# Distributed Event-Based System with Multiscoping for Multiscalability

Léon Lim and Denis Conan
Institut Mines-Télécom, Télécom SudParis
UMR CNRS SAMOVAR, Évry, France
<firstname>.<lastname>@telecom-sudparis.eu

## ABSTRACT

Distributed Event-Based System (DEBS) provides a versatile solution for asynchronously exchanging data in a distributed system, loosely-coupled in space and time. The software architecture of a DEBS is composed of an overlay network of brokers that are responsible for routing data from producers to consumers. An important issue is the cost (in terms of exchanged messages) of the installation of advertisement or subscription filters on the brokers and the cost of routing notifications. The problem is exacerbated in large and heterogeneous systems involving clouds, cloudlets, desktops, laptops, mobile phones, and smart objects of the Internet of Things (IoT). In this paper, we associate the system concept of scale (of multiscale distributed systems) with the concept of scope (of DEBS) and we introduce DEBS with multiscoping. We also extend the requirements of distributed routing to deal with multiscoping. In the context of the IoT, we show in an illustrative example that the solution allows application designers and system administrators to tag advertisements and subscriptions for semantically delimiting scopes that are superposed.

## Categories and Subject Descriptors

C.2.4 [**Computer Systems Organisation**]: Computer Communication Networks—*Distributed Systems.*

## General Terms

Algorithms, Design.

## Keywords

Middleware, Distributed Event-Based Systems, Multiscalability, Scoping, IoT.

## 1. INTRODUCTION

Among the many middleware approaches for designing emerging systems using the Internet of Things (IoT) [1], the publish/subscribe communication model [10] that is brought to play in Distributed Event-Based Systems (DEBS) [13, 16] belong to the most popular solutions to address sense and respond applications. The service architecture of a DEBS relies on an overlay network of brokers that are responsible for routing data from producers to consumers. Producers are clients that publish notifications whereas consumers are clients that react to notifications delivered to them by the system. An advertisement describes a set of notifications a producer is willing to publish. Producers initiate the communication but they do not know any consumer. A subscription describes a set of notifications a consumer is interested in. If a notification matches a subscription, it is delivered to the consumer. The access broker of the consumer is responsible for installing the subscription filter on all the brokers of the overlay network so that notifications are routed towards the consumer, regardless the access broker of the producers.

DEBS approaches exist to the distribution of context data coming from the numerous sensors of the IoT to the numerous clients that are connected to AMQP-standard[1] based brokers with topic-based filtering and with clustering facilities for scalability. These brokers are deployed on Clouds. We explore more generic content-based solutions for the following reasons: *i)* considering the diversity of the sources and consumers of context data, it is questionable to rely on agreed set of topics for all the current and future clients; *ii)* all the notifications must reach the Clouds, impairing the classical locality principle when the producers and the consumers are on the same host, the same network, the same Intranet, etc.; *iii)* end-users may be reluctant to publish the context data of the sensors in their vicinity to centralised services installed on Clouds and may prefer to choose other topologies with direct communication links between peers; *iv)* topic-based filters are hierarchical and less expressive as compared to content-based filters.

Many brokers may be involved and many messages may be exchanged when installing advertisement and subscription filters and when routing notifications. The cost in the terms of number of exchanged messages is an important issue in wide-area DEBS and especially in heterogeneous systems involving for instance clouds, cloudlets, desktops, laptops, mobile phones, and smart objects of the IoT. In these systems, system heterogeneity comes from device heterogeneity but also from the different technologies used for networking (impacting for instance latency, throughput) and computing (impacting for instance processing power, storage capacity), and from the end-users characteristics (such as social mem-


*MW4NG* '14 December 8, 2014, Bordeaux, France

http://dx.doi.org/10.1145/2676733.2676736.

[1] `http://www.amqp.org`

bership, geographic location), etc. When dealing with one type of heterogeneity, e.g. network heterogeneity, and one characteristic, e.g. network range, it is interesting to distinguish several ranges of levels or scales, e.g. PAN, LAN, WAN scales. Then, a distributed system that takes into account several of these scales is said to be a multiscale distributed system [7, 12, 15, 17].

