: dimensions, scales, and scopes

the notifications that serve to the announcement of newly freed parking places contain the scope of the producer in the *geographical area* dimension. Then, clients searching for parking places in their vicinity can specify in their subscription filter that the scope of the producer must match their *quarter*.

III. DEBS WITH MULTIPLE GRAPHS OF SCOPES

In this section, we give an overview of the underlying concepts of multiscoping that organises multiple graphs of scopes in MUDEBS[4] [27], [28].

A. Graph of scopes of a dimension

We complement the API of regular DEBS with the management of scopes. Before advertising or subscribing, system administrators "partition" the system into scopes by tagging brokers. This is done through the new JoinScope action. When clients submit an advertisement or a subscription, they can specify the scopes, at most one per dimension. Thus, their notifications are forwarded only in these scopes: This is the role of distributed routing to limit the dissemination (cf. Section IV).

As depicted in Figure 1, the edges of the graphs of scopes, one graph per dimension, are the scales (translated into scopes in a DEBS) plus the clients —i.e. a client advertising or subscribing "in" a scope is linked to the scope. The vertices define the partial order relationship "is subscope of" from children scopes to parent scopes (noted $\lhd$, with its transitive closure $\overset{*}{\lhd}$), and the inverse relation "is superscope of" (noted $\rhd$, with its transitive closure $\overset{*}{\rhd}$). As shown in Figure 1, for the sake of convenience, we introduce the two specific scopes $\bot$ (bottom) and $\top$ (top) such that the graph of scopes of a dimension contains at least $\bot$ and $\top$, and such that $\bot$ is a subscope of any other scope of the graph and $\top$ is a superscope of any other scope of the graph. By construction, the sets of scopes of different dimensions do not intersect, except for $\bot$ and $\top$.

Edge directions in the graph indicate scope membership but notifications can travel in both directions. However, before "going through" a scope boundary, a notification must match a visibility filter established between the two scopes. Like forwarding filters[5], visibility filters are content-based. The act of applying a visibility filter is called visibility matching. This way, virtually speaking, notifications are "passed along" from scope to scope by matching visibility filters.

The following rules determine the set of eligible direct subscopes and superscopes to which a notification is visible. Firstly, when published, a notification is made visible to the scopes the producer belongs to. This rule is applied recursively to make outgoing notifications visible to all further superscopes. Secondly, when an incoming notification is visible

[4]https://fusionforge.int-evry.fr/www/mudebs/index.html

[5]This is the routing filter in classical DEBS without scoping that serves to forward a notification to a destination neighbouring broker or connected client.

within a scope, it is visible to all its children. This rule is applied recursively to make notifications visible to further children. Consequently, outgoing notifications are visible to all the superscopes and to all the sibling scopes. A path of scopes that connects producer X to consumer Y is called a visibility path and is decomposed into two, possibly empty, parts: an upward part and a downward part such that $X \overset{*}{\lhd} K \wedge K \overset{*}{\rhd} Y$. For instance, as depicted in the graph of scopes of the *public transportation* dimension, notification n published in the *Phébus Versailles* scope is visible to the consumers in the *Roissy area* scope (because $ph \overset{*}{\lhd} im \wedge im \overset{*}{\rhd} ro$) but is not visible to the consumers in the *SNCF-TGV* scope (because $im \overset{*}{\not\rhd} st$).

B. Visibility in several dimensions

In order to structure the overlay of network brokers according to several dimensions, we consider more than one graph of scopes. So, when submitting an advertisement or a subscription, a client provides a set of scopes, at most one per dimension. More precisely, Φ_f is a set of scope paths, each scope path containing initially only one scope. When publishing, a producer indicates the identity of the advertisement filter f and the notification is tagged with Φ_f. Therefore, each filter f is a triple (id_f, r_f, Φ_f), where id_f is the identifier of the filter (noted f in brief), r_f is the forwarding filter function of f and Φ_f is a set of scope paths associated to f (one path per dimension).

