On Event-Based Middleware for Location-Aware Mobile Applications

René Meier, *Member*, *IEEE*, and Vinny Cahill, *Member*, *IEEE Computer Society*

Abstract—As mobile applications become more widespread, programming paradigms and middleware architectures designed to support their development are becoming increasingly important. The event-based programming paradigm is a strong candidate for the development of mobile applications due to its inherent support for the loose coupling between components required by mobile applications. However, existing middleware that supports the event-based programming paradigm is not well suited to supporting location-aware mobile applications in which highly mobile components come together dynamically to collaborate at some location. This paper presents a number of techniques including location-independent announcement and subscription coupled with location-dependent filtering and event delivery that can be used by event-based middleware to support such collaboration. We describe how these techniques have been implemented in STEAM, an event-based middleware with a fully decentralized architecture, which is particularly well suited to deployment in ad hoc network environments. The cost of such location-based event dissemination and the benefits of distributed event filtering are evaluated.

Index Terms—Distributed systems, middleware, publish subscribe, event-based communication, mobile computing, collaborative and location-aware applications, wireless ad hoc networks.

1 Introduction

Emerging pervasive and mobile computing applications comprise large numbers of interacting components distributed over large geographical areas. Examples include context-aware intelligent transportation systems [1], [2] and city-wide information systems [3], [4]. Middleware to support such applications must deal with the increased complexity that arises from such scale, from the geographical dispersion of components, and from the spontaneously changing connections between components.

Such mobile applications can be characterized as collaborative in the sense that mobile entities use a wireless network to interact with other mobile entities that have come together at some common location. Examples might include tourists visiting the same site or vehicles traveling in the same direction. Having come together in some area, collaborative entities establish connections with other collaborative entities dynamically, temporarily forming a group that has a common goal. The members of such a group may even travel together for a period of time, as in the case of a group of tourists coming together and deciding to participate in a guided tour or a group of vehicles traveling in the same direction forming a convoy to improve driver safety and reduce fuel consumption [5], [6]. Although these collaborative applications may use infrastructure networks, they will often use ad hoc networks [7] since these are immediately deployable in arbitrary environments and support communication without the need for a separate infrastructure. For pervasive and mobile computing as well as sentient computing [8], this collaborative style of application allows loosely coupled, highly mobile components to communicate and collaborate in a spontaneous manner anywhere and at any time. In many cases, such applications will be deployed in situations where wireless network infrastructure might be available. For example, museums may deploy wireless local area networks and proposals exist for large-scale deployment of wireless access infrastructure along roads such as the Vehicle Infrastructure Integration initiative in the United States [9]. However, we argue that such applications cannot rely on the presence of such infrastructure: Tourist attractions such as national parks or archaeological sites are unlikely to have such infrastructure and cost mitigates against ubiquitous wireless infrastructure being deployed on every road. Moreover, there are many collaborative mobile applications that will never be able to avail of such infrastructure such as coordination of Unmanned Aerial Vehicles (UAVs) or autonomous military vehicles deployed in hostile environments.

In principle, event-based communication [10], [11] is well suited to such mobile applications [12], [13], [14], [15] since it naturally accommodates a dynamically changing population of interacting entities and the dynamic reconfiguration of the connections between them. Event-based communication supports asynchronous interconnections between components and is particularly useful where communication relationships among components are dynamically and frequently reconfigured during the lifetimes of the entities. The event-based communication model supports a one-to-many or many-to-many communication pattern that allows one or more entities to react to a change in the state of another entity. Event notifications, or simply events, contain the data representing the change to the state of the sending entity. They are propagated from the generating entities,

- *The authors are with the Distributed Systems Group, School of Computer Science and Statistics, Trinity College Dublin, Ireland, and with Lero—The Irish Software Engineering Research Centre.*
E-mail: {rene.meier, vinny.cahill}@cs.tcd.ie.

Manuscript received 24 July 2007; revised 28 Jan. 2008; accepted 13 May 2009; published online 4 Dec. 2009.
Recommended for acceptance by W. Emmerich.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSE-2007-07-0226.
Digital Object Identifier no. 10.1109/TSE.2009.9004.

0098-5589/10/$26.00 © 2010 IEEE Published by the IEEE Computer Society

called the producers, to the receiving entities, called the consumers. Events typically have a name and may have a set of typed attributes whose specific values describe the specific change to the producer's state. A particular consumer may only be interested in a subset of the events produced in the system. Event filters provide a means to control the propagation of events. Ideally, filters enable a particular consumer to specify the exact set of events in which it is interested [15]. Essentially, a consumer's event filters are matched against the events received by the middleware and only events for which the matching produces a positive result are subsequently delivered to the consumer application [16].

Unfortunately, existing research on event-based middleware for wireless networks, such as [10], [12], [13], [17], [18], [19] has mainly focused on what may be termed nomadic applications. These applications are characterized by the fact that mobile entities make use of the wireless network primarily to connect to a fixed network infrastructure, such as the Internet, but may suffer periods of disconnection while moving between points of connectivity. Such applications typically employ infrastructure networks [7]. Much of this work has concentrated on handling disconnection while entities move from one access point to another, whereas relatively little work has addressed the distinct requirements of collaborative mobile applications, especially those that use ad hoc networks. For event systems to support collaborative mobile applications, they must enable collocated producers and consumers to be able to discover each other. Producers need to be able to advertise the events they intend to generate independently of their location and consumers must be able to subscribe to events persistently. Consumers must also be able to discover events of interest and to eventually deliver them at the relevant locations.

In this paper, we present a number of techniques that can be used by event-based middleware to support collaboration in location-aware mobile applications including *location-independent announcement and subscription*, *location-based event filtering*, and *location-dependent delivery of events*.

Location-independent announcement allows producers to advertise the (types of) events that they produce and have these advertisements persist while moving location. Likewise, consumers use location-independent subscription to subscribe to events allowing them to receive events of interest wherever they move. Such announcements and subscriptions are persistent in that they apply transparently to all locations independently of the specific location at which they have been issued and are vital to enabling the delivery of subsequently disseminated events at any location and while entities are moving. For example, an ambulance providing an emergency vehicle warning service while rushing to an accident site may use a location-independent announcement to persistently advertise this service while responding to an emergency call. Other vehicles may use a location-independent subscription to subscribe to this service, enabling them to receive emergency vehicle warning events every time their respective journeys intersect that of an ambulance. In contrast, mobile consumers using event-based middleware for nomadic applications, such as Elvin4 [18], register their interest in events with proxy servers that maintain permanent connections to producers and store the subscription information. This style of subscription requires consumers to explicitly connect to specific proxy servers and a consumer wishing to move is required to disconnect from the server prior to moving and to reconnect to the same server from its new location.

Event filters define the specific subset of events in which a consumer is interested. Our approach to event filtering allows producers and consumers to define location-based filters that may use the actual entity location when applying a filter. Producers may define filters that describe a geographical area surrounding their location. These filters bound the geographical scope in which events are to be disseminated and move location with a migrating producer. For example, an ambulance might use a filter to define a circular scope surrounding its location in order to bound event dissemination of an emergency vehicle warning service. Consumer filters may include event attributes that refer to the current consumer location or to the location of the event producer. These filters may, therefore, exploit location in addition to the meaning of events in order to describe the exact subset of events in which a consumer is interested. For example, a vehicle might define a filter that compares its location to the location of an ambulance so that events of an emergency vehicle warning service only match as long as the ambulance is driving toward the vehicle. Furthermore, filters are distributed in that they can be applied either at the producer side or at the consumer side. For example, an ambulance might define the scope of an emergency vehicle warning service while a vehicle might define a filter for matching further constraints of these events. Nomadic applications, on the other hand, usually use event filters that do not evaluate location, allowing consumers to access the same subset of events regardless of their point of connectivity [10], [12], [13], [17], [18], [19].

Event propagation is location dependent in that events generated by a particular producer will only be delivered by consumers currently residing in a specified geographical area. Consumers may deliver a particular set of events at some location and subsequently deliver a different set of events at another location; they may deliver events generated by one producer and then deliver events generated by another producer at a different location. For example, a vehicle may deliver emergency vehicle warning events generated by a certain ambulance at some stage of its journey and subsequently deliver emergency vehicle warnings generated by another ambulance at a different location.

Significantly, this concept of location-dependent event delivery allows a consumer to deliver events at the location where they are relevant and while it or indeed the event producer is moving. In contrast, Elvin4 proxy servers forward events to connected consumers and store events while a disconnected consumer is moving. However, the same set of events is expected to be delivered to a specific consumer whether or not it has moved location.

This paper describes these techniques and their implementation in STEAM, an event-based middleware for collaborative pervasive and mobile applications with an inherently distributed architecture. It outlines how STEAM supports these techniques with a fully distributed discovery mechanism for dynamic discovery of locations (and events) of interest. We have also realized several prototypical collaborative application scenarios derived from relevant areas, including search and rescue, gaming, and especially transportation, to evaluate the proposed techniques. The evaluation demonstrates the feasibility of accommodating representative scenarios from the target category of application and illustrates the trade-off of this support by assessing the cost of location-based event dissemination as well as the latency imposed by location-dependent event delivery in STEAM. As system scale is a defining characteristic of collaborative pervasive and mobile applications, we have also assessed the benefits of distributed location-based event filtering demonstrating that using distributed filters further accommodates large-scale collaborative applications as well as the geographical dispersion of application components. This is achieved by significantly reducing the potentially large number of events to be delivered while limiting the number of filters being applied at a particular location and balancing the computational load of filter matching between the nodes in a system.

The remainder of the paper is structured as follows: Section 2 surveys related work. Section 3 presents the techniques for supporting location-aware event-based applications in highly dynamic mobile computing environments. Section 4 describes how these techniques have been realized in STEAM. Section 5 presents our evaluation of this work outlining the benefits of distributed event filtering and the cost of location-based event dissemination. Finally, Section 6 concludes this paper by summarizing our work.

2 Related Work

Middleware supporting event-based communication has been developed by both industry [20], [21] and academia [10], [12], [22], [23]. Most such middleware assumes that the components comprising an application are stationary and that a fixed network infrastructure is available. Existing research on event-based middleware for mobile computing has mainly focused on supporting nomadic applications using wireless data communication based on the infrastructure network model [10], [12], [13], [17] [18], [19]. Recently, some authors [24] have begun to address the distinct requirements of collaborative mobile applications or of supporting event-based communication in ad hoc networks characterized by the absence of shared infrastructure. For example, application components using an ad hoc network cannot rely on the use of access points when discovering peers in order to establish connections to them. Event messages can neither be routed through access points nor rely on the presence of intermediate components that may apply event filters or enforce nonfunctional attributes such as ordering policies and delivery deadlines.

For example, JEDI [12] allows nomadic application components to produce or consume events by connecting to a logically centralized event dispatcher that has global knowledge of all subscription requests and events. JEDI provides a distributed implementation of the event dispatcher consisting of a set of dispatching servers that is interconnected through a fixed network. Nomadic entities may move using the `moveOut` and `moveIn` operations. The `moveOut` operation disconnects the entity from its current dispatching server, allowing it to move to another location and then to use the `moveIn` operator to connect to another dispatching server. The source dispatching server buffers all relevant information while an entity is disconnected and forwards it upon reconnection. Mobile Push [17] provides a similar approach to supporting nomadic application components in which entities do not use the event service while moving. In addition, it allows mobile application components to access the event service infrastructure through wireless connections while moving. However, like JEDI, this approach relies on the presence of a separate event service infrastructure.

Elvin4 [18] represents event-based systems that support mobility through the use of a proxy server maintaining a permanent connection to the event server on behalf of nomadic client components. The proxy server stores events while a client is temporarily disconnected and clients can specify a time to live for each subscription to prevent large numbers of events being stored indefinitely. Clients must explicitly connect to a proxy server using a URL and must reconnect to the same proxy server each time they reconnect to the event system.

Although these middleware services support mobility, their main goal is to handle disconnection while an entity moves from one access point or event broker to another. In contrast, STEAM accommodates a changing set of collaborative entities coming together at a location and supports communication between these entities without relying on a separate event service infrastructure.

2.1 Exploiting Location in Event Dissemination

Rebeca [19] allows nomadic clients to access a network of event routing brokers through local brokers. Local brokers act as access points and allow clients to disconnect from their current network broker and reconnect at the network broker to which they wish to relocate in a way similar to the approaches described above. However, in addition to handling disconnection, Rebeca also promotes a form of location awareness. Nomadic clients are explicitly aware of changes to their location and are able to exploit their current location when filtering events. Such location-aware clients can move while remaining connected to the same local broker, and use event filters with `myloc` attributes that denote their current location. The values for such attributes are drawn from an application-specific set of locations, for example, describing the rooms in a house, the places in a city, or the (coarse-grained) coordinates of a GPS system. Current values are maintained by local brokers and are updated whenever a client moves. Much like STEAM's consumer-defined filters, which may include event attributes that refer to the current consumer location, this form of location-dependent filtering supports location awareness on the consumer side of an application. However, it does not provide for producer filters that bound the geographical area in which events are to be disseminated and move location with a migrating producer, which are essential for the techniques presented in this paper. This focus on supporting location awareness on the consumer side as well as its dependence on a broker infrastructure enables Rebeca to support location awareness in nomadic applications rather than providing for location-aware collaborative applications. Moreover, Rebeca also proposes a separate abstraction that enables applications to explicitly localize the relationships between static entities. Using such scopes, applications can group the entities in a system so that producers restrict the dissemination of their events to consumers within the same scope. The purpose of this abstraction is to handle heterogeneity and provide a means for integrating membership policies possibly in applications that are based on wireless sensor networks [25].