Clearly, some of the sources of heterogeneity impact distributed notification routing. But, we claim that heterogeneity is also a part of the solution. For example, when a consumer declares an interest in the weather and living conditions of travelling family members on vacation abroad, the brokers may deduce that some parts of the overlay network are not concerned. In other words, some sources of heterogeneity delimit logical spaces and these spaces delimit scopes of data distribution, notifications being visible only in certain scopes. Then, subscription filters should be installed only on brokers belonging to these scopes and notifications should be disseminated only into these scopes.

The modelling and characterisation of the heterogeneity of distributed systems has been recently studied in the research domain of multiscale distributed systems [7, 12, 15, 17]. Our contribution builds on these works. In this paper, we associate the system concept of scale (of multiscale distributed systems) with the concept of scope (of DEBS), and we introduce muDEBS (multiscale Distributed Event-Based System) that is a DEBS with multiscoping for limiting the broadcasting of subscription filters and the forwarding of notifications only to scopes of the overlay network of brokers.

The work that is presented in this paper is conceptual in nature. It is implemented and being evaluated in french National Research Agency Project INCOME[2] whose goal is the provisioning of a generic middleware for the distribution of context data coming from the IoT. Due to the lack of space, algorithms are only sketched.

The paper is organised as follows. In Section 2, we motivate the multiscoping approach for dealing with multiscalability in event dissemination within DEBS. Then, in Sections 3 and 4, we define DEBS with multiscoping and distributed routing with multiscoping, respectively. Finally, we discuss some related works in Section 5 and we conclude in Section 6.

## 2. MOTIVATIONS AND OBJECTIVES

In recent studies of Internet computing systems, the expression "multiscale distributed system" is used to highlight the complexity due to several kinds of heterogeneity [7, 12, 15, 17]. In our study, we follow the approach of the authors of [17] who propose a definition of multiscalability, plus a characterisation framework and a vocabulary to use at design time and at run time. Multiscalability is defined as the ability to cover several scales into at least one dimension [17]. Architectural viewpoints [14] are studied through dimensions. A dimension is "a measurement of a particular characteristic for a particular viewpoint" and it is associated with a numeric or semantic measure. The important consequence is that by "using a measure, a dimension can be divided into ordered scales". The authors propose a vocabulary of viewpoints, dimensions, and scales. For instance, in the viewpoint user, it is worthwhile, e.g. for performance and data privacy issues of context data distribution, to consider the dimension membership to organise hierarchies of groups [6]. As another example, in the viewpoint network, there exist clearly several dimensions such as network range, latency, and bandwidth with their corresponding scales: PAN, LAN, MAN, WAN, etc. These discriminating scales are relevant to structure the management of resources [15]. As a last example, in the viewpoint geography, it is often interesting to make the distinction between the scales ambient, local, regional, global using the distance between entities. These scales are relevant to control the dissemination of sensor data [12]. The authors of [17] present other viewpoints, dimensions, and scales.

In the field of DEBS, the concept of "scope" is used to put the concept of visibility of notifications forward [11]. The visibility of a notification limits the set of consumers that may get access to this notification. We take the definition presented in [11]: "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". In this paper, we claim that the system concept of scale matches with the DEBS concept of scope, and we go further by abstracting the customisability of DEBS with multiscoping —i.e. distributed routing is impacted by the visibility of notifications that are analysed according to several dimensions. A client advertises or subscribes to a filter that is tagged with a set of scopes, at most one scope per dimension, and a notification is visible to a client if it is visible in all the dimensions. The objective is to limit the propagation of subscription filters and the distribution of notifications to only a part of the system.

Throughout the paper, we illustrate the solution with a context management service [9], and more particularly with context data from the IoT. As previously explained, the dimension membership of the viewpoint user is important when considering context data distribution. With the IoT, context data are used locally (e.g. in a smart space) and remotely (e.g. in another administrative areas). A dimension administrative area of the viewpoint geography is thus interesting for tagging subscriptions and forwarding notifications to specific areas. In addition, the scales of the dimension network range of the viewpoint network can help in deciding where to perform context data aggregation to limit broadcasting: e.g. in the local WiFi network on-board the bus or in the intranet of the transportation operator, or even in a public cloud.

## 3. DEBS WITH MULTISCOPING

In this section, we give an overview of the architecture of muDEBS. Then, we define the distributed system model and the interface of the system. Next, we proceed with the specification into three steps: visibility in a dimension for monoscoping, visibility in several dimensions for multiscoping, and DEBS with multiscoping.