We now denote $X \overset{n}{\rightsquigarrow}_{\Phi_{X,f}} \Phi_{Y,f'} Y$ the visibility of notification n produced by producer X and matching the forwarding filter of advertisement filter f to consumer Y that subscribes to subscription filter f'. Since it is not reasonable to let designers specify a scope for every dimension of the multiscale characterisation, we use the specific scope $\perp$ in advertisements to indicate that $\perp$ should be used for every dimension not explicitly stated in the advertisement. In other words, the producer is imposing no constraint on the routing of the notification for that dimension. Similarly, we use the specific scope $\top$ in subscriptions to indicate that $\top$ should be used for every dimension not explicitly stated in the subscription. In other words, the consumer wants to be notified regardless the scope of the notification for that dimension. Therefore, notification n published by client X through advertisement filter f is visible to client Y through subscription filter f' if and only if n is visible from X to Y in all the dimensions appearing in $\Phi_f \cup \Phi_{f'} \setminus \{(\perp), (\top)\}$, with X being replaced by $\perp$ when $(s) \notin \Phi_f \wedge (\perp) \in \Phi_f$, and with Y being replaced by $\top$ when $(s) \notin \Phi_{f'} \wedge (\top) \in \Phi_{f'}$.

In conclusion, a distributed event-based system with multiscoping exhibits only traces satisfying the following requirements: (safety) (1) A client receives only notifications it is currently subscribed to; (2) A client receives only notifications that have been previously published by other clients and from which they are visible; (3) If the filter of the publication call does not belong to the active advertisements of the publishing client or if the notification does not match the filter specified in the publication call, the notification should not be delivered to any client; (4) A client receives a notification at most once; and (liveness) (5) A client eventually receives every notification that is visible to it and that matches some forwarding filters of its subscriptions.

In the next section, we complement the requirements with the distributed routing of notifications from producers to consumers.

IV. Distributed routing with multiscoping

For the sake of simplicity, muDEBS implements the simple routing approach with advertisements. Brokers broadcast the subscriptions in the overlay network of brokers and keep local the advertisements of their local clients. Techniques such as covering-based routing [8] could also be added to muDEBS. We now describe how the flooding of subscriptions and the routing of notifications are constrained by scopes.

At first, administrators partition the system into scopes by tagging brokers through the JoinScope actions. More precisely, the role of administrators is to invoke JoinScope actions at specified brokers to project each scope graph onto the network of brokers in order to delimit scope "areas" and so that the membership relation superscope and subscope between scopes are preserved. Afterwards, a client can request for the membership to some scopes that its access broker is not aware of; the access broker then calls the corresponding JoinScope action. When the JoinScope action is called at broker B with the argument $s \lhd t$, B updates its knowledge of the graph of scopes and sends a $\langle joinscope \rangle$ message to the neighbouring brokers that know t but do not know s so that (1) every future notification n tagged with scope s is forwarded to brokers in scope t if n matches the visibility filter of the relationship $s \lhd t$, and (2) the reception of a notification n tagged with scope t is forwarded to B and then to the brokers of the scope "area" of scope s if n matches the visibility filter of the relationship $t \rhd s$.

Visibility matching is a test that precedes any subscription or advertisement matching. Before forwarding a notification to a given destination D —i.e. a neighbouring broker or a local client— the receiving broker B verifies whether a visibility path associated with that notification can be built in the direction of D. The verification must be done not only according to the point of view of the producer for satisfying the constraints of the producer, but also according to the point of view of the consumer for satisfying the constraints of the consumer. Let Φ_n be the set of scope paths associated with the notification, and Φ_s be the set of scope paths associated with the subscription. From the point of view of the producer, a visibility path should exist for each dimension d in the set of dimensions associated with Φ_n. Similarly, from the point of view of the consumer, a visibility path should exist for each dimension d in the set of dimensions associated with Φ_s.

In a dimension, the routing of a notification follows a visibility path that is computed as the concatenation of the scope path of the notification and the scope path of the subscription as follows. At broker B, a scope path $pp = (X, s_1, s_2, \ldots, s_h, \ldots, s_i, \ldots, s_j)_d$ in