ToPSS [26] supports location awareness by extending its centralized filtering engine with a location matching engine. The filtering engine mediates event dissemination and maintains content-based subscriptions while the location-matching engine maintains location information for stationary and mobile producers and consumers. Location information is expressed as latitude/longitude/altitude tuples and the location-matching engine receives periodic updates of the location of mobile entities. The filtering engine forwards events to consumers with matching subscriptions if they are located within a certain distance from the producer. The relevant distance is defined by the consumers as part of their subscriptions. Mobile entities connect to and interact through the filtering engine and use the location-matching engine for filtering events based on the locations of mobile producers and mobile consumers. For example, ToPSS has been used for a friend-finder application in which mobile users specify a mobile friend about whom they wish to be notified when in close proximity. Hence, ToPSS supports applications in which the geographical scope of interest is defined by the consumer and is solely based on distance to the producer, rather than applications where proximity is defined by producers as surrounding geographical areas. Also, as with Rebeca, ToPSS depends on infrastructure, the centralized filtering and location matching engines, that needs to be accessible by the producers and consumers in a system, which prevents it from supporting collaborative entities.

The recent work of Frey and Roman [24] is the most closely related to ours in that it also provides an approach to supporting collaborative event-based applications for ad hoc networks. Their work allows location to be used to describe both the areas in which subscriptions and events are relevant and also. separately, the areas in which the producers and consumers of events should be located for corresponding events to be delivered. In contrast to our approach, their system allows events to be delivered to subscribers even when they are not located within the area to which the producer has specified they are relevant, mitigating the ability of the middleware to use location-independent announcements to filter events. Moreover, the dual use of an event's area of relevance and the producer's location to determine whether or not to deliver events introduces additional communication overhead. An extra round of direct communication between the producer and subscriber is needed before delivering matched events that depend on the producer's location.

2.2 Alternatives to Event-Based Middleware for Collaboration

While the focus of our work is on techniques to support collaborative mobile applications that use event-based communication, we note that a number of other paradigms have also been extended to support such applications as exemplified by various extensions to the tuple space paradigm. As discussed by Schelfthout [27], the tuple space paradigm differs from the event paradigm in that it is inherently coupled in space (as entities must know which tuple space to access) and decoupled in time (collaborating entities do not need to be present at the same time). Conversely, event-based middleware is inherently decoupled in space (entities do not need to know where other entities are) and coupled in time (entities usually need to be present at the time when a message is sent).

The most influential of tuple space system supporting mobility is Linda in a Mobile Environment (Lime) [28], [29], which is also explicitly designed for use in an ad hoc network. Lime caters for physical mobility of hosts and logical mobility of agents (i.e., runtime migration of software components) by having a tuple space attached to each mobile entity. Entities then collaborate by transiently sharing their tuple spaces, creating a "global virtual data structure" [28]. While tuples from connected nodes can be accessed based on pattern matching, thereby providing a form of subject or content filtering, there is no concept of location-dependent filtering nor a concept of location-independent announcement allowing the producer of some relevant information to be identified.

EgoSpaces [30] is one of the many extensions of Lime and introduces the concept of a view that allows nodes to specify from which other nodes tuples are gathered. To offer higher level coordination support, the concept of views was extended to include reactions, which specify actions that are automatically performed in response to specified changes in a view. EgoSpaces allows receivers to choose from which nodes the data are coming and this can be defined on location. Tuples on the Air (TOTA) [31] allows the definition of tuples that are automatically disseminated by copying them to connected nodes according to an application-specific rule and propagation may be restricted based on distance (e.g., number of hops). Finally, Limone [32] is another Lime-inspired system designed for use over ad hoc networks. In Limone, each agent maintains strict control over its local data and defines an acquaintance policy that governs the agents with which it will interact. As can be seen, largely because of their focus on maintaining a shared tuple space as a central abstraction, these systems lack the support for spontaneous discovery and collaboration with unknown peers facilitated by location-independent announcement and location-dependent event delivery.

3 Location-Aware Event-Based Middleware

Event-based middleware to support pervasive and mobile applications in which collaboration between nearby entities is intrinsic must deal with the increased complexity that arises from a potentially large number of interacting entities, from their geographical dispersion, and from the spontaneously changing connections between them. Mobile entities

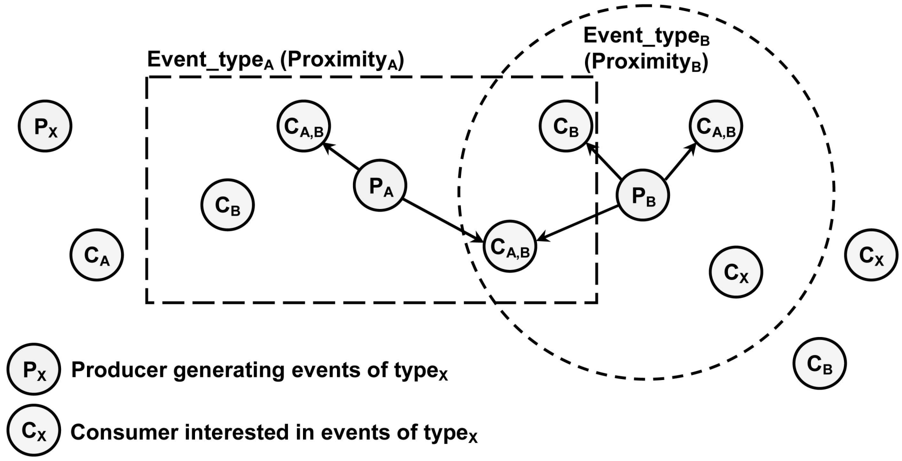


Fig. 1. Supporting collaboration using event types and proximities.

that comprise this style of application characteristically move together and apart over time. Sets of such entities typically come together at a certain location to communicate and collaborate, move apart, and then come together with other entities at a different location to collaborate there. Hence, these entities are more likely to interact when they are in close proximity. For example, a vehicle is interested in receiving emergency vehicle warnings from an ambulance only when the ambulance is within close proximity. Similarly, tourists participating in a guided tour are most interested in information about a specific site once they have arrived at its location. This means that the closer event consumers are located to a producer the more likely they are to be interested in the events that it produces. Significantly, this implies that events are likely to be most relevant within a certain geographical area surrounding a producer. This section presents techniques that can be used by location-aware event-based middleware to support such collaboration allowing entities representing real-world objects currently located within the same geographical area to discover and deliver events at the location where they are relevant.

3.1 Inherently Distributed Event Service

Event-based middleware for collaborative applications should enable entities that have come together at a certain location to communicate and collaborate through wireless connections even in the absence of any local network infrastructure, in particular by supporting ad hoc as well as infrastructure networks. Due to the characteristics of ad hoc networks, such an event service must be inherently distributed since it cannot rely on any service infrastructure. It cannot depend on logically centralized or intermediate components that are typically hosted by such an infrastructure. For example, it cannot rely on a well-known intermediate event broker to connect producers and consumers as has been proposed by SIENA [33] to support nomadic applications. Moreover, the characteristics of collaborative applications, where entities come together to collaborate, move apart, and then come together with other entities at a different location, preclude dependency on broker nodes interconnecting such locations of collaboration across an ad hoc network. For example, it cannot rely on dynamically elected cooperating directories as have been proposed to support scalable service discovery for service-oriented architectures based on ad hoc networks [34]. Hence, event-based middleware for collaborative applications must employ concepts that can support such an inherent distribution, including event types and proximities, instead of centralized components.

3.1.1 Event Types and Proximities

An implicit event-based programming model, i.e., one that does not make the present of other entities or event brokers explicit to application programmers [11], is naturally suited for applications in ad hoc environments. It allows producers to publish events of specific event types and consumers to subscribe to events of a particular type rather than having to subscribe at another entity or at an intermediate, as is required by peer-based and mediator-based event models [11]. Producers may publish events of several event types and consumers may subscribe to one or more event types. To accommodate collaborative applications, we propose an implicit event model that supports geographical scopes allowing producers to explicitly disseminate events to nearby consumers. Producers associate the type of event they intend to generate, or raise, with a geographical area, called the proximity, within which events of this type are to be disseminated. Consumers can receive events of some type if (and only if) they are located inside a proximity in which events of this type are being raised. For example, an ambulance may define a proximity, whose size may depend on its speed and prevailing road conditions, for its emergency vehicle warning events. Other vehicles will only receive these events when located within this proximity relative to the ambulance.

Proximities may be of arbitrary shape and may be defined as nested and overlapping areas. Nesting allows a large proximity to contain a smaller proximity subdividing the large area. Fig. 1 depicts two event types associated with overlapping proximities of different shape and illustrates that multiple consumers may reside inside a proximity. `Proximity`$_A$ and `Proximity`$_B$ have been defined for `Event_type`$_A$ and `Event_type`$_B$, respectively. Consumers that have subscribed will be delivered these events if they reside inside the appropriate proximity. Note that consumers located inside these areas but which are only interested in other event types will not be delivered events of either type.

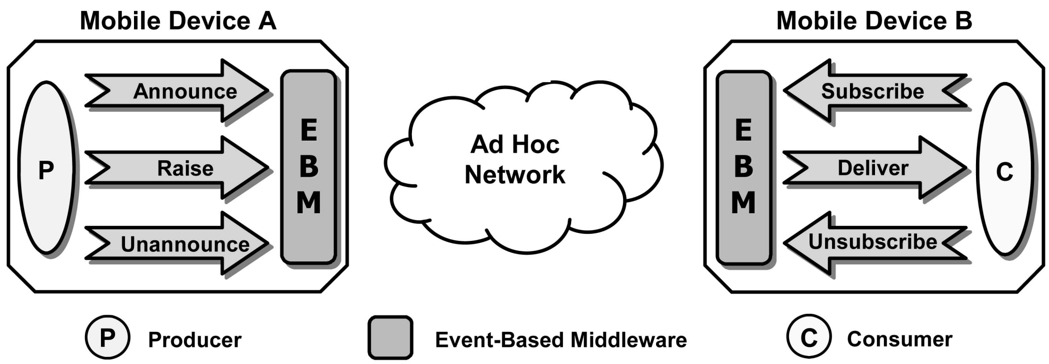


Fig. 2. Location-independent announcement and subscription.

3.1.2 Supporting Mobility

Collaborative entities can be either stationary or mobile and interact based on their geographical location. This implies that the middleware, as well as the entities hosted by a particular node, are aware of their geographical location at any given time, for example, by using an appropriate location service to determine the current geographical location of the node. A location service for outdoor applications, such as context-aware transportation systems and city-wide information systems, may exploit a GPS satellite receiver to provide latitude and longitude coordinates. One for indoor applications may exploit tag-based location mechanisms, such as [35], [36].

Proximities can be either stationary or mobile since they may be defined by both stationary and mobile event producers. The proximity definition below shows that this notion of proximity is defined by the area covered, which is described as a geometric shape with associated dimensions and a reference point that is relative to this shape, and by a "naval" location used to associate the area with the real world. The reference point of a stationary proximity is attached to a naval represented by a fixed point in space whereas the reference point of a mobile proximity is mapped to a moving naval, which is characteristically represented by the location of a specific mobile producer. Hence, a mobile proximity moves with the location of the producer to which it has been attached. For example, a platoon of vehicles traveling in the same direction might interact in a mobile proximity that might have been attached to a naval defined by the position of the leading vehicle and moving with its location.

```
Proximity = Area(Shape,Dimensions
            Reference Point),Naval}
```

3.2 Location-Independent Announcement and Subscription

Location-independent announcement allows producers to advertise their events and have these advertisements persist while moving location. As summarized in Fig. 2, a producer using location-independent announcement specifies an event type/proximity pair to associate a specific event type with a certain proximity. That producer may then generate events of the announced type until it is unannounced and have them delivered to interested consumers within the proximity. Likewise, consumers use location-independent subscription to subscribe to events allowing them to receive events of interest whenever they move into a proximity in which such events are being generated. Fig. 2 shows that consumers simply subscribe to event types in order to have the middleware deliver location-relevant events to them until they unsubscribe. Significantly, consumers subscribe to event types only. They do not need to subscribe to specific proximities; rather, they implicitly subscribe to all the proximities that have been associated with a specific event type by any producer. This enables them to receive events of interest generated inside any proximity in which they are present. Such announcements and subscriptions are persistent in that they apply transparently to all locations independently of the specific location where they have been issued. Announced event types and proximities apply to all locations that a migrating producer might visit and consumers may move from one proximity to another without reissuing a subscription when entering the new proximity.

Announcements specify both the functional and nonfunctional attributes of the events to be generated by some producer. The definition below shows that an event type consists of a *subject* and *content* representing its functional attributes, as well as of a self-describing *attribute list* representing its nonfunctional attributes. The subject defines the name of a specific event type and the content defines the names and types of a set of associated parameters. The attribute list may include a variety of attributes ranging from context used for additional filtering, such as geographical location, to temporal validity, to delivery semantics, such as event priority, delivery deadline, and delivery order. Producers and consumers must use a common vocabulary defined by the application to agree on the names of the event types. Event types that have the same subject must have an identical content structure, i.e., the set of parameter names and types must be consistent, and must have an identical list of attributes:

```
Event Type = {Subject, Content,
             Attribute_List}
```

In principle, either a consumer or a producer may define a proximity. Consumers might wish to define proximities that describe their interest in events published in certain areas depending on their (current) activities. For example, a migrating consumer might define its scope of interest according to its actual travel speed. Producers, on the other hand, might wish to define proximities describing the scopes inside which their events are raised.