### 3.1 Overview of the architecture

In order to ease the work of application designers, filters are content-based —i.e. they are constraints expressed on the whole content of notifications [10]. In addition, in order to stay as open as possible, we assume a semi-structured data model *à la* XML to interop with approaches such as sensors as a service that use standards like RDF. For the

[2] http://anr-income.fr/

sake of simplicity, distributed notification routing relies on the simple routing approach with advertisements —i.e. advertisements are kept local to the producers' access brokers and new subscriptions are flooded into the overlay network of brokers (into the limits of scopes) such that every broker (in these scopes) organises its routing table to forward notifications that match the subscription filters toward consumers. Techniques such as covering-based routing [10] are complementary to the solution that is presented in this paper.

We complement the API of regular DEBS with the management of multiscoping. Scales of a multiscale distributed system characterisation become scopes of DEBS. Before advertising or subscribing, system administrators "partition" the system into scopes by tagging brokers. This is done through the new operations `join_scope` and `leave_scope`. For instance, the system administrator may tag some brokers with the scale Europe, then others France, next Bordeaux, etc. When clients submit an advertisement or a subscription, they specify the scopes, at most one per dimension. A producer may propose context data for the scale of Bordeaux. Thus, her notifications may by default be forwarded only in the area of Bordeaux: This is the role of distributed routing to limit the dissemination. In other words, the idea is that an overlay of brokers join together producers and consumers that specify the scope Bordeaux. Secondly, scopes are organised into hierarchies and this is the role of the visibility filters of a scope to constrain the condition for the forwarding to subscopes and superscopes. Roughly speaking, notifications are visible to a client if they are visible to that client for every scope of every dimension of the set of scopes specified in the advertisement. A third impact of multiscoping is that subscription filters are only installed on brokers belonging to visible scopes.

We now define multiscoping, starting with the distributed system model.

## 3.2 Distributed system model

We consider a system made up of a finite set of brokers and clients that communicate by asynchronous message passing. Each broker is connected to a set of neighbouring brokers. A client is associated with a unique broker called its access broker. We also add the following assumptions. Processes are correct. Communication links are reliable and respect FIFO message ordering. Message delays are unbounded but finite. The topology of the network of brokers is a connected and static, and may contain cycles. The set of clients is dynamic. Finally, we assume that scope graphs are static and that calls to `join_scope` and `leave_scope` are performed before calls to `advertise`, `subscribe`, `publish`, etc.

## 3.3 Interface of the system

Clients interact with the system by invoking operations. They take parameters from different domains: the set of all the clients $\mathcal{C}$, the set of all the notifications $\mathcal{N}$, the set of all the advertisement and subscription filters $\mathcal{F}$, and the set of all the possible scopes $\mathcal{O}$ (which corresponds to the union of the sets of scopes of all the dimensions). Each filter $f \in \mathcal{F}$ is a triple $(id_f, r_f, \Phi_f)$, where $id_f$ is the identifier of the filter (noted $f$ in brief), $r_f$ is the forwarding filter function[3] of $f$ and $\Phi_f$ is a set of scope paths (defined in next section) associated to $f$. A forwarding filter is defined as follows: $r_f(n) = n$ (the notification $n$ matches the forwarding filter of $f$) or $r_f(n) = \epsilon \notin \mathcal{N}$ ($n$ does not match the forwarding filter), and $r_f(\epsilon) = \epsilon$ ($\epsilon$ matches no forwarding filter).

[3]The routing filter in classical DEBS without scoping, which serves to forward a notification to a destination.

## 3.4 Visibility in a dimension—Monoscoping

We define $\mathcal{D} = \{d, d', d_1, d_2, d_3 \dots\}$ to be the set of all the possible multiscale dimensions (or dimensions in brief). We also define $\mathcal{O}_d = \{s_1, s_2, s', s'', r, s, t, u \dots\} \cup \{\bot, \top\}$ to be the set of all the possible scopes of dimension $d$, with $\bot$ and $\top$ being two specific scopes introduced for the sake of convenience. By convention, the scope set of a dimension contains at least $\bot$ and $\top$. We assume that the sets of scopes of different dimensions do not intersect, except for $\bot$ and $\top$. We also define the set of all the components $\mathcal{K}_d$ to be the union of the disjoint sets of clients $\mathcal{C}$ and scopes $O_d$. The membership relation "superscope" between scopes in a dimension $d$ is defined by a directed acyclic graph (DAG) of components such that any two clients are not directly connected, and no scope, except $\bot$, is a subscope of a client. In addition, by convention, we state that $\bot$ is a subscope of any component and that $\top$ is a superscope of any component.

We define the partial order $\lhd$, which means "is subscope of", over the set of vertices $\mathcal{V}_d$ of the scope graph $G_d$. $\overset{*}{\lhd}$ denotes the transitive closure of the relation $\lhd$. The inverse relation of $\lhd$ corresponds to the relation "is superscope of" and is noted $\rhd$, and its transitive closure is denoted $\overset{*}{\rhd}$. Notice that edge directions in the scope graph indicate scope membership but notifications can travel in both directions.

Figure 1 depicts three illustrative scope graphs for the dimensions *membership*, *geographical administrative area* and *network range*, respectively. In the dimension *membership*, the scope graph is composed of clients $W$, $X$, $Y$ and $Z$, and scopes $es$, $us$, $fs$, $ir$, $ls$, and $is$. For the sake of simplicity, we do not draw all the edges ($scope$, $\top$). In this dimension, $X \lhd ls$, $ls \lhd us$, $us \lhd es$, and $X \overset{*}{\lhd} es$ hold. But, neither $us \overset{*}{\lhd} ir$ nor $ir \overset{*}{\lhd} us$ do hold.

Notifications may cross scope boundaries. However, before "going through" a scope boundary, a notification must match a visibility filter established between the two scopes. Like forwarding filters, visibility filters are content-based. The act of applying a visibility filter is called visibility matching. Visibility matching is a test that precedes any subscription or advertisement matching. A scope possesses an input interface and an output interface. The *input interface* (resp. *output interface*) is composed of a visibility filter such that a notification that matches the filter is visible to subscopes or scope members (resp. superscopes). In Figure 1, the notification $n$ is visible from scope $ls$ to scope $us$, and from scope $fs$ to scope $is$.

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, as depicted in Figure 1, 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 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.

In dimension $d$, the visibility of notification $n$ from $X$ to $Y$, which is noted $(X \overset{n}{\rightsquigarrow} Y)_d$ holds, if and only if one of the following requirements is satisfied: *a)* $X = Y$; *b)* There exists a component $s$ such that $X \lhd s$, $n$ matches the visibility filter from $X$ to $s$, and $n$ is visible from $s$ to $Y$; *c)* There exists a component $s$ such that $s \rhd Y$, $n$ is visible from $X$ to $s$, and $n$ matches the visibility filter from $s$ to $Y$. When the context is clear, we omit the subscript $d$. In addition, by convention, we state that $\forall K \in \mathcal{K}_d \backslash \{\bot, \top\}, \forall n \in \mathcal{N}, \neg(K \overset{n}{\rightsquigarrow} \bot) \wedge (\bot \overset{n}{\rightsquigarrow} K) \wedge (K \overset{n}{\rightsquigarrow} \top) \wedge \neg(\top \overset{n}{\rightsquigarrow} K)$.

A path of scopes that connects a producer $X$ to a consumer $Y$ is called a visibility path and is decomposed into two, possibly empty, parts: an upward part and a downward part such that $X \overset{*}{\lhd} K \wedge K \overset{*}{\rhd} Y$. In Figure 1, the visibility path $(X, ls, us, es, fs, is, Y)$ is composed of the upward part $(X, ls, us, es)$ and the downward part $(es, fs, is, Y)$.

## 3.5 Several dimensions—Multiscoping

The multiscalability issue leads to consider more than one dimension, and thus 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, $\Phi_f$ 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 $\Phi_f$.