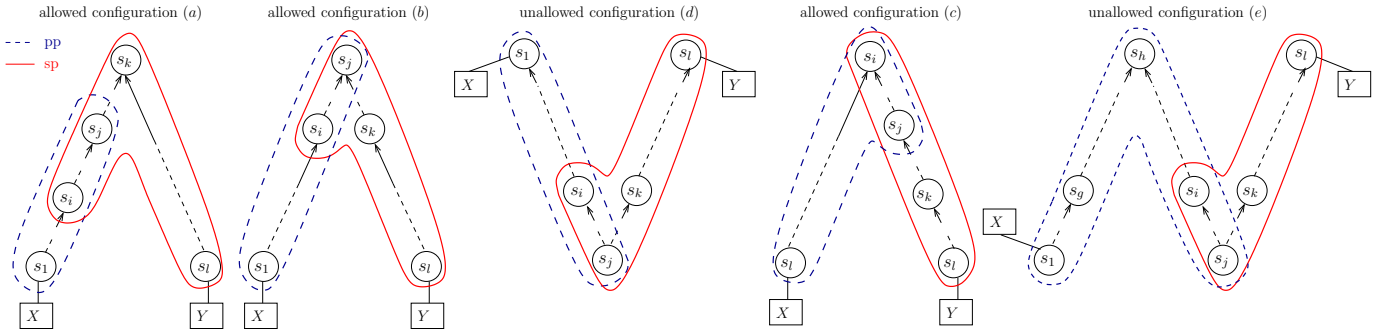

Fig. 2: Construction of a visibility path during routing

$\Phi_n \setminus \{(\perp),(\top)\}$ can be used to construct a visibility path $vp = (X, s_1, s_2, \ldots, s_i, \ldots, s_j, \ldots, s_{l-1}, s_l, Y)$ from X to Y if there exists a scope path $sp = (Y, s_l, s_{l-1}, \ldots, s_k, \ldots, s_j, \ldots, s_i)_d$ in $\Phi_s \setminus \{(\perp),(\top)\}$ such that s_{j+1} is an eligible next-hop scope of s_j. In other words, there exists a suffix of pp that is a prefix of the reverse path[6] of sp such that the rule "eligible next-hop scope" can be applied. Figure 2 depicts the three allowed configurations a, b, and c, and the unallowed configurations d and e. pp is surrounded by a dashed line and sp is surrounded by a continuous line. In configurations a and b, $(s_i...s_j)$ is in the upward part of vp and s_{j+1} is an eligible next-hop scope of s_j. In the configurations c and d, $(s_i...s_j)$ is in the downward part of vp and s_{j+1} is an eligible next-hop scope of s_j. In the configurations d and e, $(s_i...s_j)$ is in the downward part of vp, but s_{j+1} is not an eligible next-hop scope of s_j. Therefore, the construction of vp with pp and sp in configurations a, b, and c is allowed, but is not allowed in configurations d, and e.

V. DEBS WITH MULTISCOPING IN ACTION

The objective of scoping DEBS with one or multiple graphs of scopes is to limit the installation of subscription filters as well as the dissemination of notifications. In this section, we evaluate the impact of multiscoping on the overall message traffic with different topologies of overlay networks of brokers. The comparison is with "no scoping" and monoscoping —i.e. scoping with one graph of scopes.

The overall message traffic depends upon the way the projection of the graphs of scopes onto the overlay network of brokers is performed. Hence, we start with some rules for the projection of the graphs of scopes onto the graph of brokers before presenting the experiments.

A. Projection of the graphs of scopes onto the graph of brokers

The projection is specific to application domains, but we can provide indications of how it can be done. Administrators partition the system into scope areas by tagging brokers with the JoinScope action. Of course, when misused, this latter functionality may degrade the overall performance of the system. We propose the following rules of thumb for building the scope areas. Let $\mathcal{G}_d = (\mathcal{V}_d, \mathcal{E}_d)$ be a graph of scopes:

- $Rule_1$: $|\mathcal{E}_d|$ JoinScope actions are necessary.
- $Rule_2$: The calls JoinScope$(t \lhd \top)$ are the first calls to perform. The choice of the brokers for these calls may have a great impact on performance.
- $Rule_3$: Consider the "root" t such that a call JoinScope$(t \lhd \top)$ has already been executed at broker B_m. Assume that there is a client C with a scope path such that $X = k_1 \lhd k_2 \ldots k_i \lhd k_{i+1} \ldots k_{n-1} \lhd k_n = t$, and such that there exists a path of network links $(\theta(X) = B_1, B_2, \ldots B_j, \ldots B_m)$. Before calling JoinScope$(k_i \lhd k_{i+1})$ at broker B_j, a call JoinScope$(k_{i+1} \lhd k_{i+2})$ at broker $B_{j'}$ with $1 \leqslant j \leqslant j' \leqslant m$ should be performed.