However, we believe that, in many collaborative applications, the semantics of disseminating events depends on the producer, and therefore, it is the producer that would define proximities. A producer may assess its local conditions, which likely apply to all consumers within its vicinity, and may define, based on application requirements and these circumstances, an appropriate proximity. For example, a traffic light propagating its status to approaching vehicles (see also the "Informed Driver" [37] and the "Cooperative Intersection Collision Avoidance System" [38] initiatives for applications that depend on such interaction) defines its proximity based on the location of the next traffic light and on the local speed limit. This enables each traffic light in an urban environment to define a proximity that is tailored according to its local conditions (and thereby to the conditions of passing vehicles), ensuring that vehicles suitably receive the light changes that are relevant at their current location. Irrelevant information, for example, a change to the state of a light at the far end of town, is being filtered and vehicles are prevented from having to adjust their filters to the prevailing conditions, such as a change to the speed limit, along their journeys. Nevertheless, we admit the possibility of applications in which consumers might wish to determine their proximities. A vehicle exceeding the local speed limit, for example, a police car on a call, may require a larger scope for receiving a traffic light's status compared to an "ordinary" vehicle traveling within the speed limit. However, location-independent announcement currently supports proximities defined by producers only, while support for proximities defined by consumers remains for future work.

3.3 Location-Dependent Event Delivery

Event propagation is location dependent in that events generated by a particular producer are only delivered by consumers currently residing in the appropriate geographical area. Producers announce proximities to specify the locations at which their events are relevant. Consumers discover these areas of interest and subsequently deliver events at the locations where they are relevant. Mobile (and stationary) consumers transparently discover the proximities and ultimately the events of interest that are available at their current location regardless of the dynamics of the producers, i.e., whenever they enter a proximity or a proximity (attached to some mobile producer) arrives at their location. Consumers then deliver these events for as long as they reside inside the proximity.

Consumers may deliver a particular set of events at some location and subsequently deliver a different set of events at another location. They may deliver events generated by some set of producers and then deliver events generated by another set of producers at a different location. This implies that a subscription to a specific event type applies to all proximities, where such events are generated. A single subscription may result in events of a particular event type raised by different producers in multiple proximities being delivered over time. Hence, the set of events received by a consumer at a certain time depends on its movements as well as on the movements of producers and proximities. For example, a vehicle may deliver status events generated by a certain traffic light at some intersection and subsequently deliver status events generated by another traffic light at the next intersection.

Significantly, location-dependent event delivery exploits the announcement concept to support mobility and enable consumers to discover proximities (and events) rather than producers. This provided a notion of anonymity in which entities are anonymous to each other but known by the middleware.

3.4 Location-Based Event Filtering

An event system consists of a potentially large number of producers [18], [22], [39], all of which produce events containing different information. As a result, the number of events propagated in an event-based system may be very large. However, a particular consumer may only be interested in a subset of the events propagated in the system or even within its current locality. Event filters provide a means to control the propagation of events. Ideally, filters enable a particular consumer to receive only the exact set of events in which it is interested. Events are matched against the filters and are only delivered to consumers that are interested in them, i.e., for which the matching produced a positive result.

Location-based event filtering is a distributed approach to filtering that allows an application to define multiple event filters, which may use the actual location of a producer or a consumer, and to apply them at both the producer side and the consumer side. Producer-side filters may describe a proximity surrounding a producer's location and consumer-side filters may include event attributes that refer to the current consumer location or to the location of the event producer.

3.4.1 Distributing Location-Based Filters

Location-based event filtering allows an application to specify multiple event filters, each of which may apply to a different attribute of a specific event. Such filters may be combined and a particular event is only delivered to a consumer if all filters match. Combining filters is beneficial to the precision of filtering allowing a consumer to define the subset of events in which it is interested using multiple criteria, including not only the meaning of an event, but also criteria such as time and geographical location. Event filtering at both the consumer and the producer side implies that a relatively small number of filters are applied on a specific node compared to traditional approaches in which an arbitrarily large number of filters are evaluated sequentially on a node hosting an event broker or a producer. For example, considering that applications often consist of more consumers than producers [10], [39], applying the filters defined by many consumers on a single (producer) node, may result in significant computational load for that node. This is in contrast to a distributed approach, where filtering can be shared between many nodes, thereby lightening the load of individual nodes. Distributing event filters allows mobile devices, which typically have limited computational resources, to concurrently evaluate the filters that apply to a particular event. The computational load of filter matching can therefore be distributed between several mobile devices.

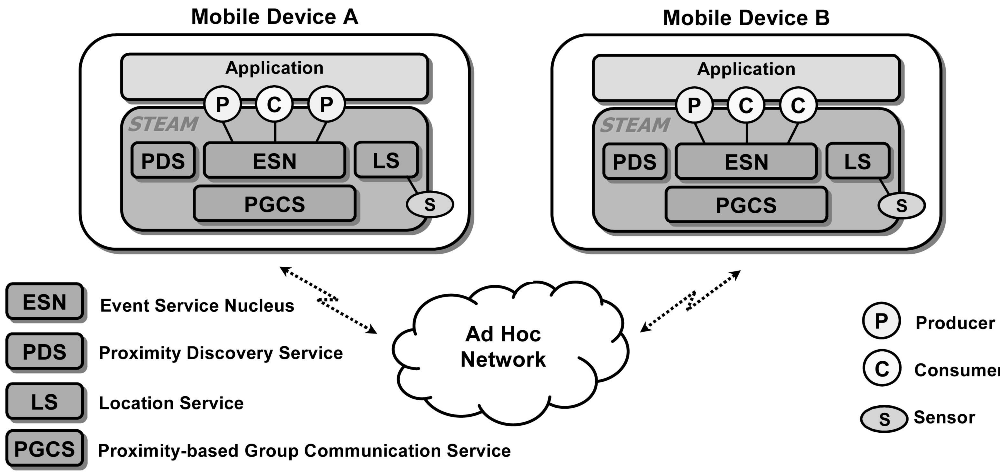


Fig. 3. Mobile devices hosting the inherently distributed architecture of the STEAM middleware.

3.4.2 Defining Location-Based Filters

Location-based event filtering supports three classes of event filters: *subject filters*, *content filters*, and *proximity filters*. Subject filters match the subject of events and allow a consumer to specify the event type in which it is interested. As shown below, subject filters can be defined by giving the name of the relevant event type. Content filters contain a filter expression that can be matched against the values of the parameters of an event. As shown below, content filters may consist of an arbitrary number of self-describing filter terms. Each term specifies the parameter to which it applies along with an operator and a value representing the second operand. These filter terms are matched against the relevant parameters of an event either in a conjunctive or a disjunctive manner, thus defining whether *all* or *at least one* of the terms that comprise a filter must be true for the filter to match. These filter expressions can contain equality, magnitude, and range operators, as well as consumer local information, such as the consumer's geographical location. Proximity filters define the geographical scope within which certain events are relevant and correspond to the proximities that producers may announce. As shown below, proximity filters specify the area and naval of a proximity and may be defined as either stationary or mobile. Proximity filters bound event dissemination and serve as the basis for location-dependent event dissemination and delivery. They act as implicit event filters and use location information to determine whether or not to deliver an event to a particular consumer.

```
Subject Filter={Subject}

Filter Term={Content Parameter Name,
Operator,Value}
Content Filter=
(Conjunctive | Disjunctive),
Filter Term,[Filter Term], ...}

Proximity Filter={(Stationary | Mobile),
Area (Shape,Dimensions,
Reference Point), Naval}
```

Event filtering based on geographical context enhances the ability of a system to accommodate the dynamically changing population of mobile entities by dividing the system into bounded geographical scopes. An entity entering a proximity causes other entities in the same area to reconfigure, i.e., to update routing and subscription information, without affecting entities residing outside the area. Consequently, location-based filtering bounds the propagation range of events and of event filters. In particular, location-based filtering limits the forwarding of events and of filters to a confined geographical area.

4 The STEAM Event Service

The STEAM event-based middleware implements the techniques introduced in the previous section using group communication. STEAM provides location-aware event dissemination for collaborative pervasive and mobile applications running on mobile devices that interact through IEEE 802.11b-based ad hoc wireless local area networks [7], [40]. Depending on the application areas in which they are used, such portable computing devices may range from handheld devices, such as personal digital assistants, to notebook computers. This section outlines the architecture and most important implementation techniques used by STEAM.

4.1 Inherently Distributed Service Architecture

The STEAM event service is based on an inherently distributed architecture in which the middleware is exclusively collocated with the application components and does not depend on any separate centralized or intermediate components. As illustrated in Fig. 3, the architecture essentially consists of four key components that reflect the main features of the event service. The Event Service Nucleus (ESN) implements STEAM's application programming interface and therefore is explicitly exposed to applications. The event service nucleus can be regarded as STEAM's central component since it interconnects the remaining components and because it provides a filter engine that applies and maintains the various event filters that producers and consumers may define. The event service nucleus exploits a Proximity-based Group Communication Service (PGCS) to disseminate events depending on the locations of the relevant producers to consumers. The Proximity Discovery Service (PDS) provides the means for potentially mobile entities to persistently announce and

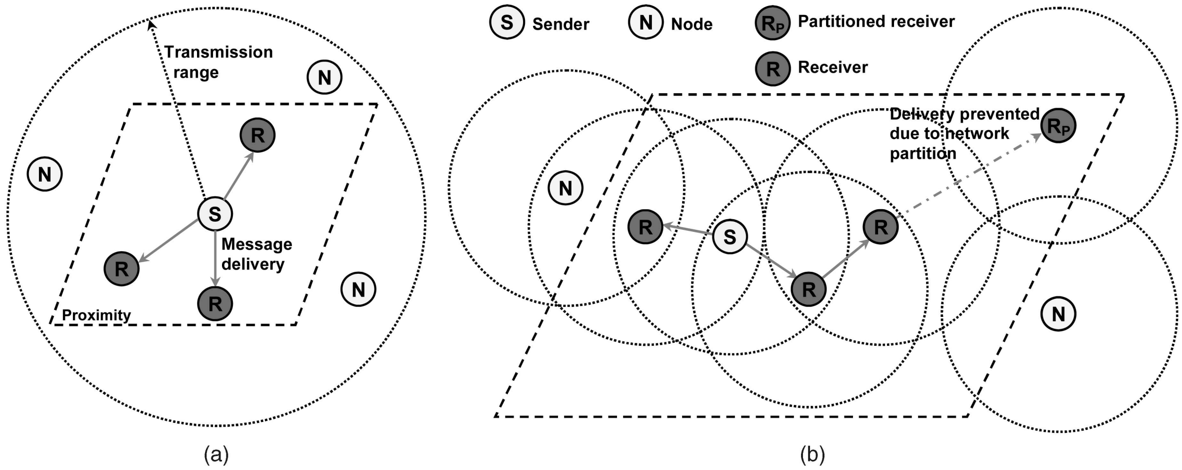


Fig. 4. Location-based event dissemination. (a) Single-hop event dissemination. (b) Multihop event dissemination.

subscribe to events and eventually to discover events of interest. The discovery mechanism uses location information to map proximities to the subscriptions of its consumers and as a result, manages the proximity groups that are relevant at its current location. This implies that the proximity discovery service is responsible for maintaining a consistent notion of the relationship between proximity groups and subscriptions at any given time while considering the migration of entities and indeed of proximities. STEAM depends on a Location Service (LS) to supply geographical location information. The current location service uses GPS-based sensor data to compute the current geographical location of the mobile device and provides this location information to the middleware and to producers and consumers hosted by the device.

Fig. 3 illustrates the fact that every mobile device has identical STEAM capabilities. The middleware may support a variable number of consumers and producers on each mobile device, thereby allowing individual devices to both produce and consume events. The event type and proximity information announced by producers are exploited by the PDS to establish communication relationships between mobile entities rather than to optimize event routing as suggested by Carzaniga et al. [22]. Subscriptions are used *locally* to map consumer interests to the sets of currently available events being disseminated within the discovered proximities.

As shown below, the operations of the STEAM application programming interface reflect the fact that STEAM is based on an implicit event model as they refer neither to explicit entities nor to designated components of any kind. Instead, the operations for announcing and subscribing to events refer to event types, the former indicating the actual type and the latter using a subject filter to name the type.

```
announce(eventType et, proximityFilter pf)
unannounce(eventType et)
subscribe(subjectFilter sf,
  deliveryHandler dh,
contentFilter cf)
unsubscribe(subjectFilter sf,
  deliveryHandler dh,
contentFilter cf)
raise(event e)
```

The STEAM application programming interface also illustrates how producers and consumers specify their respective event filters. Producers specify their proximity filters and announce them, together with their event types, thereby grouping them into associated pairs, while consumers specify both their subject filters and their content filters together with their delivery handlers. Consumers may omit content filters, allowing them to express their interest in events solely using their types, thereby employing a classic, topic-based subscription mechanism [11].

4.2 Mapping Location onto Process Groups

Group communication [41] has been recognized as a natural means to support event-based communication models [42]. Groups provide a one-to-many communication pattern that can be used by producers to propagate events to a group of subscribed consumers. STEAM's proximity groups have been designed to support mobile applications using wireless local area networks [43]. To apply for group membership, an application component must first be located in the geographical area corresponding to the group, and second, be interested in the group in order to join, i.e., a group is identified by both geographical and functional aspects. In contrast, classical group communication defines groups solely by their functional aspect. STEAM maps event subject and proximity to the functional and geographical aspect, respectively, of proximity groups. Furthermore, proximity groups can be either absolute or relative. An absolute proximity group is geographically fixed; it is attached to a fixed point in space and is used to support stationary proximities. In contrast, a relative proximity group is attached to a moving point represented by a specific mobile node and is used to support mobile proximities. Management of proximity groups is discussed in detail in the service discovery section below.

4.3 Exploiting Multicast Groups

STEAM allows entities to define geographical scopes independently of the physical transmission range of their wireless radio transmitters. This implies that STEAM supports multihop event dissemination for scenarios in which proximity exceeds the radio transmission range of the sender. Fig. 4a outlines a single-hop event propagation scenario, where the radio transmission range of the sender covers the entire scope of the proximity. Event messages are propagated within this radio transmission range and travel exactly one hop in order to reach all potentially interested nodes. Proximity group members recognize the event messages that are relevant to them and may subsequently deliver these events. Nodes that are not members of such a proximity group ignore these event messages.