We can now define the visibility of a notification $n$ produced by a producer $X$ and matching the forwarding filter of the advertisement filter $f$ to a specific consumer $Y$ that subscribes to a subscription filter $f'$, denoted $X \overset{n}{\rightsquigarrow}_{\Phi_{X,f}, \Phi_{Y,f'}} Y$. The usage of the pairs $(X, f)$ and $(Y, f')$ is for allowing several advertisements and subscriptions per client. In addition, since it is not reasonable to let designers specify a scope for every dimension of the multiscale characterisation of the distributed system, we use the specific scope $\bot$ in advertisements to indicate that $\bot$ should be used for every dimension not explicitly specified in the advertisement —i.e. the evaluation of the expression $X \overset{n}{\rightsquigarrow} Y$ is replaced by the evaluation of the expression $\bot \overset{n}{\rightsquigarrow} Y$. 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 specified in the subscription —i.e. $\top$ is substituted to $Y$ in the evaluation of the expression $X \overset{n}{\rightsquigarrow} Y$. In other words, the consumer wants to be notified regardless the scope of the notification for that dimension. Therefore, a notification $n$ published by client $X$ through the advertisement filter $f$ is visible to client $Y$ through the subscription filter $f'$ if and only if $n$ is visible from $X$ to $Y$ in all the dimensions used in the set $\Phi_f \cup \Phi_{f'} \setminus \{(\bot), (\top)\}$, with $X$ being replaced by $\bot$ when $(s) \notin \Phi_f \wedge (\bot) \in \Phi_f$, and with $Y$ being replaced by $\top$ when $(s) \notin \Phi_{f'} \wedge (\top) \in \Phi_{f'}$.

In Figure 1, let $W$ and $X$ be two producers, and $Y$ and $Z$ be two consumers. Given the following sets of scope paths for the advertisements and the subscriptions: $\Phi_{W,f} = \{(ls), (ch)\}$, $\Phi_{X,f} = \{(ls), (ch), (\bot)\}$, $\Phi_{Y,f'} = \{(is), (lo), (tm)\}$, and $\Phi_{Z,f'} = \{(is), (tm), (\top)\}$, we can deduce that: $W \overset{}{\rightsquigarrow}_{\Phi_{W,f}, \Phi_{Y,f'}} Y$ does not hold because $\neg(W \rightsquigarrow Y)_{d_3}$; $W \overset{}{\rightsquigarrow}_{\Phi_{W,f}, \Phi_{Z,f'}} Z$ does not hold because $\neg(W \rightsquigarrow Z)_{d_3}$; $X \overset{}{\rightsquigarrow}_{\Phi_{X,f}, \Phi_{Y,f'}} Y$ does hold because $(X \rightsquigarrow Y)_{d_1} \wedge (X \rightsquigarrow Y)_{d_2} \wedge (\bot \rightsquigarrow Y)_{d_3}$; and $X \overset{}{\rightsquigarrow}_{\Phi_{X,f}, \Phi_{Z,f'}} Z$ does hold because $(X \rightsquigarrow Z)_{d_1} \wedge (X \rightsquigarrow \top)_{d_2} \wedge (\bot \rightsquigarrow Z)_{d_3}$.

## 3.6 DEBS with multiscoping

With or without multiscoping, a notification matches an advertisement filter if it matches the forwarding filter. With multiscoping, at the access broker of the producer, the set of scope paths of the advertisement filter becomes the set of scope paths of the notification. A notification matches a subscription filter at the access broker of the consumer if a visibility path can be deduced from the set of scope paths of the notification $\Phi_n$ and of the subscription $\Phi_s$, and if the notification matches the forwarding filter of the subscription.

A distributed event-based system with multiscoping exhibits only traces satisfying the following requirements: (safety) *a)* A client receives only notifications it is currently subscribed to; *b)* A client receives only notifications that have previously been published; *c)* A client receives only notifications published by other clients and from which they are visible; *d)* A client receives a notification at most once; and (liveness) *e)* A client eventually receives every notification that is visible to it and that matches forwarding filters of its subscription.

These requirements define the properties of muDEBS when seen as a black box. This specification does not suit when designing the routing part of the system because it does not specify the central role of brokers in the dissemination of notifications. In other words, we need a property that only depends on the configurations of neighbouring brokers for taking decisions on routing matter. In the next section, we focus on distributed routing with multiscoping.

# 4. DISTRIBUTED ROUTING WITH MULTISCOPING

In this section, we introduce the requirements for distributed routing with multiscoping that satisfy muDEBS, and then we provide details on the main algorithms.

## 4.1 Requirements

We reify a scope as an overlay of brokers and clients, with the specific scopes $\bot$ and $\top$ being virtual. Scope overlays are built such that two overlays of brokers corresponding to two scopes related by the relationship "superscope" intersect[4]. At a broker, say $B_s$, a notification $n$ tagged with the set of scope paths $\Phi_n$ can be either forwarded to broker $B_r$, or "mapped up or down". In the first case, $n$ is routed within an overlay of brokers belonging to the same scope and $n$ does not go across any scope boundary —i.e. $\Phi_n$ does not change. In the second case, $\Phi_n$ is transformed to $\Phi'$ by $B_s$, which is a broker at the boundary of several scopes, so that $n$ becomes visible to its superscope(s) or subscope(s). Thus, $n$ may be routed from broker to broker or transformed several times before reaching a given destination.