$Rule_1$ stems from the fact that a graph of scopes is projected onto the network of brokers by adding edges one after the other. $Rule_2$ is specified in order to consider the top-down construction so that the edges at the top level are built first. The use of $Rule_3$ in conjunction with $Rule_2$ preserves the superscope and subscope relationships between scopes.

B. Overlay topologies of brokers and graphs of scopes

We use the GT-ITM Tool [20] to generate three overlay topologies Net_{16}, Net_{52} and Net_{100} composed of 16, 52, and 100 nodes, respectively. Figure 3 depicts for instance the topology used for 52 brokers. Each overlay topology has one transit domain built upon $x = 4$ nodes that form a ring. There are on average $y = 3$ connected graphs —i.e. stub domains— per transit node, without any extra transit-stub or stub-stub edges. Each stub domain has on average $\frac{|Net_i|-x}{y \times x}$ nodes, where $i \in \{16, 52, 100\}$. After the generation of the topology by GT-ITM, we adapt the connections between nodes inside each stub domain so that each stub domain contains a ring. For instance, in Figure 3, we identify the four transit nodes a–d. The transit node a is connected to three stub domains, where (a, a_1), (a, a_2) and (a, a_3) are transit-stub edges.

For all the topologies, there is one client per broker and the client plays both the roles of producer and consumer. Before advertising, subscribing, and publishing, each client obtains the set of scopes known by its access broker. For each combination of scopes, with at most one scope per dimension,

[6] sp is reversed because it has been built during the broadcasting of the subscription starting from the access broker Θ of the consumer, thus Θ being the first element of sp.

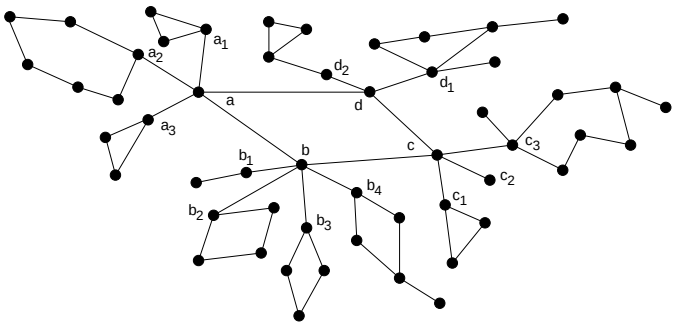

Fig. 3: Topology of the overlay of brokers Net_{52}

the client advertises a filter with a forwarding filter that always returns $true$ and with a visibility filter associated with the corresponding set of scope paths (at most one singleton scope path per dimension). Next, the client does the same with a call to subscribe. Finally, the client publishes one notification per advertisement filter.

If not stated otherwise, we assume that each graph of scopes is a hierarchy with 3 levels (without counting the levels corresponding to the clients and to the specific scopes $\top$ and $\bot$). The projection of the graphs of scopes follows $Rule_1$, $Rule_2$ and $Rule_3$. In addition, let s_1, s_2, s_3, s_4 and s_5 be five scopes such that $s_1 \neq \top$ is the root of the graph such that $s_1 \rhd s_2 \rhd s_4$ and $s_1 \rhd s_3 \rhd s_5$: i) action JoinScope$(s_1 \lhd \top)$ is called at broker d in the transit domain, ii) action JoinScope$(s_2 \lhd s_1)$ is called at neighbouring broker c in the transit domain, iii) action JoinScope$(s_4 \lhd s_2)$ is called at stub domain node d_1 ((d_1, d_2) is a transit-stub edge), iv) action JoinScope$(s_3 \lhd s_1)$ is called at the neighbouring broker a in the transit domain, and so on. When considering two or more graphs of scopes, we take the next transit node of c to be the broker at which action JoinScope$(t_1 \lhd \top)$ is called, where t_1 plays the same role as s_1.

Finally, if not stated otherwise, all the visibility filters return $true$.

C. Dissemination of subscriptions

System administrators can control the calls to the JoinScope actions so that some parts of the overlay network of brokers have no knowledge of the scopes of a certain dimension (as exemplified in the third dimension of Figure 1). In the following, we call a subset of brokers that manages the scopes of a dimension a "partition".