Fig. 4b shows a multihop event propagation scenario in which the proximity exceeds the radio transmission range of the sender. Proximity group member nodes must forward event messages for them to reach other members of the group. Similarly to single-hop event dissemination, group members recognize relevant event messages and may subsequently deliver these events. However, group members also forward relevant event messages, thereby expanding their dissemination range. The maximum number of hops such event messages may travel to reach any member of the group is bounded by the proximity. Nongroup member nodes ignore these event messages and consequently do not forward them. Multihop event dissemination generally increases the range within which event messages can be propagated but characteristically imposes additional transmission latency compared to single-hop transmissions.

STEAM uses a producer's geographical location and radio transmission range in conjunction with the proximity in which specific events are to be published to optimize the propagation of specific events by transparently determining whether to use single-hop or multihop messages. STEAM therefore supports both single-hop and multihop event dissemination and typically uses cost-efficient single-hop messages for publishing events in proximities that are likely to be covered by a producer's wireless transmission range and employs multihop messages only when transmitting events beyond this range.

The current version of STEAM uses a proximity-based group communication service that is based on IP multicast. Such multicast-based group communication is naturally suited to support the one-to-many interaction style of location-based event dissemination. It allows a consumer to join a proximity group of interest without having to explicitly establish a direct communication relationship with the producer as is the case for unicast-based group communication. This is especially beneficial for collaborative mobile applications, where mobile entities come together to dynamically form temporary groups. Exploiting IP multicast for disseminating messages provides best-effort delivery semantics, which does not guarantee that any subscriber will necessarily receive a specific event or an individual announcement will be received by nearby devices. The current implementation of STEAM therefore provides a best-effort event service but has been designed to use proximity groups that are based on the TBMAC protocol [44]. Hence, a future version of STEAM will provide strong guarantees in terms of event delivery reliability and timeliness.

Coverage can be defined as the geographical area to which a particular sender can send messages using either single-hop or multihop communication. A proximity is said to be covered if the associated set of nodes is a subset of the nodes to which event messages can be propagated. Network partitions may occur in cases where an area defined by a proximity is not covered by a specific sender. They may therefore occur in the kind of multihop scenario illustrated in Fig. 4b, but not in single-hop scenarios where the radio transmission range covers the whole proximity. Our approach to message forwarding in which nongroup member nodes are not expected to designate resources to unrelated messages may result in a scenario where nongroup member nodes might help increase coverage or prevent network partitioning. However, we argue that such a scenario is unlikely given the nature of collaborative applications and that depending on the willingness of an unrelated node to provide additional resources is impractical.

Techniques for preventing message loss due to coverage limitations must first anticipate a network partition. Such partition anticipation can be based on a means for detecting link failure between individual components in ad hoc networks [45], [46] or on an approach that assesses the quality of wireless connections to predict their future level of connectivity [47]. Consumers that are able to anticipate network partitions can then employ a means to recover missed events once they reestablish their connections to the members of a group. Such consumers recover missed events by requesting a retransmission of previously sent events. A producer may forward the events it has cached to these consumers depending on their persistence level.

The design is optimized for delivery of events to consumers that are expected to be present in the proximity. This contrasts with approaches used in the so-called delay-tolerant networks, where the intended receivers of messages are expected to be uncontactable for extended periods of time. This leads inevitably to different design decisions usually based on some notion of "custody" whereby intermediate nodes take responsibility for each message until it can be delivered to its intended next hop—an approach that is exemplified by the Bundle Protocol [48].

4.4 Distributed Hashing

There are two essential issues that need to be addressed when mapping announcements and subscriptions to proximity groups. First, an addressing scheme for uniquely identifying groups is required, and second, a means for producers and consumers to obtain the correct group identifiers needs to be provided. An approach to addressing these issues, based on statically generating a fixed number of unique and well-known group identifiers, has been described by Orvalho et al. [49]. Another approach might involve using a centralized lookup service for generating and retrieving group identifiers. However, neither of these approaches suffices for applications that need to accommodate a dynamically changing number of communication groups and depend on an inherently distributed architecture.

STEAM exploits a decentralized addressing scheme in which identifiers representing groups are computed from event type and proximity pairs. Each combination of event type and proximity (shape dimensions, reference point, and naval location) is considered to be unique throughout a system under the assumption that there is no justification for applications to define multiple identical subject and

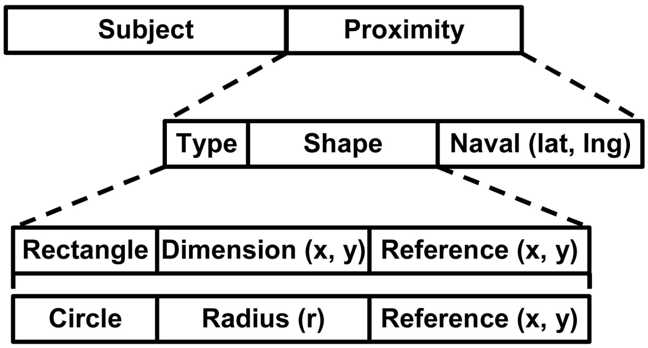


Fig. 5. Computing group identifiers from event type and proximity pairs.

proximity pairs for different sets of events. A textual description of such a pair is used as the stimulus for a hashing algorithm to dynamically generate hash keys that represent identifiers using node-local rather than global knowledge. Producers and consumers compute the corresponding group identifier upon discovery of a proximity and the associated event type and subsequently use these group identifiers to join groups in which relevant events are disseminated. This scheme prevents entities that are not interested in certain events from joining irrelevant groups, and consequently, from receiving unwanted events even though they might reside inside the proximity associated with a group.

This distributed addressing scheme replaces the kind of centralized approach traditionally used for identifying peers of interest and therefore represents a key enabling mechanism for the inherently distributed architecture of STEAM. It enables mobile devices to recognize proximities of interest and *locally* compute proximity group identifiers from serialized event type and proximity descriptions.

STEAM uses a hashing algorithm that generates 24-bit group identifiers from variable length character strings. This algorithm is based on a combination of a hash function for such stimuli proposed by Preiss [50] and the use of a hash function multiplier [51] in order to reduce the chance of collisions. Fig. 5 depicts the structure of the character strings that describe event type/proximity pairs with rectangular and circular proximities. Proximity filters are described by their shapes and by the coordinates that specify the location of their navals. Note that a mobile proximity filter always describes its initial naval location since its actual naval location might change over time, therefore enabling mobile devices to generate consistent group identifiers.

Depending on a number of factors, including the quality of the hash function and the ratio of stimuli to potential identifiers, this approach might lead to colliding identifiers. Such collisions occur when there exist stimulus pairs x and y such that $x \neq y$, for which $h(x) = h(y)$. Collisions may result in different sets of events using the same identifiers. This does not affect a system provided that such sets are only used in different geographical scopes, i.e., their proximity groups do not overlap. Overlapping proximity groups with colliding identifiers can lead to unwanted events being received by certain mobile devices. Such devices can be prevented from delivering unwanted events to their applications by a runtime-type checking mechanism. Such a mechanism can detect and discard these events. Hence, colliding group identifiers may lead to additional use of communication and computational resources, but will not cause delivery of unwanted events and are in any case unlikely due to geographical separation.

4.5 Fully Decentralized Service Discovery

The PDS runs on every node that hosts STEAM, regardless of whether local entities act only as producers or consumers or as both. The PDS uses beacons to periodically announce relevant proximities (and the associated event types) on behalf of the producers. The discovery service announces the event type and proximity pairs that have been defined by producers within the scope of the proximity using a well-known discovery multicast group. This implies that the location at which these announcements are disseminated can change when the node migrates and that the set of adjacent devices is likely to change as well. Moreover, this also implies that other stationary or mobile devices may need to forward announcements for them to reach the boundaries of the proximity. The frequency of the beacons announcing relevant proximities can be defined depending on the requirements of specific collaborative applications. A higher beacon frequency helps in refreshing the PDS more often and reduces the latency for discovering relevant proximities but also increases the computational and communication overhead of the PDS. Senart et al. [52] provide a further discussion on how to derive the configuration of a proximity from application requirements.

4.5.1 The Discovery Mechanism

The PDS (Fig. 6) allows its producers to announce and its consumers to discover relevant proximities and their event types according to the following discovery mechanisms:

1. Initially, a PDS instance recognizes the proximities defined by its local producers.
2. Each PDS instance uses a type repository to maintain a list describing all proximities that are relevant at its current location. A specific proximity description is discarded when the hosting node leaves the geographical area associated with this proximity description.
3. Each PDS instance periodically broadcasts messages describing the proximities defined by its local producers within the proximity relative to its current location. The broadcast period is configurable on an application-specific basis.
4. Each PDS instance that receives an announcement adds a proximity description to its own list if it is currently located inside the associated geographical area.
5. Each PDS instance uses a repository to maintain a list of the subscription by its local consumers.
6. The PGCS is informed whenever a proximity of interest is added to or discarded from the list of relevant proximities. This enables the PGCS to join and leave appropriate proximity groups, and eventually, to receive relevant events.

The proximity discovery mechanism essentially comprises two algorithms, one for advertising locally defined

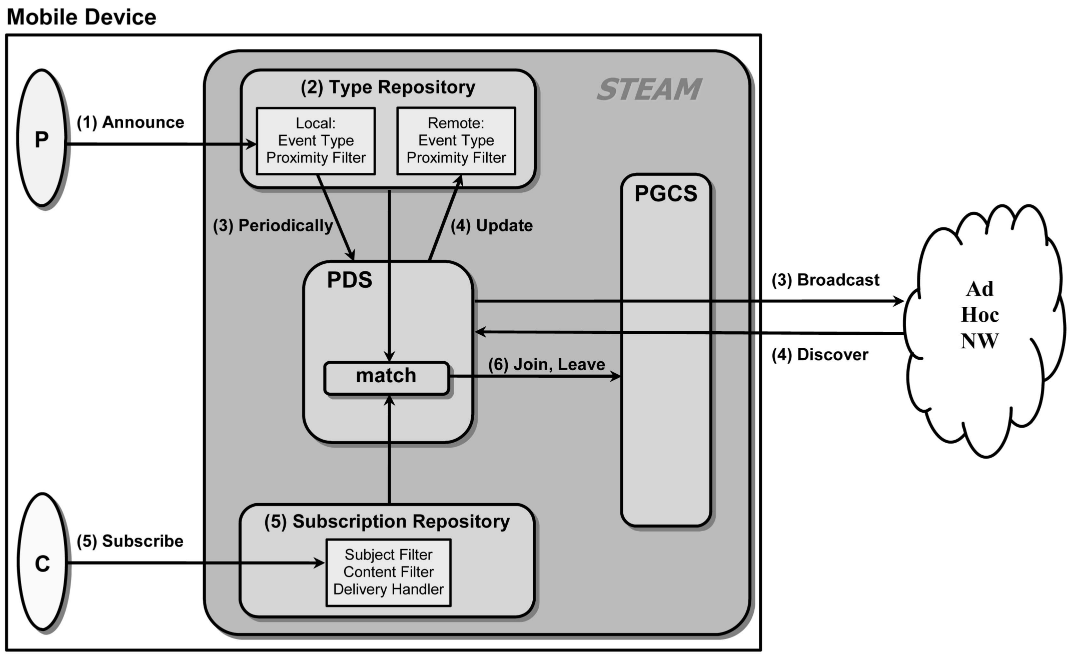


Fig. 6. The mechanism for fully decentralized service discovery.

announcements and another for handling remotely defined announcements on reception. Both algorithms operate on the relevant proximity lists of a particular mobile device and collaborate in order to maintain the relevant locally and remotely specified proximities. Hence, the relevant proximity lists are maintained according to geographical relevance and are independent of the set of issued subscriptions.

4.5.2 Advertising Event Types

The algorithm for broadcasting and, to a certain extent, maintaining event type/proximity pairs on a particular node periodically traces through all descriptions stored in a relevant proximity list in order to advertise the proximities that have been defined by local producers and to verify the geographical validity of both stationary and mobile proximities at the node's current location.

Stationary proximities remain in a list as long as their scope includes the current location regardless of whether they have been specified locally or remotely. A proximity is removed from the list once the node has left the associated geographical area. Removing a relevant proximity may lead to a change in the set of events to be delivered to local consumers. However, the subscriptions of such consumers remain in the subscription repository.

The dynamics of mobile proximities cause some variation in the means by which they are maintained and advertised. Mobile proximities specified by local producers always (by definition) include the actual location of the mobile device and migrate together with the device. Consequently, the migration of a mobile device does not cause mobile proximities to expire. These proximities can therefore be advertised without verifying their validity. In addition to enabling remote nodes to discover mobile proximities, these advertisements provide a means for disseminating location updates describing the migration of mobile devices and their proximities. The naval of every mobile proximity is therefore updated with the latest device location prior to it being advertised. The validity of remotely specified mobile proximities is verified when updates on their latest locations are received. This prevents validity checks using cached, and therefore potentially obsolete (due to migration), location information.

4.5.3 Locating Event Types

The algorithm for handling received advertisements adds newly discovered event type/proximity pairs to the relevant proximities list of any mobile device residing inside the geographical area while proximities whose geographical areas no longer include the node's location are removed from the list. The PGCS is informed of newly discovered proximities (join) as well as of canceled proximities (leave) if a matching subscription exists.

Remotely specified event type/proximity pairs are handled similarly regardless of whether they describe stationary or mobile proximities. Both stationary and mobile proximities that are no longer relevant at the current location are removed from the repository. Mobile proximities are removed based on their latest naval location, although it will be recalled that they are identified based on their initial naval location.

5 Evaluation

This section evaluates the techniques used by the STEAM event-based middleware to support collaboration in location-aware mobile applications proposed in this paper. The first experiment assesses the cost of location-dependent event dissemination, and more specifically, how exploiting proximity can limit event forwarding without the need for additional control messages. Note that the costs of disseminating events are similar to those of disseminating announcements as both techniques rely on group communication for disseminating messages. The experimental results demonstrate that the costs of location-dependent event dissemination for stationary entities are comparable to those for mobile entities and that cost neither depends on the speed of consumers nor on the speed of the producer. This evaluation, therefore, demonstrates that using proximity to bound multicast-based event dissemination is well suited for highly dynamic networks as it limits the complexity of the underlying dynamic network topology without introducing overhead for maintaining dissemination routes and overlay network structures.

The second experiment aims to assess the benefits of distributed event filtering. The experimental results demonstrate that bounding the propagation range of events using location-dependent filtering can significantly reduce the potentially large number of events to be delivered in a collaborative application. A collaborative application scenario has been realized as part of this experiment that uses real data to simulate interaction between a producer and consumers. This experiment demonstrates that using distributed filters significantly reduces the number of events delivered in a system, thereby accommodating large-scale collaborative applications while limiting the number of filters being applied at a particular location and balancing the computational load of filter matching between the nodes in a system.

The third experiment assesses the latency imposed by location-dependent event delivery. The experimental results show the magnitude of event delivery latency for a canonical application scenario using location-dependent filtering and furthermore demonstrates that location-dependent event delivery based on our approach to maintaining distributed event filters limits the effect of varying system scale on the latency of producers raising and consumers delivering location-dependent events.

Finally, the fourth experiment assesses the accuracy of location-dependent event dissemination. The experimental results show that using proximity can limit the effect of network partitions, even in sparsely populated areas, and that optimization techniques can be applied to reduce the cost of location-dependent event dissemination without compromising event delivery accuracy.

All experiments were conducted by deploying prototypical realizations of representative collaborative application scenarios. The producers and consumers that comprise these applications are distributed on between two and four notebook computers (depending on the specific experiment) running the Microsoft Windows XP operating system on a 1 GHz Intel Pentium III processor and interact with other entities using a live ad hoc network using Lucent Orinoco Gold WiFi (IEEE 802.11b) PCMCIA cards with a channel capacity of 11 Mbit/s. The third experiment, where the latency of location-dependent event delivery is assessed, also hosts producers and consumers on an embedded machine designed to control a mobile robot. This robot controller runs the Microsoft Windows XP Embedded operating system on a 1.3 GHz Intel Celeron M processor and interacts with other entities using a live IEEE 802.11g ad hoc network with a channel capacity of 54 Mbit/s. One or more entities may reside on each machine and the individual application deployment is further outlined below for each experiment. The hosting notebook computers were placed within ad hoc communication reach of each other, approximately five meters apart, and their physical location remained stationary. Entity locations and mobility are simulated throughout these experiments using the simulated GPS-based location service. The radio transmission range $\mathtt{T}_{\mathrm{R}}$ for each machine was emulated to be $\mathtt{T}_{\mathrm{R}} = 200$ meters. This ensures that an entity discards all communication messages received from entities located beyond $\mathtt{T}_{\mathrm{R}}$, even though the distance between the physical locations of their host machines is less than $\mathtt{T}_{\mathrm{R}}$. Multiple runs, 1,000, were conducted for each latency experiment and the data collected were averaged over those runs.

5.1 Cost of Location-Dependent Event Dissemination

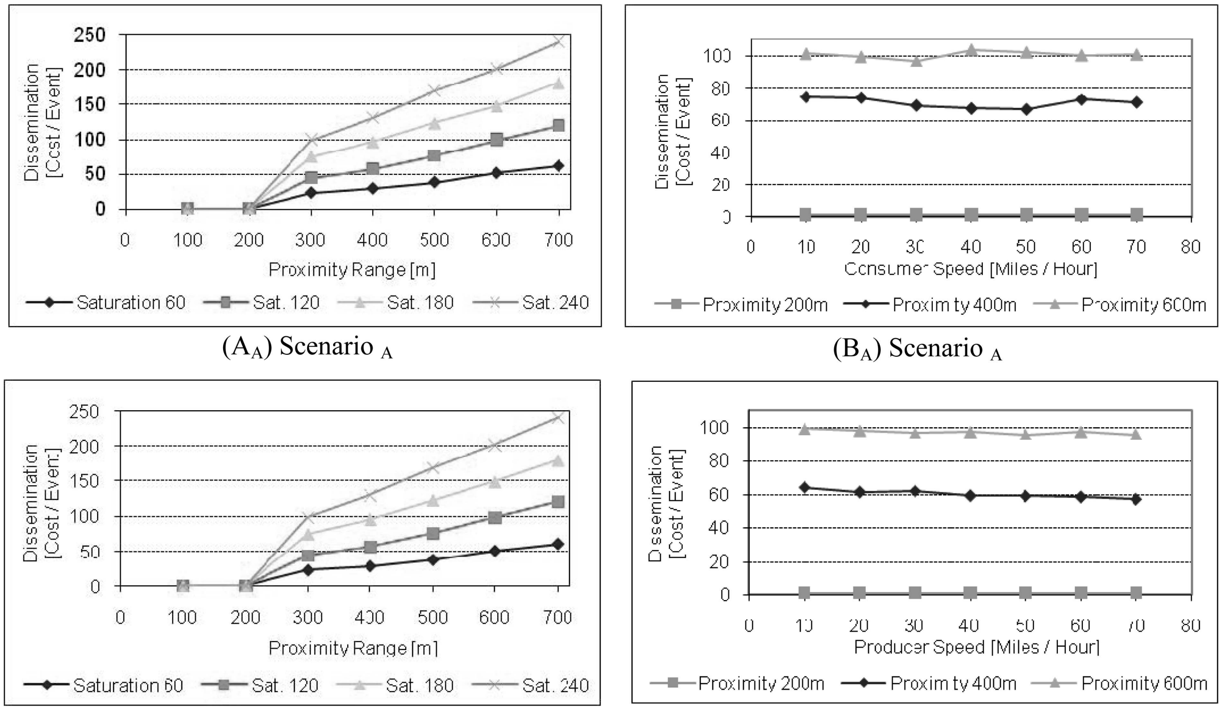
The primary measurement of interest in this experiment is a quantity we refer to as *cost*. We assign a relative cost to the dissemination of a single location-dependent event. Cost $\mathtt{C}_{\mathrm{d}}$ describes the number of messages required when propagating an event from a producer to the consumers residing within its radio transmission reach $\mathtt{T}_{\mathrm{R}}$ and the forwarding of this message to consumers beyond this range. Hence, cost depends on the number of connected consumers residing within a particular proximity $\mathtt{N}_{\mathrm{CP}}$ and provides a qualitative indication of the bandwidth required for disseminating a location-dependent event (or for disseminating an announcement) within a proximity range $\mathtt{P}_{\mathrm{R}}$, as shown below. For example, for a proximity range that exceeds radio transmission reach (2), the cost of a producer disseminating an event to three consumers, each of which forwards the message that describes the event once, is described as 4; one message sent by the producer and three messages forwarded by the consumers. In the same scenario, the cost of a producer disseminating an announcement is also described as 4, one message sent by the producer and three messages forwarded by the consumers. This allows us to measure the cost independently of the actual frequency of event publishing while varying a number of application parameters. These parameters include the migration speed of the entities, the range of the proximity within which events are disseminated, and the number of consumers.

$$\mathtt{C}_{\mathrm{d}} = 1 \qquad \mathtt{iff}\ (\mathtt{P}_{\mathrm{R}} \leq \mathtt{T}_{\mathrm{R}}) \tag{1}$$
$$\mathtt{C}_{\mathrm{d}} = 1 + \mathtt{N}_{\mathrm{CP}} \qquad \mathtt{iff}\ (\mathtt{P}_{\mathrm{R}} > \mathtt{T}_{\mathrm{R}}) \tag{2}$$

Note that (2) describes the cost for disseminating a location-dependent event (or an announcement) to the consumers in a proximity that is covered. Hence, it outlines cost for event dissemination to all consumers located in a proximity if the proximity is covered. In the presence of network partitions, it describes cost for event dissemination to the consumers within the producer's partition.

5.1.1 *The Application Scenarios*

We have selected and realized two collaborative application scenarios from the traffic management domain for this evaluation. ScenarioA models a broken-down car providing a stationary warning service to vehicles within its vicinity. This scenario is set on a two-way road with a producer



(A_A) Scenario $_A$ (B_A) Scenario $_A$

(A_B) Scenario $_B$ (B_B) Scenario $_B$

Fig. 7. (A_A, A_B) Dissemination cost as a function of proximity range and saturation. (B_A, B_B) Dissemination cost as a function of migration speed and proximity for a saturation of 120. All published events have been delivered to the consumers as no event loss has been recorded in any experiment.

acting as the broken-down car disseminating its events to consumers representing approaching vehicles. These vehicles are randomly distributed on the lanes of the road. $\texttt{Scenario}_B$ models an ambulance providing a mobile warning service to nearby vehicles for them to yield the right of way. This scenario is set similarly to the previous scenario on a two-way road and comprises an ambulance disseminating its events to a number of randomly distributed consumers representing vehicles. Both scenarios include a circular-shaped proximity with a central reference point that is mapped to the location of the respective producer and involve a road section of 1,400 meters with the producer being located at the center of the section. All consumers move on this road section and their number defines the saturation of the area. The application scenarios were deployed in turn on four notebook computers placed on the sides of a $5 \times 5\ \mathrm{m}^2$. One machine has been hosting the producer and the remaining three machines have accommodated an equal share of the consumers.

5.1.2 Results and Analysis

Fig. 7 (A_A and A_B) shows event dissemination cost as a function of proximity range $\texttt{P}_R$ for $\texttt{Scenario}_A$ and $\texttt{Scenario}_B$, respectively. The respective proximities define the set of consumers residing inside the area of interest. These ranges have been selected to include proximities in which all consumers can be reached using a single-hop radio transmission ($\texttt{P}_R \leq \texttt{T}_R$) as well as those that require multihop routing ($\texttt{P}_R > \texttt{T}_R$). The largest proximity covers the whole scenario area enabling all consumers on the road section to receive events. The results essentially demonstrate how proximities bound event dissemination cost by bounding the number of consumers that forward a certain event. They show a significant difference when comparing dissemination cost within the single-hop reach to the cost beyond this range.

Beyond the single-hop range, all results show similar tendencies of increasing cost with expanding proximities and rising saturations as every consumer residing inside a certain proximity forwards events. This illustrates that exploiting proximity enables STEAM to transparently select the appropriate protocol when disseminating events. STEAM uses its cost-efficient single-hop protocol for disseminating events within proximities that are covered by the producer's $\texttt{T}_R$ and employs the multihop version only when transmitting events beyond $\texttt{T}_R$. Other middleware platforms [49], [53], [54] typically use either a single-hop protocol with propagation range limitations or a more expensive multihop protocol for both short and long-range dissemination. However, Fig. 7 (A_A and A_B) most significantly illustrates that the results recorded for $\texttt{Scenario}_A$ and $\texttt{Scenario}_B$ are virtually identical. This is consistent with our design that does not require additional control messages for handling service mobility, and hence, given the similar configuration, the dissemination costs for the stationary service scenario, are comparable to those of the mobile service scenario.

Fig. 7 (B_A and B_B) depicts the event dissemination costs recorded for $\texttt{Scenario}_A$ and $\texttt{Scenario}_B$, respectively, as a function of migration speed. These results show the effect of migration speed, and thus, of a dynamically changing network topology that reflects entity movement, on dissemination cost. Similar results were recorded for $\texttt{Scenario}_A$, where mobile consumers deliver events from a stationary service, and for $\texttt{Scenario}_B$ in which a mobile

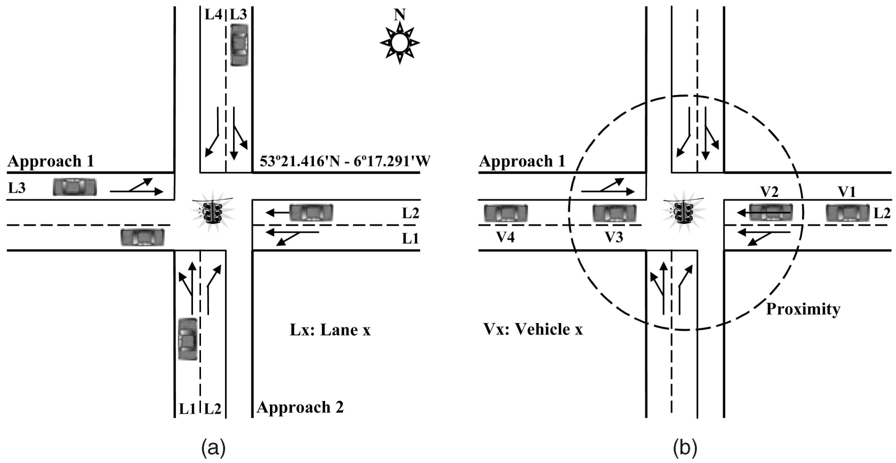


Fig. 8. Modeling the intersection. (a) North Circular Road and Prussia Street. (b) Routes and filters.

service provided by an ambulance moving along the road is being used by stationary consumers representing stopped vehicles. These results essentially show that the cost is low within single-hop reach and increases for proximities expanding beyond single-hop range. Significantly, the results recorded for $\texttt{Scenario}_{\mathrm{A}}$ and $\texttt{Scenario}_{\mathrm{B}}$ demonstrate that event dissemination cost does not depend on the speed of consumers or on the speed of the producer. This is due to the fact that STEAM exploits proximities to control multicast-based flooding, and consequently, does not introduce extra overhead for maintaining routing information that needs to be updated more frequently with increasing relative migration speed. Hence, event dissemination cost is independent of the relative migration speed between producer and consumers, regardless of whether such relative migration speed is the result of producer migration, consumer migration, or the simultaneous migration of producer and consumer. The study of flooding-based multicast protocols for ad hoc networks presented by Lee et al. [54] presents similar conclusions, and hence, argues that neither the number of transmitted messages nor the associated delivery ratio is a function of the relative speed of the interacting entities. Nonetheless, relative migration speed is an important factor along with the size of a proximity and the broadcast period for disseminating announcements that might prevent a possibly fast moving consumer from detecting a proximity and ultimately from receiving events of interest. We expect to further evaluate this issue in a future version of STEAM that supports collaborative applications with guaranteed quality of service requirements.

5.2 Benefits of Distributed Event Filtering

In this case, the scenario evaluated models the interaction between vehicles passing through an intersection and the intersection's traffic light disseminating its light status. The scenario has been realized according to the intersection of North Circular Road (NCR) and Prussia Street (PST), located in Dublin's inner city. It is based on real data gathered by induction loops, which have been provided by the Dublin City Council, describing vehicle movements and light status at the intersection over a period of 24 hours.

Fig. 8a illustrates the intersection and outlines how the traffic flow can be broken up into two distinct phases. The intersection comprises two *approaches*: Approach 1 describes the traffic flows arriving from east and west, whereas approach 2 describes the traffic flows arriving from north and south. Approach 1 consists of three lanes and approach 2 consists of four lanes. The traffic light for both approaches is considered to be located in the center of the intersection at the stated latitude and longitude. The *cycle* time defines the duration for the two approaches to complete their combined sequence of light changes. The proportion of the cycle length that is assigned to one particular approach is called the *split*. The split between the phase of approaches 1 and 2 is 45 to 55 percent. The intersection data were acquired over a period of 24 hours starting on the 3rd of December 2002 at 6 pm. It consists of a sequence of records, each describing a cycle duration and the number of vehicles passing through the intersection on each individual lane during the cycle.

The setup includes three notebook computers placed five meters apart, one machine hosting the traffic light and the other two hosting the vehicles arriving on approaches 1 and 2, respectively. The traffic light raises an event every second for each approach to disseminate the light status, approach name, and light location. Vehicles approach the intersection in their respective lanes at an average speed of 25 miles per hour (MPH) (the intersection is located in a 30 MPH zone). Each vehicle follows a predefined route according to its approach lane simulated by its location service. Fig. 8b depicts an example route of a vehicle in lane 2 of approach 1. The available intersection data do not describe the behavior of an approaching vehicle in terms of queuing; these indicate the number of vehicles passing the intersection during a green light sequence. Hence, vehicles are modeled to reflect this behavior arriving at the intersection in time to pass the light during a green light sequence.

Fig. 8b also illustrates the use of distributed event filters in this scenario. The traffic light announces events of type "Traffic Light" and the associated proximity, which defines the radius of the area of interest surrounding the traffic light. The radius has been set to 40 meters to allow for vehicle breaking distance (16 meters) and update rate (once

TABLE 1
Results of the Experiment

	Approach 1			Approach 2			
	Lane 1	Lane 2	Lane 3	Lane 1	Lane 2	Lane 3	Lane 4
Number of Vehicles	6652	3320	3728	3038	1383	2802	1135
No Event Filter	469945	235093	264557	210114	95743	193931	78730
Distributed Event Filter	24139	11905	13553	9436	4488	8805	3543
Relative Decrease	94.9%	94.9%	94.9%	95.5%	95.3%	95.5%	95.5%

per second) of the location service. This radius guarantees that an approaching vehicle receives at least two events before having to decide whether or not to stop at the light. Vehicles subscribe to "Traffic Light" events and define a content filter that matches events on their approach when they are moving toward the traffic light. This combination of filters causes vehicles 1 and 4 to discard "Traffic Light" events as they reside outside the scope of the proximity. Even though vehicles 2 and 3 are inside the proximity, only vehicle 2 will deliver "Traffic Light" events. The content filter of vehicle 3 prevents event delivery since the vehicle is moving away from the traffic light. The actual proximity filter and the content filter are shown below. The proximity filter is stationary and defines the circular-shaped area while the content filter uses a filter term on the event attribute describing the location of the traffic light to match events for approaching vehicles. The realization of these filters has been discussed in further detail in [55].

```
//proximity filter of the traffic light
SP_Shape cir = new SP_Circle(proxRadius);
SP_ProximityFilter pf = new
SP_ProximityFilter(cir, SP_ABSOLUTE,
                    navalLoc);
//content filter of the vehicle
SC_ConjunctiveContentFilter cf = new
SC_ConjunctiveContentFilter();
cf->addTermPOS(TrafficeLightLoc,
SC_POS_DISTANCE_DECREASES, null);
```

5.2.1 Results and Analysis

This experiment comprises two runs using the same stimuli. The first run applies distributed events filters as described above whereas the second run lacks any filters assuming the communication range of the traffic light's wireless transmitter to limit event dissemination. As described above, we assume the radio transmission range to be 200 meters.

Table 1 summarizes the number of vehicles passing through the intersection on each lane as well as the total number of events delivered to these vehicles in each experiment. It also includes the relative decrease of delivered events between the two experiments. The data in Table 1 show a substantial reduction, averaging at around 95 percent, in the number of events delivered to the vehicles when applying distributed event filters. This is hardly surprising considering the bounding of the propagation range and the content filter discarding events once a vehicle has passed the traffic light. Applying distributed event filters in experiment 1 causes additional overhead due to the proximity discovery. The traffic light announces its proximity by propagating beacons within the proximity area. Vehicles discover the proximity upon entering the area on delivering a single beacon message. Assuming this to be equivalent to delivering an additional event per vehicle would reduce the relative decrease by approximately 1.4 percent. However, the overall number of events delivered would still substantially decrease, on average, by over 93 percent.

The realization of an application scenario such as this may also help to illustrate the usability of our middleware and of its programming abstractions. Our techniques provide high-level programming abstractions that enable the realization of a wide range of collaborative applications, for example, ranging from social services, to traffic management applications, to mobile gaming, without depending on a comprehensive understanding of the underlying middleware. Application programmers rely on their understanding of the specific application requirements to derive appropriate configurations for our high-level abstraction, for example, to configure the shape and size of the proximity of a location-based filter or to define the dissemination frequency of a location-independent announcement [52].

5.3 Latency of Location-Dependent Event Delivery

We have realized an application scenario for this evaluation that models the interaction between a producer disseminating location-dependent events through a wireless ad hoc network and one or more consumers delivering these events. The producer has been deployed on one notebook computer and the consumers have been hosted by another. The locations of the notebook computers hosting these entities have been chosen so that the producer and all consumers are within transmission range T_R for single-hop communication and the location-based filter has been defined to allow event dissemination within this range. This enables the experiment to assess the latency of

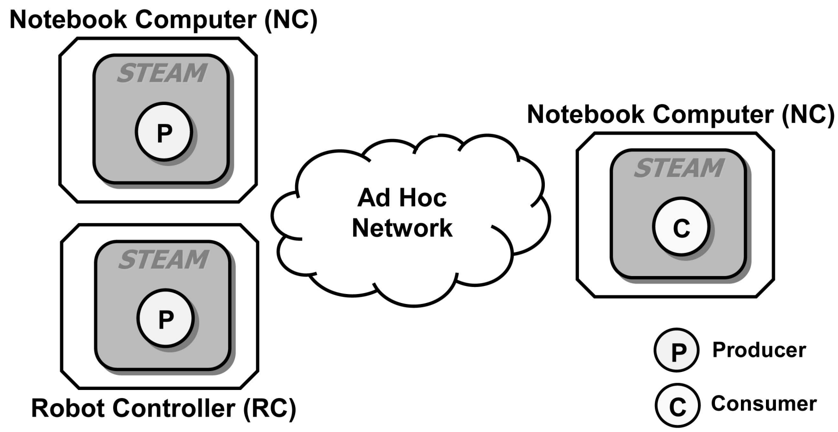


Fig. 9. Application scenario for assessing the latency of a producer raising a location-dependent event on either a notebook computer or a controller for a mobile robot.

location-dependent event delivery by illustrating the performance behavior of producers and consumers when raising and delivering location-dependent events, respectively. The experiment was repeated to assess the performance behavior of producers and consumers hosted on an embedded controller for a mobile robot. The latency of a producer deployed on the robot controller interacting with notebook-based consumers was measured as well as the latency of a consumer deployed on the robot controller interacting with a notebook-based producer. Figs. 9 and 10 illustrate the experiment for assessing the performance of a producer raising a location-dependent event and of a consumer delivering a location-dependent event, respectively. This enables the experiment to assess the effect of the underlying computing platform on the latency of location-dependent event delivery. The latency imposed by a low-performance notebook computer, whose performance we consider similar to those of today's handheld devices, is compared to the latency of the kind of designated embedded controller that might be deployed in the stationary and mobile entities described in our scenarios. The effect of parameters describing the scale of a system on delivery latency, including the latency of retrieving and matching event filters, is measured while excluding the latency of wireless communication, which is beyond the control of our techniques. Hence, the conducted measurements have focused on determining *producer latency* and *consumer latency* of location-dependent event delivery. Producer latency represents the time for a collaborative application to invoke the raise operation, and thus, includes processing (filtering and marshalling) and sending of a location-dependent event, thereby indicating the throughput of a producer. The content of the event type used specifies a set of five parameters, one each of type `Integer`, `Time`, `Location`, `String`, and `Double`. Consumer latency specifies the time for processing (marshalling and filtering) a received location-dependent event and for passing it to the application by invoking the delivery handler. Consumers employ a conjunctive content filter with a filter expression that consists of a set of filter terms, one of which is applied to each of the event parameters. The actual values of these filter terms have been chosen to match all the disseminated events.

As mentioned above, these measurements have been conducted as a function of a number of application parameters, including the numbers of announcements, subscribers, and subscriptions, which typically describe the scale of a system. In addition to announcing and raising events for the purpose of these latency measurements, the producer announces a number of other event types (and

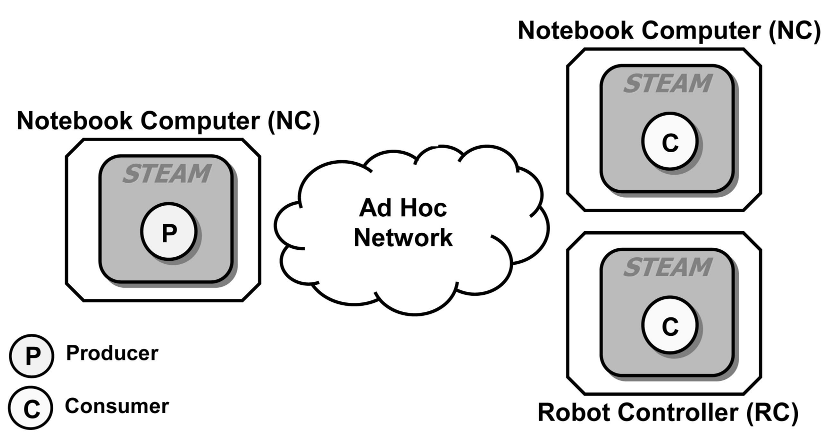


Fig. 10. Application scenario for assessing the latency of a consumer delivering a location-dependent event on either a notebook computer or a controller for a mobile robot.

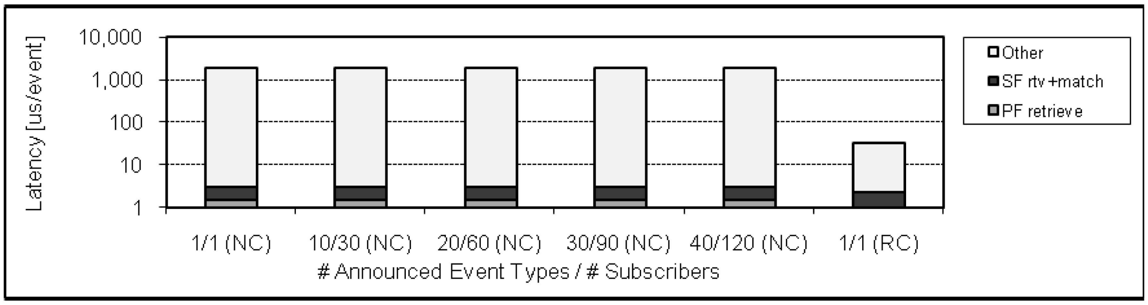


Fig. 11. Latency of a producer raising a location-dependent event as a function of the number of announced event types and the number of subscribers in a system. Producers are hosted on either a notebook computer (NC) or a robot controller (RC). As producer latency has been shown to be independent of the numbers of announced event types and subscribers, only one set of measurements is shown for RC.

their location-based filters) modeling a scenario in a (large) system consisting of several event types. This traditionally imposes extra computational load on producers. Moreover, the producer disseminates its location-dependent events to a number of interested consumers, which have subscribed to these events. Similar to adding announcements, adding subscribers models the effect of increasing system scale on producers. And finally, consumers may subscribe to several event types, causing them to maintain multiple subscriptions and filters. This models the effect of increasing system scale on consumers.

5.3.1 Results and Analysis

Fig. 11 depicts the measured producer latency as a function of the number of announced event types and associated location-based filters, i.e., proximity filters, and the number of subscribed consumers. The ratio of announced event types to subscriptions has been chosen to reflect scenarios such as the emergency vehicle warning service and the traffic light status service, where the proximities of event types might overlap and services are provided to subscribed vehicles. Producers are hosted on either a notebook computer or an embedded robot controller. The latency for raising a location-dependent event on a notebook computer was found to be approximately 1.8 milliseconds, which is equivalent to a throughput of just above 550 events per second. The latencies recorded for subject filter retrieval and matching (SF rtv + match) and proximity filter retrieval (PF retrieve) are 1.55 and 1.45 microseconds, respectively. Marshalling and sending events accounts for the remaining latency (Other). These experimental results demonstrate that the latencies for raising location-dependent events and, in particular, for processing event filters are independent of the numbers of announced event types and subscribers in a system. However, increasing the number of proximity groups in a system also increases the number of obligatory announcements and is likely to result in the dissemination of a larger number of events. Such an increase in scale, while not affecting the local latency of individual producers or consumers, may impact the dissemination latency of the broadcast medium where it is shared by multiple proximity groups. The latency for raising a location-dependent event on a robot controller was found to be approximately 0.03 millisecond, significantly less compared to the notebook-based producer latency. While the latencies recorded at 1.13 and 1.14 microseconds for processing subject filter and proximity filter, respectively, represent a latency of about 75 percent compared to that recorded for the notebook computers, the most significant reduction was found for the latency for marshalling and sending events, which can be attributed to the increased ad hoc network capacity. The comparison of these experimental results demonstrates the dramatic impact that the communication and computation resources associated with different devices can have on throughput. We also argue that this comparison of the performance of the types of devices supported by STEAM is to be preferred over comparing STEAM performance with the performance achieved by other systems with vastly different computing resources and network characteristics in addition to different application requirements and configurations.

Fig. 12 illustrates the recorded consumer latency on a notebook computer as a function of the number of subscriptions to other event types maintained by the consumer. The latencies for retrieving and matching subject filters, content filters, and proximity filters, as well as the overall latency for delivering events, are independent of the varying number of subscriptions. Marshalling and invocation of the application delivery handler accounts for the remaining latency (Other). These experimental results demonstrate that the latencies for delivering location-dependent events and, in particular, for processing event filters are independent of the number of subscriptions in a system. Fig. 12 also illustrates the latency for consumers hosted on a robot controller, which was found to be approximately 65 percent compared to the notebook-based consumer latency and with linear reductions to all latencies measured. Unlike the producer latency measurements, these consumer latency measurements exclude latency for accessing the ad hoc network, and as a result, the shown latency reduction can be attributed solely to the increase in computing power rather than to the increase in both computing power and network capacity.

Some of the latencies outlined in Figs. 11 and 12 are specific to application parameters. Content filter matching latency depends on the number of filter terms and on the type of the parameter to which these terms apply; proximity filter evaluation latency may vary with the geographical shape of the proximity area, and passing events to an application depends on the realization of the delivery handler. Even though evaluating these application-specific latencies is straightforward, such an evaluation has been

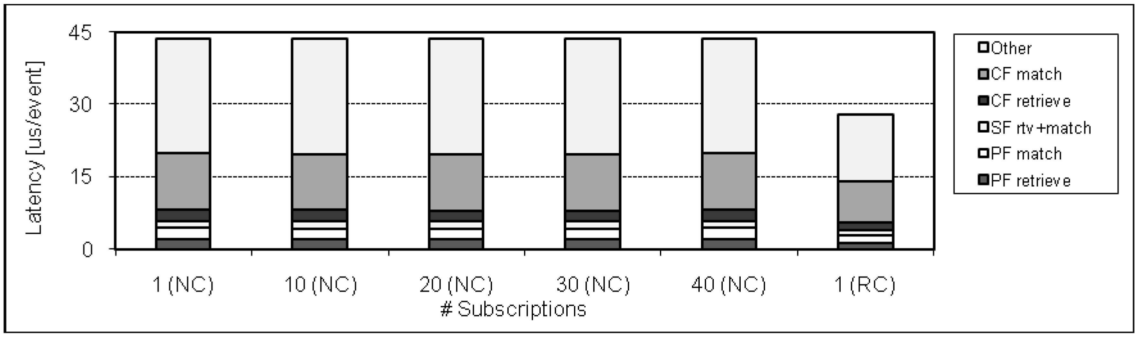


Fig. 12. Latency of a consumer delivering a location-dependent event as a function of the number of subscriptions to other event types in a system. Consumers are hosted on either an NC or an RC. As consumer latency has been shown to be independent of the number of subscribers, only one set of measurements is shown for RC.

omitted as it represents an insignificant contribution in the context of this paper.

Furthermore, Fig. 11 outlines a scenario in which a significant number of event types and associated location-based filters have been announced in the same geographical area. Up to 40 announcements have been disseminated concurrently without incurring collisions when computing the hash-based group identifiers. Each of these announcements comprises a different event type of between 14 and 18 characters in length and the same proximity filter defining a stationary, circular area with a radius of 200 meters.

5.4 Accuracy of Location-Dependent Event Dissemination

This experiment is based on five collaborative application scenarios used to evaluate the accuracy of location-dependent event dissemination under different network conditions. $\texttt{Scenario}_{\mathrm{A}}$, the broken-down car, and $\texttt{Scenario}_{\mathrm{B}}$, the approaching ambulance, are taken from Section 5.1 using the same proximity configuration, vehicle distribution, and saturation range. These proximity configuration and saturation range are used for all scenarios in this experiment. However, the scenarios vary in their entity distributions, and as a result, in their network conditions. $\texttt{Scenario}_{\mathrm{C}}$ models the traffic light intersection scenario from Section 5.2 in which vehicles are randomly distributed on the two approaches. $\texttt{Scenario}_{\mathrm{D}}$ models an augmented reality game, where a particular player informs nearby players of her game status. Such players can reside anywhere in the game space, and hence, are randomly distributed. And finally, $\texttt{Scenario}_{\mathrm{E}}$ models a search and rescue operation, where a coordinator directs a group of distributed searchers, each responsible for examining an equal part of the search area. These searchers are homogeneously distributed in the search area. Fig. 13 summarizes these five application scenarios used for assessing the accuracy of location-dependent event dissemination under different network conditions. The

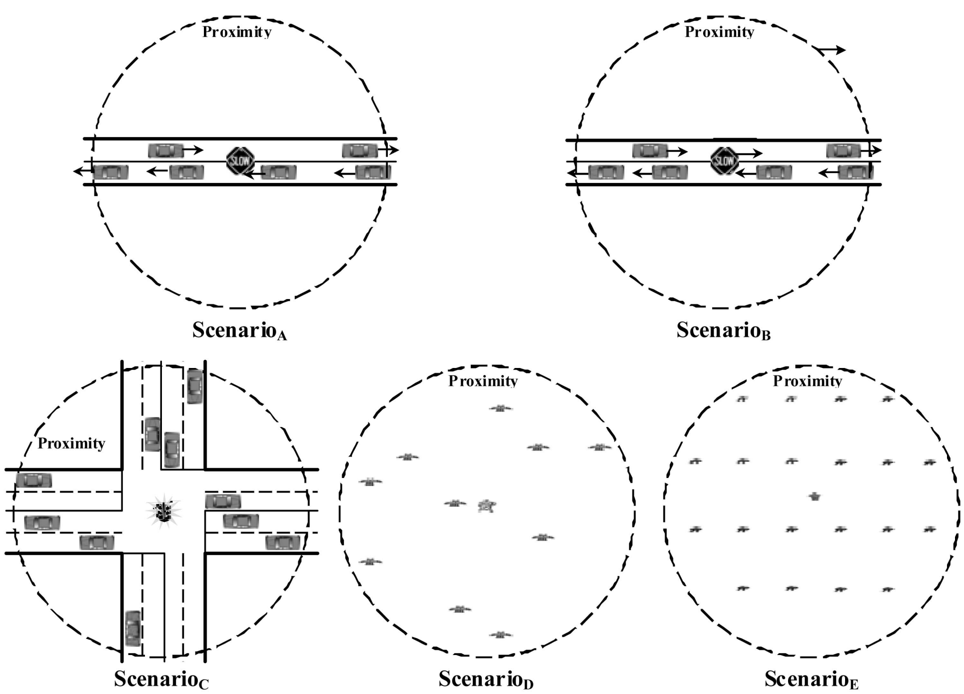


Fig. 13. The five collaborative application scenarios used to evaluate the accuracy of location-dependent event dissemination under different network conditions.

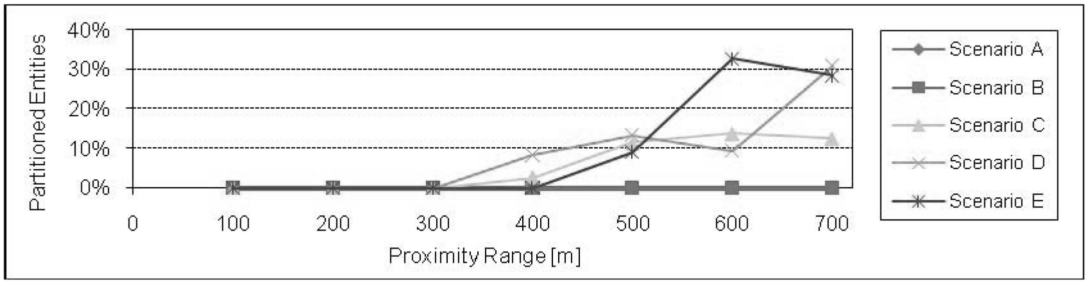


Fig. 14. Partitioned entities as a function of proximity range for event dissemination in sparsely populated areas with saturation 60.

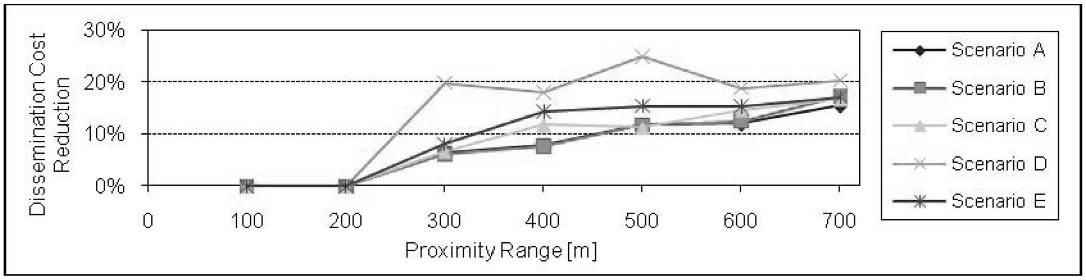


Fig. 15. Dissemination cost reduction due to a gossiping technique as a function of proximity range and averaged for saturations 120, 180, and 240.

prototypical realizations of the application scenarios were deployed, in turn, on four notebook computers as in the first experiment. One machine has been hosting the producer and the remaining three machines have accommodated an equal share of the consumers.

5.4.1 Results and Analysis

Measurements were conducted for all five scenarios disseminating an event from the pivotal entity to the entities located inside the proximity ranging from 100 to 700 meters and for saturations 60, 120, 180, and 240.

The first measurement analyzes the impact of network partitions on the accuracy of event dissemination. It quantifies the ratio of entities situated within the proximity that are prevented from receiving the event due to the network partition. To further increase the probability of network partitions emerging in this experiment, 20 percent of all entities are configured to restrain from forwarding received event messages, thereby, emulating entity failure. Of the measurements conducted for saturations between 60 and 240, network partitions are found only for saturation 60. Fig. 14 shows the result for saturation 60 illustrating the percentage of entities located inside the proximity that did not receive the event. Only very sparse populations in scenarios `C`, `D`, and `E`, where entities are distributed over large parts of the proximity, cause the ad hoc network to partition and prevent accurate dissemination of location-dependent events. Moreover, partitions are recorded only for proximity ranges above 300 meters demonstrating that location-dependent event dissemination can limit the effect of the network partitions on the entities in sparsely populated areas.

The second measurement analyzes the impact of possible optimization techniques on the accuracy of event dissemination. These measurements are based on the previously described configuration, while applying gossip-based message forwarding [56] with parameters $\mathrm{G}(\mathrm{d} = 200.0, \mathrm{p} = 0.8)$ instead of imposing entity failure. These gossip parameters compel entities outside the single-hop distance of $\mathrm{d} = 200.0\ \mathrm{meters}$ to retransmit messages with a probability of $\mathrm{p} = 0.8$. Event dissemination for saturations 120, 180, and 240 is found to be accurate in the presence of gossip-based optimization. All entities situated inside the proximity received the event even though fewer messages are used to disseminate the event. Fig. 15 summarizes the reduction to the cost of event dissemination as a result of applying gossip-based message forwarding while preserving accurate delivery. This demonstrates that optimization techniques can be used in many collaborative application scenarios to significantly reduce the dissemination cost without compromising delivery accuracy. Note that the relatively high gossip probability and entity failure rate used in this experiment have been chosen to demonstrate the effect of optimizations techniques and network partitions. Discovering ideal gossip parameters and tolerable entity failure rates for the respective scenarios is considered beyond the scope of this paper.

6 Conclusion

As mobile computing becomes increasingly common and the event-based communication paradigm is increasingly adopted for engineering distributed applications, it is natural that many researchers have considered extensions to event-based communication models to support mobile applications. To date, the bulk of this work has addressed the requirements of what we characterize in this paper as *nomadic* mobile applications, i.e., those whose components move between points of attachment to the network being used to interact with other static or mobile components. Such applications are characterized by the use of fixed network infrastructure. More recently, event-based communication researchers have begun to consider the requirements of applications composed of collections of interacting mobile components, which we characterize here as *collaborative* mobile applications. These applications are characterized by sporadic collaboration between collocated mobile entities in the possible absence of any fixed network infrastructure. In this paper, we have introduced a number of techniques, including location-independent announcement and subscription coupled with location-dependent filtering and event delivery, that can be used by event-based middleware to support such applications. In particular, we have demonstrated that these techniques can be implemented in a fully decentralized manner and have shown the benefit of exploiting location to improve event filtering for this class of applications. We expect that these techniques will be increasingly incorporated in future middleware platforms deployed to support collaborative mobile applications, for example, in the automotive industry where the emergence of standardized intervehicle and vehicle-to-roadside communication (based, for example, on 802.11p) will give the development of such applications significant impetus in the future. Other emerging application domains ranging from assisted living to mobile gaming will likely also benefit from the availability of such middleware.

While our techniques provide the basis for supporting a wide range of mobile applications, important work remains to extend this work to support applications with guaranteed quality of service requirements in terms of event-delivery latency and this is the topic of our future work. We intend to exploit the concept of proximity introduced here as the basis for performing admission control to allocate the necessary communication resources for timely event delivery within a dynamically varying population of mobile components.

ACKNOWLEDGMENTS

The authors would like to thank the Dublin City Council's Traffic Office for providing the traffic data that made their evaluation of STEAM possible. This work was partly supported by the Irish Higher Education Authority's Program for Research in Third Level Institutions cycle 0 (1998-2001) and by the FET Program of the Commission of the European Union under research contract IST-2000-26031 (CORTEX).

REFERENCES

[1] T. Sivaharan, G. Blair, A. Friday, M. Wu, H. Duran-Limon, P. Okanda, and C.-F. Sørensen, "Cooperating Sentient Vehicles for Next Generation Automobiles," *Proc. First ACM Int'l Workshop Applications of Mobile Embedded Systems,* 2004.

[2] J. Kjeldskov, S. Howard, J. Murphy, J. Carroll, F. Vetere, and C. Graham, "Designing TramMateña Context-Aware Mobile System Supporting Use of Public Transportation," *Proc. 2003 Conf. Designing for User Experiences,* pp. 1-4, 2003.

[3] K. Cheverst, N. Davies, K. Mitchell, A. Friday, and C. Efstratiou, "Experiences of Developing and Deploying a Context-Aware Tourist Guide: The GUIDE Project," *Proc. MobiCom,* pp. 20-31, 2000.

[4] G.D. Abowd, C.G. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton, "Cyberguide: A Mobile Context-Aware Tour Guide," *ACM Wireless Networks,* vol. 3, pp. 421-433, 1997.

[5] DaimlerChrysler, "Vision of Accident-Free Driving," DaimlerChrysler HighTech Report #2/2003, 2003.

[6] M. Michaelian and F. Browand, "Field Experiments Demonstrate Fuel Savings for Close-Following," California PATH Research Report UCB-ITS-PRR-2000-14, Univ. of California, Berkeley, 2000.

[7] B.P. Crow, I. Widjaja, J.G. Kim, and P.T. Sakai, "IEEE 802.11 Wireless Local Area Networks," *IEEE Comm. Magazine,* vol. 35, no. 9, pp. 116-126, Sept. 1997.

[8] P. Verissimo, V. Cahill, A. Casimiro, K. Cheverst, A. Friday, and J. Kaiser, "CORTEX: Towards Supporting Autonomous and Cooperating Sentient Entities," *Proc. European Wireless Conf.,* pp. 595-601, 2002.

[9] US Dept. of Transportation—Research and Innovative Technology Administration, "Vehicle Infrastructure Integration," http://www.vehicle-infrastructure.org, Jan. 2009.

[10] J. Bacon, K. Moody, J. Bates, R. Hayton, C. Ma, A. McNeil, O. Seidel, and M. Spiteri, "Generic Support for Distributed Applications," *Computer,* vol. 33, no. 3, pp. 68-76, Mar. 2000.

[11] R. Meier and V. Cahill, "Taxonomy of Distributed Event-Based Programming Systems," *The Computer J.,* vol. 48, pp. 602-626, 2005.

[12] G. Cugola, E.D. Nitto, and A. Fuggetta, "The JEDI Event-Based Infrastructure and Its Application to the Development of the OPSS WFMS," *IEEE Trans. Software Eng.,* vol. 27, no. 9, pp. 827-850, Sept. 2001.

[13] I. Burcea, H.-A. Jacobsen, E.d. Lara, V. Muthusamy, and M. Petrovic, "Disconnected Operation in Publish/Subscribe Middleware," *Proc. IEEE Int'l Conf. Mobile Data Management,* pp. 39-50, 2004.

[14] Y. Huang and H. Garcia-Molina, "Publish/Subscribe in a Mobile Environment," *Proc. Second ACM Int'l Workshop Data Eng. Wireless and Mobile Access,* pp. 27-34, 2001.

[15] R. Meier, "Communication Paradigms for Mobile Computing," *ACM SIGMOBILE Mobile Computing and Comm. Rev.,* vol. 6, pp. 56-58, 2002.

[16] G. Coulouris, J. Dollimore, and T. Kindberg, *Distributed Systems, Concepts and Design,* fourth ed. Pearson Education Limited, 2005.

[17] I. Podnar, M. Hauswirth, and M. Jazayeri, "Mobile Push: Delivering Content to Mobile Users," *Proc. Int'l Workshop Distributed Event-Based Systems,* pp. 563-570, 2002.

[18] P. Sutton, R. Arkins, and B. Segall, "Supporting Disconnectedness—Transparent Information Delivery for Mobile and Invisible Computing," *Proc. IEEE Int'l Symp. Cluster Computing and the Grid,* pp. 277-285, 2001.

[19] L. Fiege, F.C. Gartner, O. Kasten, and A. Zeidler, "Supporting Mobility in Content-Based Publish/Subscribe Middleware," *Proc. ACM/IFIP/USENIX Int'l Middleware Conf.,* pp. 103-122, 2003.

[20] *CORBAservices: Common Object Services Specification—Notification Service Specification, Version 1.0,* Object Management Group, 2000.

[21] *Jini: Distributed Event Specification,* Sun Microsystems, Inc., 1999.

[22] A. Carzaniga, D.S. Rosenblum, and A.L. Wolf, "Design and Evaluation of a Wide-Area Event Notification Service," *ACM Trans. Computer Systems,* vol. 19, pp. 283-331, 2001.

[23] P.R. Pietzuch, B. Shand, and J. Bacon, "A Framework for Event Composition in Distributed Systems," *Proc. Fourth ACM/IFIP/USENIX Int'l Conf. Middleware,* pp. 62-82, 2003.

[24] D. Frey and G.-C. Roman, "Context-Aware Publish Subscribe in Mobile Ad Hoc Networks," *Proc. Ninth Int'l Conf. Coordination Models and Languages,* pp. 37-55, 2007.

[25] J. Steffan, L. Fiege, M. Cilia, and A. Buchmann, "Scoping in Wireless Sensor Networks," *Proc. Second Int'l Workshop Middleware for Pervasive and Ad-Hoc Computing,* 2004.

[26] I. Burcea and H.-A. Jacobsen, "L-ToPSS—Push-Oriented Location-Based Services," *Proc. Fourth Very Large Data Base (VLDB) Workshop Technologies for E-Services,* pp. 131-142, 2003.

[27] K. Schelfthout, "Supporting Coordination in Mobile Networks: A Middleware Approach," PhD thesis, Dept. of Computer Science, Katholieke Universiteit Leuven, Nov. 2006.

[28] A.L. Murphy, G.P. Picco, and G.-C. Roman, "Lime: A Middleware for Physical and Logical Mobility," *Proc. 21st Int'l Conf. Distributed Computing Systems,* p. 524, 2001.

[29] A.L. Murphy, G.P. Picco, and G.-C. Roman, "LIME: A Coordination Model and Middleware Supporting Mobility of Hosts and Agents," *ACM Trans. Software Eng. Methodology,* vol. 15, pp. 279-328, 2006.

[30] C. Julien and G.-C. Roman, "EgoSpaces: Facilitating Rapid Development of Context-Aware Mobile Applications," *IEEE Trans. Software Eng.,* vol. 32, no. 5, pp. 281-298, May 2006.

[31] M. Mamei and F. Zambonelli, "Self-Maintained Distributed Tuples for Field-Based Coordination in Dynamic Networks," *Proc. 2004 ACM Symp. Applied Computing,* pp. 479-486, 2004.

[32] C.-L. Fok, G.-C. Roman, and G. Hackmann, "A Lightweight Coordination Middleware for Mobile Computing," *Proc. Sixth Int'l Conf. Coordination Models and Languages,* pp. 135-151, 2004.

[33] M. Caporuscio, A. Carzaniga, and A.L. Wolf, "Design and Evaluation of a Support Service for Mobile, Wireless Publish/Subscribe Applications," *IEEE Trans. Software Eng.,* vol. 29, no. 12, pp. 1059-1071, Dec. 2003.

[34] F. Sailhan and V. Issarny, "Scalable Service Discovery for MANET," *Proc. Third IEEE Int'l Conf. Pervasive Computing and Comm.,* pp. 235-244, 2005.

[35] N. Priyantha, A. Chakraborty, and H. Balakrishnan, "The Cricket Location-Support System," *Proc. ACM/IEEE MobiCom,* pp. 32-43, 2000.

[36] P. Steggles and S. Gschwind, *The Ubisense Smart Space Platform,* Ubisense Limited, 2005.

[37] A.G. Audi, "The Informed Driver," http://www.audi.com/audi/com/en2/about_audi_ag/news/unternehmen/monthly_summary/Travolution_promotes_eco-friendly_driving.html, Jan. 2009.

[38] US Dept. of Transportation—Research and Innovative Technology Administration, "Cooperative Intersection Collision Avoidance Systems (CICAS)," http://www.vehicle-infrastructure.org/applications/, Jan. 2009.

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

[40] B. O'Hara and A. Petrick, *The IEEE 802.11 Handbook: A Designer's Companion.* Standards Information Network IEEE Press, 1999.

[41] F. Cristian, "Synchronous and Asynchronous Group Communication," *Comm. ACM,* vol. 39, pp. 88-97, 1996.

[42] G. Banavar, T. Chandra, R. Strom, and D. Sturman, "A Case for Message Oriented Middleware," *Proc. 13th Int'l Symp. Distributed Computing,* 1999.

[43] M.O. Killijian, R. Cunningham, R. Meier, L. Mazare, and V. Cahill, "Towards Group Communication for Mobile Participants," *Proc. ACM Workshop Principles of Mobile Computing,* pp. 75-82, 2001.

[44] R. Cunningham and V. Cahill, "Time Bounded Medium Access Control for Ad Hoc Networks," *Proc. Second ACM Int'l Workshop Principles of Mobile Computing,* pp. 1-8, 2002.

[45] T. Goff, N.B. Abu-Ghazaleh, D.S. Phatak, and R. Kahvecioglu, "Preemptive Routing in Ad Hoc Networks," *Proc. MobiCom,* pp. 43-52, 2001.

[46] K. Paul, S. Bandyopadhyay, A. Mukherjee, and D. Saha, "Communication-Aware Mobile Hosts in Ad-Hoc Wireless Network," *Proc. Int'l Conf. Personal Wireless Comm.,* pp. 83-87, 1999.

[47] G. Gaertner, E. O'Nuallain, A. Butterly, K. Singh, and V. Cahill, "802.11 Link Quality and Its Prediction—An Experimental Study," *Proc. Ninth IFIP Int'l Conf. Personal Wireless Comm.,* pp. 147-163, 2004.

[48] K. Scott and S. Burleigh, *Bundle Protocol Specification: IETF RFC 5050,* 2007.

[49] J. Orvalho, L. Figueiredo, and F. Boavida, "Evaluating Light-Weight Reliable Multicast Protocol Extensions to the CORBA Event Service," *Proc. Third Int'l Conf. Enterprise Distributed Object Computing,* pp. 255-261, 1999.

[50] B. Preiss, *Data Structures and Algorithms with Object-Oriented Design Patterns in C++.* John Wiley & Sons, Inc., 1999.

[51] P.S. Wang, *C++ with Object-Oriented Programming.* PWS Publishing Company, 1994.

[52] A. Senart, M. Bouroche, and V. Cahill, "Modelling an Emergency Vehicle Early-Warning System Using Real-Time Feedback," *Int'l J. Intelligent Information and Database Systems,* special issue on information processing in intelligent vehicles and road applications, vol. 2, pp. 222-239, 2008.

[53] L. Opyrchal, M. Astley, J. Auerbach, G. Banavar, R. Strom, and D. Sturman, "Exploiting IP Multicast in Content-Based Publish-Subscribe Systems," *Proc. IFIP/ACM Int'l Conf. Distributed Processing,* pp. 185-207, 2000.

[54] S.-J. Lee, W. Su, J. Hsu, M. Gerla, and R. Bagrodia, "A Performance Comparison Study of Ad Hoc Wireless Multicast Protocols," *Proc. IEEE INFOCOM,* pp. 565-574, 2000.

[55] R. Meier, "Event-Based Middleware for Collaborative Ad Hoc Applications," PhD thesis, Dept. of Computer Science, Trinity College, Univ. of Dublin, Sept. 2003.

[56] Z.J. Haas, J.Y. Halpern, and L. Li, "Gossip-Based Ad Hoc Routing," *Proc. IEEE INFOCOM,* pp. 1707-1716, 2002.

René Meier is a Lecturer in Computer Science at Trinity College Dublin. His interests as a researcher in distributed systems cover a variety of overlapping areas related to very large-scale, context-aware mobile and pervasive computing systems as well as self-organizing systems. These include mobile and message-oriented middleware, context and location-aware services, self-organizing peer-to-peer systems, and the application of middleware architectures to global smart spaces, especially in the transportation domain. He is an editorial board member for the Mobile and Pervasive computing community of the IEEE Computer Society's online magazines home, *Computing Now*, and an associate editor for the *IGI International Journal of Ambient Computing and Intelligence*. He is a member of the IEEE and the IEEE Computer Society.

Vinny Cahill holds a Personal Chair in Computer Science at Trinity College Dublin, where he also serves as the Head of the Discipline of Computer Systems and Director of Research for Computer Science and Statistics. His research interests include many aspects of distributed systems, in particular, middleware and programming models for ubiquitous and mobile computing with application to intelligent transportation systems, global business systems, and personal healthcare/independent living. He has published more than 100 peer-reviewed publications in international conferences and journals. He is a member of the IEEE Computer Society.

▹ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.