The routing of a notification involves two kinds of treatments: forwarding between neighbouring brokers and scope transformation at a broker. The forwarding is the routing of classical DEBS without scoping. The transformation of a set of scope paths leads to a new set of scope paths that differ by a single scope path, the two dissimilar scope paths being related by the relationship "superscope"[5]. Subsequently, we model the distributed routing of a notification

[4]This assumption must be taken into account by the administrator when using `join_scope` and `leave_scope`.

[5]$\bot \lhd s$ and $s \lhd \top$ are particular cases to take into account.

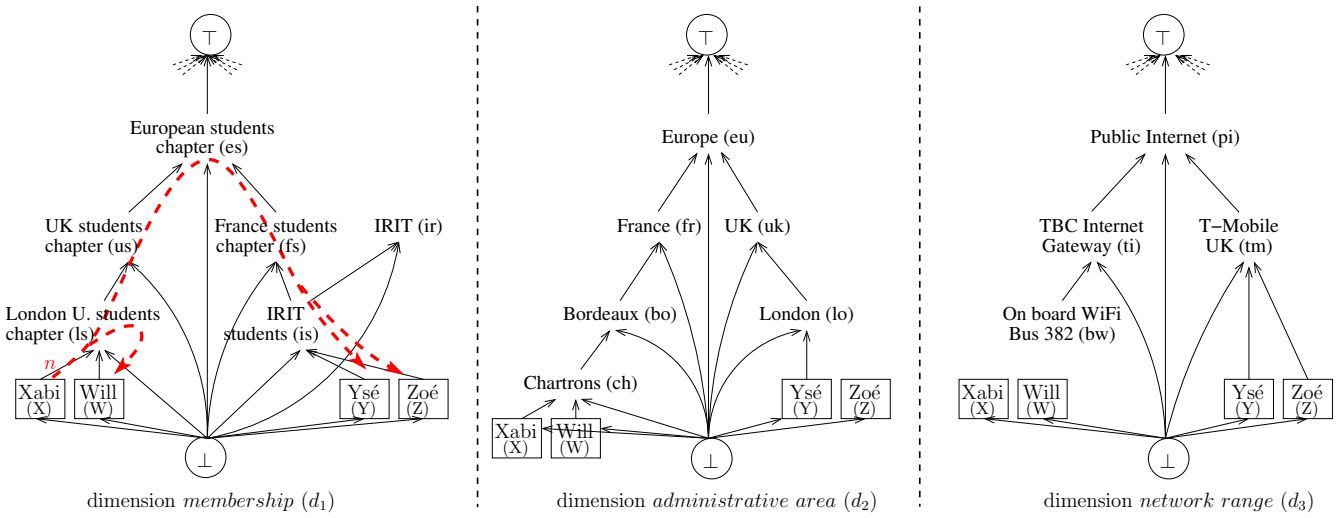


**Figure 1: Dimensions, scopes, clients, and visibility** (superscope relations $k \lhd \top$ are not drawn)

as a graph of transitions, which are either broker forwardings or scope transformations. The graph is an edge-labelled DAG of nodes that are pairs (broker, set of scope paths) and of edges that are of two kinds: either "$\rightarrow$" to represent the act of forwarding from a broker to another one within the same set of scope paths, or "$\curvearrowright$" to represent the act of transforming the set of scope paths at a broker. The node (access broker of the consumer, set of scope paths of the subscription) is the root of the graph. A transition path of a notification is a path in the transition graph.

A transition path can be reduced to a forwarding path by removing the set of scope paths at each node, and then removing the repetitions of consecutive identical brokers. A forwarding path conforming to the topology of the overlay network of brokers is called a delivery path. Similarly, a transition path can be reduced to a transformation path by removing the broker at each node, and then removing the repetitions of consecutive identical sets of scope paths. A transformation path is a visibility path if all the transformation paths per dimension are visibility paths. In order to control the transformation path at a broker so that only valid transformations are applied, we complement the set of scope paths of a notification $n$ with the set $\Lambda$ of booleans stating, for each dimension, whether $n$ is in the upward or downward part of a visibility path.

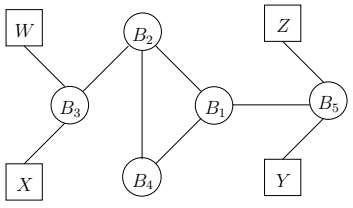


**Figure 2: Network of brokers**

Figure 2 depicts a network of brokers. Let consider that $Y$ subscribes to a filter with the set of scope paths $\{(is),(lo)\}$, and that $X$ advertises a filter with the set of scope paths $\{(ls),(ch)\}$ and publish a notification $n$ through this advertisement, and that $n$ matches the forwarding filters of the advertisement and the subscription filters. A possible transition path from $(B_3,\{(ls),(ch)\})$ to $(B_5,\{(is),(lo)\})$ is $((B_3,\{(ls),(ch)\})$, $(B_3,(\{(ls,us),(ch)\})$, $(B_3,\{(ls,us),(ch,bo)\})$, $(B_3,\{(ls,us),(ch,bo,fr)\})$, $(B_2,\{(ls,us),(ch...fr)\})$, $(B_2,\{(ls,us,-es),(ch...fr)\})$, $(B_2,\{(ls...es),(ch...fr,eu)\})$, $(B_1,\{(ls...es),(ch,...-eu)\})$, $(B_1,\{(ls...es,fs),(ch...eu)\})$, $(B_1,\{(ls...fs),(ch...eu,uk)\})$, $(B_5,\{(ls...fs),(ch...uk\})$, $(B_5,\{(ls...fs,is),(ch...uk)\})$, $(B_5,\{(ls...-is),(ch...lo)\}))$. This transition path can be reduced to the delivery path $(B_3,B_2,B_1,B_5)$ and to the visibility path of dimension *membership* to $((is),(is,ls),(is,ls,us),(is...us,es),-(is...es,fs),(is...fs,is))$.

While being routed, a notification is tagged with the set of scope paths $\Phi_n$, one scope path per dimension. For instance, in the above example of transition path, the scope path of the notification when arriving at broker $B_2$ is $(ls,us)$ for $d_1$. In order to route a notification tagged with a set of scope paths, each broker classically maintains a routing table. The multiscoping aspect is taken into account by organising routing table entries into triples (set of scope paths $\Phi_s$, forwarding or visibility filter $f$, destination $D$). In the case of a forwarding filter, an entry is read as follows: If $D$ is a client and if a visibility path can be built from $\Phi_n$, and if $n$ matches $f$, then $n$ is forwarded to $D$; If $D$ is a broker and if a visibility path can be built from $\Phi_n$ and $\Phi_s$, and if $n$ matches $f$, then $n$ is forwarded to $D$. In the case of a visibility filter, $\Phi_s$ contains only one scope path $sp$ and $sp = (s)$; an entry is read as follows: there exists a scope path $sp_n$ in $\Phi_n \setminus \{(\bot),(\top)\}$ such that a new path $sp'$ can be built with $sp_n$ and $sp$ —i.e. $s$ is an eligible next-hope scope of $last(sp_n)$— and if $n$ matches $f$; then $\Phi_n$ is transformed and the destination $D$ is the same broker. Therefore, a distributed routing algorithm with multiscoping exhibits only traces satisfying the following requirements: (local validity) *a)* Only notifications that match the forwarding filter of their advertisement filter at the producer's access broker are forwarded; *b)* Only notifications that match one of the forwarding filter of the subscription filters at the consumer's access broker are delivered and they are delivered immedi-

ately; (eventual monotone remote validity) *c)* The routing table entries with forwarding filters build delivery paths; and *d)* The routing table entries with visibility filters build visibility paths.

## 4.2 Principles of algorithms

Since brokers forward the notifications according to a simple routing mechanism, we only sketch the handling of *joinscope* and *subscribe* messages.

A `join_scope(s,t)` operation is invoked by a system administrator on any broker or by a client on the access broker. The broker updates its knowledge of the scope graph and forwards the new scope relation to interested neighbours in *joinscope* messages. To do so, each broker maintains a scope look-up table (SLT) that indicates in which directions (in terms of neighbouring brokers) clients belonging to a scope can be found. An entry of SLT is a pair (scope, destination broker). The broker forwards the *join_scope* message to the neighbouring broker $D$ if there is no SLT entry $(t,E)$ satisfying $s \lhd t \wedge D \neq E$.

A `subscribe(f)` operation is invoked by a client on its access broker. The broker adds a "forwarding" entry directed towards the sender of the subscription and also computes all the transformations of the set of scope paths of the subscription filter to update the routing table. Then, it forwards the *subscription* message to neighbouring brokers such that there exists a SLT entry $(t,D)$ satisfying $(s = t \vee s \overset{*}{\lhd} t)$ and $D$ has not already seen the subscription.

# 5. RELATED WORKS

We define a relationship between multiscalability [7, 15, 17] and multiscoping, and show how that relationship can be fruitful for dealing with heterogeneity issues in the distributed algorithms for DEBS routing.

The concept of scope was introduced in [11], but the requirements of distributed routing were not explicited. We formalise them, extend them to multiscoping. In addition, contrarily to [11], our solution does not impose the tagging of advertisements and subscriptions for every dimension, thus the interoperability with scope-agnostic applications is insured.

The scoping approach for improving event distribution is complementary to the solutions that scale up or down the internal infrastructure of a broker (e.g. publication partitioning and subscription partitioning [5], parallel filtering [4]), that aggregates subscriptions [8], that reorganise the overlay network of brokers [3], or that reorganise the overlay of hierarchical agent-based nodes [2].

# 6. CONCLUSION

DEBS provides a versatile solution for asynchronously interconnecting clients that exchange data in a distributed system, loosely-coupled in space and time. Some sources of heterogeneity such as network range capabilities, geographic location, or social membership that characterise multiscale systems impact distributed routing. In this paper, we have shown that the DEBS concept of scope is beneficial to the management of DEBS that target emerging multiscale Internet communication architecture (e.g. on the Internet of Things). We have introduced the specifications of a distributed routing with multiscoping. Due to the lack of space, the algorithms have only been sketched. They are implemented as an open source solution[6], and we are currently planning their performance evaluation.

# 7. REFERENCES


[1] L. Atzori, I. Antonio, and G. Morabito. The Internet of Things: A survey. *Computer Networks*, 54(5), May 2010.

[2] D. Balakrishnan and A. Naya. Adaptive Context Dissemination in Heterogeneous Environment. *IEEE Transactions on Mobile Computing*, 13(6), 2014.

[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*, 50(4), July 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. ACM DEBS*, July 2013.

[5] R. Barazzutti, P. Felber, H. Mercier, E. Onica, J. Pineau, E. Rivière, and C. Fetzer. Infrastructure Provisioning for Scalable Content-based Routing: Framework and Analysis. In *Proc. IEEE NCA*, Aug. 2012.

[6] P. Bellavista, A. Corradi, M. Fanelli, and L. Foschini. A Survey of Context Data Distribution for Mobile Ubiquitous Systems. *ACM CS*, 44(4), Aug. 2012.

[7] G. Blair and P. Grace. Emergent Middleware: Tackling the Interoperability Problem. *IEEE Int. Comp.*, 16(1), Jan. 2012.

[8] R. Chand and P. Felber. Scalable Distribution of XML Content with XNet. *IEEE TPDS*, 19(4), Apr. 2008.

[9] J. Coutaz, J. Crowley, S. Dobson, and D. Garlan. Context is Key. *COMM. ACM*, 48(3), Mar. 2005.

[10] P. Eugster, P. Felber, R. Guerraoui, and A.-M. Kermarrec. The Many Faces of Publish/Subscribe. *ACM CS*, 35(2), June 2003.

[11] L. Fiege, M. Cilia, and B. Mühl. Publish-subscribe grows up: Support for management, visibility control, and heterogeneity. *IEEE Int. Comp.*, 10(1), Jan. 2006.

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

[13] A. Hinze, K. Sachs, and A. Buchmann. Event-Based Applications and Enabling Technologies. In *Proc. ACM DEBS*, July 2009.

[14] ISO/IEC/IEEE. Systems and software engineering — Architecture description. International Standard ISO/IEC/IEEE-42010:2011, Dec. 2011.

[15] 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.

[16] G. Mühl, L. Fiege, and P. Pietzuch. *Distributed Event-Based Systems*. Springer, 2006.

[17] R. Rottenberg, S. Leriche, C. Taconet, C. Lecocq, and T. Desprats. MuSCa: A Multiscale Characterization Framework for Complex Distributed Systems. In *3rd MDASD*, Sept. 2014.


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