Scenario 1: Increasing the number of partitions. Every graph of scopes is projected onto a set of stub domains that are connected directly to the same transit node. We study in this scenario the intersection of partitions. Let D_1, D_2, and D_3 be the three dimensions considered. In experiment P_1, no partition intersect: D_1 is present in broker d, and in stub domains of d_1 and d_2; D_2 is present in broker a, and in stub domains of a_1, a_2, and a_3; D_3 is present in broker c, and in stub domains of c_1, c_2, and c_3. In experiment P_2, the two partitions D_1 and D_2 intersect: they spread over the same brokers (a, d, and those of stub domains of a_1, a_2, a_3, d_1 and d_2). In experiment P_3, the three partitions intersect.

Metric. The following metric is computed: the average number of $\langle$*subscribe*$\rangle$ messages per call to Subscribe. For each experiment P_1 to P_3, we compare the results with the case of no scoping —i.e. as if the clients specify $\bot$ in their advertisements and $\top$ in their subscriptions.

Results. In Figure 4, we observe that in the three experiments P_1, P_2 and P_3, the number of messages exchanged is less than 56% compared to the case of no scoping. In addition, using multiscoping is efficient to drastically diminish the number of $\langle$*subscribe*$\rangle$ messages: a 22% of decrease when using a second dimension, and a 36% of decrease when using two additional dimensions. In the next section, we study the impact of the projection of the graphs of scopes onto the topology with the placement of clients.

D. Dissemination of notifications

In the rest of the experiments, we do not "partition" the overlay network according to dimensions, but study the benefit of using visibility filters and of mixing several dimensions. The solution is evaluated with respect to i) the number of brokers, ii) the number of graphs of scopes, and iii) the level at which visibility filters always return $false$. We conduct the following series of experiments.

Scenario 2: Increasing the number of brokers and changing the level at which visibility filters return $false$**.** The size of the overlay network is successively increased by considering the three topologies Net_{16}, Net_{52}, and Net_{100}. Only one graph of scopes is considered. We study the impact of the placement of visibility filters that return $false$: i) all the visibility filters return $true$, ii) all the visibility filters at level 1 ($L1$) return $false$ (and only these filters), iii) all the visibility filters at level 2 ($L2$) return $false$ (and only these filters), and iv) all the visibility filters at level 3 ($L3$) return $false$ (and only these filters). The results are expressed using percentages: the experiment used as the reference is the "no scoping" one.

Scenario 3: Increasing the number of dimensions and changing the level at which visibility filters return $false$**.** For a given topology, namely Net_{52}, the number of dimensions is successively increased (1, 2 and 3 graphs of scopes).

Metrics. The following metrics are computed: the average number of $\langle$*notification*$\rangle$ messages per call to Publish, the average number of brokers involved per call to publish, and the average size of the routing table of brokers —i.e. entries with forwarding or visibility filter.

Results. In Figure 5, we observe that the number of $\langle$*notification*$\rangle$ messages with $false$ visibility filters at $L1$ is the least important, and is the most important at $L3$. In

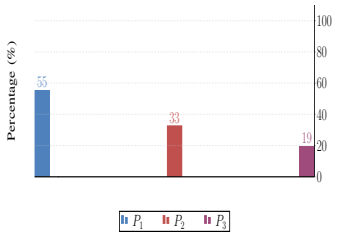

Fig. 4: Number of ⟨*subscribe*⟩ messages per call to Subscribe, three graphs of scopes, Net_{52}.

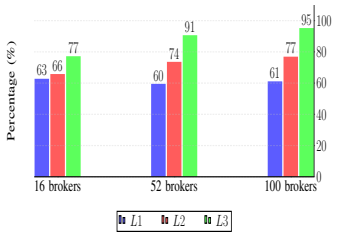

Fig. 5: Number of ⟨*notification*⟩ messages per call to Publish, one graph of scopes.

