x

A Survey of Context Data Distribution for Mobile Ubiquitous Systems

PAOLO BELLAVISTA, ANTONIO CORRADI, MARIO FANELLI, AND LUCA FOSCHINI
University of Bologna, Bologna, Italy

The capacity to gather and timely deliver to the service level any relevant information that can characterize service-provisioning environment, such as computing resources/capabilities, physical device location, user preferences, and time constraints, usually defined as context-awareness, is widely recognized as a core function for the development of modern ubiquitous and mobile systems. Much work has been done to enable context-awareness and to ease the diffusion of context-aware services; at the same time, several middleware solutions have been designed to transparently implement context management and provisioning in the mobile system. However, to the best of our knowledge, an in-depth analysis of the context data distribution, namely the function in charge of distributing context data to interested entities, is still missing. Starting from the core assumption that only effective and efficient context data distribution can pave the way to the deployment of truly context-aware services, this paper aims at putting together current research efforts to derive an original and holistic view of the existing literature. We present a unified architectural model and a new taxonomy for context data distribution, by considering and comparing a large number of solutions. Finally, based on our analysis, we draw some of the research challenges still unsolved and we identify some possible directions of future work.

Categories and Subject Descriptors: A.1 [**General Literature**]: Introductory and Survey; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*; D.4.4 [**Operating Systems**]: Communications Management—*Network communication, Message sending*

General Terms: Design, Management

Additional Key Words and Phrases: ubiquitous system, context, context-aware, context-awareness, context access, context data distribution, context data distribution infrastructure, quality of context, QoC, survey, research challenge

ACM File Format:
BELLAVISTA, P., CORRADI, A., FANELLI, M., AND FOSCHINI, L. xxxx. A Survey of Context Data Distribution for Mobile Ubiquitous Systems. *ACM Computing Surveys*, xx, xx, Article xx (xxx xxx), xx pages. DOI = xxx

1. INTRODUCTION

The popularity of wireless devices and the increasing availability of heterogeneous wireless infrastructures, spanning from IEEE 802.11 (Wi-Fi sometimes shortened as WiFi) and Bluetooth to cellular 3G and beyond, are stimulating new service provisioning scenarios. A growing number of users require any-time and any-where access to their Internet services, such as email, printing, Voice over IP, social computing, and many others more, while moving across different wireless infrastructures. In the so-called

Authors' addresses: Dipartimento di Elettronica, Informatica e Sistemistica (DEIS), University of Bologna, Italy; email: {paolo.bellavista, antonio.corradi, mario.fanelli, luca.foschini}@unibo.it.

Internet of Things vision, mobile users will be able to dynamically discover and impromptu interact with heterogeneous computing and physical resources encountered during their roam [Gershenfeld *et al.* 2004].

To fully enable the great potential of all above service provisioning scenarios, *context-awareness*, broadly defined as the ability to provide services with full awareness of current execution environment, is widely recognized as one of the cornerstones to build modern mobile and ubiquitous systems [Bolchini *et al.* 2009; Dey and Abowd 2000a; Jones and Grandhi 2005; Schilit *et al.* 1994]. Several research efforts have been devoted to middleware solutions aimed to transparently implement the main context management phases, such as production, processing, storage, and distribution to mobile nodes, to foster the diffusion of context-aware services. *Context data distribution*, namely the capability to gather and to deliver *relevant context data* about the environment *to all interested entities* connected to the mobile ubiquitous system, is emerging as a new research area in context-aware systems [Baldauf *et al.* 2007]. In fact, context data distribution is extremely significant from both the service and the middleware perspectives. On the one hand, service adaptation is triggered by received context data: hence, context data have to be timely delivered to let services promptly adapt to the current execution context. On the other hand, the middleware has to transparently manage and route huge amounts of context data, while ensuring timely delivery to mobile nodes: especially in wide-area mobile networks, that can lead to non-negligible overhead, thus hindering both system scalability and reliability. Finally, the relevance of the context data distribution is also proved by the evidence at several context-aware solutions, and some seminal survey activities include context data distribution among core support functions [Baldauf *et al.* 2007; Chen and Kotz 2000; Gaddah and Kunz 2003; Hightower and Boriello 2001; Kjær 2007; van Sinderen *et al.* 2006].

Differently from surveys already existing in literature, our work focuses on context data distribution to deeply study main requirements, implementation primitives, and identify research challenges. Following an order of increasing similarity with our work, in [Gaddah and Kunz 2003], authors highlight four different types of middleware paradigms for mobile computing, namely reflective, tuple space, context-aware, and event-based ones; however, just few context-aware middleware implementations are considered, and context data distribution is not presented. Instead, [Chen and Kotz 2000] surveys several context-aware applications to detail which context information are usually necessary and how they can be sensed. Similarly, [Hightower and Boriello 2001] focuses on location-aware applications, and details location sensing techniques with associated benefits/shortcomings. As regards surveys more focused on the middleware layer, and closer to our work, in [Baldauf *et al.* 2007], the authors detail the main requirements and functionalities of general context-aware middlewares: however, neither they focus on a particular middleware function nor they present a taxonomy for the context data distribution. Finally, [Kjær 2007] is specifically at the middleware-level, and introduces a taxonomy to classify existing solutions: however, as regards the context data distribution, the proposed taxonomy is rather limited and high-level oriented, so it does not consider the impact that different network deployment overlays and context data dissemination strategies can have on context provisioning.

Consequently, this survey aims to fill the gap that, in our opinion, is currently hindering the realization of context-aware services in wide-area wireless networks, i.e., the context data distribution. In fact, a clear formulation of context data distribution requirements and its core components is still missing due to several reasons. First, context data distribution should be least intrusive as possible, thus requiring an integrated

management of mobile ubiquitous systems. That poses several technical challenges due to high heterogeneity, severe communication and computation constraints, and high variability of mobile ubiquitous deployment environments (mobility, volatility of the wireless medium, …). Hence, even if a multitude of partial ad-hoc solutions have been developed for specific service/deployment scenarios, it is difficult to identify common design guidelines. Second, past context-aware research has mainly focused on small-scale deployments, typically limited to homes and buildings, where context data distribution has affordable run-time overhead. Consequently, previous works mainly addressed local middleware functionalities to support context provisioning to the service layer, while using rather simple and centralized approaches to implement the distribution process. Finally, context data distribution crosscuts different protocol layers (from network to application layer), and covers various emerging research fields, including traditional (infrastructured) mobile ubiquitous systems, Mobile Ad-hoc NETworks (MANET), Vehicular Ad-hoc NETworks (VANET), and Delay Tolerant Networks (DTN) [Conti and Giordano 2007a; Conti and Giordano 2007b; Fall 2003]. The dispersion of research efforts in these different areas also complicates the definition of a clear model. In addition, different network and middleware deployments, e.g., based on fixed wireless infrastructures or on ad-hoc communications, centralized or decentralized, etc., have a great impact on context provisioning: hence, additional research is required to understand how they influence and limit context data distribution.

Our survey addresses the above mentioned issues by proposing four perspectives: i) a unified architectural model for context data distribution, ii) a new taxonomy to settle terminology and concepts useful to compare existing solutions in literature, iii) a thorough comparison of a large number of supports and infrastructures for context data distribution, and iv) a discussion about open issues and future research trends in the field.

The paper is organized in sections as follows. Section 2 defines context and context data distribution. Section 3 provides an architectural model for context data distribution. In Section 4, we present our taxonomy, and we compare existing solutions against it. Section 5 and 6 detail respectively the comparisons among surveyed solutions and future research directions. Finally, Section 7 presents some concluding remarks.

2. CONTEXT AND CONTEXT DATA DISTRIBUTION

Context-awareness has now a very wide meaning and it can be considered even a contradictory word that may express several and different senses according to the specific scenario and author. Since there is no agreed definition, next sub-sections settle required terminology and definitions. The aim of these sub-sections is also to motivate and to better point out the scope of this work.

2.1. Context & QoC Definition

Context is still a vague concept to identify the aspects the designer considers useful to model and describe the environment where a given service is to be deployed and executed. Many different authors presented their own context definition: in [Schilit *et al.* 1994], service context contains "where you are, who you are with, and what resources are nearby"; in [Dey and Abowd 2000a], it contains "any information that can be used to characterize the situation of an entity"; finally, in [Zimmermann *et al.* 2007], authors say that "elements for the description of this context information fall into five categories: individually, activity, location, time, and relations". In a common sense meaning, context is the "set of variables that may be of interest for an agent and that influence its actions" [Bolchini *et al.* 2009].

For the sake of clarity, in the following we adopt the context definition presented in

[Chen and Kotz 2000] because it is able to cover the main context aspects with a rather straightforward classification; in addition, this definition has many important similarities with the one presented in [Schilit *et al.* 1994], hence it is overall well-accepted by the community. According to [Chen and Kotz 2000], context is a four-dimensional space composed by: *computing context*, *physical context*, *time context*, and *user context*.

Computing context deals with all those technical aspects related to computing capabilities and resources. This category has a two-fold aim. First, it expresses all those heterogeneities that are usually present in mobile environments, like different device capabilities and connectivity [Bartolini *et al.* 2009; Ceri *et al.* 2007]. Second, it also takes into account the different resources that a mobile device encounters while roaming [Schilit *et al.* 1994]. Many existing systems already exploit these attributes to trigger management functions and to adapt services. For instance, Google and Facebook dynamically adapt to the current characteristics of mobile devices, Web clients, and connectivity (such as the available bandwidth).

The *physical context* groups all those aspects that represent real world and that are accessible by using sensors/resources deployed in the node surroundings. Device/user location is a notable basic example of physical context; other aspects include traffic condition, people speed, noise level, temperature, and lighting data [Kim *et al.* 2006]. Physical models and laws, e.g., mechanics laws to help predicting future physical states of the system, are also part of the physical context. Due to its nature, physical context is intrinsically very prone to measurement errors (due to several sources of incorrectness and imprecision and the stochastic nature of physical processes, …). Many solutions already use this kind of context to perform environmental monitoring: for instance, some driver-assistant systems use sensors deployed on vehicles to perform traffic jam monitoring and to redirect vehicles to alternatives routes.

Time context captures the time dimension, such as time of a day, week, month, and season of the year, of any activity performed in the system (either real-world or computing). Let us also remark that these context items can be of two main types: *sporadic* and *periodic*. *Sporadic events* model (unexpected) occurrences triggered occasionally, even only once. *Periodic events* describe expected events that present themselves in a repeated and predictable way. In addition, these two main types can also be combined to build complex context events based both on event sequence, number of events in a particular time slice, and so forth [Dey and Abowd 2000a]. For instance, a sporadic activity can automatically reduce video quality during network congestion, while a periodic activity can automatically switch off cell phone ringing tone always at the same time-of-the-day, from 11pm to 7am, to avoid waking up users. In addition, a complex activity, based on the monitoring of network congestion occurrences, can decide to switch between different network interfaces for the sake of better transmission quality.

Finally, *user context* contains high-level context aspects related to the social dimension of users, such as user's profile, people nearby, and current social situation [Adams *et al.* 2008]. In fact, we are considering distributed mobile systems that are the result of the aggregation of multiple end-user devices. Hence, as noticed in [Eugster *et al.* 2009], each node context has both an *individual dimension*, descending from its own egocentric view (such as user profile and preferences) and a *social dimension*, descending from the awareness of being an actor part of a whole system (such as other people around in the proximity and current social situation descriptions). Many systems already use this kind of context to perform automatic recommendation and situation-based adaptation. In particular, some systems use co-localization patterns to infer common interests and to recommend possible friends, while others infer the current situation, for instance, to

switch off the cell phone ringing tone during a business dinner.

Hence, by considering all the aforementioned context dimensions, different context-aware behaviors can be realized to adapt services so to make them satisfactory for final user and to fit current execution environment characteristics. Toward this goal, the quality of the context data is a fundamental issue since it can compromise the correctness of adaptation operations. In fact, on the one side, systems can use physical sensors (such as temperature and pressure sensors) that, due to their nature, introduce errors and approximations associated with their resolution. On the other side, systems can use virtual sensors (such as data retrieved by database), but those supplied data do not ensure total correctness even if usually more polished than physical ones. Consequently, the emerging notion of *Quality of Context (QoC)* – usually defined as the set of parameters that express quality requirements and properties for context data (e.g., precision, freshness, trustworthiness, …) – is overwhelming important to control and manage all the possible context inaccuracies [Buchholz *et al.* 2003; Krause and Hochstatter 2005].

Delving into finer details, several works studied both context quality parameters and their effects on the context data distribution. In [Manzoor *et al.* 2008], authors associate context data with four QoC parameters: i) *up-to-dateness* to deal with data aging; ii) *trustworthiness* to rate the belief we have in context correctness; iii) *completeness* to consider that context data could be partial and so incorrect; and iv) *significance* to express differentiated priorities. Successively, the same authors use these QoC parameters to resolve context conflicts: a conflict resolution policy can be based either on one particular parameter or on a weighted combination of them, and selects the data to be saved, i.e., the data with the highest QoC [Manzoor *et al.* 2009a; Manzoor *et al.* 2009b]. Instead, [Neisse *et al.* 2008] presents a new QoC framework based on three main QoC parameters, namely *up-to-dateness*, *precision*, and *resolution*; authors exploit a standard ISO vocabulary for measurements to define their own framework, and show that their approach is general enough to cover the main QoC parameters. Finally, they report a new dynamic schema to evaluate trustworthiness parameter based on users' feedbacks.

Hence, similarly to context-awareness in itself, a well-accepted QoC definition is still missing. Several authors presented their own QoC framework, also introducing and using the same concepts with different names. However, despite these differences, a common thought can be highlighted: QoC is not requiring perfect context data, such as all data with the highest possible precision and up-to-dateness, but having and maintaining a correct estimation of the data quality [Buchholz *et al.* 2003]. In fact, if the context data distribution is not aware of data quality, possible service reconfigurations could be completely misled by low quality data.

In addition to this traditional notion of QoC, extremely focused on data quality, especially in the last years, much research has recognized the approach of introducing the quality of the context data distribution (e.g., data delivery time, reliability, …) to ensure *the availability of the context data with the right quality, in the right place, and at the right time*. In other words, if Quality of Service (QoS) permits service consumers and service providers to negotiate their requirements at acceptable service levels by considering the network available underneath [Tanenbaum 2002], QoC has to consider the quality of both the exchanged context data and the distribution process to ensure user satisfaction. In fact, context data distribution usually exploits best-effort wireless infrastructures that could introduce delays and droppings, thus leading to additional inaccuracies in the final context received by mobile devices.

To the best of our knowledge, [Buchholz *et al.* 2003] is the first work that presented an in-depth analysis of the quality problem in context-aware settings. The authors

decompose the quality problem along three main directions to consider i) the quality of the physical sensors; ii) the quality of the context data; and iii) the quality of the delivery process. Differently from those authors, we think that it is not always possible to clearly separate these three quality dimensions. For instance, some data quality parameters are dynamic, and their value meaning (and quality) depends on the time elapsed from data generation. Hence, possible delivery delays can affect these data quality parameters, thus violating the assumption that the data quality parameters do not depend on the quality parameters of the distribution process [Buchholz *et al.* 2003].

Consequently, we consider a broader QoC definition, at the same time dealing with both the quality of the context data and of the context data distribution: based on all above definitions, we exemplify some of the parameters mostly concerned in both those aspects of QoC. *Context data validity* specifies the field of validity that any data of a given type must comply with; for instance, a month time context data must conform to the Gregorian calendar format. *Context data precision* evaluates the degree of adherence between real, sensed, and distributed value of a context data; for instance, depending on received subscriptions, the context data distribution support can either deliver more precise ultra-wide-band-based location data or more standard GPS-based information. *Context data up-to-dateness* expresses how the usefulness of particular data changes over time; for instance, the up-to-dateness of location information of a fixed resource (e.g., a GPRS antenna) is higher than the one of a mobile entity (e.g., a user) and context data distribution can use that additional knowledge to suppress several heavy measurements. To summarize, we claim that the above QoC parameters must be taken into account in the QoC agreement specified at the service level and, at the same time, used by the context data distribution to measure and achieve the fulfillment of the QoC requirements.

2.2. Context Data Distribution in Mobile Ubiquitous Environments

Even if context-aware solutions have appeared in different research areas, context-awareness reaches its maximum usefulness when applied to mobile ubiquitous systems. Context-awareness permits mobile services to dynamically and efficiently adapt both to the current situation, such as current physical place and/or social activity, and to the challenging and highly variable deployment conditions typical of mobile environments (resources scarcity, unreliable and intermittent wireless connectivity, …). The central role of context data distribution in mobile computing is evidenced also by the plethora of research efforts proposed in the last years in this area. Therefore, we have decided to focus on mobile ubiquitous systems to provide privileged examples of context-aware systems. At the same time, we believe that the taxonomy and analysis of context data distribution systems proposed here apply also to other context-aware solutions in different research areas (fixed Internet services computing, distributed data base management systems, …).

The realization of real-world context-aware services in mobile ubiquitous environments is a complex task that requires a deep understanding of many technological details and includes several non-trivial operations, spanning different layers and depending on executing platforms. The considerable efforts required to manage all those technological aspects could be blamed for the slowdown of context-aware services deployment in mobile ubiquitous systems [Chen and Kotz 2000]. Consequently, to tackle all these issues and to ease the diffusion of context-aware services, there is the need for proper context-aware middleware solutions aimed at transparently addressing all the main management phases involved in context provisioning to the service layers, i.e., representation, memorization, aggregation, distribution, notifications to running services,

etc. [Baldauf *et al.* 2007; Kjær 2007]. In other words, context-aware services should only have to produce and publish context data and to declare their interests in receiving them from the support middleware, while an internal middleware function takes over distribution responsibility and transparently executes specific management operations to distribute context data.

More formally, we define *context data distribution* the (distributed) middleware function that makes possible *the injection of context data in the system and their automatic delivery to all those entities that have expressed any form of interest* in those context data. At the same time, we distinguish two main types of context data distribution depending on how they manage data distribution. We call *(uninformed) context data distribution* the first type of implementations that simply route context data according to context needs expressed by mobile nodes: like traditional pub/sub systems, those systems blindly route data without inspecting their content. The second type, instead, groups other implementation approaches that take advantage of exchanged context data to dynamically adapt and self-manage the distribution process itself. We call these proposals *informed context data distribution* due to their increased context-awareness, and while they are more recent and less widespread, they represent already a fundamental new and emerging area of current research. In particular, they especially suit to those deployment scenarios that do not assume a fixed wireless infrastructure to rely upon, such as MANET, VANET, and DTN: in this case, context-awareness is crucial to improve the effectiveness and the efficiency of context data distribution.

2.3. Context Data Distribution: Main Requirements

In the last decade, much research has been done in the design, the realization, and the deployment of context-aware middleware solutions. Previous research mainly focused on rather small-scale deployments, such as smart homes and smart university campus, with the main goal of studying the (local) middleware infrastructure useful to support context provisioning to service level. However, in the very last years, an increasing number of systems are requiring context provisioning for wide-area wireless deployments up to the Internet scale. Of course, the context data distribution becomes a first concern that deserves more attention due to the system dimension; in particular, additional research is required to explicitly deal with the problem of effective and efficient context data distribution with respect to negotiated QoC. To effectively support context-aware services in wide-area wireless networks, we claim that the context data distribution has to fulfill several main requirements: *context data production/consumption decoupling*, *adaptation to mobile and heterogeneous environments*, *context data visibility scopes enforcement*, *QoC-based context data distribution*, and *context data lifecycle management.*

First, the context data distribution has to transparently route produced context data to all the interested sinks connected to the mobile systems. To foster system scalability and context availability, context data production and consumption should be possible at different times (time decoupling), and sinks and sources do not have to know each other (space decoupling); in other words *communication should be asynchronous and anonymous among context producers and consumers*, the same as in traditional pub/sub systems.

Second, the context data distribution has to support mobile heterogeneous wireless scenarios. Mobile nodes requiring context-aware services move in and out, sometimes randomly, by also introducing sudden variations in context needs; hence, *the context data distribution has to promptly adapt to mobility*, so to distribute only currently required

context data. At the same time, this function has to comply with heterogeneous systems including nodes with different computational capabilities, wireless standards, and wireless modalities; hence, *the adaptation to currently available resources* is fundamental to avoid system saturation.

Third, the context data distribution has to introduce, preserve, and enforce differentiated visibility scopes for context data. In fact, context data have typically a limited visibility scope that depends on physical/logical locality principles. For instance, physical context of a place is likely to be visible only to the nodes in the same place (physical locality); similarly, user context data associated with a particular event should be visible only to its participants (logical locality). In other words, *context data intrinsically have visibility scopes that the context data distribution must enforce to avoid useless management overhead*.

Fourth, the context data distribution has to enforce QoC-based constraints to enable correct system management. QoC constraints on context data specify the quality of received data; in addition, considering that real-world wireless systems have to deal with frequent topology changes, limited delivery guarantees, and temporary disconnections, QoC constraints on context data distribution allow enforcing data delivery with particular timeliness and reliability guarantees. In addition, the context data distribution could be deployed in distributed architectures in which several servers, each one with its own local context repository, process and route context data; without proper coordination protocols, context data could be present in multiple and conflicting copies into the system. Therefore, as context data consistency can become costly to handle, it is advisable to *avoid strong consistency semantics by preferring best-effort approaches driven by QoC constraints*.

Finally, the context data distribution has to handle data life cycle, starting from data construction to data destruction [Chang *et al.* 2007]. At the same time, it has to implement context aggregation and filtering techniques required to reduce the final management overhead. In fact, aggregation techniques are useful to reason about context data, so to obtain more high-level and concise information. In addition, filtering techniques are necessary to shape context data distribution, so to reduce the management overhead depending on service needs. Both those techniques must be supported in a distributed manner so, for instance, to filter the distribution of a data as close as possible to the node that had generated it. Moreover, the context data distribution should offer some degrees of *availability* (with a certain probability), namely influencing the degree of distribution and replication of the data into the system. Hence, *the context data distribution should be able to self-control the distribution process*.

3. CONTEXT DATA DISTRIBUTION SYSTEM

This section details the context data distribution from both a local and a distributed perspective. First, we present a unified architecture model for the context data distribution with the main goal of clarifying its macro-components and their interactions. Second, as the adopted network deployment deeply affects and influences the implementation of the context data distribution, we give some background about network deployment useful for the discussions of Section 4. Finally, for the sake of clarity, we compare the context data distribution with some other related models well spread in literature (such as the publish/subscribe model) to better understand how these models differ in requirements, facilities, and supported services.

3.1. Unified Architectural Model and Facilities

Of course, both the heterogeneity and the complexity of the requirements enumerated in Section 2.3 claim for complex context data distribution solutions that transparently distribute context information to all the interested entities, while monitoring available resources and ensuring QoC constraints. Since the wider the system scale, the higher the overhead introduced by context distribution, novel decentralized solutions are required to implement the context distribution function into the mobile ubiquitous system.

First of all, we envision context data distribution systems as data-centric architectures that encompass three principal actors: *context data source*, *context data sink*, and *context data distribution function* (see Figure 1). *Context source* masks back-end sensors access operations and enables context data publication. *Context sink* permits the service level to express its context data needs by using either context data *queries* (pull-based interaction) or *subscriptions* (push-based interaction); *context data matching* is the satisfaction of sink requests, both query and subscription, to achieve a correct fulfillment of both types. Finally, the *context data distribution function* distributes context data by mediating the interaction between context data sources and sinks; for instance, it automatically notifies subscribed context sinks upon context data matching. For the sake of brevity, in the following, we use context subscriptions to indicate both pull-based and push-based approaches; in fact, pull-based interactions can be easily mimicked by using short-lived push-based ones.

With a closer view to the organization, the only main phase executed directly by the service level is *context data sensing* that involves the access to either physical or virtual sensors. While context data sensing is out-of-the-scope of this paper because we consider it part of context data generation process, here we focus mainly on the internal context data distribution function at the middleware level. Given its central role, the efficiency of this function is overwhelming important to ensure system scalability and reliability; at the same time, the context data distribution function has to deal with several key requirements. Directly stemming from the five main context data distribution requirements detailed in the previous section and considering related issues together, Figure 1 details the internal architecture of the context data distribution function. It contains three main facilities organized in two horizontal layers – *Context Data Management* and *Context Data Delivery*, starting from the uppermost to the lowest one – and one cross-layer vertical facility – *Run-time Adaptation Support*.

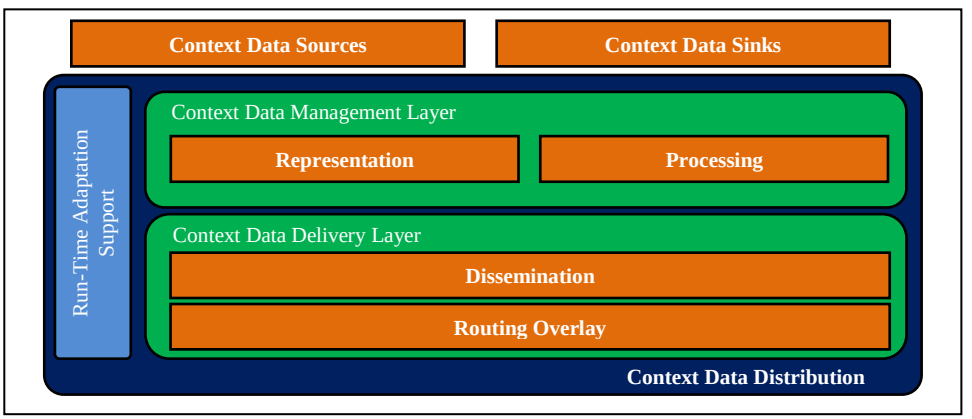

Fig. 1. Context data distribution system logical architecture

The *Context Data Management Layer* takes care of the local context data handling, by defining context data representation and by expressing processing needs and operations. Context data representation includes all different models and techniques, spanning from simple and flat name-value pairs to ontology, proposed to represent raw context data at the middleware level [Bolchini *et al.* 2007; Strang and Popien 2004]. Context data processing includes both i) the production of new knowledge from pre-existing context data by using aggregation techniques (such as simple data matching, first-order logic aggregation, semantic-based techniques, …); and ii) simple filtering techniques to adapt context data distribution to currently available resources, so to foster system scalability [Baldauf *et al.* 2007]. Other important context data processing component aspects are memorization and organization of past context data history and context data security. Let us remark that local context-aware services interact directly with this layer through their own sinks that takes proper management decisions according to expressed context needs. Service context needs are usually expressed as context data filters that can include data QoC constraints apart from other constraints. QoC constraints, for instance based on data precision, are i) locally used to filter the context data supplied to the final services; and ii) remotely used to avoid the distribution of out-of-QoC data that will not be used by requesting node.

The *Context Data Delivery Layer* realizes both methods and algorithms to properly deliver the context data inside the system. It implements all the required coordination and dissemination protocols (flooding, selection, gossiping, …) to carry the published context data to the interested context-aware services; several solutions are possible with a deep impact on the final system scalability and context data availability. At the same time, this layer organizes the nodes that take part to the context data distribution, called brokers in the remainder, to build a particular overlay structure useful to drive both context data and subscriptions routing at run-time. Finally, this layer has to exploit QoC constraints on the data distribution process to tailor context data delivery. Of course, let us remark that the specific context delivery solution must map onto the integrated wireless communication platform available underneath, and this can limit the feasible solutions.

Finally, the *Run-Time Adaptation Support* enables the dynamic management and tailoring of the other layers according to the current run-time conditions, e.g., deployment environment, monitored resource conditions, and QoC requirements with a typical cross-layer perspective. Let us remark that the run-time adaptation support uses QoC constraints, both on the context data and on the distribution process, to assess the feasibility of possible run-time reconfigurations: for instance, a conflict could arise if we would impose tight filter operators to reduce the number of exchanged data, and these filters would violate required QoC constraints. In addition, we assume this facility as cross-layer to better stress that it crosscuts several different aspects and may require to execute coordinated operations at different distribution support layers. As discussed in the next sections, the adaptation can deeply influence the performance of the context data distribution; inappropriate decisions and/or reconfigurations could lead to both system and QoC degradation, thus inducing unwanted and noisy side-effects in context-aware services provisioning.

Once understood the logical architecture of the context data distribution, let us stress that the above three facilities have to carefully collaborate to ensure the main requirements presented in Section 2.3. First, to ensure context production/consumption decoupling, the context data management layer has to store context data, so to make them

available for subsequent usage (time decoupling), while the context data delivery layer has to exploit sinks subscriptions to transparently route the context data inside the system (space decoupling). Second, to achieve adaptation to mobile situations, the context data delivery layer has to reconfigure to suit according to current local services and neighbors requests; in addition, to address heterogeneous environments, the run-time adaptation support has to monitor current available resources (CPU, memory, available bandwidth, …), and to proper command both the context data management and delivery layers to tailor resource usage. Third, to handle context data visibility scopes, the context data delivery layer has to tailor the context data routing into the system, in particular, by carefully avoiding the uncontrolled propagation of context data and subscriptions; in other words, the routing of both context data and subscriptions has to suit to physical/logical locality principles. Fourth, to implement QoC-based context distribution, all the above facilities have to coordinate and reconfigure themselves according to the agreed QoC constraints: QoC constraints on data require proper context data filters useful to shape the context data supplied to services; QoC constraints on the context data distribution process have to be supported by the context data delivery layer that has to somehow affect the context data routing both at the local node and in the whole distributed architecture; at the same time, the run-time adaptation support needs to consider current available resources and QoC policies to i) limit and drive possible facilities reconfigurations; and ii) warn either the system or the user whether current resources do not enable respecting current QoC requests. Finally, to manage context data lifecycle, the context data management layer has to consider context data generation time and lifetime to trigger possible context data elimination, and, at the same time, to supply context data processing techniques, i.e., by aggregation and filtering operators. Let us remark that these techniques may depend conjunctly on several different context data, perhaps with different QoC constraints; hence, triggered by service requests, both the context data management and delivery layers have to collaborate to retrieve required context data and make them available with the right QoC.

3.2. Network Deployments

By considering that the network deployment affects the implementation of the context data distribution in real-world systems, this section discusses existing and emerging network deployment scenarios. Of course, the adopted network deployment can lead to different degrees of connectivity among nodes, and it could implicitly either favor or hinder the coordination and the communication among context data sources and sinks. For the sake of clarity, this section presents the main categories of possible network deployments; Section 5 compares surveyed solutions to highlight the main effects of the network deployment on the context data distribution.

First of all, we consider three principal broad categories of network deployment: i) fixed, that simply extends the traditional (wired) Internet with wireless Access Points (APs); ii) ad-hoc, where mobile entities communicate directly (without infrastructure); and iii) hybrid, that combines the two previous approaches. In *fixed infrastructure*, the context data distribution uses some service reachable through the wireless infrastructure. The usage of a fixed infrastructure ensures high context availability, but also imposes strong constraints on provisioning environments, since the system is unable to work in infrastructure-less scenarios, and nodes that do not host the wireless technology adopted by the infrastructure cannot join the system. In *ad-hoc infrastructure*, the context data distribution must be implemented in a decentralized way, while ad-hoc links support transmissions among the different mobile nodes. These approaches well fit all those

deployment scenarios that do not present fixed infrastructures, such as battlefield and emergency response scenarios, but worsen data availability and related management issues. Finally, *mixed infrastructure* approaches strive to obtain the best from previous ones, with fixed infrastructures that ensure high data availability for those nodes able to communicate through it, and ad-hoc communications that may reduce the infrastructure overhead and permit to reach nodes unreachable otherwise. For instance, a multi-homed laptop that has both a WiFi and a Bluetooth card can act as a router for Bluetooth-only cellular phones toward WiFi-based infrastructures.

In addition, we intend to introduce three emerging ad-hoc-based network models, MANET, VANET, and DTN, as typical network deployment use cases. We focus on these three ad-hoc deployments because interesting informed context data distribution solutions have been proposed for those specific deployment scenarios. In particular, we identify the following main areas of widespread and significant applicability: data replication support for MANET, information dissemination and coordinated driving support for VANET, and context-aware routing in DTN [Derhab and Badache 2009; Pelusi *et al.* 2006; Senart *et al.* 2009; Sichitiu and Kihl 2008; Zhang 2006].

With a finer degree of details, a MANET is a collection of mobile nodes that use wireless ad-hoc links to communicate without using existing wireless network infrastructures [Conti and Giordano 2007a; Conti and Giordano 2007b] and where nodes are free to move randomly, thus creating network partitions and causing disconnections. In these general scenarios, different data replication solutions have been proposed to ensure data availability despite of network partitions and nodes departure. A VANET is a specialized MANET where mobile nodes are vehicles [Conti and Giordano 2007b]. Those scenarios present high mobility, and exploit communications both between nearby vehicles and between vehicles and available wireless infrastructures. Different context data distribution scenarios have been proposed both for i) environmental monitoring; and ii) coordinated driving [Caveney 2010]. In the first case, each vehicle enacts as sensor by publishing data concerning the surroundings, like air pollution and plate numbers of near vehicles. In the second case, vehicles coordinate to enable driver-assistant services, such as adaptive traffic lights, car accident prevention, and traffic scheduling. While MANET/VANETs mimic fixed infrastructure by assuming that the path between the source and the destination exists when a message has to be routed, DTNs accept longer latency and do not assume that the whole source-destination routing path always exists at the same time. The message is forwarded on a hop-by-hop basis and by following a store-carry-and-forward paradigm, in which each node tries to select the current best forwarder toward the destination, i.e., the node that has the highest probability to bring the message close to the destination [Fall 2003].

The above scenarios offer very good cases for informed context data distribution. Data replication solutions for MANET exploit context data, usually physical context data such as received signal strength or relative location measurements, to evaluate the number and the location of data replicas to spread within the system [Derhab and Badache 2009]. VANETs usually exploit computing and physical context to adapt data production, spreading, and harvesting [Lochert *et al.* 2010]. Finally, DTNs exploit context information, especially time and user context, to select hop-by-hop best forwarders [Jain *et al.* 2004].

3.3. Understanding the Context Data Distribution

In the past years, much research has addressed the general problem of data distribution in mobile heterogeneous environments and several solutions have been

designed and implemented with the main goal of counteracting scarce system resources and unstable network connectivity. Hence, after the presentation of the main requirements, facilities, and network deployments of the context data distribution, this section introduces three main emerging technical areas very close to context data distribution, namely, mobile databases in MANET, multicast and group communication protocols in MANET, pub/sub in mobile environments, and explains why we consider their requirements different from context data distribution ones. Of course, the proposed comparison does not have any pretence of being exhaustive, but we believe it can help understanding the original aspects of this new research area.

Starting with brief research area descriptions, mobile database solutions enhance data availability over MANET settings by overcoming possible node disconnections and network partitions. Existing solutions copy data at different mobile nodes by using either replication or caching techniques [Derhab and Badache 2009; Padmanabhan *et al.* 2008]: replication techniques proactively copy all local data to remote nodes, and keep them until explicitly deleted [Hara 2001; Shaheen and Gruenwald 2010]; instead, caching solutions reactively maintain data in response to queries, and usually keep them until deletion by replacement operations mainly due to memory saturation [Chow *et al.* 2007; Yin and Cao 2006]. Multicast and group communication protocols in MANET well fit the context data delivery facility. These techniques allow to create different groups and to distribute data to all the interested entities that have previously joined a group; it comes without saying that this model is suitable for distributing context data produced by a context data source to a group of context data sinks. Finally, context data distribution model may seem close to a pure pub/sub model because it is based on sources, sinks, and data distribution function [Eugster *et al.* 2003]. Many different solutions for pub/sub in mobile environments have been already proposed in literature, and we anticipate that a certain number of the systems analyzed in this survey adopt pub/sub implementations to perform context distribution themselves.

Even if these areas are close to context data distribution, some important differences outstand. By analyzing the five main requirements highlighted in Section 2.3 (following the same presentation order), this section aims at explaining better the original need for and the identity of context data distribution infrastructures for mobile ubiquitous systems.

The first two requirements relate to mobile systems in general and hence are common to context data distribution and to the above three research areas as well. In fact, mobile systems in which nodes freely join and leave the system make strong coupling between communication entities absolutely unsuitable. Consequently, context data production/consumption decoupling is intrinsic due to the mobile nature of the system and several solutions belonging to close research fields, such as pub/sub systems, can ensure this requirement [Eugster *et al.* 2003]. At the same time, also the capability of adapting to mobility and heterogeneity mainly derives from dealing with mobile systems in general, because these systems group several mobile devices, spanning from cell phones and PDAs to full-fledged laptops, with extremely different resources. Adaptation to heterogeneity is essential and several solutions in the above three areas already support it.

Hence, if the first two requirements are mainly connected with mobile systems in general, and do not allow to clearly differentiate context data distribution from other approaches, the remaining three requirements carefully suggest that context data distribution, despite some similarities, cannot be fully addressed by other approaches.

In fact, starting with the enforcement of the context data visibility scopes, mobile database approaches do not usually enforce locality principles, and try to spread data in

the whole system to increase availability; this is against the locality principles of the context data distribution. Similarly, both multicast and group communication protocols in MANET and mobile pub/sub architectures strive to build system-wide communication primitives that do not usually enable the enforcement of context data visibility scopes. Of course, differentiated visibility scopes could be mimicked depending on the specific system; however, these solutions are system-dependent and can lead to increased management overhead. At the same time, it is worth stressing that some pub/sub systems, usually called location-aware in literature, can also constrain the message/subscription match depending on the current location so to enforce limited visibility scopes associated with physical locality principles.

In consideration of QoC-based context data distribution, several problems can arise because mobile databases do not usually consider quality constraints, neither on the data nor on the distribution process. Even if QoC constraints on context data could be mimicked by local filtering operations, they do not tailor the distributed data delivery process, thus possibly introducing unneeded overhead for caching/replicating out-of-QoC data. In addition, since replication techniques aim to ensure system-wide data consistency, they work effectively only with very slow change rates (close to null), and this is against the fact that context data could change very rapidly according to the represented physical phenomenon [Derhab and Badache 2009]. Similarly, multicast and group communication protocols tend to ensure consistency between produced and received data: this is against both production/consumption decoupling and QoC-based data filtering. In addition, they focus more on delivering the data as soon as possible, while leaving out the tailoring of the distributed data delivery process: hence, QoC constraints on the distribution process are usually not supported. Finally, also pub/sub solutions do not usually consider quality-based delivery [Mahambre *et al.* 2007]. On the one side, QoC constraints on data can be obtained via message filtering; however, the usage of these filters to tailor the (possible) distributed message routing depends on the specific implementation. In addition, context data distribution has to deal with both uncertain data and subscriptions, while the subscriptions made to pub/sub systems consider only perfect subscription/data matches. On the other side, QoC constraints on the distribution process have to be directly supported by the implementation since they affect the dispatching process itself. To the best of our knowledge, previous research on these systems mainly focused on reliability, i.e., reliable message delivery notwithstanding node mobility, by means of explicit sign-in/sign-off application-layer mechanisms and caching proxy servers running over the fixed infrastructure [Cugola and Di Nitto 2001; Cugola *et al.* 2001; Muhl *et al.* 2004; Sutton *et al.* 2001]; these solutions, instead, do not consider other quality objectives, such as the message delivery time.

Finally, all these approaches do not explicitly handle (context) data lifecycle. Even if mobile databases and pub/sub systems offer some solutions to deal with data/message removal, they do not offer more complex operations, such as data/message aggregation. Of course, as long as the system merely delivers data/message driven by additional and external routing information, the final payload could also adopt complex representation techniques, e.g., first-order logic; however, if the system cannot inspect payloads, different management operations, for instance QoC-based filtering, cannot be implemented. Similar to QoC-based data filtering, aggregation functions could be obtained by external services running on top the data delivery infrastructure; however, this limits possible operations and final system efficiency.

To conclude, although context data distribution exhibits some similarities with different research areas and works in the literature, none of these approaches is able to

fulfill the context data distribution requirements all together, especially i) locality principles; ii) QoC-based constraints both on received data and on distribution process; and iii) context data lifecycle management. With these observations in mind, we claim that context data distribution for context-aware system is different from all other traditional data distribution architectures. Consequently, our survey focuses on context data distribution, and we tend to exclude all those systems that, although belonging to close research fields, have not been specifically designed to perform context data distribution, because they do not address many issues highlighted in the previous sections; in addition, the neighbor areas have already received much attention elsewhere, where interested readers can refer to [Baldoni *et al.* 2009; Costa *et al.* 2009].

4. CONTEXT DATA DISTRIBUTION: A TAXONOMY

This section proposes an original taxonomy for clarifying main characteristics and components used in the context data distribution. The goal is to explain better our classification that stems directly from our architectural model; in addition, it shares some organization ideas with the few taxonomies already present in literature [Baldauf *et al.* 2007; Kjær 2007; Strang and Popien 2004]. To make the comprehension simpler, together with the taxonomy, we also introduce and present some state-of-the-art context data distribution systems so to exemplify step-by-step and populate the proposed classification.

Rather than reporting a more exhaustive list of the plethora of solutions in the literature, we purposely present a limited set of 37 systems to grant enough space for an in-depth analysis of each of them. We suitably put in our collection a selection of systems that cover all our main taxonomy directions, being representatives of the wide spectrum of context data distribution aspects.

The presentation order of the three main taxonomy parts derives directly from the proposed logical architecture: Subsection 4.1 opens the section by introducing the taxonomy for the context data management layer aspects, Subsection 4.2 deeply analyzes the context data delivery layer ones, and Subsection 4.3 concludes the section classifying run-time adaptation supports.

4.1. Context Data Management Layer

The context data management layer offers two fundamental components useful to locally handle context data: context data representation and processing (see Figure 1). First, context data need to be represented via a chosen representation technique; Subsection 4.1.1 presents several different approaches currently present in literature. Second, context data have to be processed according to service needs; Subsection 4.1.2 analyzes the main processing operators that a context data management layer has to offer to let services retrieve needed context data. For the sake of clarity, Figure 2 summarizes the proposed taxonomy with the possible choices for any of the components.

4.1.1 Representation

Several different models have been proposed to represent context information; they differ in expressiveness, memorization cost, and processing overhead. By focusing on expressiveness, we can divide context data models in *general*, *domain-specific*, and *no model* ones. *General models* offer a wide design space to enable the specification and the representation of any known application domain; they are concerned with the generic problem of knowledge representation. *Domain-specific models*, instead, represent only

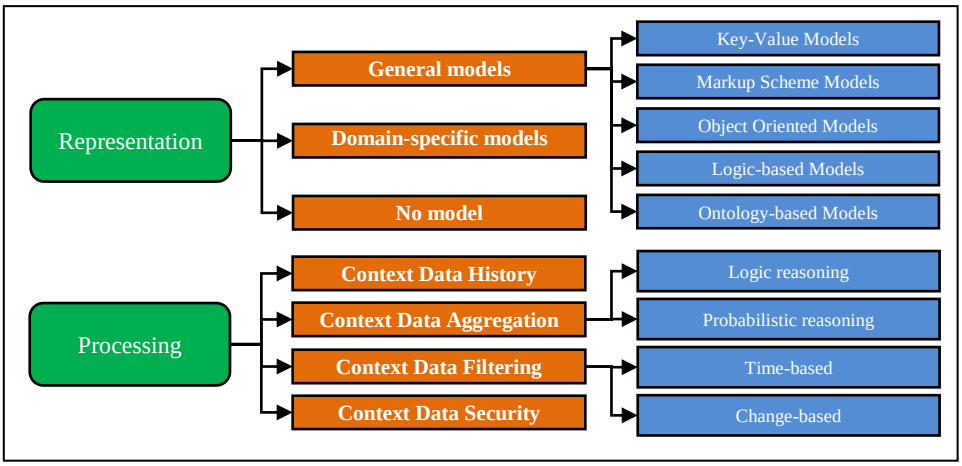

Fig. 2. Taxonomy for the classification of context data management layer

data belonging to a specific vertical domain and do not enable the specification of generic data: because of their reduced scope, they can introduce complex operations on data. Hybrid models based on two and even more models, either general or domain-specific, at the same time are feasible as well. Finally, several context data distribution systems do not address data representation aspects, by focusing more on other aspects; we define those systems *no model* ones.

In the following of this section, we focus mainly on general models since context-aware middlewares are widely based on them due to their more immediate applicability; in addition, we introduce also some seminal examples of domain-specific models.

General models offer different degree of formalism and expressiveness. Since model expressiveness strictly relates to offered data operations, more complex models tend to standardize and to supply additional data operations, like aggregation operators to derive new context data and quality operators to specify and manage QoC constraints. However, a general agreement about standard operations (not achieved yet) is needed to foster the development and the widespread adoption of reusable data management automatic tools. With an increasing order of complexity, context data distributions have adopted one of the main accepted and followed models: *key-value models*, *markup scheme models*, *object oriented models*, *logic-based models*, and *ontology-based models* [Bettini *et al.* 2010; Strang and Popien 2004]; we briefly present those models in the following with a finer degree of details.

Key-value models represent the simplest data structure for modeling context by exploiting pairs of two items: a key (attribute name) and its value; simplicity is the main reason for this approach popularity. Unfortunately, they tend to lack capabilities for structuring context data, and do not provide mechanisms to check data validity. Context Toolkit, one of the most important seminal works on context-awareness, adopts this approach to represent both context data and the metadata associated with context sources [Dey and Abowd 2000b]. Pervasive Autonomic Context-aware Environments (PACE) relies upon key-value pairs to represent context data used to determine which actions the user prefers in the current pervasive context [Henricksen *et al.* 2005]. History-Based routing protocol for Opportunistic networks (HiBOp) and Context-aware Adaptive Routing (CAR), two representative solutions for DTN, use computing, time, and user

context to evaluate and to select the best forwarder; to build the routing infrastructure, each mobile node distributes required context data, described as key-value pairs, to neighbors [Boldrini *et al.* 2008; Musolesi and Mascolo 2009].

Markup scheme models use XML-based representations to model a hierarchical data structure consisting of markup tags, attributes, and contents. These approaches overcome some of the limitations of key-value models; for instance, they support the possibility of i) validating context data by means of XML-schemas; and ii) structuring data via nested XML structures. Context-Aware Resource Management ENvironment (CARMEN) exploits XML-based profiles to describe both computing and user context information [Bellavista *et al.* 2003]. Context CASTing (C-CAST) is explicitly focused on context provisioning aspects, and defines a lightweight XML-based Context Meta Language (ContextML) to distribute context data into the system [Knappmeyer *et al.* 2009]. Cooltown uses Web presence (identified by an URL) to offer access to required context information: Web presences are rich Web-based interfaces linked via Web hyperlinks that can be navigated to obtain more information [Debaty *et al.* 2005]. COntext Provisioning for AL (COPAL) focuses on context data provisioning and processing, and represents data by means of XML documents [Li *et al.* 2010]. COntext Sharing In uNreliable Environments (COSINE) builds a modular context share in which context data are represented by XML and can be queried by using XPath queries [Juszczyk *et al.* 2009]. Finally, MANet Information Planet (MANIP) does not impose a context data model, but suggests some candidates, like Common Information Model (CIM) and Resource Description Framework (RDF), that belong to this kind of approach [Macedo *et al.* 2009; Powers 2003; Sweitzer *et al.* 1999].

Object-oriented models take advantage of the benefits of the object-oriented approach, typically encapsulation and reusability: each class defines a new context type with associated access functionalities; type-checking and data validity can be ensured both at compile- and at run-time, while QoC elements can also be easily mapped as other objects. At the same time, these models ease interactions between services and context data: the usage of the same abstractions provided by object-oriented programming languages simplifies the deployment of context handling code. By following this model, in COntext entitieS coMpositiOn and Sharing (COSMOS), each context data is reified as an object comprehending several built-in mechanisms to ensure both push- and pull-based change notifications [Conan *et al.* 2007]. Hydrogen represents each context data type with a sub-class of ContextObject [Hofer *et al.* 2003]. Reconfigurable Context-Sensitive Middleware (RCSM) exploits directly an Interface Definition Language (IDL) approach [Yau *et al.* 2004]: by using it, the developer can specify context/situations relevant to the application, the actions to trigger, and the timing of these actions. Similarly, Mobile Social Computing (MobiSoC) and Mobile Collaboration Architecture (MoCA) exchange traditional data objects defined by developers to represent useful context data, mainly belonging to the user context dimension [Gupta *et al.* 2009; Sacramento *et al.* 2004].

Logic-based models take advantages of the high expressiveness intrinsic to the logic formalism: context contains facts, expressions, and rules, while new knowledge can be derived by inference. Traditionally, these models focus on inference mechanisms by providing also proper formalisms to specify inference rules. Unfortunately, usually they do not offer simple functionalities to deal with data validity: validation can be ensured, but associated rules are not straightforward to specify and depend on the adopted type of

logic. Mobile Gaia and Gaia represent context using first-class predicates [Chetan *et al.* 2005; Ranganathan and Campbell 2003]. In both systems, context is represented through quaternary predicates like Context(<ContextType>, <Subject>, <Relater>, <Object>), where <ContextType> is the context type that the predicate is describing; <Subject> is the person, place, or physical object the context is concerned; <Object> is the value associated with the <Subject>; and <Relater> links <Subject> and <Object> by means of a comparison operator (=, >, or <), a verb, or a preposition. CORTEX and Context-Awareness Sub-Structure (CASS) employ a similar context data model [Duran-Limon *et al.* 2003; Fahy and Clarke 2004]. EgoSpaces adopts a tuple-space model, i.e., a logic-based approach, but it does not predefine the content of each tuple [Julien and Roman 2006].

Ontology-based models use ontologies to represent context and take advantage of the capability of expressing even complex relationships: data validity is usually expressed by imposing ontology constraints. By focusing on relationships between entities, ontologies are very suitable for mapping every-day knowledge within a data structure easily usable and manageable automatically. In addition, the wide adoption of ontology enables the reuse of previous works and the creation of common and shared domain vocabularies. Service-Oriented Context-Aware Middleware (SOCAM) composes a generic ontology with domain-specific ones [Gu *et al.* 2005]; when necessary, the specific ontology is bounded with the generic one. In addition, SOCAM classifies data as direct – either sensed directly by sensors or defined by users – and indirect – derived by inference. Context Broker Architecture (CoBrA) uses a context knowledge base and its own OWL-based ontology (CoBrA-Ont) to memorize available knowledge [Chen *et al.* 2003]. Context Management Framework (CMF) can aggregate different domain-specific ontologies, also defined by different administration domains [van Kranenburg *et al.* 2006]. Even if first-order logic and ontology approaches seem very competitive, mobile environments usually avoid them since the required computing resources (memory and CPU usage) could be not acceptable for resource-constrained mobile devices.

Finally, even if using a single data model can ease data management operations, some systems tend adopting hybrid models to grant the best of two (or more) models. Between surveyed solutions, only Solar and Scalable context-Aware middleware for mobiLe EnviromentS (SALES) represent data by using both a key-value and an object-based model [Chen *et al.* 2008; Corradi *et al.* 2010a]. The use of key-value pairs reduces management overhead (especially required bandwidth), while the adoption of the object-oriented approach facilitates the design and implementation of the systems by supporting also extendibility.

We have already stated that *domain-specific models* are less flexible since focused on a particular application domain; at the same time, due to the restricted flexibility, more complex aggregation operators are usually offered by the system. For instance, *spatial data models* are widely adopted by localization systems to represent real world objects location and to perform queries on containment, intersection, and so forth in a fast and efficient way. In this case, data validity is easier to ensure, and automatic tools are usually available to specify validation rules. Among surveyed solutions, MiddleWhere is a context data distribution system strictly related with location-aware scenarios, hence, it specifically focuses on physical context dimension [Ranganathan *et al.* 2004]. It uses a *spatial data model* assuming that real world objects can be only points, lines, and polygons, and that localization data form a hierarchy; based on this spatial representation

of the world, MiddleWhere can answer to location-based queries.

Finally, a good part of surveyed solutions comes *with no data model*, typically because they focus more on other specific technical aspects. For instance, Pervaho uses events for context data distribution without specifying their internal format [Eugster *et al.* 2008]; Aura focuses on task migration and uses all the four context dimensions, i.e., computing, physical, time, and user context, to tailor migration decisions [Sousa and Garlan 2002]; HiCon tackles context data aggregation by also assuming any type of data representation [Cho *et al.* 2008]. Context-Aware Reflective mIddleware System for Mobile Applications (CARISMA) and Mobile Platform for Actively Deployable Service (MobiPADS) propose general frameworks for context-aware service adaptation and do not impose any data representation [Capra *et al.* 2003; Chan and Chuang 2003]. Other solutions, typically MANET-, VANET-, and DTN-based systems, are more focused on protocol design and assessment than on data modeling: Habit uses physical and user context to create data distribution routes [Mashhadi *et al.* 2009]; Migratory Services deals with agents migration driven by context [Riva *et al.* 2007]; REplication in Dense MANet (REDMAN) ensures data replication with user-defined degrees [Bellavista *et al.* 2005]; Adaptive Traffic Lights and Active Highways use physical context to realize driver-assistance services through data exchanges between vehicles [Gorgorin *et al.* 2007; Iftode *et al.* 2008]; MobEyes proposes environmental data harvesting protocols for VANET [Lee *et al.* 2009].

To conclude, it is generally possible to infer that most systems tend adopting very simple general models, such as key-value and markup scheme models; nonetheless, adopted data model mainly depends on the supported scenarios and on the aggregation to perform. At the same time, to the best of our knowledge, although almost all above models offer sufficient abstractions to represent QoC constraints, none of them supports mature tools to declare and to enforce them. The huge design space, the different semantics associated with represented data, and the absence of generic QoC frameworks could be blamed for this lack. Consequently, due to above problems, real systems tend more to introduce ad-hoc solutions and to exploit additional metadata for QoC treatment.

4.1.2. Processing

The processing component is in charge of all those operations needed to locally shape retrieved context data according to service level needs. In particular, usual context data processing covers four main context data management aspects: context data history, aggregation, filtering, and security. Hence, by following this order, we sketch our taxonomy along those four main directions and we introduce more details about the processing component; let us anticipate that, depending on the processing operator, those techniques are prone to introduce very different overhead.

The *Context Data History* module captures the possibility of maintaining all relevant past events and retrieving the history of a particular context data. Of course, context data history imposes requirements on memory resources; depending on data sizes and on production rate, it could be difficult to maintain the whole history, especially in mobile deployment scenarios. However, despite required resources, the history is very useful, as demonstrated by new emerging solutions that more and more include it. SOCAM maintains the history of localization data [Gu *et al.* 2005]. In CORTEX, each context source predicts future context based on current state and on history [Duran-Limon *et al.* 2003]. Cooltown maintains a history of events related to a physical entity [Debaty *et al.*

2005]. In C-CAST, the context broker can implement a context data history useful to retrieve previous context events [Knappmeyer *et al.* 2009]. HiBOp and CAR maintain summaries of the history of encountered user context to drive context-aware routing decisions [Boldrini *et al.* 2008; Musolesi and Mascolo 2009]. Habit keeps track of time context information, such as node inter-contact times and frequencies, to elaborate distribution paths [Mashhadi *et al.* 2009]. Finally, SALES, CASS, and Gaia allow the maintenance of whatever data history [Corradi *et al.* 2010a; Fahy and Clarke 2004; Ranganathan and Campbell 2003].

The *Context Data Aggregation* module provides all the fusion and merging operations capable of managing different context data. Specific operations strictly depend on the adopted context data model and, since context data can be imprecise, affected by errors, and even stale, must be deeply concerned with QoC. The available aggregation techniques can be classified as *logic* and *probabilistic reasoning techniques* depending on whether the system considers the context data either simply correct or correct with a specified probability (typically smaller than 1); in addition, hybrid schemes that combine those two techniques are also followed. Probabilistic reasoning techniques could also derive the correctness of composed and complex context data from the correctness of single involved context data. Even if aggregation techniques can be very resource-demanding, they are fundamental to enable context-awareness since: i) context can be not usually defined explicitly due to the huge amount of possible context directions; and ii) context changes claim for continuous updates that must be carried on automatically by the system. Above all, Artificial Intelligence provides techniques, as well as standard logic-based representations and inference engines, that can simplify the usage of aggregation techniques. Consequently, most systems that require dynamic data aggregation adopt either logic- or ontology-based models that are simpler to manage and integrate with those engines.

By considering *logic reasoning* (aggregation) functions, in C-CAST, different context providers, also running on different devices (user terminals, servers, etc.), can be configured to exchange context data and to perform distributed context data aggregation [Knappmeyer *et al.* 2009]. Context Toolkit meta-widgets provide the ability to aggregate low-level context data, acquired directly from sensors by using so-called Context Toolkit widgets, into higher-level ones [Dey and Abowd 2000b]. COPAL enables the specification of complex context processing operations: in particular, it provides the Aggregator operator with the main goal of merging more context data in a new high-level one [Li *et al.* 2010]. In COSINE, Aggregator Services are introduced to explicitly handle context data aggregation; in addition, if a context query needs to correlate multiple types of context data, an aggregator service is able to determine which context should be aggregated and to transparently retrieve them [Juszczyk *et al.* 2009]. COSMOS middleware differentiates three main layers, i.e., collection, processing, and adaptation, and offers context processors to perform context aggregation [Conan *et al.* 2007]. HiCon introduces a hierarchical context aggregation framework based on three principal hierarchical levels, i.e., PocketMon (personal), HiperMon (regional), and EGI (global), to reduce bandwidth requirements and/or data visibility [Cho *et al.* 2008]. Finally, MANIP uses logic rule-based aggregation to infer high-level context data [Macedo *et al.* 2009]. In the same way, solutions that adopt first-order logic and ontology-based approaches easily integrate aggregation functionalities. CoBrA and CASS derive high-level context by using reasoning techniques based on inference engines [Chen *et al.* 2003; Fahy and

Clarke 2004]. Mobile Gaia infers high-level contexts by means of first-order logic operations (such as conjunction, disjunction, negation, implication, and quantification) on other predicates [Chetan *et al.* 2005]. PACE memorizes user context, especially preferences, and evaluates them to determine the preferred user actions in the current context, while RCSM manages context acquisition and elaboration to derive feasible descriptions of current situations [Henricksen *et al.* 2005; Yau *et al.* 2004]. SOCAM supports context aggregation by means of appropriate interpreters [Gu *et al.* 2005]. Finally, due to the limited memory resources of mobile devices, HiBOp, Habit and CAR use hard-coded aggregation algorithms to derive reduced context summaries from context data history [Boldrini *et al.* 2008; Mashhadi *et al.* 2009; Musolesi and Mascolo 2009].

By considering *probabilistic reasoning* (aggregation) functions, MiddleWhere incorporates a wide range of localization data aggregation techniques [Ranganathan *et al.* 2004]. Data supplied by different sensors are stored in a spatial database, and a reasoning engine merges them to retrieve location data with a certain probability; in addition, a location service ensures access to localization data by resolving conflicts, and answering to queries for spatial regions and physical objects. MobiSoC focuses on social state learning and introduces different solutions to discover people relations, social groups, and group-place relations: the authors introduce the Group-Place Identification (GPI) algorithm to aggregate user context information and to infer geo-social patterns with a certain probability [Gupta *et al.* 2007; Gupta *et al.* 2009].

Finally, two of surveyed solutions adopt hybrid approaches. Gaia uses a first-order logic model, and provides aggregation by means of both logical rule-based and probabilistic machine-learning techniques [Ranganathan and Campbell 2003]. Similarly, CMF applies the same two techniques to ontology-based models [van Kranenburg *et al.* 2006].

To conclude, even if very neglected in surveyed solutions, we claim that probabilistic reasoning techniques are fundamental to deal with context uncertainty and errors. However, the definition of proper aggregation algorithms is not straightforward, and usually results in different parameters, difficult to tune and situation dependent (for an example, see GPI [Gupta *et al.* 2009]).

The *Context Data Filtering* module strives to increase system scalability by controlling and reducing the amount of transmitted context data. These techniques are fundamental since: i) some context aspects change very often, and their associated sources can produce data with very high rates; and ii) context provisioning to services has to be managed according to granted QoC; if services can accept reduced QoC, that produces less management overhead, context data distribution can apply these techniques to enhance system scalability. In finer details, filtering techniques can be classified as *time-based*, i.e., data sending is suppressed until certain time conditions become true, and *change-based*, i.e., data sending is suppressed as long as context data is equal or similar to the previous transmitted ones; of course, also any combination of time-/change-based techniques is feasible as well. For instance, the service can ask to receive localization updates only if the new localization is different from the previous one and if at least an interval of 10 seconds is elapsed.

Among surveyed solutions, C-CAST context providers can perform sensor data filtering and pre-processing [Knappmeyer *et al.* 2009]. COPAL introduces filter components able to exclude data that do not match a particular combination of criteria

applied on context types, values and attributes [Li *et al.* 2010]. In COSMOS, every context data is an object that can introduce also complex filtering mechanisms on the propagation of context data changes [Conan *et al.* 2007]. Solar focuses on scalability for pervasive environments, and supplies a wide-range of time-based and change-based filtering techniques to avoid bandwidth congestion and client overload [Chen *et al.* 2008]. It exploits operator graphs deployed over the Context Fusion Network (CFN) to provide data filtering and to distribute context data. Each graph is composed in terms of sources, sinks, and operators, respectively, sensors, services, and components responsible for data filtering that act both as sources and sinks. Each service composes its own operator graphs by providing the associated filters, while Solar completely takes care of its allocation over the CFN. HiBOp, instead, adopts a hybrid filtering approach to exchange user context data useful to the maintenance of the DTN routing infrastructure [Boldrini *et al.* 2008]. Mobile nodes send their context data to neighbors exploiting a combination of a time-based approach, i.e., context is exchanged every beacon period, and of a change-based approach, i.e., context data are sent either if different from the one emitted in the previous period or if node neighborhood has changed.

Finally, the *Context Data Security* module includes all mechanisms to grant privacy, integrity, and availability of data (such as to counteract to Denial of Service attacks). Real deployment scenarios deeply ask for them because context data could carry sensible information. For instance, while temperature data exchanged in text may be not perceived by users as a privacy violation, other data containing user localization may require appropriate mechanisms to ensure privacy. However, we remark that the context data security is still much neglected an issue: one main reason is that security issues have been already tackled and solved in literature, and efficient solutions to address security problems, e.g., by exploiting access control and encryption mechanisms, are available and usable. Between surveyed solutions, only CoBrA, Mobile Gaia, Context Toolkit, CORTEX, CASS, SOCAM, MobiSoC, Gaia, and CMF support some security and privacy primitives [Chen *et al.* 2003; Chetan *et al.* 2005; Dey and Abowd 2000b; Duran-Limon *et al.* 2003; Fahy and Clarke 2004; Gu *et al.* 2005; Gupta *et al.* 2009; Ranganathan and Campbell 2003; van Kranenburg *et al.* 2006]. Unfortunately, despite traditional security issues, we stress that an important part of the privacy loss problem related to the usage of localization data is still open: in particular, indirect inferences of users identity/relations performed on those data represent a real problem that is currently mining the diffusion of these systems [Hengartner and Steenkiste 2005; Jones and Grandhi 2005].

4.2. Context Data Delivery Layer

The context data delivery layer takes care of routing the context data into the mobile ubiquitous system. Of course, because this layer sits right above the real network infrastructure available underneath, possible solutions at this layer can be limited by the adopted network deployment. This layer can be organized in two main components. The first one, *dissemination*, presented in Section 4.2.1, considers the main policies the context data distribution can adopt to decide i) which context data have to be distributed; and ii) which destination nodes will receive the distributed data. The second one, *routing overlay*, detailed in Section 4.2.2, considers that the context data distribution could exploit different types of overlay networks to connect and organize the involved brokers. Figure 3 shows the proposed taxonomy with the possible choices for the sake of clarity.

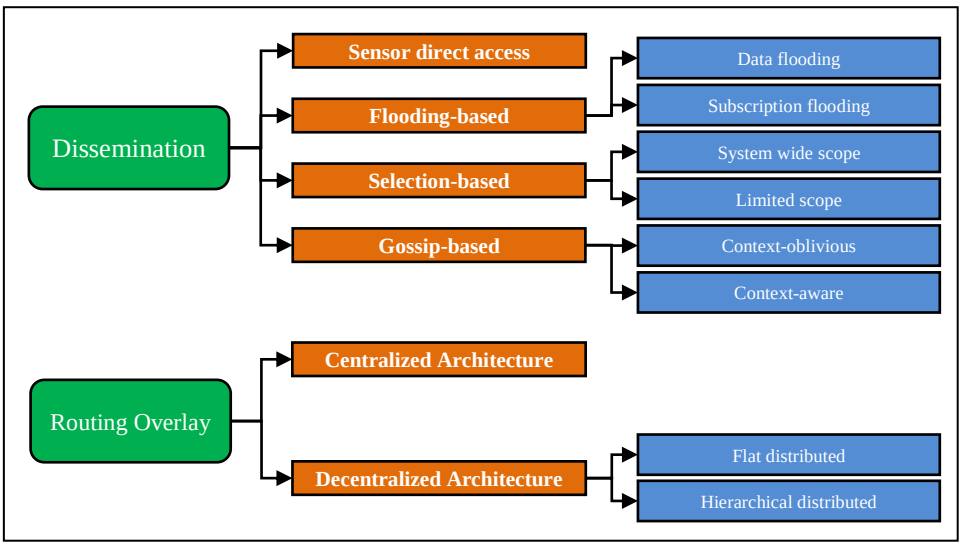

Fig. 3. Taxonomy for the classification of context data delivery layer

4.2.1 Dissemination

The *Dissemination* module enables data flow between sources and sinks. Hence, it is a core function in enabling context access with great impact on scalability and data availability. A notable extreme condition is when no dissemination support is needed and sinks directly access sources; we define that category *sensor direct access*. Apart from that dissemination strategy, dissemination solutions belong to three different categories: *flooding-based*, *selection-based*, and *gossip-based*. The first two categories characterize deterministic approaches where, except during system reconfigurations, a sink definitely receives matching data produced by sources belonging to the same context data distribution system. The last category is typical of the probabilistic approaches where a sink could miss some matching data. Systems adopting a hybrid approach that mixes these three main dissemination solutions are also notable and followed.

Given dissemination crucial role, we have decided to devote some more space to this section by presenting an in-depth analysis of the dissemination modules adopted by surveyed solutions; reported system descriptions are also useful to better clarify the taxonomy presented in Figure 3, and to draw comparisons and future research directions in next sections. In the following, we present the solutions adopted by surveyed systems, and we better detail *flooding-/selection-/gossip-based* categories by introducing additional taxonomy elements that can help in analyzing real systems.

Sensor direct access approaches may induce low data availability and clash with time/space decoupling because sinks have to communicate directly with sources to access data; however, as main benefit, they usually result in low complex organization and support: several seminal research proposals relied upon this approach. Cooltown services access directly Web servers to find context data [Debaty *et al.* 2005]. Context Toolkit introduces discoverers to handle registration from context data sources and to enable device mobility [Dey and Abowd 2000b]. By exploiting a Web service-oriented architecture, also COSINE subscriptions directly reach either physical or virtual sensors deployed in the network [Juszczyk *et al.* 2009]. COSMOS is completely focused on

context data processing, and assumes that all the context data are produced by local sensors [Conan *et al.* 2007]. SOCAM supplies a naming service to identify context data sources, and uses sensor direct access to retrieve data [Gu *et al.* 2005]. Finally, RCSM implements a context discovery protocol to manage registrations of local sensors and to discover remote sensors: when the application starts up, the discovery protocol looks for local/remote sensors to satisfy context requests, and then it enables direct access [Yau *et al.* 2004].

Flooding-based algorithms realize context data dissemination via flooding operations, in other words operations that reach all the nodes contained in a particular scope (e.g., the entire network, the one-hop neighborhood in an ad-hoc network, ...), and are typically complete within that scope. They operate either by flooding context data (*data flooding*) or by flooding context data subscriptions (*subscription flooding*). In data flooding, each node broadcasts known data to spread them inside the entire system by letting receiver nodes locally select data to be received. For instance, Adaptive Traffic Lights exchanges those context data useful to coordinate red/yellow/green times between vehicles near to an intersection using this kind of approach [Gorgorin *et al.* 2007]. HiBOp and CAR flood the one-hop physical neighborhood with those data useful for the maintenance of the DTN routing infrastructure [Boldrini *et al.* 2008; Musolesi and Mascolo 2009]. In MANIP, each data comes with a physical locality tag that limits the physical neighborhood to which the data is flooded [Macedo *et al.* 2009]. Instead, in subscription flooding, each node broadcasts its context data subscriptions to all nodes to build dissemination structure. This schema propagates subscriptions to all network nodes and assumes that each node memorizes subscriptions from all other nodes to perform local matching on produced data. This can reduce bandwidth overhead by disseminating only needed data; however, this schema requires very large routing tables, and that limits scalability. As a consequence, none of the surveyed context data distribution solutions uses subscription flooding.

Selection-based algorithms are typically organized on two phases. In the first one, they deterministically build dissemination backbones by using context data subscriptions; in the second one, data dissemination takes place only over the backbones, and is limited by granting that context data reach only interested nodes. To build backbones, nodes must exchange control information, thus introducing additional overhead-prone communications. Selection-based approaches can offer two different visibility scopes to each subscription: *system wide scope* and *limited scope*. In the first case, the dissemination process ensures that each subscription is visible in the whole distributed system, so to ensure that all the matching data will be retrieved. In the second case, the dissemination process limits subscription visibility to a subset of nodes, for instance the two-hop neighborhood in ad-hoc networks, so to ensure locality principles and increase system scalability; however, due to the limited visibility, it is possible that some matching data will not be found.

Starting with *system wide scope* approaches, CASS uses one central server that memorizes context data, while proper subscriptions let mobile nodes listen for context events [Fahy and Clarke 2004]. Both C-CAST and COPAL employ centralized brokers in which all subscriptions have system-wide visibility scopes [Knappmeyer *et al.* 2009; Li *et al.* 2010]. CMF adopts a fixed infrastructure in which distributed brokers coordinate to supply data to mobile nodes according to specific requirements [van Kranenburg *et al.* 2006]. Pervaho uses a Location-based Publish/Subscribe System (LPSS) by imposing

location-based constraints: each publication and each subscription has a visibility scope, and a publication is delivered to an active subscription only if publisher and subscriber lie in the intersection of these two scopes [Eugster *et al.* 2008]. Hence, Pervaho dissemination module performs context-/location-based filtering to limit received events. At the same time, the adopted network layer ensures that subscriptions have system wide visibility scopes. Solar adopts a selection-based network based on mobile nodes and CFN [Chen *et al.* 2008]. The CFN contains fixed hosts, so called Planets, used to allocate single operators at run-time. Context data dissemination exploits application-level multicast trees in which each Planet filters data according to node needs. Finally, also MobiSoC and MiddleWhere use selection-based approaches to disseminate context data to mobile nodes; at the same time, since based on central servers, they both ensure system-wide visibility to context subscriptions [Gupta *et al.* 2009; Ranganathan *et al.* 2004].

Instead, considering *limited scope* approaches, CORTEX sentient objects interact by using a service discovery and a pub/sub module: due to the deployment layer, interactions among sentient objects are limited, thus realizing a limited query visibility scope [Duran-Limon *et al.* 2003]. EgoSpaces also introduces a selection-based approach based on agents [Julien and Roman 2006]. Each agent operates over multiple views that include data/resources associated with hosts/agents in the physical locality. Each view imposes constraints on contained data/resources metadata; at the same time, each view can span close neighbors, thus leading to a limited visibility scope approach. Habit realizes data dissemination in a selection-based manner by exploiting time and user context, in particular, nodes physical proximity and user social relationships [Mashhadi *et al.* 2009]. It exploits a regularity graph that keeps trace of when and how often two nodes come into contact, and an interest graph that keeps trace of nodes interests, to build dissemination paths based on nodes interested in the data they are willing to route. By using the above graphs, each source node calculates final dissemination paths and sends data when next relay becomes visible; however, the data useful to build the interest graph are disseminated only to close neighbors belonging to the regularity graph. Mobile Gaia groups nodes into clusters, and it uses an event service to disseminate context data: because every cluster has its own event service, the final approach is selection-based with limited visibility scope [Chetan *et al.* 2005]. SALES exploits a selection-based approach on a structured hierarchical architecture that imposes physical locality principle [Corradi *et al.* 2010a]. To route data, SALES introduces context queries in the sense of subscriptions. Each query captures particular context needs, and is disseminated in part of the SALES distributed architecture, thus leading to limited visibility scopes. In addition, SALES adopts Bloom filters [Bloom 1970; Broder and Mitzenmacher 2005], i.e., a space-efficient data structure for membership tests, to reduce query size.

Gossip-based algorithms disseminate data in a probabilistic manner by letting each node resend the data to a randomly-selected set of neighbors. Since these approaches do not need complex routing infrastructures to be constructed and maintained, but rather simple and local views of the network to choose the neighbors to which gossip data to, gossip-based protocols well suit fast-changing and instable networks, such as MANETs [Friedman *et al.* 2007]. It is worth stressing that, if properly tuned, these proposals are able to ensure high reliability and low latency despite their own simplicity: however, at the same time, they exhibit a run-time behavior that strictly depends on node density and

mobility, and this could lead to not so stable performance. In a more detailed view, gossip-based protocols can be classified in two broad categories: *context-oblivious* and *context-aware* approaches [Friedman *et al.* 2009; Kermarrec and van Steen 2007].

Context-oblivious protocols usually rely on random retransmission probabilities and do not consider any external context information to tailor their behavior [Sasson *et al.* 2003]. Among context-oblivious approaches, pure probabilistic gossip systems simply resend each received data with a retransmission probability that can be different for each neighbor node and depends either on the local node density or neighborhood information [Cartigny and Simplot 2003; Drabkin *et al.* 2007; Haas *et al.* 2002; Tilak *et al.* 2003]. In counter-based gossip systems, instead, every time a node receives a new data, it waits a random delay to overhear possible retransmissions by neighbors: at the end of the delay, the node resends the data if and only if it has overheard a number of total retransmissions lower than a threshold [Cartigny and Simplot 2003; Haas *et al.* 2002]. One important finding about context-oblivious approaches in [Haas *et al.* 2002] and [Sasson *et al.* 2003] is that probabilistic gossip with equal retransmission probabilities at every node has a threshold behavior for data dissemination: the percentage of nodes that will receive the data suddenly increases when approaching a specific threshold that depends on node density. Hence, we conclude that main benefit of these approaches is in the fact that they involve neither heavy computation nor state on traversed nodes that simply select randomly in the neighborhood. Unfortunately, since they can waste wireless bandwidth uselessly by gossiping unneeded data, none of the surveyed systems adopts them.

Context-aware protocols, typically used by informed context data distribution, select neighbors for data gossiping by using some external context data potentially belonging to very different context dimensions. For instance, some approaches use distance between nodes (physical context) to position replicas far away [Miranda *et al.* 2009]; other approaches use social similarity, such as membership to the same class (user context), to select neighbors to gossip data to. In summary, context-aware approaches reduce the number of useless gossiped data, but they require heavier coordination to exchange and analyze context data used to make gossip decisions. Among surveyed solutions, REDMAN implements a context-aware gossip approach that exploits hop-count and replication degrees to drive replication process [Bellavista *et al.* 2005]. Besides, HiBOp and CAR exploit context-awareness for message routing purpose: above all, to select the best forwarder during message routing [Boldrini *et al.* 2008; Musolesi and Mascolo 2009].

Finally, hybrid approaches are present as well. Active Highways collects data from sensors local/remote to the vehicles, and relies upon fixed servers to assist vehicles in transit; a selection-based approach with limited visibility scopes is used to disseminate data among servers [Iftode *et al.* 2008]. Gaia uses both sensor direct access and selection-based policies with system wide visibility scopes [Ranganathan and Campbell 2003]. In the same way, HiCon exploits a sensor direct access to produce data, and a selection-based approach based on its hierarchical distributed architecture to disseminate data by imposing both (physical and logical) locality principles; in addition, even if not explicitly stated by the authors, we can safely assume that the three-level hierarchical architecture of HiCon enforces limited query visibility scopes to foster scalability [Cho *et al.* 2008]. Also Hydrogen uses a mixed approach in which each node can retrieve data through local sensors (sensor direct access approach) and by contacting with other mobile nodes (the policy used to exchange data during contacts is not clarified by authors) [Hofer *et al.*

2003]. Finally, in MobEyes, data can be collected either by sensors or by other vehicles during contacts: close vehicles exchange data by using a flooding-based approach where each vehicle broadcasts context data either locally sensed or received by other vehicles according to system configuration [Lee *et al.* 2009].

4.2.2 Routing Overlay

The routing overlay module takes care of organizing the brokers involved in context data dissemination. Different architectures can be classified as *centralized* and *decentralized*; the centralized approach includes any possible concentrated deployment (i.e., both single host and clustered), while we classify decentralized architectures into two main subcategories: *flat distributed* and *hierarchical distributed*. These latter two architectural alternatives can help in satisfying the physical locality principle, for instance by ensuring that each broker handles only close and easily reachable physical places, and can enhance scalability even if they introduce additional management overhead. The adopted network deployment limits the feasible solutions of the overlay module; for instance, ad-hoc network deployments are extremely decentralized with possible network partitions and node departures, hence they clash with the realization of centralized overlays. Consequently, for the sake of clarity, in the remainder we consider every single routing overlay type and, for each one of them, we group solutions depending on the adopted network deployment.

Centralized architectures are usually adopted in conjunction with fixed wireless infrastructures at the network deployment. With a fine degree of details, CoBrA, CASS, and COPAL adopt a wireless infrastructure with a centralized server that stores data from sensors and supplies them to context-aware services on mobile nodes [Chen *et al.* 2003; Fahy and Clarke 2004; Li *et al.* 2010]. In C-CAST, a centralized server stores and supplies context data to mobile nodes: even if authors present the introduction of a distributed broker overlay as a future research direction, to the best of our knowledge this feature is still lacking in the C-CAST project [Knappmeyer *et al.* 2009]. Pervaho LPSS also exploits a wireless infrastructure with a central JMS-based server [Chappell and Monson-Haefel 2000; Eugster *et al.* 2008]. Even if authors suggest that Pervaho LPSS can be realized in a complete decentralized ad-hoc manner (in which nodes enact as brokers), to the best of our knowledge, LPSS ad-hoc-based implementation is not available yet. Gaia uses a wireless infrastructure with a centralized server that realizes context lookup and context data access [Ranganathan and Campbell 2003]. MobiPADS also implements a centralized approach: it exploits Mobilets, i.e., active entities that can be migrated to install new code and to transfer computational tasks, and adopts context-awareness to perform service-level adaptation [Chan and Chuang 2003]. Also SOCAM, MobiSoC, MiddleWhere, and MoCA exploit a fixed centralized architecture in which a single server takes care of supplying access to available context data [Gu *et al.* 2005; Gupta *et al.* 2009; Ranganathan *et al.* 2004; Sacramento *et al.* 2004]. Instead, for ad-hoc settings, in Adaptive Traffic Lights, each traffic light enacts as coordinator by receiving physical context data from near vehicles and by elaborating the red/yellow/green times for the next period [Gorgorin *et al.* 2007]. Finally, COSMOS focuses mainly on context processing issues, and assumes that all the context sources are locally deployed at the node executing the context-aware services; hence, we categorize it among centralized architectures since every node can be seen as a central server building its own context data distribution system [Conan *et al.* 2007].

Moving toward *flat distributed* architectures, several systems have been proposed in literature both for infrastructure-based and ad-hoc based settings. Starting with wireless fixed infrastructures, Active Highways is based upon servers that receive data collected by vehicles/sensors deployed in the highways, and that coordinate to obtain time constraints on journeys [Iftode *et al.* 2008]. Aura exploits an infrastructural environment and migrates applications between different hosts to support users in performing tasks [Sousa and Garlan 2002]. CARMEN exploits a similar approach in which mobile proxies follow the associated user while roaming [Bellavista *et al.* 2003]. CMF has a distributed set of brokers that coordinate to supply context data according to services needs [van Kranenburg *et al.* 2006]. Cooltown expects wireless Internet connection to access different Web applications that supply context data [Debaty *et al.* 2005]. COSINE adopts distributed Web services that can provide access to both sensors and aggregation functionalities; the final network is a flat distributed one [Juszczyk *et al.* 2009]. PACE uses different servers organized in a flat distributed architecture to memorize user context [Henricksen *et al.* 2005]. Solar adopts a wireless flat infrastructure composed by Planets: each Planet manages associated sensors, participates to the CFN application-level multicast trees, and distributes data to registered services [Chen *et al.* 2008]. Moreover, for ad-hoc-based settings, HiBOp, CARISMA, Context Toolkit, CORTEX, Hydrogen, EgoSpaces, MobEyes, MANIP, Habit, CAR, Migratory Services, and RCSM share context information with devices in physical proximity, thus realizing the (ad-hoc) flat approach [Boldrini *et al.* 2008; Capra *et al.* 2003; Dey and Abowd 2000b; Duran-Limon *et al.* 2003; Hofer *et al.* 2003; Julien and Roman 2006; Lee *et al.* 2009; Macedo *et al.* 2009; Mashhadi *et al.* 2009; Musolesi and Mascolo 2009; Riva *et al.* 2007; Yau *et al.* 2004]. In addition, even if they adopt clustering protocols to manage the network, from the routing overlay viewpoint, both REDMAN and Mobile Gaia adopt a flat distributed architecture to manage context data [Bellavista *et al.* 2005; Chetan *et al.* 2005].

Finally, among surveyed systems, we found *hierarchical distributed* architectures for the routing overlay only when the system adopts a mixed wireless network, i.e., based on both a fixed infrastructure and ad-hoc wireless communications. In fact, SALES exploits both ad-hoc and wireless infrastructure communications, and organizes nodes belonging to the system in a hierarchical architecture to ensure physical locality principle: the final overlay used for the context data distribution is a tree-like three-level one [Corradi *et al.* 2010a]. Similarly, HiCon realizes a tree-like overlay architecture in which intermediate nodes perform context data aggregation to reduce the number of exchanged context data [Cho *et al.* 2008].

4.3. Run-time Adaptation Support

The run-time adaptation support is in charge of dynamically managing and modifying context data distribution. Even if not many solutions have investigated the dynamic adaptation of data distribution so far, we claim that this is already a core component of a very significant set of informed context data distribution systems; moreover, it will receive an increasing attention due to the growing number of solutions that require adaptive and efficient context distribution in large scale networks.

Our proposed taxonomy, shown in Figure 4, stresses a crucial adaptation support aspect: service level can affect the adaptation process by influencing the decisions of the run-time adaptation support with different levels of control; we classify it as i) *unaware*, ii) *partially-aware*, and iii) *totally-aware*. In *unaware adaptation*, the service level

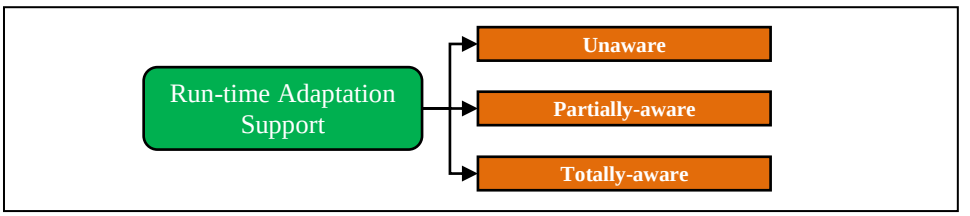

Fig. 4. Taxonomy for the classification of run-time adaptation support

neither reaches nor influences run-time adaptation support strategies. In *partially-aware adaptation*, there is more collaboration between the two: the service level supplies profiles that describe the required kind of service, while the run-time adaptation support modifies context data distribution facilities to meet those requests. Finally, in *totally-aware adaptation*, the run-time adaptation support does not perform anything on its own, and it is the service level that completely drives reconfigurations.

Since adaptation is a crosscutting concern with many complex management goals, let us present a more expressive diagram (Figure 5) to better clarify how the run-time adaptation support works. The support exploits both context data inputs (*computing*, *physical*, *time*, and *user context*) and QoC parameters, and, after an elaboration, produces specific reconfiguration commands for both context data management and delivery layers. To be more concrete, adaptations can follow five main directions. First, *computing context*: the run-time adaptation support triggers and executes management functions aimed to overcome changes in the execution environment, such as wireless AP handoff and wireless technology modifications. Second, *physical context*: the run-time adaptation support modifies data distribution according to physical constraints, such as by exploiting localization to save unneeded data forwarding. Third, *time context*: the run-time adaptation support modifies data distribution according to specific events or time-of-the-day, for instance in actions from slowing down to suspending the context distribution during night. Fourth, *user context*: the run-time adaptation support tailors data distribution to user preferences, for instance choosing low-cost connections even if they offer lower bandwidth. Fifth, *QoC parameters*: the run-time adaptation support dynamically modifies context data dissemination, for instance, by applying proper filtering criteria and differentiated data priorities according to required QoC. Let us note that the run-time adaptation support should consider all these aspects since reconfigurations can depend from complex conditions, spanning different context aspects

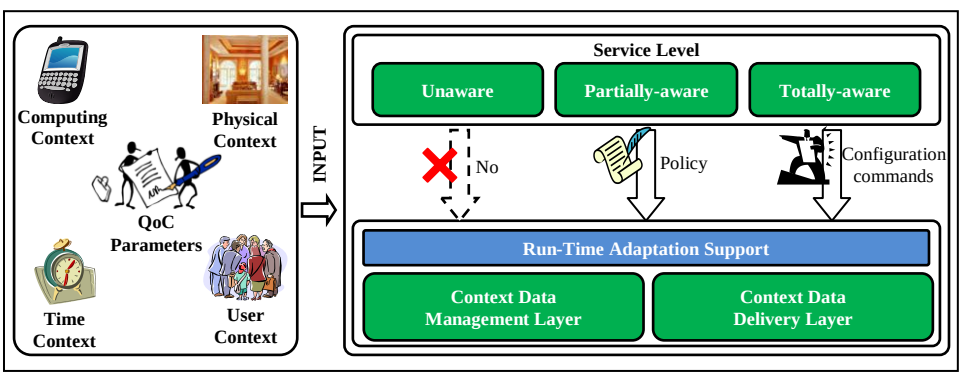

Fig. 5. Run-time adaptation support: taxonomy and interactions with other layers

and QoC parameters.

After these inputs elaboration, the run-time adaptation support can command suitable management actions at all different layers. For context data management layer, common reconfigurations deal with the usage of filtering techniques to finely tune received context data. The emerging notion of QoC is also encouraging the introduction of mechanisms suited to data quality; for instance, QoC-based data filtering is a valuable component, but it usually assumes that data chunks carry a quality descriptor attached at the production by the source and, perhaps, modified by brokers involved in data dissemination. Instead, for context data delivery layer, common reconfigurations try to adapt dissemination algorithms according to current run-time conditions. For instance, if a gossip-based protocol is adopted, the gossip period between two different gossip operations must be adapted to available bandwidth and mobility. In this case, the most interesting solutions are those that realize the context data distribution in a completely decentralized way, mainly flooding-/gossip-based supports for MANET/VANETs, since those systems need to adapt and to optimize distribution to overcome resource-constrained mobile nodes limitations.

Delving into surveyed systems details, and starting with *unaware adaptation* approaches, CMF automatically chooses among multiple instances of a certain source by using QoC information, like most accurate, reliable, and fresh context information [van Kranenburg *et al.* 2006]. Similarly, COSINE analyzes each context subscription, expressed as XPath query, to understand which sources can provide requested context data: the different opportunities are ranked depending on QoC parameters, and the subscription is sent to the best one without service intervention [Juszczyk *et al.* 2009]. At the same time, this approach has a very interesting outcome when a context subscription requires data from multiple context sources: if there exists an aggregator service that already collects all the required context data, the subscription is routed directly to it; otherwise, the initial subscription is automatically translated into a set of fine-grained subscriptions, each one for each required context source. Both HiBOp and CAR dynamically adapt data distribution depending on current context information [Boldrini *et al.* 2008; Musolesi and Mascolo 2009]. The adaptation actions can be tuned by modifying weighting parameters used in context processing; however, these parameters are hard coded, and cannot be adapted from the service level, thus leading to an unaware approach. In MobEyes data retrieval exploits multiple agents that carry subscriptions and travel the network so to harvest as much interesting data as possible [Lee *et al.* 2009]. To ensure efficient, effective, and fast data harvesting, MobEyes adopts biological-inspired algorithms to mark information-productive regions, and to keep away agents from already harvested regions. The adaptation actions are automatically imposed during agents roaming by the context data distribution function without service level intervention, thus realizing an *unaware* approach. Finally, MANIP is a MANET-based information repository useful to share information between different nodes and where each node can publish its local context computing conditions, such as queue lengths, available bandwidth, and so on [Macedo *et al.* 2009]. The cluster coordinator monitors those context data to decide the current subscription distribution technique, and imposes its choice on all other cluster nodes. Authors present i) a flooding approach where subscriptions are flooded by each node; and ii) a random-walk approach where each node propagates subscriptions to only few randomly-selected neighbors (thus realizing a context-oblivious gossip-based approach). The system can switch at run-time between the

two approaches to ensure good performance even with high load conditions close to congestion.

Moving toward *partially-aware adaptation* solutions where services can influence the run-time adaptation support, REDMAN implements a partially-aware adaptation approach by granting data dissemination in a MANET with certain data replication degrees [Bellavista *et al.* 2005]. Each cluster has a local coordinator that maintains the current replication status by monitoring the other nodes in the cluster (and their own local data). When the sensed replication degree is not equal to the required one, the coordinator commands a new data dissemination by using a gossip-based context-oblivious protocol. SALES allows services to declare and force specific constraints on system resource usage, such as the maximum number of context data to memorize and the percentage of CPU for different context data priorities, so to drive and constrain the distribution process [Corradi *et al.* 2010b]. At run-time, these parameters affect the context data matching and dispatching to the distributed architecture, while the SALES distribution function exploits them to self-adapt to currently available resources. Similarly, Solar exploits an adaptive approach to counteract buffer overflows, consequence of high production rates [Chen *et al.* 2008]. Each service supplies a policy that comprehends different filtering levels, each one composed by a chain of filters. These levels must be arranged to reflect the willingness of dropping context data under increasing overflow conditions; hence, the higher the filtering level, the tighter the filtering conditions. Then, if a buffer overflow is detected at run-time, Solar automatically determines the filtering level to use and applies all the filters, starting from the lowest level up to the current one, to the queue buffering the context data. Optionally, a policy can also define methods, such as the average value, to summarize dropped data in digests, thus enabling the provisioning of context with lower quality during congestion.

Finally, some examples of notable *totally-aware adaptation* approaches; compared to partially-aware ones, they are fewer because they require continuous actions and participation to the adaptation process by the service level, thus complicating the service development except for very specific application scenarios. MoCA services can specify constraints, such as temporal utility and validity, to apply on received context data [Endler and Da Rocha 2006]. Similarly, MiddleWhere provides localization data filtering based on freshness and temporal utility, precision (quality), and validity [Ranganathan *et al.* 2004]. Hence, both MoCA and MiddleWhere services can directly control the adaptation process by limiting the decisions of the run-time adaptation support at both context data management and delivery layers.

5. COMPARISONS OF SURVEYED SOLUTIONS

In this section, we compare surveyed systems to stress the main benefits and the shortcomings of the possible solutions that can be adopted for context data distribution. As in the previous section, we organize our comparison according to our logical architecture (see Figure 1). Let us anticipate that the most important result is that all aspects related to QoC, starting from QoC constraints definition, to QoC management (e.g., aggregation, filtering, and QoC usage for data distribution), and more complex dynamic QoC-driven system adaptation actions, are the ones that deserve more exploration and for which clear models are still missing. In fact, notwithstanding our extensive investigations for systems supporting both QoC constraints specification and adaptive data distribution, we found only few seminal works (11 out of 37).

Table I. Comparison between surveyed solutions

Network Deployment	System	References	Context Data Management					Context Data Delivery						Run-time Adaptation Support		
			Context Data Model	Processing				Dissemination				Routing Overlay				
				History	Aggregation	Filtering	Security	Sensor access	Flooding-based	Selection-based	Gossip-based	Centralized	Decentralized	Unaware	Partially-aware	Totally-aware
Hybrid	**SALES**	Corradi *et al.* 2010a	K.-V. P. / O.B.	✓	×	×	×	×	×	✓	×	×	✓	×	✓	×
	HiCon	Cho *et al.* 2008	-	×	✓	×	×	✓	×	✓	×	×	✓	×	×	×
	Pervaho[1]	Eugster *et al.* 2008	-	×	×	×	×	×	×	✓	×	✓	×	×	×	×
Ad-Hoc Infrastructure	**COSINE**	Juszczyk *et al.* 2009	M.S.	×	✓	×	×	✓	×	×	×	×	✓	✓	×	×
	CAR	Musolesi and Mascolo 2009	K.-V. P.	✓	✓	×	×	×	✓	×	✓	×	✓	✓	×	×
	Habit	Mashhadi *et al.* 2009	-	✓	✓	×	×	×	×	✓	×	×	✓	×	×	×
	MANIP	Macedo *et al.* 2009	M.S.	×	✓	×	×	×	✓	×	✓[2]	×	✓	✓	×	×
	MobEyes	Lee *et al.* 2009	-	×	×	×	×	✓	✓	×	✓[2]	×	✓	✓	×	×
	HiBOp	Boldrini *et al.* 2008	K.-V. P.	✓	✓	✓	×	×	✓	×	✓	×	✓	✓	×	×
	Adaptive Traffic Lights	Gorgorin *et al.* 2007	-	×	×	×	×	×	✓	×	×	✓	×	×	×	×
	Migratory Services	Riva *et al.* 2007	-	×	×	×	×	-	-	-	-	×	✓	×	×	×
	EgoSpaces	Julien and Roman 2006	-	×	×	×	×	×	×	✓	×	×	✓	×	×	×
	Mobile Gaia	Chetan *et al.* 2005	F.O.L.	×	✓	×	✓	×	×	✓	×	×	✓	×	×	×
	REDMAN	Bellavista *et al.* 2005	-	×	×	×	×	×	×	×	✓	×	✓	×	✓	×
	RCSM	Yau *et al.* 2004	O.B.	×	✓	×	×	✓	×	×	×	×	✓	×	×	×
	CARISMA	Capra *et al.* 2003	-	×	×	×	×	-	-	-	-	×	✓	×	×	×
	CORTEX	Duran-Limon *et al.* 2003	F.O.L.	✓	×	×	✓	×	×	✓	×	×	✓	×	×	×
	Hydrogen	Hofer *et al.* 2003	O.B.	×	×	×	×	✓	-	-	-	×	✓	×	×	×
	Context Toolkit	Dey and Abowd 2000b	K.-V. P.	×	✓	×	✓	✓	×	×	×	×	✓	×	×	×
Fixed Infrastructure	**COPAL**	Li *et al.* 2010	M.S.	×	✓	✓	×	×	×	✓	×	✓	×	×	×	×
	C-CAST	Knappmeyer *et al.* 2009	M.S.	✓	✓	✓	×	×	×	✓	×	✓	×	×	×	×
	MobiSoC	Gupta *et al.* 2009	O.B.	×	✓	×	✓	×	×	✓	×	✓	×	×	×	×
	Active Highways	Iftode et al. 2008	-	×	×	×	×	✓	×	✓	×	×	✓	×	×	×
	Solar	Chen *et al.* 2008	K.-V. P. / O.B.	×	×	✓	×	×	×	✓	×	×	✓	×	✓	×
	COSMOS	Conan *et al.* 2007	O.B.	×	✓	✓	×	✓	×	×	×	✓	×	×	×	×
	CMF	van Kranenburg *et al.* 2006	On.B.	×	✓	×	✓	×	×	✓	×	×	✓	✓	×	×
	PACE	Henricksen *et al.* 2005	K.-V. P.	×	✓	×	×	-	-	-	-	×	✓	×	×	×
	SOCAM	Gu *et al.* 2005	On.B.	✓	✓	×	✓	✓	×	×	×	✓	×	×	×	×
	CASS	Fahy and Clarke 2004	F.O.L	✓	✓	×	✓	×	×	✓	×	✓	×	×	×	×
	MiddleWhere	Ranganathan *et al.* 2004	Spatial Model	×	✓	×	×	×	×	✓	×	✓	×	×	×	✓
	MoCA	Sacramento *et al.* 2004	O.B.	×	×	×	×	-	-	-	-	✓	×	×	×	✓
	CARMEN	Bellavista *et al.* 2003	M.S.	×	×	×	×	-	-	-	-	×	✓	×	×	×
	CoBrA	Chen *et al.* 2003	On.B.	×	✓	×	✓	-	-	-	-	✓	×	×	×	×
	Cooltown	Debaty *et al.* 2003	M.S.	✓	×	×	×	✓	×	×	×	×	✓	×	×	×
	Gaia	Ranganathan and Campbell 2003	F.O.L.	✓	✓	×	✓	✓	×	✓	×	✓	×	×	×	×
	MobiPADS	Chan and Chuang 2003	-	×	×	×	×	-	-	-	-	✓	×	×	×	×
	Aura	Sousa and Garlan 2002	-	×	×	×	×	-	-	-	-	×	✓	×	×	×

Legend:
K.-V. P.) Key-Value Pairs M.S.) Markup Scheme O.B.) Object-based F.O.L.) First-Order Logic On.B.) Ontology-based
1) Ad-hoc/infrastructure mutually exclusive 2) During data harvesting process
√) Function Supported x) Function Not Supported -) Not explicitly defined

With a closer view to details, Table I summarizes all 37 surveyed solutions. We decided to group surveyed systems according to adopted network deployment since that permits to neatly divide all solutions and also because it is a fair indicator of time frame a solution has been proposed. We start from the, typically older, fixed-based infrastructures (18 solutions at the bottom); in the middle we report more recent ad-hoc solutions (16 solutions likewise); then, we end with the most recent and fewer hybrid ones (only 3 seminal solutions). Within each group, we order solutions in increasing time order, from the oldest to the newest one.

5.1. Context Data Management Layer

The context data management layer is fundamental to locally manage context data and a wide range of mechanisms is required to elaborate and process retrieved context data and to supply them in a proper form at the service level. As a matter of fact, past research related to context-awareness mainly focused on these issues; in fact, among surveyed systems, it is possible to find several different solutions for both the context data representation and the context data processing. In Section 5.1.1 and 5.1.2, we compare respectively the different context representation and processing techniques with associated benefits and shortcomings; at the same time, when needed, we also highlight possible relationships between adopted solutions and deployment scenarios.

5.1.1. Representation

Because context data representation is a well-known and studied issue, much literature has already addressed it by also providing different techniques with associated pros and cons in terms of complexity, overhead, support for imperfect data modeling, and so forth. Given a generic context-aware problem, we can represent the domain knowledge by using one by one the different approaches: of course, some choices could be more appropriate since different techniques offer different modeling levels of abstraction.

Above all, the context data model should be adopted according to available resources and need of aggregation techniques. First, if we have scarce resources, such as very bandwidth-constrained links, we cannot afford the exchange of complex data, and we have to optimize their representation. Second, if the system exchanges only statically defined context data without performing any kind of aggregation, simple models, such as key-value pairs and XML-based, are sophisticated enough and also ensure the best price/performance ratio. Let us stress that more evolved but complex models, such as ontology-based, should be carefully used since they usually introduce a wide design space that can confuse designers, thus increasing the probability of introducing modeling errors.

The above considerations are widely verified in all surveyed solutions. As shown in Table I, ad-hoc infrastructures characterized by system deployments with strict resource constraints, and that do not require particular aggregation on context data, i.e., many MANET/VANET/DTNs-based systems [Boldrini *et al.* 2008; Dey and Abowd 2000b; Juszczyk *et al.* 2009; Macedo *et al.* 2009; Musolesi and Mascolo 2009], adopt simple models, i.e., key-value pairs or XML-schema based. Fixed infrastructure solutions for context-aware pervasive environments, like CoBrA, PACE, Gaia and CMF [Chen *et al.* 2003; Henricksen *et al.* 2005; Ranganathan and Campbell 2003; van Kranenburg *et al.* 2006], instead, are typically tailored for full-fledged fixed infrastructure deployments where one of the main goals is to support context changes: in this case, ontologies represent a good choice due to the supplied aggregation instruments. In the middle,

between these two extreme deployment scenarios, there is a wide spectrum of design context modeling possibilities.

In any case, let us remark that context modeling is still a complex, human-based and time-consuming task very prone to errors: even the usage of powerful tools that help designers during modeling phase does not succeed in granting that the final model well represents context.

5.1.2. Processing

Context data processing strictly depends on the context-aware scenarios we are going to support.

As shown in Table I, even if context data history is a very resource-demanding function, many context-aware systems (also belonging to different scenarios) assume and require its services. In fact, context-aware pervasive systems, like PACE [Henricksen *et al.* 2005], require history to perform reasoning over past events and users' choice to adapt future automatic decisions. In addition, context-aware routing protocols for DTN, such as HiBOp and CAR [Boldrini *et al.* 2008; Musolesi and Mascolo 2009], modify routing by considering the temporal dimension, i.e., the history, of context data used to route message. In conclusion, the introduction of context history strictly descends from system requirements and from the reasoning the system will perform.

About aggregation, even if in slightly different manners, many solutions adopt those techniques with two principal goals: i) to reduce memory requirements by obtaining summary of past context data history (and substituting old data with produced summary); ii) to increase the system knowledge by introducing new high-level context. The former requirement applies especially to systems with resource-constrained environments, such as MANET/VANET/DTN-based solutions, while the latter describes systems for pervasive context-aware environments, usually supported by wireless fixed infrastructure. Either way, as also demonstrated by the number of surveyed systems that introduce aggregation operators, these techniques are required to support context-aware systems in next-generation ubiquitous systems. At the same time, let us remark that aggregation techniques have to be supported and implemented in the context data distribution itself, so to enable further distribution process optimizations and to reduce the final management overhead. For instance, COSINE exploits aggregator services to collect context data from multiple data sources, and automatically routes context subscriptions to these services at run-time to avoid multiple and expensive fine grained subscriptions [Juszczyk *et al.* 2009].

Filtering techniques seem to be rarely adopted. However, we argue that this lack is not due to the fact that systems could not take advantage of them, but to the fact that many solutions adopt them without explicitly assessing it in the description of systems themselves. In this case, the usage of these techniques strictly depends on three main factors: i) QoC; ii) supported deployment scenarios; and iii) system size. First, if the system introduces QoC constraints on data, filtering techniques should be introduced to correctly handle data management and delivery: as already stated for data aggregation, these primitives should also be implemented in the distribution function to avoid unneeded overhead. Second, when the system targets resource-constrained environments, filtering techniques are fundamental to avoid system and resource overloading: above all, context data sources have to be carefully controlled to avoid heavy context data distributions (for instance, see Solar [Chen *et al.* 2008]). Finally, solutions that support

services with city/regional/worldwide scope should adopt these techniques since whatever context data distribution solution would be completely overwhelmed by the traffic introduced without filtering.

Finally, only very few systems adopt any form of security (see Table I). In our opinion, this lack stems from the fact that research on context-awareness, in the last decades, has been concentrated on problems mainly related with high-level context modeling, representation, and aggregation. Consequently, security has been neglected in favor of other concerns even if that can become an obstacle to the diffusion of context-aware systems. However, it is worth stressing that some recent systems, like SOCAM and MobiSoC [Gu *et al.* 2005; Gupta *et al.* 2009], have introduced security mechanisms to support real deployments. Even in this case, by considering that security mechanisms (such as cryptography) may introduce high computational overhead, they should be carefully introduced depending on i) deployment scenarios; and ii) privacy of exchanged data. In fact, the availability of fixed infrastructure that could implement Public Key Infrastructure (PKI) deeply affects security mechanisms design. Moreover, context data have different level of privacy and only some of them require proper security mechanism. In conclusion, security in real context-aware scenarios is fundamental, but it should be tailored to the specific deployment scenario and exchanged data by trading-off security requirements and consumed resources, thus avoiding unnecessary and costly overheads.

5.2. Context Data Delivery Layer

As we can note from Table I, surveyed systems cover almost all the possible solutions that can be adopted in the dissemination and in the routing overlay module. Even if a multitude of solutions could be adopted for each module, it is possible to highlight some preferences for each particular deployment scenario. For instance, when a fixed infrastructure is adopted as network deployment, selection-based approaches with either centralized or flat distributed routing overlay are the most common ones; instead, when an ad-hoc network is used as network deployment, flooding-/gossip-based protocols with flat decentralized routing overlays are preferred due to their intrinsic properties.

In the following sections, we compare the main approaches that can be adopted respectively for the dissemination and for the routing overlay module, so to highlight main benefits and shortcomings. In addition, considering that these modules have a great impact on the final QoC perceived by users, we will spend some more space to detail how both these modules can affect context provisioning to mobile devices.

5.2.1. Dissemination

Several approaches adopt and implement the dissemination module to route context data to all the interested sinks. From a general viewpoint, we can highlight the following main considerations (Table II concisely reports them for the sake of clarity).

Sensor direct access approach now has been relegated only to the initial phase of context data production. In fact, the resulting strong coupling between context data production and consumption is against system scalability and context data availability; hence, this approach is unfeasible to address the context dissemination in real deployments. For instance, MobEyes uses this approach to acquire data, but then it uses a flooding-based approach to enable the final context data dissemination [Lee *et al.* 2009].

For flooding-based approaches, data flooding ensures high dependability, small dissemination times, and no routing information on intermediate nodes, but it introduces an overhead unsuitable for wide-area dissemination. In fact, if we consider both the large

number of sources and the production rates usually associated with real physical phenomena, it comes without saying that the context data dissemination could saturate all the available resources; that, in its turn, would lead to increasing routing delays and message droppings, thus affecting the final QoC perceived by mobile users. However, notwithstanding these issues, data flooding has been widely adopted because it is feasible in several deployments: first, since this schema does not require routing information on intermediate nodes, it suits well small networks composed by highly-mobile devices; second, it suits also the case of context data with limited and local dissemination scope. Subscription flooding is a viable solution to reduce the introduced network overhead; unfortunately, it induces scalability problems for memory since every node has to maintain all the subscriptions of remote nodes. Hence, as data flooding approaches, they do not scale well in wide-area wireless systems, and can easily lead to QoC degradation due to system saturation. In conclusion, in both approaches (data and subscription flooding), tight locality principles (either physical or logical) are necessary to limit flooding scope and avoid system saturation [Baldoni *et al.* 2009].

Selection-based approaches strive to increase system scalability by routing only the required context data. On the one hand, approaches with system wide visibility scope hinder system scalability: memory and bandwidth can suddenly become bottlenecks if every query reaches the whole system, thus triggering an unviable number of responses. On the other hand, selection-based approaches with limited visibility scopes can increase system scalability, but do not ensure the retrieval of all required context data. Hence, both selection-based approaches have some disadvantages about QoC: approaches with system wide visibility scopes are more prone to scalability problems, and this can lead to reduced QoC; instead, approaches with limited visibility scopes could be more feasible, but they do not ensure the delivery of all potentially matching data, thus possibly leading to reduced QoC due to inaccuracies into retrieved context.

Finally, gossip-based approaches, despite their probabilistic delivery guarantees, are emerging as good alternatives to the previous ones due to their good price/performance ratio; these systems are still (relatively) a few, because gossip-based strategies have been accepted more recently than other ones. Main benefits of these approaches are the small (even null) state on intermediate nodes and the good reliability, at the expense of a usually tolerable overhead in message dissemination. Unfortunately, they do not ensure context data dissemination, and have a run-time behavior strictly dependent on run-time conditions, such as node density and mobility. Consequently, QoC guarantees over the data dissemination are extremely difficult, if not impossible, to enforce when adopting these approaches.

From the analysis of Table I we can draw some considerations, most surveyed solutions adopt selection-based approaches over other ones, and this could seem against the technical soundness of other approaches. Above all, the dominance of selection over other approaches is justified by two main reasons: i) it easily addresses data dissemination with a good balance between performance and overhead in small scale deployments; and ii) the largest part of surveyed solutions assumes fixed wireless infrastructures as network deployment. Of course, the adopted network deployment affects the feasibility of the possible dissemination solutions detailed above: in the following we try to highlight the hidden relationships between dissemination and adopted network deployment.

First, if the network deployment adopts a fixed infrastructure with high performance

Table II. Dissemination module comparison

Category	Sub-category	Pros	Cons
Sensor Direct Access	-	• No state on mobile nodes • Low network overhead • Sink always receive interesting data • Dissemination reaches only interested nodes	• Strong coupling between sources/sinks
Flooding-based	**Data flooding**	• Low state on mobile nodes • Loose coupling between sources/sinks • Sink always receive interesting data	• High network overhead • Dissemination can reach not-interested nodes
	Subscription flooding	• Loose coupling between sources/sinks • Sink always receive interesting data • Dissemination reaches only interested nodes	• High state on mobile nodes • High network overhead
Selection-based	**System wide scope**	• Loose coupling between sources/sinks • Sink always receive interesting data • Dissemination reaches only interested nodes	• Medium state on mobile nodes • Medium network overhead
	Limited scope	• Loose coupling between sources/sinks • Dissemination reaches only interested nodes	• Sink could miss interesting data • Medium state on mobile nodes • Low network overhead
Gossip-based	**Context-oblivious**	• Low state on mobile nodes • Loose coupling between sources/sinks • Low network overhead	• Sink could miss interesting data • Dissemination can reach not-interested nodes
	Context-aware	• Low network overhead • Loose coupling between sources/sinks	• Medium state on mobile nodes • Sink could miss interesting data • Dissemination can reach not-interested nodes

servers, selection with either system wide or limited visibility scopes can address the problem of context data dissemination with delivery guarantee and affordable overhead. Flooding- and gossip-based approaches are obviously neglected by these scenarios because they introduce additional management problems, respectively scalability bottlenecks and lack of data delivery. Moreover, traditional pub/sub systems can be adapted with a minimal effort to realize selection-based approaches: for instance, Gaia adopts a standard pub/sub system, specifically the CORBA Notification Service, to perform context data dissemination [Ranganathan and Campbell 2003; Siegel 2000].

Instead, for network deployment with an ad-hoc infrastructure, the feasibility of selection-based approaches strictly depends on i) the visibility scope offered to context subscriptions; and ii) mobility. In fact, while the realization of selection-based approaches with system wide visibility and highly mobile nodes could result in unaffordable overhead, it could be feasible with limited visibility scopes (for instance, the one-/two-hops neighborhood) and rather static nodes. In the past, much research has been done to realize pub/sub systems in MANET environments with system wide visibility

scopes, i.e., subscriptions visible in the entire ad-hoc networks, but they showed very different run-time overhead depending on node mobility [Mottola *et al.* 2008]. In other words, if applied to perform system wide data dissemination in ad-hoc networks, these solutions can have very different behavior at run-time. Consequently, research on ad-hoc systems started to introduce flooding-/gossip-based approaches to increase dissemination guarantees. Flooding-based approaches ensure more predictable run-time behavior, even if they may lead to scalability issues. Gossip-based approaches are more scalable but also exhibit probabilistic delivery. Notwithstanding these problems, these two approaches are generally more naturally followed in ad-hoc environments than selection-based ones because of their properties [Baldoni *et al.* 2005; Rezende *et al.* 2008].

5.2.2. Routing Overlay

Once chosen a particular data dissemination approach, the routing overlay takes care of organizing the broker nodes into the final mobile system: depending on running context-aware services, these nodes will handle context data dissemination and routing toward the user mobile nodes.

From a general viewpoint, the routing overlay can adopt either a centralized architecture or a decentralized one (flat and hierarchical distributed architecture). Of course, similarly to what already happen for the dissemination module, the routing overlay approach depends on the adopted network deployment; at the same time, given a particular network deployment, some routing overlay approaches are more suitable according to the adopted dissemination approach. For the sake of clarity, Table III briefly summarizes the main benefits and shortcomings for each particular routing overlay approach.

Delving into finer details, the usage of a routing overlay built by a unique central broker in charge of routing context data is appealing due to the guarantees on context data availability. In fact, due to its centralized nature, all the mobile nodes can easily find the needed context data by contacting the unique broker. Unfortunately, it exhibits two main shortcomings: low scalability and low reliability; hence, it is usually suitable only for small-scale deployments, such as homes or buildings, where the whole context data distribution system has to serve only a limited and small number of sources and sinks. In addition, the feasibility of this approach is extremely biased by the adopted network

Table III. Routing overlay module comparison

Category	Sub-category	Pros	Cons
Centralized Architecture	-	• Context data access is always ensured • No management overhead for routing overlay maintenance	• Limited scalability and reliability • Locality principles difficult to apply
Decentralized Architecture	**Flat distributed architecture**	• Increased scalability and reliability • Locality principles easy to apply	• Context data access could not be always ensured • Additional management overhead for routing overlay maintenance
	Hierarchical distributed architecture	• Increased scalability and reliability • Locality principles easy to apply	• Context data access could not be always ensured • Additional management overhead for routing overlay maintenance

deployment. When fixed wireless infrastructures are available at the network deployment, this approach can be easily supported with a single physical server enacting as broker: in fact, among survey systems based on fixed wireless infrastructures, 10 out of 18 systems rely upon a centralized broker. In addition, let us remark that selection-based dissemination protocols with system wide query visibility scope take great advantage from this overlay organization: in fact, obtaining system wide query visibility is as simple as routing the context subscription to the unique central broker. Instead, when mobile ad-hoc networks are adopted at the network deployment, this approach is difficult to apply due to the lack of a static and always available physical node: in fact, in the surveyed systems, only Adaptive Traffic Lights relies upon this approach since it exploits each traffic light as a central and localized broker [Gorgorin *et al.* 2007].

Decentralized approaches, either flat or hierarchical, have two main benefits: i) they exploit multiple brokers for the sake of scalability and reliability; and ii) the routing overlay can be exploited to enforce locality principles on the context data dissemination. Unfortunately, decentralized routing overlays trade off system scalability and reliability with context availability: the usage of multiple brokers could introduce partial context views; hence, additional management protocols are required to build and maintain a consistent view over available context data. In addition, hierarchical architectures can be preferred to flat ones since i) they can better match the organization of context data that have strict physical locality principles; and ii) they can better drive context subscription routing into the distributed architecture; unfortunately, some hierarchical architectures, such as trees, can lead to unfair load distribution.

To conclude, it is worth stressing that the final routing overlay strictly depends on both choices of network deployment and dissemination module. On the one hand, ad-hoc networks claim for distributed routing overlays (both flat and hierarchical), while fixed wireless infrastructures can exploit all the routing overlay approaches. On the other hand, ad-hoc networks match flooding-/gossip-based approaches since they do not require the maintenance of heavy routing information, while fixed wireless infrastructures prefer selection-based approaches to avoid useless context data distribution and ensure context availability.

5.3. Run-time Adaptation Support

The run-time adaptation of the context data distribution function is still far from being embodied in most surveyed systems. In fact, only few of them perform some kind of run-time adaptation, on either the management or the delivery layer. However, although older context data distribution solutions do not usually adopt context-awareness, several seminal newer systems start considering it, so to have already demonstrated that it could greatly enhance system performance and scalability.

First of all, it is worth analyzing the main pros/cons associated with the different levels of control available at the service level (see Table IV). Moving from unaware to totally-aware adaptation approaches, we progressively shift the control from the data distribution function to the services, thus giving them the chances of finely tuning self-adaptation actions at run-time. This capability, if carefully handled, can lead to better run-time performance due to more specific and fine-grained reconfigurations that would be impossible otherwise.

By dealing into finer details, unaware approaches have been adopted to perform both context source selection and context data distribution reconfigurations. COSINE and

Table IV. Comparison of levels of control for Run-time Adaptation Support

Category	Pros	Cons
Unaware	• Low complexity for services • No conflicts between services	• High complexity of run-time adaptation support • Services cannot drive reconfigurations
Partially-aware	• Services can drive reconfigurations • Conflicts between services can be solved by the run-time adaptation support	• Medium complexity of run-time adaptation support • Medium complexity for services
Totally-aware	• Low complexity of run-time adaptation support	• Services must completely drive reconfigurations • High complexity for services • Potential conflicts between services-driven configurations

CMF perform automatic context source selection depending on the QoC ensured by context sources [Juszczyk *et al.* 2009; van Kranenburg *et al.* 2006]. Instead, HiBOp, MobEyes, MANIP, and CAR adapt context data distribution at run-time to maximize system performance, while avoiding service intervention [Boldrini *et al.* 2008; Lee *et al.* 2009; Macedo *et al.* 2009; Musolesi and Mascolo 2009]. Partially-aware adaptation approaches seem to well suit delivery layer reconfigurations. Driven by service profiles, Solar and SALES use computing context to dynamically adapt and avoid system saturation due to context data routing [Chen *et al.* 2008; Corradi *et al.* 2010b]. Also REDMAN adopts this approach to drive data distribution and avoid configurations clashing with service requirements [Bellavista *et al.* 2005]. Finally, totally-aware adaptation approaches are usually adopted to specify QoC constraints on received context data. Both MoCA and MiddleWhere adopt a totally-aware adaptation approach to let services apply QoC constraints on received context data [Endler and Da Rocha 2006; Ranganathan *et al.* 2004].

Even if some preferences between the control level and the context data distribution function layer to be reconfigured could be highlighted, in our opinion the final approach adopted by the run-time adaptation support mainly depends on: i) the reference context-aware scenarios; ii) the possibility to execute multiple services on the same context distribution function; and iii) the willingness of service developers in dealing with configuration details and defining proper distribution policies. First, distribution functions aimed to manage generic context information usually need service intervention to perform meaningful adaptations. For instance, QoC-based adaptations are possible only by agreeing on what is QoC and how to express QoC constraints. In fact, MiddleWhere distributes only localization data and introduces a particular QoC model based on sensor errors and localization up-to-dateness; hence, due to this reduced scope, it can introduce run-time aggregation algorithms that also automatically define the QoC parameters of the aggregated data [Ranganathan *et al.* 2004]. Instead, the usage of generic QoC constraints would not enable more intelligent adaptations of the aggregation component. Second, multiple services executed atop of the same context distribution function can require conflicting reconfigurations that, in their turn, require ad-hoc mechanisms to solve additional conflicts. In this case, partially-aware adaptation

approaches are well suitable since they enable run-time adaptations while leaving to the adaptation support the responsibility of final decisions. Hence, if required, partially-aware adaptation approaches can solve conflicts between different services, while this is extremely difficult to ensure in totally-aware adaptation approaches. Finally, the higher the control to the services, the more the technicalities the service developer needs to know to properly configure the context distribution function. Moving from unaware to totally-aware adaptation approaches, we reduce the complexity of the run-time adaptation support at the expense of the service level one. In addition, giving higher control to service developers tends to eventually make run-time performance less stable and predictable.

In conclusion, the surveyed solutions with run-time adaptation clearly show its importance toward a correct management of system run-time behavior. We believe that partially-aware adaptation approaches represent a good tradeoff between overhead and flexibility, because they are able to influence context distribution reconfigurations at the service level and, at the same time, to keep the final adaptation decisions into the context distribution function itself.

6. FUTURE RESEARCH DIRECTIONS AND OPEN ISSUES

To truly enable the development and the deployment of context-aware services in wide-scale wireless networks, several open issues have still to be faced and resolved. By focusing on the specific context data distribution function, we claim that many of the solutions required by distributed, scalable, and QoC-based context data distribution are still widely unexplored: here, with the same presentation order adopted both in Section 4 and 5, we detail the future research directions and open issues that we think still deserve additional research.

QoC Frameworks Definition – Several works have looked at the problem of QoC by also presenting new valid definitions and real case studies [Buchholz *et al.* 2003; Krause and Hochstatter 2005; Manzoor *et al.* 2008; Manzoor *et al.* 2009a; Manzoor *et al.* 2009b; Neisse *et al.* 2008]. However, the intrinsic ambiguity of the QoC concept has not promoted yet the definition of general and widely adopted QoC frameworks that could help context-aware service designers to understand QoC representation, sensing, and run-time usage [Krause and Hochstatter 2005]. Even if some general QoC parameters, such as context data up-to-dateness, can be easily applied to every context data, the definition of more complex QoC parameters for particular context aspects is still an open problem. In fact, data-specific parameters are difficult to standardize since strictly related to the context data aspect they refer to; however, even if widely adopted definitions do not exist, let us remark that some data-specific parameters are emerging as standard-de-facto for specific context data.

For instance, by focusing on localization as part of physical context, many solutions in literature, see MiddleWhere [Ranganathan *et al.* 2004], exploit a quality attribute usually defined as resolution. Resolution captures the difference between real and sensed localization (typically expressed in meters or centimeters): obviously, localization data errors strictly depend on the used localization technique; therefore, many systems agree upon using the maximum possible error guaranteed by the localization technology to quantify resolution.

Unfortunately, for many other context aspects (computing, physical, time, and user), a general agreement about data-specific parameters is difficult to reach. For instance, if we

consider co-localization as part of the user context, there is no common quality attribute to characterize the difference between the real and the sensed value. Different systems adopt different sensing strategies (e.g., based on APs associations, on received beacons between devices, …), and exploit different aggregation techniques (e.g., probabilistic, history-based, ...) to better estimate people co-localization in the same place. That makes impossible to agree on a unique attribute for resolution in physical localization.

To draw a conclusion, while general QoC parameters are already available in literature, much work is still required to define proper QoC data-specific parameters, at least for all main context aspects (computing, physical, time, and user context). Data-specific QoC parameters could also potentially enable even more complex and effective adaptation management operations. Hence, the standardization of one (or more) common QoC framework(s), including both general and data-specific parameters, is one of the most challenging open issues in this area.

Context Data Aggregation and Filtering – Focusing on the context data management layer, we think that two processing modules, namely aggregation and filtering, still deserve additional research.

Starting with aggregation techniques, current researches should be directed along two main directions. On the one hand, aggregation algorithms should be easy to implement and efficient, and should have affordable space/time requirements to execute. Let us note that some techniques can result in NP complexity, i.e., problems whose solution time increases intolerably as the size of the problem grows [Arora and Barak 2009]. Consequently, in this direction, further studies need to optimize aggregation algorithms. On the other hand, aggregation techniques should also consider QoC data parameters to i) minimize the introduction of errors in aggregated context; and ii) express in a quantitative way the QoC of the derived context. Strictly related with above research issue, further studies should aim at defining proper algorithms useful to combine context data with different QoC characteristics.

Focusing on filtering techniques, they affect data transmission by applying either time- or change-based selections. These techniques foster system scalability by suppressing useless transmissions but, unfortunately, affect also perceived QoC. In fact, by limiting exchanged data, services have more chance to use stale and even invalid data, thus undermining reasoning and adaptation effectiveness. However, according to the specific scenario and granted QoC, the usage of older data is feasible and significant as well. First, if the service does not require the most up-to-date context data, we can use those techniques to enhance scalability by respecting service requests at the same time. Second, if the context data assume continuous values, we can use time-/change-based filtering with history-based integration techniques to obtain an estimation of the current value. As an example, CORTEX already provides this kind of facility, but the associated literature does not clarify how history is used [Duran-Limon *et al.* 2003]. To draw a conclusion, given the aforementioned problems, further work is needed to study the relationship between QoC degradation and the cost of filtering techniques to clarify better their interactions and mutual influences.

Adaptive Context Data Dissemination – Even if all the surveyed solutions adopt a context data dissemination component belonging to one specific category presented in Figure 3, at the expense of a more complex implementation, we remark that hybrid solutions that use different dissemination algorithms together, can lead to better performance. For instance, fixed infrastructure could use selection-based and flooding-

/gossip-based algorithms at the same time: while the former approach ensures context access, the latter ones can replicate data in a probabilistic manner, so to reduce context access time and to increase reliability. Similarly, mobile infrastructure could use gossip-based protocols to enable context visibility in far away areas, and can use selection-based protocols with tight physical constraints (for instance, in the one-/two-hops neighborhood) to disseminate only required data.

Above all, we stress that flooding- and gossip-based algorithms seem to be very promising. Even if flooding-based schemas have scalability issues, they can be suitable if flooding is constrained by locality principles; especially in small-scale distribution, data flooding algorithms address distribution with high availability, null state on involved nodes, and small response times. Instead, gossip-based approaches improve scalability by reducing the delivery guarantees. Apart from context-oblivious approaches (that have many chance of wasting resources in a useless manner), the control of the probabilistic nature of context-aware gossip-based protocols represents an interesting research direction. First, HiBOp and Habit demonstrate that user social state and relationships represent good hints to drive gossip decisions [Boldrini *et al.* 2008; Mashhadi *et al.* 2009]. New researches should investigate how user context social relationships can be inferred and dynamically maintained in a lightweight manner. Second, CAR demonstrates that the utilization of low-level time context information, e.g., inter-contact times and frequencies of contacts, produces good solutions as well [Musolesi and Mascolo 2009]. Another interesting research direction goes in the sense of extending DTN context-aware protocols to context data dissemination scenarios [Boldrini *et al.* 2008].

Toward the main goal of adopting and adapting different context data dissemination algorithms at run-time, additional researches should be directed toward the definition of meaningful attributes useful to i) drive the selection of the proper context data dissemination algorithms at the run-time adaptation support; and ii) adapt their own run-time behavior to maximize system performance. Of course, these attributes represent context aspects of the system that have to be modeled and processed; for instance, in [Taherkordi *et al.* 2008], authors model the context attributes of a wireless sensor network, and use them to adapt the processing of the context data.

To conclude, to the best of our knowledge, no solution is able to adopt and to switch different context data dissemination algorithms at run-time to maximize system performance depending on the current status. At the same time, a particular dissemination algorithm could be adapted to current run-time conditions; these conditions should also consider current QoC requirements to self-adapt dissemination structures and suppress useless context data transmissions.

7. CONCLUSIONS

This paper presents a survey of the issues and the challenges of context data distribution: to better understand the current state-of-the-art, we provide a new logical architecture and taxonomy for context data distribution in mobile ubiquitous environments. In addition, we report an in-depth analysis of a number of solutions that span different research areas and cover all our taxonomy directions by showing that all those scenarios need efficient and effective context data distribution.

From our analysis, we distil the subsequent main findings useful to enable context data distribution in wide-area real deployments. First, context data distribution should take

into account node requests and QoC requirements to reduce management overhead. Second, context data distribution requires adaptive and crosscutting solutions able to orchestrate the principal internal facilities according to specific management goals. Third, informed context data distribution can benefit from their increased context-awareness to further enhance system scalability and reliability.

Moreover, we have identified and stressed some open issues. In particular, an important part of the context data distribution research, namely the development of informed context data distribution solutions able to use traversing context data, QoC requirements, and run-time conditions to self-adapt data processing (aggregation and filtering) and delivery is still not addressed and not solved. This is for sure a very fruitful future research area.

Along the same line, we foresee another wider and still unexplored future research field: the design and development of context data distribution able *to self-adapt autonomously by dynamically combining most suitable data distribution methods and techniques at different (above all, processing and delivery) mechanisms depending on current management conditions*. The lack of these solutions stems from the complexity of the problem, but we argue we are very close to an effective realization and appearance of self-adaptive systems. They will be able to suitably orchestrate all different context data distribution facilities with no user intervention at run-time and that will be a turning point in context data distribution research.

REFERENCES

ADAMS, B., PHUNG, D., AND VENKATESH, S. 2008. Sensing and using social context. *ACM Transactions on Multimedia Computing, Communications and Applications*. 5, 2 (November 2008), