Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 9 (1): 1-14, 2009
ISSN 1812-5654

Trends in Middleware Abstarction for Context Dissemination in Mobile Ad-Hoc Network

V. Hakami and M. Dehghan
Wireless Network Laboratory, Department of IT and Computer Engineering,
Amirkabir University of Technology, P.O. Box 15875-4413,
No. 424 Hafez Street, Tehran, Iran

Abstract: Context dissemination completes the value chain of contextual information procurement and is identified as a precondition to actual context use by enlarging the visibility scope of context sources beyond the local acquisition or fusion entity and towards a network neighborhood. When leveraged for context-aware services in MANETs (Mobile Ad-hoc NETworks), an often-stated goal is to cater for personalized refinement and constant monitoring of a high volume of heterogeneous data with lower chance of longevity. Context-aware middleware abstractions pave the way by masking the physical distribution of data and by working out the appropriate logic for identifying the most relevant subset of context for use by an application component. This study compiles the state-of-the-art trends in programming abstractions built around the notion of context and present an exhaustives a detailed survey of their relevant middleware incarnations. Also, some key design considerations are identified for MANET-based context dissemination and the study investigates how these requirements have been approached from a variety of directions by the reviewed systems.

Key words: Context-aware computing, MANETs, Information dissemination, programming abstractions, middleware systems

INTRODUCTION

Systems sentience, as conceivable today, is growing in dimension owing to the upward trend in technological viability of sensory and perception artifacts: virtual, social, physical and physiological, to name a few. This makes context-awareness, the associated computing paradigm witness integral adoption in the emerging ICT paragons e.g., Autonomic Communication (Dobson *et al*., 2006) and Ambient Intelligence (AmI) (Remagnino and Foresti, 2005). Exercising the idea in the presence of the indispensable issue of ad hoc mobility in B3G/4G wireless evolution as one promising ambience enabler helps yield adaptation, unobtrusiveness and versatility for the array of solutions meant to operate in such a state of flux.

Much of the challenge in context-aware ad-hoc networking can be investigated from the context dissemination perspective wherein the repercussions of ad-hoc mobility on managing contextual information are acknowledged as the key concern, an issue barely touched in exploitation schemes featuring context as a secondary or auxiliary element. The collection and distribution of context information involves several complex problems such as data integrity, discovery, real-time update, secure storage, caching and replication. Although these tasks have been studied under various circumstances and for different types of networks, their execution from an ad-hoc networking point of view is little known and worth examining. This study takes the initiative to provide an exhaustive detailed survey on the latest trends in middleware efforts towards addressing context dissemination specific problems in MANETs. The classification is driven by the programming abstraction adopted in each system and the style of communication it promotes. The researchers believe this fundamental perspective yields a more accurate insight into the issues and intricacies involved in context dissemination middleware design. It is important to mention that no related study exists on surveying solutions for context-aware ad-hoc networking in general and the dissemination problem in particular due to the complexity and the non-obvious taxonomy of the available schemes. This study is will be among the first to provide a selection of well studied models and approaches in the field.

Corresponding Author: Mehdi Dehghan, Department of IT and Computer Engineering, Wireless Network Laboratory, Amirkabir University of Technology, P.O. Box 15875-4413, No. 424 Hafez Street, Tehran, Iran
Tel: +98 21 6454 2749 Fax: +98 21 6649 5521

IMPLICATIONS AND REQUIREMENTS

On top of the distinct challenges ad-hoc mobility poses for every system development effort (i.e., un-instrumented coincidental organization, topological fluidity, episodic connectivity and resource scarcity, etc.), providing for context-sensitivity brings about even more as well novel requirements. This section identifies the fundamental design principles and outlines the salient features sought in solutions mediating context dissemination for MANETs.

Contextual scoping: Contextual information is distinguishable from other conceivable contents by its situational semantics, i.e., a particular context item only makes sense relative to some notion of situation characterizing its provider and consumer entities. In effect, context can be considered as a function of time and environment; the environment is in turn a function of the users, services, resources and other entities in the network (Dobson *et al*., 2006). Driven by this property, distribution of context naturally calls for a mechanism to operate beyond traditional content-centric meta-data to also capture as many environmental factors as per required by application components. Contextual scoping, accordingly, refers to the versatility of a dissemination scheme in terms of scoping constraints it caters for to customize context-related propagations in MANETs.

Update management: The collection of information available in the ad-hoc network is constantly changing, due to both mobility of hosts and dynamicity of applications. The volatility typical of contextual values compounds the problem especially when entities or activities need to be monitored in real-time. In such systems, operation on stale data typically leads to wrong decisions for context-aware services. An integral facet of a dissemination solution is then its provision for efficient maintenance constructs that satisfy data temporal consistency and are responsive to environmental changes in general.

Caching/replication: The non-uniform distribution of nodes in the ad-hoc network implies the possibility for the accumulation of context sources in some place, thus forming regions with a high coverage of available data, while at the same time the gaps in between suffer from scarcity of providers. Moreover, topological perturbations due to mobility result in the quick disappearance of a connected path between producer and consumer entities. A reasonable approach to mitigate data unavailability is to equip the dissemination scheme with a caching (or replication) mechanism, which can also prove handy in reducing costly propagation of repeated requests for fairly stable context. It should be noted however that introducing replication, apart from typical inconsistency issues, might in turn lead to implications in terms of Quality of Context (QoC) (Buchholz *et al*., 2003) demands; for instance, it might be required by the interested service that the stored context always meet certain freshness or accuracy conditions.

Security/privacy: In provisioning of context-aware applications, a well amount of sensitive information is subject to communication. Since context data can disclose a lot about an entity, unsafe interactions naturally pose as a breach of confidentiality. Moreover, the heterogeneity of users' privacy demands suggests the need for some personalization and trust mechanism to let them control the extent of access to their context information. Consequently, the dissemination solution should also account for the functionality to express policies and to define ownership of context information, which is communicated with other users/entities.

Scalability: Context dissemination must scale to large population of participating nodes (i.e., the inclusion of new context users and providers), to the increase in the number of context categories and to varying degrees of mobility.

CONTEXT DISSEMINATION ABSTRACTIONS

For the purpose of this study, context dissemination middleware is classified based on the programming abstraction adopted. A common ground for context dissemination abstractions is that they intend to facilitate the extension of the availability of context information beyond a host's immediate scope by providing a logical view of the available resources while at the same time unifying context-sensitive interactions. In dynamic or unpredictable deployment settings such as ad-hoc networks, an abstraction level should also mask the complexity of mobility and churn resultant from the P2P-style of interactions among the application components. Two major categories of context dissemination abstractions are recognized (Table 1), namely: The Global Virtual Context Space (GVCS) (Huang, 2002) and Computational Fields (Co-Fields) (Mamei and Zambonelli, 2006a).

The GVCS abstraction aims at offering an illusion of a centralized information pool in which the request for a local invocation of a query operation will in effect be disseminated on a global scale. Depending on whether

Table 1: Context dissemination middleware abstractions

Context	Objectives
Global Virtual Context Space (GVCS)	To maintain a uniform illusion of a global dynamic context space and to exploit it for making system-wide inquiries for contextual information
Symmetric	To offer the same complete perception of the maximal context to all participating parties
Asymmetric (Egocentric)	To give each application direct control to narrow its observations and operations on its specific context space, i.e., a projection of the GVCS
Computational Fields (Co-Fields)	To achieve flexible scoping in context-related propagations and to reduce components interaction to indirect and local communications

this dissemination includes the whole community of the connected components or only a designated portion of the ad hoc network, the GVCS abstraction can be further sub-classified into symmetric and asymmetric (or egocentric) context sharing, respectively.

Field-assisted context dissemination promotes a radically different perspective on context-awareness. In this model, a sort of spatially distributed data structure, namely, Computational Fields (Co-Fields) (Mamei and Zambonelli, 2006a) is leveraged to achieve customizable scoping of context-related propagations and to reduce application interaction and information exchange into indirect and local communications.

GLOBAL VIRTUAL CONTEXT SPACE (GVCS)

As evidenced by a number of recent middleware efforts (Murphy *et al*., 2006; Cugola and Picco, 2001; Julien and Roman, 2006; Schelfthout and Holvoet, 2004; Payton, 2006), a holistic data-centric view over the entire ad-hoc network forms the basis to devise and support several variations of a mobility-viable shared data space abstraction which can also serve as a meta-model of context and thus be leveraged to disseminate contextual information. Within this perspective, the maximal context as conceivable by a particular application component takes the form of a global virtual data repository whose content is contributed by all sensors, both from the application hosts or from the environment (Huang, 2002). The main drive for the GVCS abstraction is to allow the application developers to interact with spatially distributed context as if it were local, i.e., a programmer simply queries the virtual repository to gain access to context; behind the scenes, queries will be executed in a distributed fashion over the ad hoc network (Payton, 2006) and the accessible GVCS is actively maintained in concert with dynamic changes in topology.

Context-aware middleware that incarnates the GVCS concept can generally be divided into two categories. Symmetric context sharing systems such as Lime (Murphy *et al*., 2006) and PeerWare (Cugola and Picco, 2001) essentially mimic traditional shared data space models in that they intend to offer the same complete perception of the maximal context to all participating parties; this way, an application sees the world in its entirety limited in extent only by the physical connectivity. Accordingly, queries are issued over the same virtual repository which is determined by the logically merged contents of all the local data spaces held by mutually reachable components.

However, as the extent of the underlying network grows, the amount of context information which has the potential to influence an application's behavior becomes large and unmanageable (Roman *et al*., 2007). Obviously, maintaining a consistent up-to-date view for all the observers is expensive in terms of communication and processing costs. The need to scale has given rise to an asymmetric (also called egocentric) notion of context (Julien and Roman, 2006), i.e., each application component observes and operates on its specific context space, which is a projection of the GVCS (Huang, 2002). The key ramification of this decision is that not every component sees the same world, but rather, a personally tailored representation. This asymmetry in viewable context is justified by the observation that while all aspects of the operational environment have the potential to influence the behavior of an application, only a subset there of is actually relevant to its behavior (Roman *et al*., 2007). To utilize this idea in an ad-hoc setting, each application should be given direct control over the scope of its queries to specifically target a subset of the entire network. Such a logical subnet is often defined by measures of distance including physical distance, bandwidth, throughput, or latency (Julien and Roman, 2006). Further constraints are applicable to the attributes of the applications possessing the context items and/or to the properties of their hosting device (location, speed, direction, processing load, memory, battery power, etc.). Figure 1a (Huang, 2002) shows the idea of GVCS and Fig. 1b shows the reification of an egocentric context wherein an application recruits its providers from among nodes in its two-hop neighborhood whose CPU load is less than 50%.

In what follows, we discuss middleware efforts built around the concept of GVCS and will inspect how each system accounts for the peculiarities of context dissemination in the presence of ad-hoc mobility.

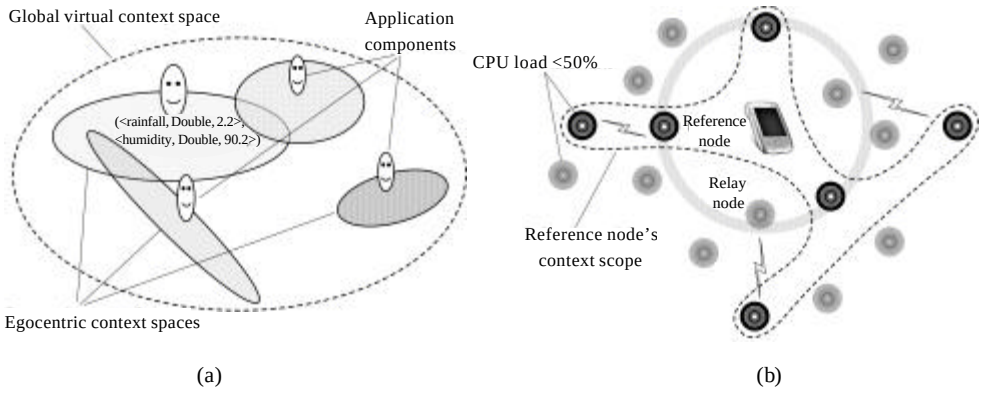

Fig. 1: (a) Global virtual context space and its application-specific projections (b) A node's context scope

Symmetric context sharing using transiently shared data spaces: In order to create an illusion of a centralized information pool, Lime (Murphy *et al*., 2006) and PeerWare (Cugola and Picco, 2001) put forth the notion of transiently shared data spaces (Busi and Zavattaro, 2001). In this model, each application unit is permanently associated with a local data repository which contains public data for use as context by others. When applications are able to communicate, the model leverages on precise rules to engage their data spaces in a sharing relationship. In case these rules are uniform and universally applied to all connected components, applications will engage indiscriminately with the whole community of their (transitively) reachable peers and thus they will eventually perceive the same interaction space. The engagement normally occurs in an atomic fashion and entails the joining of a group of hosts by a mobile unit followed by transient merging of its local data space with those of the overall group's (Murphy *et al*., 2006). This merging is dubbed transient since it occurs only at a perceptual level, i.e., the data spaces belonging to the applications are not actually merged, but rather, the results for all active queries will be re-computed so that the content of the newly joined space is taken into account. Similar considerations hold for the departure of a mobile unit, resulting in the disengagement of the corresponding data space; i.e., the content of the unit's data space is removed atomically from the transiently shared space perceived by the remaining units.

The Lime's approach to context dissemination (Murphy *et al*., 2006; Murphy and Picco, 2004): Lime fosters a Linda-like (Gelernter, 1985) instantiation of a global transient sharing space through what is referred to as a federated tuple space. Basically, the Linda tuple space is divided into many subspaces, each permanently attached to mobile hosts or agents. Transient sharing accordingly allows dynamic re-configurability of the individual tuple space contents based on changes in connectivity. A proof-of-concept adaptation of the middleware for physical context dissemination is shown in (Murphy and Picco, 2004) which requires that context and application data be insulated from one another and stored in separate local spaces.

The Lime's primitives to interact with context are basically those of Linda's (read, in, out) supplemented with their non-blocking (rdp and inp) and bulk/group (rdg and ing) counterparts; the bulk operations which return a group of matching tuples are especially helpful in retrieving historical context (e.g,. a user's movement itinerary) (Murphy and Picco, 2004). While these primitives cater for proactive (pull-based) querying, Lime's reactive constructs suit contextual monitoring and immediate adaptation to changes. A reaction R(s, p) is defined by a code fragment s specifying the callback function to be executed when a tuple matching the pattern p appears in the federated space. As for pattern matching needed for its operation set, Lime enjoys Linda's content-based data access, while at the same time extending its usual exact matching semantics to allow for the provision of more flexible constraints over tuple fields; in particular, the LIGHTS (Balzarotti *et al*., 2007) package at the core of Lime affords range queries, fuzzy matching and data aggregation at the tuple level. These extensions can be resorted to formulate complex descriptive queries coupled with several QoC meta-data constraints.

Contextual scoping in Lime is provided by, besides tuple templates, the option of annotating Linda primitives with host or agent identifiers to single out a specific provider from among the current contributors of the federated space. To cater for these operations a presence service maintains, on each host, a list of all accessible hosts in the same partition. This service is implemented in the form of a special group management protocol

(Huang *et al*., 2004) which also guarantees a consistent community-wide update diffusion in the events of engagements and disengagements. Key to this protocol is the notion of safe distance among hosts which is constantly calculated by a designated leader as a function over the speed and direction of the nodes involved in the communication as well as the maximum time necessary to complete a requested transaction (atomic dis/engagement). If transactions can have longer durations, the safe distance that defines allowable network links becomes shorter and vice versa. In effect, the goal is to prevent any harmful link failure by creating the illusion of announced disconnections (from the group) before a link failure affecting the group could happen; that is, in case hosts are close enough, according to some safe distance, disconnection is not possible and if they are just far enough there is plenty of time to carry out a configuration change before disconnection actually occurs (Huang *et al*., 2004). However, the protocol's scalability will degrade as it strives to maintain Lime's symmetric worldview while the number of devices, connections, or degree of mobility grows. Lime's scalability is also affected by its lack of an underlying protocol support to effectively steer the requests in the network (e.g., based on tuple templates) and unless specifically addressing a certain provider, a query must flood the accessible partition at each data search.

In terms of replication, a lightweight veneer on top of the federated tuple space of Lime is outlined in (Murphy and Picco, 2006) which locally copies tuples according to user-specified profiles and consistency modes. The profiles specify the tuples to be replicated and the consistency modes come up with three possibilities: never, which never updates a replica, master, which updates only from the master version of the tuple and any, which updates indifferently from master or replica versions. The core of the implementation is to exploit Lime's reactive constructs to watch for master tuples and depending on the consistency modes, possibly for replica tuples as well. When a reaction fires with a new tuple, the listener for that reaction must take the appropriate measures to keep the replicas inside the tuple space in line with the replication profile.

As for security/privacy, the studies by Handorean and Roman (2003) has presented a way to add on security features to the original Lime model. The extensions allow applications to protect selected tuple spaces and even individual tuples through the use of passwords. The same passwords can also be used to encrypt communication among hosts when exchanging messages related to sharing of specific tuple spaces. Bravetti *et al*. (2005) investigates, in detail, the security issues for TupleSpace-based systems.

PeerWare (Cugola and Picco, 2001): Of other studies promoting the symmetric and transient sharing of (primarily computational) context is PeerWare. Compared to Lime's flat tuple-based model, PeerWare introduces a richer hierarchical tree-like data organization for the global and individual spaces. Specifically, data is structured like a standard file system with multiple roots where, the tree nodes play the role of directories and the leaves, or documents in PeerWare's language, represent the files. The global content is then perceived as the superposition of all directories and documents shared by currently connected hosts; i.e., if for instance a directory node N_1 on peer A contains a single document D_1, while its homologous node on peer B contains a single document D_2, the logically merged content will feature N_1 parenting both D_1 and D_2. PeerWare leverages on a synergic combination of both TupleSpace and Publish/Subscribe communication models to enable querying the global space and subscribing to events occurring in it.

The API to access the global context includes an execute operation which runs an arbitrary action on the projection of the data space identified by two filter functions over nodes and documents. An interesting feature is the use of mobile code technology to ship an action's application-specific code along with each query to be fetched and executed on remote hosts, thus achieving both bandwidth reduction and an appropriate level of customization; in case no action is explicitly specified, the operation simply returns matching contents, unchanged. Similarly, a subscribe primitive exists which allows peers to register interest for events of a particular property occurring on a filtered set of items and to execute an associated code fragment in case one of such events occurs. In (Cugola and Picco, 2001), the expressiveness of the language to formulate these filters is not specified and the study does not go much beyond discussing the usability of regular expressions or XML-like queries as possible candidate solutions. In effect, PeerWare is meant to be a conceptual middleware design and the specifics of its run-time infrastructure are left up to real-life implementations. Thus, for instance, the model does not prescribe anything about how the routing of system messages (i.e., queries, subscriptions and events) must be performed in a MANET, e.g., what is the topology of the network interconnecting the peers and what algorithms are used to perform routing on top of it. A potential scalability benefit, however, lies in the hierarchical structure of data which can be leveraged to naturally scope the query operations, possibly through a content-based routing scheme steering requests only toward nodes that actually contain the relevant data. PeerWare serves as the core of the MOTION platform (Kirda *et al*., 2002) which builds a framework of collaborative services

for mobile users of a mesh setting. The hierarchical approach adopted assumes the existence of several thick nodes in the network, which may not always be the case for MANETs.

Egocentric context computation using declarative specifications: While the symmetric sharing paradigm best suits (small-scale) Collaborative Working Environments (CWEs), the egocentric-style context-awareness targets more individualized interactions, where applications tend to have their specific and independent contextual needs from the environment. To support application-controlled scoping of query execution, middleware systems in this direction allow for an application entity to abstractly, or rather, declaratively specify its desirable context as well as the set of providers featuring that context. Such declarative specification is typically comprised of a group of constraints over properties of data, applications, hosts and network links, effectively restraining the application's perception to a pruned version of the global repository. Typically, these specifications can be redefined at run time as needs or expectations change.

EgoSpaces (Julien and Roman, 2006): In the computational model presented in EgoSpaces, software agents are the units of modularity and mobility and mobile hosts are only containers characterized by their geographic location. Similarly to Lime, data is stored in a local tuple space and provisions are made to move this space along with its owning agent upon migration. EgoSpaces structures context in terms of fine-grained units called views. An agent may request one or more of these views by providing declarative specifications, each consisting of three tuple patterns (over data, agent profiles and host profiles) and a set of network constraints (including a link weight metric, a cost function and a cut-off bound). It can be clearly seen that the context definition in Fig. 1b easily fit in a view's constraint-based declaration. EgoSpaces, again, is founded on the notion of transient sharing, ruled by view specifications upon engagements and disengagements; accordingly, views and the data belonging to them are not actually calculated until or unless an application actively queries the view. This way the overhead of constructing and maintaining a view is incurred only when access operations are issued; however, the application always benefits from the perception of a persistent data structure that reflects the current contents of its view(s), i.e., a filtered set of tuples.

EgoSpaces comes with roughly the same primitive set as that of Lime's, but an operation's scope is restricted to the extent of a particular view whose id is passed on to it as an argument. A key component of the middleware is a protocol that given the view's networking constraints calculates the qualifying subnet of hosts in the actual network over which the view's operations are issued. The SICC protocol (Source-Initiated Context Construction) discussed by Julien *et al.*(2008) operates by building and maintaining a bounded minimum-cost spanning tree rooted in the view initiating host and entailing all nodes such that the cost of the shortest path connecting them to the root satisfies the stipulated cut-off bound. Upon invocation of a query operation, the query manager component of the middleware uses SICC to construct a shortest path tree and at the same time disseminates the request to every host which happens to lie within the span of this tree; however, the query will be processed only by those peers meeting the specified agent-level constraints running on nodes meeting the specified host constraints. In case of a blocking operation, the query remains registered on eligible hosts until the issuer explicitly deregisters it. Also, if new hosts move into the tree boundary while the query remains active, they will receive the request; likewise, the query is removed from those walking out. When it comes to reply propagation, the resultant structure essentially serves as a reverse multicast tree that allows desired information to funnel back to the requesting agent (Julien and Stovall, 2006). As with Lime, long-lived context monitoring is made viable thanks to the middleware's asynchronous reactive callbacks whose semantics is backed by SICC's tree maintenance. It might also be of note that in order to take maintenance measures, SICC relies on the information derived from an environmental monitoring package, namely CONSUL (Hackman *et al.*, 2005), which maintains on each host a registry of sensors available locally and of those on neighboring nodes. As monitor values change, the query manager is notified of and the metric is re-evaluated leading possibly to a topological-level reaction from SICC.

SICC is basically a flooding protocol whose deterministic behavior may achieve strongly consistent results. A caveat, however, exists when concurrent possibly overlapping view declarations grow in number, in which case the protocol's egocentric design basis plays havoc with the middleware's scalability. Scaling with SICC is more of an issue in dense networking scenarios, wherein the protocol's lack of a mechanism to effectively curb the unnecessary forwarding may give rise to the notorious broadcast storm problem. Moreover, in order to ensure boundedness, the distance metric used for building the view is assumed to always increase monotonically the further the query message gets propagated from the root. Defining the context in this way

makes it hard, or rather ineffective to realize inherently destination-based views (e.g., to find one or all components at a specific coordinate in space). In sum, although EgoSpaces in principle caters for nearly the finest-grained contextual scoping as per required by a context-aware application, yet the efficacy of the model is practically limited by the fact that SICC performs the actual dissemination based solely on link-level properties.

EgoSpaces is equipped with a basic support for data duplication through a special built-in behavior construct, the semantics of which essentially reduces to a high priority reaction which copies remote tuples (belonging to a specific view) and leaves the originals unaffected. Unlike Lime, the middleware does not manage consistency of the replicas.

An agent-specified access control function and a set of per view credentials form the basis for the middleware's provision of security. The credentials can be a standard set of attributes (e.g,. the agent's id), or a third-party authentication and are presented as a tuple. The access control functions can be described by a set of policies defined as patterns, or templates, which are evaluated on an individual basis for each tuple adhering to the view constraints; that is, a tuple belongs to a view only if it satisfies the view constraints and as well, the requesting agent's credentials meet the requirement of the access control function of the agent owning the tuple. The function can also account for the type of operation requested, e.g., some data should be restricted to read-only access. EgoSpaces, however, should rely on some third party cryptography scheme for the secure transmission of credentials.

Object places (Schelfthout and Holvoet, 2004): Declarative context specification is also pursued in ObjectPlaces (Schelfthout and Holvoet, 2004) which offers an analogous notion of egocentric views for automatic gathering of data objects distributed across a set of MANET nodes. Each application component maintains viewable data in a local collection of objects called an objectplace, which is basically a tuplespace variant and can be accessed by roughly the same associative operations such as put, read and take. As is the case in EgoSpaces, a remote application expresses its context of interest in the form of a view declaration essentially designated by the following four parameters: a distance metric and a bound, a host constraint, an objectplace name and an object template. However, unlike EgoSpaces which allows remote removal of tuples, views as presented in ObjectPlaces are purely observational, i.e., the contributing objectplaces are only subject to local manipulations and remote applications can only observe the associated collection and stay current on changes. Also, while in the baseline design for EgoSpaces, a view always has the interface of a tuplespace, ObjectPlaces tailors the representation of the resultant view at the whim of the application components (e.g., a sorted collection, an accumulation to a value, etc.). It is further noticeable that ObjectPlaces does not adopt the transient sharing model, but rather, views are realized from the very moment they are declared and are actively maintained ever since.

As for view construction and maintenance, ObjectPlaces again relies on a bounded spanning tree protocol, namely View (Schelfthout *et al*., 2005), which can be considered as an alternative to SICC of EgoSpaces (Julien and Roman, 2006; Julien *et al*., 2008). In order to observe a view, an application component executes a watch operation with the given object template on all qualifying objectplaces residing on nodes satisfying the stipulated host constraints and lying within the scope of the tree. In effect, the watch directive is the only method in the middleware's API for issuing a distributed query, basically implementing the semantics needed for long-lasting context monitoring. To keep its viewers posted with respect to its changing content, an objectplace provides an event-based asynchronous interface which, compared to the reactive constructs of Lime's and EgoSpaces', triggers notifications upon both insertion and deletion of data objects.

Though both SICC of EgoSpaces and the View protocol of ObjectPlaces study by building an overlay structure on demand of an application's explicit request, the View protocol account for topological reconfigurations in a proactive fashion; that is, while in SICC changes in the network are only propagated when it is detected that a link is out of specs, View handles this by regularly sending update messages, essentially achieving more robustness at the expense of some overhead. This proactive behavior also proves handy in detecting network partitions in that when a subset of nodes becomes disconnected from the root, they will no longer receive stabilizing distance updates, resulting in their measures to eventually grow out of bound. Also, while changes in the profiles of hosts and applications may affect their qualification status for being a member of an application's context over time, SICC makes no explicit attempt to report such changes to the viewing component. In ObjectPlaces, however, a node disqualifies itself by stopping the expectedly regular unicast of a so-called member message which, once past a specific time-out period, will lead to its removal from the root's acquaintance list; needless to say that the newly qualified nodes will announce their membership in a similar manner. Finally, it might be of note that the View protocol, as is,

does not factor in any non-uniform weights with regard to communication links and 'hop count' is the sole metric used to build contextual subnets.

Query ME (Payton, 2006): Yet another middleware effort to be assimilated into egocentric-style context data dissemination is presented in (Payton, 2006). QueryME envisions a database-like GVCS abstraction and provides for the execution of highly customized SQL-like queries over the virtual repository in order for the applications to acquire a tailored portion of the global context. Code mobility forms the mainstay of QueryME's design in that it renders an application capable of bundling with each query an assortment of policy specializations dictating the details of its execution. Specifically, mobile code fragments can be installed over the network to encapsulate three customizable elements of a typical query execution: a context policy declaring an application's desired context scope in terms of constraints imposed on properties of the physical network as well as on hosts and applications within the associated sub-network, a propagation policy to further truncate the spread of query over the designated scope (using e.g., controlled flooding, random sub-tree, random path, etc.), and finally, a reply processing policy, an option to encode some in-network processing scheme in order for the nodes to aggregate query responses and cut down on the costs by communicating only the aggregate result as the query reply.

Despite its SQL-like interface, QueryME again uses tuples as the baseline representation for context data items; yet, unlike EgoSpaces where each application is associated with its own tuplespace, QueryME stores the items contributed by all the applications residing on a particular node in a single host-level space, accessible only by the components of the middleware's query service. A typical query invocation in QueryME is then specified by, apart from the aforementioned trio of policies, a data constraint function in the form of a tuple template along with a tag, identifying the kind of operation to be performed (e.g., GET, EXISTS, MIN, MAX, etc.). Both QueryME and EgoSpaces draw on a semantically enhanced tuplespace engine which essentially adds on complex constraint functions to Linda's classic content-based retrieval.

Upon issuance from a given application, a query is passed on to the manager component of the QueryME's service which first processes the network constraints portion of its context policy in order to begin constructing the overlay data structure. To do so, the middleware relies on a distributed spanning tree protocol, almost akin to SICC of EgoSpaces (Julien and Roman, 2006; Julien *et al*., 2008), which will include only those nodes whose path cost satisfies the specified context bound. Once a set of neighbors have been determined to be potential candidates for belonging to the tree, the propagator component applies the propagation scheme to specify a subset of these neighbors that are eligible for propagation and returns this set as the actual context children. The service disseminates the query along with its mobile code specializations to the duly elected neighbors. Hosts and application constraints that comprise the remainder of the context policy are then imposed on the corresponding profiles, (also captured as a tuple) to see if the recipient is qualified to further process the query. A processor component on those peers eligible to initiate a response will utilize the local results from the host-level repository and, in case dictated by the in-network aggregation policy, replies from the peer's set of actual context children.

Featuring a TupleSpace core, QueryME enjoys a similar notion of reactions as in EgoSpaces to provide for reactive evaluation of long-lived queries over a topologically maintained context. The middleware introduces the concept of anti-tuples as a way to also signal the deletion of a reported context item. In particular, when a piece of data is removed from the tuplespace, a corresponding dummy (anti) tuple is inserted into the space effectively coaxing its associated reaction to fire.

As is the case in its previously discussed counterparts, QueryME's both contextual scoping and scalability are limited by its underlying spanning tree protocol, yet its ability to account for several propagation policies brings in customization superiority.

FIELD-ASSISTED CONTEXT DISSEMINATION

A Computational Field (Co-Field) (Mamei and Zambonelli, 2006a) can be considered a simple, yet spatially spreading data structure characterized by a unique identifier, a propagation rule, a location-dependent numeric value (expressing the strength of a field at a specific location and taking values that depend on the propagation rule) and any needed number of additional data (to encode and convey any information that can be of use to application components) (Mamei and Zambonelli, 2006b). With the support of a particular middleware infrastructure, fields are injected into the network and get propagated driven by their propagation rule. A field can rely on any computable rule for how its strength value will have to vary as it gets propagated; the strength can simply increase from hop to hop in the network, either linearly or according to other monotonic functions. It may increase up to a specific distance from

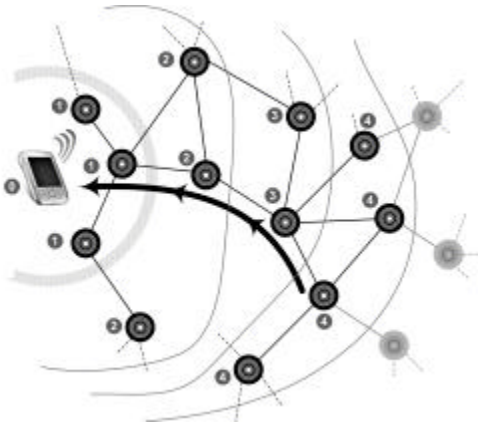

Fig. 2: Context dissemination using field-based gradient routing

the source and then start decreasing or it could even vary according to some periodic function of the distance (Mamei and Zambonelli, 2006b). By assuming different values at different nodes, fields can effectively build a distributed overlay data structure, which can be further used by application components to communicate (Mamei *et al*., 2003). In effect, applications will only have to locally sense propagated fields to acquire contextual information, exchange information with each other and react accordingly to the configuration and shape of the perceived fields, much the same as a physical mass moves in accord with a locally sensed gravitational field (Mamei and Zambonelli, 2006a).

Figure 2 sketches the very basic idea of a field-assisted context querying scheme, using hop count as the sole mechanism to control the propagation. A requesting device can inject a field-like data structure conveying an application query for a particular set of sensorial events up to a maximum of 4 hops. The propagation rule accordingly demands that the field diffuse to all peers and have its strength value incremented as it hops. The source nodes lying within the bound can then locally perceive the income of a query and react by injecting an answer field, which will simply follow downhill the query field's strength gradient reaching back the requester.

Through properly formulated propagation rules, the model has the potential to shape the field structure so as to meet a broad range of scoping constraints (e.g., distance, velocity, spatial direction, etc.). Moreover, the spatial overlays induced by fields' self-routing mechanism can also self-maintain their structures (Mamei and Zambonelli, 2005), given that the fields' propagation patterns are supposed to be dynamically re-shaped by the supporting middleware in order to reflect network and application dynamics.

The abstraction of Co-Fields is instantiated in TOTA (Tuples On The Air) (Mamei *et al*., 2003) via tuple objects of the form: T = <content, propagation rule>. The content is an ordered set of typed elements representing the information carried by the tuple (i.e., its strength value as well as any arbitrary semantic information), while the propagation rule will be encoded by implementing an abstract propagate method. Each node executing an instance of the TOTA middleware hosts a single (non-mobile) application process for which TOTA maintains a local tuple space to store the arriving tuples. The TOTA API provides functionalities allowing the application to inject and delete tuples in the local middleware and to read tuples both from the local tuple space and as well as from those of the node's one-hop neighborhoodneighbors'. Applications may also subscribe and unsubscribe to TOTA's event interface and react accordingly to the income of new tuples matching specific templates. However, there is no primitive notion of distributed queries in TOTA, which means application code must be provided to have the target peers interpret a distributed tuple as a query at the application level.

When a tuple arrives in a node (either as a result of injection into the local middleware or propagation through the network), TOTA begins processing by first evaluating its propagate method. The propagation schema at the core of the middleware's Tuple class hierarchy (Mamei *et al*., 2003) prescribes the following sequence of execution: The middleware first decides on the tuple's acceptance into the system based on any application-specific policy (e.g., the standard implementation discards the duplicates). Next, the tuple's propagation condition is verified to determine if it needs to be further propagated. Following this verification, the tuple's content goes through possible modification, typically to update its strength value. A tuple is then provided with the option of placing subscriptions in the middleware so as to be able to react to events when they happen after the tuple completes its execution. Finally, TOTA stores the tuple in the local tuple space and also propagates it to neighboring nodes only if the tuple has formerly satisfied the propagation condition. Since each node will only have to propagate the tuple to its immediate neighbors, the global effort to spread the tuple is fairly partitioned between the constituting nodes, making the propagation phase practically scalable in MANETs. Also, given that the tuple's content is duplicated in each visited host (assuming a read-only mode), data availability would be maintained in case of disconnections or even partitioning; however, TOTA comes with no particular support for

replica reconciliation. Also, the middleware lacks any kind of security policy to rule accesses to distributed tuples and their updates.

Tuples leverage on subscriptions they make to TOTA's event interface in order to take self-maintenance measures in the face of topological perturbations; specifically, each local tuple can subscribe to the income or the removal of other tuples (belonging to its same type) in its one-hop neighborhood. Upon a removal, each tuple reacts by checking if it is still in a safe-state, i.e. whether its predecessor is alive, or else, it is the source of the distributed overlay. An unsafe tuple erases itself from the local tuple space, leading eventually to a cascading tuple deletion until a safe-state tuple can be found, the source is ultimately reached, or even all the tuples in the associated subnet are deleted. It might also be the case that a safe-state tuple, observing a deletion in its neighborhood, fills the gap and reacts by propagating to that node (Mamei and Zambonelli, 2006b). Similar considerations apply with respect to arrival events; for instance, tuples may re-propagate, overwriting outdated values, and effectively cutting the distributed shape into a steeper gradient towards the source. Though experimental results in (Mamei and Zambonelli, 2006b) show that updates are almost invariably confined within a locality scope from where they took place, the algorithm could not converge if the network topology changes faster than the time required by the self-maintenance process to complete. Also, in order to alleviate the critical race condition likely to happen as a result of a spurious propagation cycle, the self-maintenance algorithm should introduce artificial delays before each deletion (Mamei and Zambonelli, 2006b), which inevitably prolongs the convergence time and can lead to scalability problems, especially in highly mobile scenarios.

TOTA's main superiority over the previous tuple-based systems would be from the very customizability of field-based dissemination schemes, i.e., the potential to map TOTA peers onto a useful assortment of virtual space topologies. Whether TOTA would be an efficient alternative in each case, then, depends and on the actual requirements of the applications and its usability should be weighed against the overheads involved. A proof-of-concept application of TOTA for a GHT-like (Ratnasamy *et al*., 2002) storage and retrieval is outlined in (Mamei and Zambonelli, 2005), where GPSR (Karp and Kung, 2000) is embedded in tuple's propagation rule, effectively conveying data (and or queries) towards geographical coordinates resultant from the application of a pre-defined hash function over the data (or query) key words.

DISCUSSION

Devising programming abstractions to properly capture, disseminate and exploit context is an open research problem. In the dynamic and self-organizing milieu of ad-hoc networks, Tuple or space-based computing has one very strong advantage. It de-couples two orthogonal dimensions involved in context exchange: time and space. This spatio-temporal uncoupling accounts for the model's particular support for context-aware interactions than for more bulky data sharing operations. It should be noted however that while a tuple space can act as a natural repository of context resulting in environmental awareness for software components, the flat structure of tuples impedes complex data organization. More expressive contextual representations as propositioned by some of the approaches (e.g., PeerWare and QueryME) were highlighted throughout our review. However, the emerging trends (Sudha *et al*., 2007) in the area of ubiquitous systems promote a synergistic blend of Tuple Space and Semantic Web namely, Semantic Space (Sudha *et al*., 2007), primarily to enhance tuple's basic representation as well as to augment Linda with vocabulary decoupling; still, an efficient transplantation of this architecture in the fluid world of MANETs is an open topic and thus forms an interesting direction for future research.

The field-theoretic approaches also prove promising for adoption in ad-hoc systems in that the resultant interactions involve only indirect and local communications; moreover, the coCo-field's Field's customizable propagation strategy is ideal for achieving flexible scoping in context-related transmissions. However, probably the biggest issue with field-based abstractions lies in the mismatch between the underlying basic model and the solution they offer at the application level, which can result in complex and tricky implementations for field-assisted services.

As can be seen in the Table 2, replication and security are the most lightly treated areas. The lack of study along the line of replication might in part be attributable to the nature of context (as opposed to ordinary data) since the proposed systems mostly deal with context featuring locality in terms of spatial and temporal semantics whose availability across the entire network is by no means guaranteed. With respect to security, EgoSpaces provides for fine-grained access control policies at the tuple level. Lime's security extension in (Handorean and Roman, 2003), introduces password protection measures to tuple spaces and allows for encrypted communication of messages between hosts.

Table 2: Summary of context dissemination middleware

	Global Virtual Context Space (GVCS)					
	Symmetric			Asymmetric (egocentric)		Field-assisted context dissemination
Middleware criterion	Lime's proof-of-concept C-A adaptation	PeerWare	EgoSpaces	ObjectPlaces	QueryME	TOTA
Features	Insulation of context from application tuples, agent migration	Directory-like representation, Code mobility for on-site primitive execution	Context as tuples, agent migration	Customizable context view representation	Code mobility to customize query execution policies	Encapsulation of context-related information in fields tuples
Communication n model	Linda, transient sharing of context tuple spaces	Linda in synergy of Pub/Sub transient Superposition of directions and documents	Linda, transient sharing of tuple spaces ruled by view declarations	Linda synergy of Pub/Sub	Linda-like communication at the core of the middleware	Co-Fields
Supporting protocols	Unicast routing for ID-annotated operations, A group service to manage (dis) engagements	N.A.	SICC: composition metric minimum spanning tree	View: single metric minimum spanning tree	Composite metric minimum spanning tree	None (application-level self-routing)
Distributed query operations	(Non) blocking Linda primitives (RIO)+bulk modes of read and in for historical context	A core execute primitive with programmable actions	(Non) blocking Linda primitives (R)+bulk modes of read and in	A non-blocking Linda-like watch directive	SQL-like ApI (GET, MAX, AVG, etc.)	Requires application code
Scoping constraints	Tuple templates, host/agent Ids	Data and event filters	Constraints over links, host, agents, and tuples	Constraints over hosts, objectplaces and objects	Constraints over links, hosts, applications, and data, Propagation Policies	Depends on propagation rule
Context update management	Reactive callbacks	Subscriptions	Reactive callbacks	Subscriptions	Reactive callbacks	Subscriptions
Caching/ Replication	Yes	No	Duplication, no replica managing	No	No	Duplication, no replica managing
Security and privacy	Yes	No	Yes	No	No	No
Scalability	Low (symmetric sharing, partition-wide flooding for data search)	Depends on MANET implementation symmetric sharing	Limited by SICC (broadcast storm, tree maintenance)	Limited by view (broadcast storm, tree Maintenance)	Limited by SICC-like SPT protocol	Good (Not in highly mobile scenarios)

The other systems do not use security concepts, which is one of their major shortcomings.

In terms of scalability, Lime is designed for small ad-hoc networks and draws on flooding at the network level which can result in low performance and high searching overhead. It also adopts a symmetric sharing paradigm which cannot scale to large communities of users. While this is also the case in PeerWare, the structural organization of data can still be leveraged to devise more scalable solutions in general. Middleware systems based on Egocentric notion of GVCS have a better chance of scaling in MANETs, yet current implementations rely on spanning tree overlays whose construction and maintenance often proves overkill in dynamic settings. An efficient alternative would be the use of gossiping protocols (Friedman *et al*., 2007) to reduce costly multi-hop communications into opportunistic pair-wise interactions. Such interactions can be seen in TOTA's tuple propagation strategy, yet the self-maintenance algorithm based on cascaded subscriptions is not guaranteed to converge quickly especially in highly dynamic scenarios.

As discussed earlier, the versatility of a dissemination scheme in terms of its provision for contextual scoping is a significant property to have. In an ideal solution, to cater for different scoping constraints would suggest the need for a multi-protocol architecture (e.g., geographic routing, content-based routing, QoS routing, etc.) as well as the viability to dynamically switch between these protocols at run-time. One promising direction would be the adoption of an ActiveNet-like (Tennenhouse and Wetherall, 2007) architecture in middleware's design; in particular, the ActiveNet's underlying idea of putting intelligence in the network by

letting messages execute snippets of code to dynamically compute their next hop destinations can also be utilized in a dissemination architecture to flexibly route context-dependent propagations. An added value also lies in ActiveNet's interposed computing paradigm, which can be adapted to perform in-network processing or information fusion as has been the case in research studies for the parallel area of WSNs (Wireless Sensor Networks) (Levis *et al*., 2005). In effect, by having query replies processed in the network, nodes aggregate responses and communicate only the collective result as the query reply, thus achieving performance benefits compared to when naively contacting each node on the route individually and then locally aggregating the responses. Initial thoughts and prototypical implementations of ActiveNet-inspired approaches to context dissemination in mobile settings can be explored by Kang *et al*. (2004), Riva (2006) and Stovall and Julien (2007).

Yet another important direction of research is concerned with the relatively new notion of quality of context (QoC) (Buchholz *et al*., 2003), which requires that context be provided together with describing metadata (e.g., accuracy of the information), primarily with the intention of allowing for the estimation of the reliability of the resultant values. The reviewed systems in this study provide only partial support for QoC-bound services in that they merely consider the definition of metadata metrics to resolve a flavor of quality-constrained contextual queries. A more demanding aspect of QoC support can be viewed with reference to update management requirements, i.e., the ability to schedule context-related transmissions so as to maximize data freshness, while at the same time imposing the least possible burden over the MANET and its participants. In other terms, it is necessary to somehow establish a proper trade-off between QoC and QoS and in fact between up-to-dateness and performance (overhead). Striking a balance between timing and resource constraints can effectively be achieved through the incorporation of adaptive scheduling mechanisms (Böse *et al*., 2006; Yau and Huang, 2004). A common ground for adaptive schemes is to let the providers periodically conduct some sort of a divergence analysis on their local provision of context and trigger update diffusion only when the estimated deviation exceeds a negotiated (or enforced) threshold (e.g., as specified by the node itself or through feedback from consumers). Resource-aware policies can be incorporated into the calculation of divergence and the threshold values can be tuned adaptively to suit varying network conditions or in line with changes in nodes' behavior.

Finally, given the heterogeneous nature of a MANET setting, an open issue would be the design of contextual data formats and appropriate network algorithms to enable interoperability by supporting various types of sources as well as finding the right balance of developing a universal context model and intelligent infrastructures for future context-aware services.

CONCLUSIONS

In mobile ad-hoc networks, context dissemination involves managing personalized access to a dynamic collection of distributed and heterogeneous contextual information. In this study, context dissemination has been investigated through the prism of context-aware middleware abstractions. A common ground is to facilitate the sharing of and interaction with context data by providing a logical view of the available resources while at the same time unifying context-sensitive interactions. The study conducted a thorough investigation of the latest trends in middleware efforts towards providing for such abstractions and highlighted their key properties with reference to context dissemination specific requirements. The researchers believe that the design and implementation of middleware abstractions that fully satisfy all the implications and challenges imposed by context-aware ad-hoc networking entail trade-offs among usability, flexibility, efficiency and scalability.

A scrupulous observation of the prior art in this area suggests that a comprehensive taxonomy of the existing strategies would best fit in a trilogy of abstractions, architectures and protocols; hence, a part of our future work would be to undertake investigations covering the other two.

ACKNOWLEDGMENT

The research described in this study has been supported in part by the Iran Telecommunications Research Center (ITRC) under Grant No. 86-2670.

REFERENCES

Balzarotti, D., P. Costa and G.P. Picco, 2007. The LighTS tuple space framework and its customization for context-aware applications. J. Web Intell. Agent Syst., 5: 215-231.

Bravetti, M., N. Busi, R. Gorrieri, R. Lucchi and G. Zavattaro, 2005. Security issues in the tuple-space coordination model. Int. Federat. Inform. Process., 173: 1-12.

Buchholz, T., A. Kupper and M. Schiffers, 2003. Quality of context: What it is and why we need it. Proceedings of the 10th Workshop of the HP OpenView University Association, July 6-9 2003, University of Geneva, Switzerland, pp: 1-14.

Busi, N. and G. Zavattaro, 2001. Some thoughts on transiently shared dataspaces. Proceedings of the Workshop on Software Engineering and Mobility, Co-Located with the 23rd International Conference on Software Engineering, (ICSE'01), USA., pp: 1-5.

Böse, J.S., K. Hahn, M. Scholz, H. Schweppe and A. Voisard, 2006. Using moving object databases to provide context information in mobile ad-hoc networks. Proceedings of the 7th International Conference on Mobile Data Management, May 10-12, IEEE Computer Society, Washington, DC., USA., pp: 75-82.

Cugola, G. and G.P. Picco, 2001. Peerware: Core middleware support for peer-to-peer and mobile systems. Technical Report, Politecnico di Milano. http://citeseer.ist.psu.edu/479980.html.

Dobson, S., S. Denazis, A. Fernandez, D. Gaiti and E. Gelenbe *et al*., 2006. A survey of autonomic communications. ACM Trans. Auton. Adapt. Syst., 1: 223-259.

Friedman, R., D. Gavidia, L. Rodrigues, A.C. Viana and S. Voulgaris, 2007. Gossiping on MANETs: The beauty and the beast. ACM SIGOPS Operat. Syst. Rev., 41: 67-74.

Gelernter, D., 1985. Generative communication in linda. ACM Trans. Programm. Languages Syst., 7: 80-112.

Hackmann, G., C. Julien J. Payton and G.C. Roman, 2005. Supporting generalized context interactions. Lecture Notes Comput. Sci., 3437: 91-106.

Handorean, R. and G.C. Roman, 2003. Secure sharing of tuple spaces in ad hoc settings. Elect. Notes Theoret. Comput. Sci., 85: 1-20.

Huang, Q., 2002. Supporting context-aware computing in ad hoc mobile environments. Technical Report WUCS-02-36, Washington University, Department of Computer Science.

Huang, Q., C. Julien and G.C. Roman, 2004. Relying on safe distance to achieve strong partitionable group membership in ad hoc networks. IEEE Trans. Mobile Comput., 3: 192-205.

Julien, C. and G.C. Roman, 2006. EgoSpaces: Facilitating rapid development of context-aware mobile applications. IEEE Trans. Software Eng., 32: 281-298.

Julien, C. and D. Stovall, 2006. Enabling ubiquitous coordination using application sessions. Lecture Notes Comput. Sci., 4038: 130-144.

Julien, C., G.C. Roman and Q. Huang, 2008. SICC: Source-initiated context construction in mobile ad hoc networks. IEEE Trans. Mobile Comput., 7: 401-415.

Kang, P., C. Borcea, G. Xu, A. Saxena, U. Kremer and L. Iftode, 2004. Smart messages: A distributed computing platform for networks of embedded systems. The Comput. J., 47: 475-494.

Karp, B. and H.T. Kung, 2000. GPSR: Greedy perimeter stateless routing for wireless networks. Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (Mobicom), August 6-11, Boston, Massachusetts, USA., pp: 243-254.

Kirda, E., H. Gall, P. Fenkam and G. Reif, 2002. MOTION: A peer-to-peer platform for mobile teamwork support. Proceedings of the 26th Annual International Conference on Computer Software and Applications, (COMPSAC'02), Vienna, Austria, pp: 1115-1117.

Levis, P., D. Gay and D. Culler, 2005. Active sensor networks. Proceedings of the 2nd Conference on Symposium on Networked Systems Design and Implementation, May 2-4, USENIX Association, Berkeley, CA., USA., pp: 343-356.

Mamei, M., F. Zambonelli and L. Leonardi, 2003. Tuples on the air: A Middleware for context-aware computing in dynamic networks. Proceedings of the 23rd International Conference on Distributed Computing Systems, May 19-22, USA., pp: 342-347.

Mamei, M. and F. Zambonelli, 2005. Location-based and content-based information access in mobile peer-to-peer computing: The TOTA approach. Lecture Notes Comput. Sci., 2872: 162-173.

Mamei, M. and F. Zambonelli, 2006a. Field-Based Coordination for Pervasive Multi-agent Systems. 1st Edn., Springer, Germany.

Mamei, M. and F. Zambonelli, 2006b. Theory and practice of field-based motion coordination in multiagent systems. J. Applied AI., 20: 305-326.

Murphy, A.L. and G.P. Picco, 2004. Using coordination middleware for location-aware computing: A LIME case study. Lecture Notes Comput. Sci., 2949: 263-278.

Murphy, A. and G.P. Picco, 2006. Using lime to support replication for availability in mobile ad hoc networks. Lecture Notes Comput. Sci., 4038: 194-211.

Murphy, A., G.P. Picco and G.C. Roman, 2006. LIME: A coordination model and middleware supporting mobility of hosts and agents. ACM Trans. Software Eng. Methodol., 15: 279-328.

Payton, J., 2006. A query-centered approach to supporting the development of context-aware applications for mobile ad hoc networks. Ph.D Dissertation, Department of Computer Science and Engineering, Washington University in Saint Louis, MO, USA.

Ratnasamy, S., B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan and S. Shenker, 2002. GHT: A geographic hash table for data-centric storage. Proceedings of the International Workshop on Wireless Sensor Networks and Applications, September 28-28, Atlanta, Georgia, USA., pp: 78-87.

Remagnino, P. and G.L. Foresti, 2005. Ambient intelligence: A new multidisciplinary paradigm. IEEE Trans. Syst. Man Cybernet., 35: 1-6.

Riva, O., 2006. Contory: A middleware for the provisioning of context information on smart phones. Lecture Notes Comput. Sci., 4290: 219-239.

Roman, G.C., C. Julien and J. Payton, 2007. Modeling adaptive behaviors in Context UNITY. Theoret. Comput. Sci., 376: 185-204.

Schelfthout, K. and T. Holvoet, 2004. ObjectPlaces: An environment for situated multi-agent systems. Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multi-Agent Systems, July 19-23, New York, USA., pp: 1500-1501.

Schelfthout, K., T. Holvoet and Y. Berbers, 2005. Views: Customizable abstractions for contextaware applications in MANETs. Proceedings of the 2005 Workshop on Software Engineering for Large-Scale Multi-Agent Systems, May 15-16, St. Louis, Missouri, pp: 1-8.

Stovall, D. and C. Julien, 2007. Resource discovery with evolving tuples. Proceedings of the International Workshop on Engineering of Software Services for Pervasive Environments, September 04-04, Dubrovnik, Croatia, pp: 1-10.

Sudha, R., M.R. Rajagopalan, M. Selvanayaki and S.T. Selvi, 2007. Ubiquitous semantic space: A context-aware and coordination middleware for ubiquitous computing. Proceedings of the 2nd International Conference on Communication Systems Software and Middleware, Jan. 7-12, Bangalore, pp: 1-7.

Tennenhouse, D.L. and D.J. Wetherall, 2007. Towards an active network architecture. ACM SIGCOMM Comput. Commun. Rev., 37: 81-94.

Tim, B.L., J. Hendler and O. Lassila, 2001. The semantic web. Scienti?c Am., 284: 34-43.

Yau, S.S. and D.D. Huang, 2004. An adaptive, lightweight and energy-efficient context discovery protocol. Proceedings of the 10th IEEE International Workshop on Future Trends of Distributed Computing Systems, May 26-28, Suzhou, China, pp: 261-267.