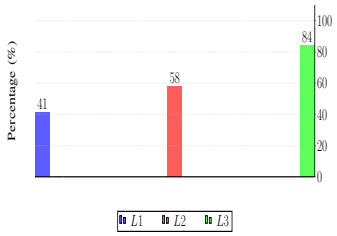

Fig. 6: Number of ⟨*notification*⟩ messages per call to Publish, three graphs of scopes, Net_{52}.

experiment $L3$, the clients that are connected to brokers that are the borders of the topology are isolated. The clients that are connected to brokers that are situated in the transit domain, that are transit-stub domain nodes, etc. are linked through visibility paths. Therefore, isolating clients that are at the borders is meaningful as long as the number of clients that are "in the centre" is not high. Since there is one client per broker in all the experiments, the number of messages exchanged in $L3$ is the most important. In addition, the difference between $L1$ and $L2$, and $L2$ and $L3$ with the Net_{16} topology are smaller than the ones with the Net_{52} and Net_{100} topologies. This is because the number of duplicate forwarding paths increases with the size of the network.

In Figure 6, the same phenomenon can be observed. Nonetheless, the number of exchanged messages is less important: for example, the percentage is 91 in the experiment $L3$ with one dimensions and 84 in the experiment $L3$ with three dimensions. This results from the mixing of several dimensions, as already observed in Scenario 1 for the ⟨*subscribe*⟩ messages.

VI. Related works

The concept of scope in [17] allows to organize a DEBS with a graph of scopes. This corresponds to what we can call monoscoping where only one scope graph is used at a time. [32] defines "divisions" as different sets of scopes (finance, makerting, production...). Divisions are subsumed into a global graph such that each division is represented by a top-level scope that is a subscope of the root scope of the global graph. This scope hierarchy does not allow to distinguish divisions as clearly as we propose with the concept of dimension. In addition, the root scope is different from the virtual scope $\top$ in two ways: 1) the root scope is the superscope of only top-level scopes whereas $\top$ is a superscope of every scope, 2) the root scope is defined as the "authority" scope that has access to notifications published inside a division and forwards to another division whereas $\top$ is not considered as a central authority. Indeed, in our solution, $\top$ is specified in order to ease the use of multiscoping, not for a global control of the visibility between dimensions. In addition, we do not require advertisements and subscriptions to be tagged for every dimension, thus allowing interoperability with scope-agnostic applications.

With Self-Organizing Broker Overlay [3], the authors propose to dynamically reorganise the overlay topology by avoiding pure forwarders involved in the routing but having no clients. The principle is to have a connection between brokers that match similar events. Brokers therefore run a self-organising algorithm in order to measure the similarity of interest with another broker. The filtering is content-based and a structured data model is used. In the context of the IoT wih edge computing, we rather consider another organisation where brokers have different roles: e.g. border brokers *versus* inner brokers, or proximity brokers *versus* brokers in Cloudlets or in Clouds.

Similarly to [3], the authors of [23] propose techniques for rewiring the broker overlay in order to avoid involving pure forwarder brokers. The solution introduces the notion of "subscription anchor" used to store subscription information in brokers' routing tables. From the perspective of a broker, an anchor for a subscription is a broker located up to Δ hops closer to the issuing subscriber (the anchor of a local subscriber points to the broker itself). Then, the authors present the anchor placement algorithm to propagate subscriptions — i.e. to build the overlay. We can notice that, as highlighted by [4], a bad placement may result in a high number of messages being propagated between brokers. Some optimisations are proposed to address this problem: by connecting subscribers with similar subscriptions to the same brokers [11], or by linking publications and their expected subscribers to the same brokers [26]. Again, we are in favor of a more distributed solution in which brokers are present for example to manage visibility filters.

VII. Conclusion

We advocate that the high potential of the Internet of Things can be reached only with applications built following fully decentralised software architectures enabling decoupled communication among dynamic and heterogeneous producers and consumers. Decoupling is required in time, space and synchronisation and should not be limited by physical constraints. Therefore such application software architectures should be deployed on highly distributed infrastructures like with edge computing and not on centralised systems such as clouds [19]. Distributed event-based systems are a good

candidate for providing decoupled communication as required by the IoT. However, the unprecedented scales envisioned in the IoT call for additional features.

This paper instantiates the semantically rich localised scalability concept [35] through a multiscale approach where multiple dimensions of multiple viewpoints may be modelled. Communication may then take place among geographically close entities, entities sharing common interests or entities with any kind of application-defined relationship without having to know each other in a direct way. A multiscale characterisation step allows to specify a graph of multiple scopes providing multiscoping and limiting event dissemination to some parts of the IoT. We propose an implementation of multiscoping in the MUDEBS framework that relies on content-based filtering with a semi-structured data model and we measure its performance gain on event dissemination. First experiments on micro-benchmarks show that multiscoping is efficient to drastically diminish the number of $\langle subscribe \rangle$ messages (a decrease of 45%, 67% and 81% when using 1, 2 and 3 dimensions).

Future research directions concern the enforcement of quality of service where our multiscale approach enables flexible and locality-aware QoS management as well as multi-level QoS management [6]. We are also investigating high-availability solutions such as in [34]. We promote a flexible approach in which the brokers self-reorganise to arrange redundant communication paths.

REFERENCES

[1] M. Altınel and M. Franklin, "Efficient Filtering of XML Documents for Selective Dissemination of Information," in *Proc. of VLDB*, 2000, pp. 53–64.

[2] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[3] R. Baldoni, R. Beraldi, L. Querzoni, and A. Virgillito, "Efficient Publish/Subscribe Through a Self-Organizing Broker Overlay and its Application to SIENA," *The Computer Journal*, vol. 50, no. 4, Jul. 2007.

[4] R. Barazzutti, P. Felber, C. Fetzer, E. Onica, J.-F. Pineau, M. Pasin, E. Rivière, and S. Weigert, "StreamHub: A Massively Parallel Architecture for High-Performance Content-Based Publish/Subscribe," in *Proc. 7th ACM DEBS*, Jul. 2013.

[5] P. Bellavista, A. Corradi, M. Fanelli, and L. Foschini, "A Survey of Context Data Distribution for Mobile Ubiquitous Systems," *ACM CS*, vol. 44, no. 4, Aug. 2012.

[6] P. Bellavista, A. Corradi, and A. Reale, "Quality of Service in Wide Scale Publish-Subscribe Systems," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1591–1616, Third Quarter 2014.

[7] G. Blair and P. Grace, "Emergent Middleware: Tackling the Interoperability Problem," *IEEE Internet Comput.*, vol. 16, no. 1, Jan. 2012.

[8] A. Carzaniga, D. Rosenblum, and A. Wolf, "Design and Evaluation of a Wide-area Event Notification Service," *ACM TOCS*, vol. 19, no. 3, pp. 332–383, Aug. 2001.

[9] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron, "Scribe: a large-scale and decentralized application-level multicast infrastructure," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 8, pp. 1489–1499, Oct. 2002.

[10] C.-Y. Chan, P. Felber, M. Garofalakis, and R. Rastogi, "Efficient filtering of XML documents with XPath expressions," in *Proc. 22nd IEEE ICDE*, Mar. 2002, pp. 235–244.

[11] A. Cheung and H.-A. Jacobsen, "Design and Evaluation of a Wide-area Event Notification Service," *ACM TOCS*, vol. 28, no. 4, Dec. 2010.

[12] C. Esposito, A. Castiglione, F. Palmieri, M. Ficco, and K.-K. Choo, "A Publish/Subscribe Protocol for Event-Driven Communications in the Internet of Things," in *Proc. 14th IEEE PICom*, Aug. 2016, pp. 376–383.

[13] P. Eugster, P. Felber, R. Guerraoui, and A.-M. Kermarrec, "The Many Faces of Publish/Subscribe," *ACM CS*, vol. 35, no. 2, Jun. 2003.

[14] P. Eugster, R. Guerraoui, A. Kermarrec, and L. Massoulie, "Epidemic information dissemination in distributed systems," *Computer*, vol. 37, no. 5, pp. 60–67, May 2004.

[15] E. Fidler, H. Jacobsen, G. Li, and S. Mankovski, "The PADRES Distributed Publish/Subscribe System," in *Proc. of the International Conference on Feature Interactions in Telecommunications and Software Systems*, 2005, pp. 12–30.

[16] L. Fiege, M. Cilia, and B. Mhl, "Publish-subscribe grows up: Support for management, visibility control, and heterogeneity," *IEEE Internet Comput.*, vol. 10, no. 1, pp. 48–55, Jan. 2006.

[17] ——, "Publish-subscribe grows up: Support for management, visibility control, and heterogeneity," *IEEE Internet Comput.*, vol. 10, no. 1, pp. 48–55, Jan. 2006.

[18] M. Franklin, S. Jeffery, S. Krishnamurthy, and F. Reiss, "Design Considerations for High Fan-in Systems: The HiFi Approach," in *Proc. 2nd Conference on Innovative Data Systems Research*, Jan. 2005.

[19] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric Computing: Vision and Challenges," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 5, pp. 37–42, Sep. 2015. [Online]. Available: http://doi.acm.org/10.1145/2831347.2831354

[20] GT-ITM Tool, "Modeling Topology of Large Internetworks," College of Computing, Georgia Tech, http://www.cc.gatech.edu/projects/gtitm/.

[21] A. Hinze, K. Sachs, and A. Buchmann, "Event-Based Applications and Enabling Technologies," in *Proc. 3rd ACM DEBS*, Jul. 2009, pp. 1–15.

[22] S. Hou and H. Jacobsen, "Predicate-based Filtering of XPath Expressions," in *Proc. 22nd IEEE ICDE*, Apr. 2006.

[23] R. Kazemzadeh and H.-A. Jacobsen, "Opportunistic Multipath Forwarding in Content-based Publish/Subscribe Overlays," in *Proc. 13th ACM/IFIP/USENIX Middleware*, Montreal, Quebec, Canada, Dec. 2012, pp. 249–270.

[24] A.-M. Kermarrec and P. Triantafillou, "XL Peer-to-Peer Pub/Sub Systems," *ACM CS*, vol. 46, no. 2, Nov. 2013.

[25] M. Kessis, C. Roncancio, and A. Lefebvre, "DASIMA: A Flexible Management Middleware in Multi-Scale Contexts," in *Proc. 6th International Conference on Information Technology: New Generations*, 2009.

[26] W. Li, S. Hu, J. Li, and H.-A. Jacobsen, "Community Clustering for Distributed Publish/Subscribe Systems," in *Proc. IEEE Cluster*, Sep. 2012, pp. 81–89.

[27] L. Lim and D. Conan, "Distributed Event-Based System with Multiscoping for Multiscalability," in *Proc. 9th Middleware Workshop on Middleware for Next Generation Internet Computing*, Bordeaux, France, Dec. 2014.

[28] ——, "Poster: Concept of Multiscoping for Distributed Event-based Systems," Jun. 2015, pp. 348–351.

[29] L. Lim, P. Marie, D. Conan, S. Chabridon, T. Desprats, and A. Manzoor, "Enhancing Context Data Distribution for the Internet of Things using QoC-awareness and Attribute-based Access Control," *Annals of Telecommunications*, vol. 71, no. 3/4, pp. 121–132, 2016.

[30] G. Mühl, "Generic Constraints for Content-Based Publish/Subscribe," in *Proc. 9th CoopIS*, ser. Lecture Notes in Computer Science, vol. 2172, Trento, Italy, Dec. 2001, pp. 211–225.

[31] B. Oki, M. Pfluegel, A. Siegel, and D. Skeen, "The information Bus: An Architecture for Extensible Distributed Systems," in *Proc. 14th ACM SOSP*, Dec. 1993, pp. 58–68.

[32] H. Parzyjegla, "Engineering Publish/Subscribe Systems and Event-Driven Applications," Ph.D. dissertation, University of Rostock, Germany, 2012.

[33] R. Rottenberg, S. Leriche, C. Taconet, C. Lecocq, and T. Desprats, "MuSCa: A Multiscale Characterization Framework for Complex Distributed Systems," in *Proc. 3rd Workshop on Model Driven Approaches in System Development*, Sep. 2014.

[34] P. Salehi, C. Doblander, and H.-A. Jacobsen, "Highly-available content-based publish/subscribe via gossiping," in *Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems*, ser. Proc. 10th ACM DEBS. ACM, 2016, pp. 93–104. [Online]. Available: http://doi.acm.org/10.1145/2933267.2933303

[35] M. Satyanarayanan, "Pervasive Computing: Vision and Challenges," *IEEE Personal Communications*, vol. 8, no. 4, pp. 10–17, Aug. 2001.

[36] D. C. Schmidt, "Guest editor's introduction: Model-driven engineering," *IEEE Computer*, vol. 39, no. 2, pp. 25–31, Feb. 2006. [Online]. Available: http://dx.doi.org/10.1109/MC.2006.58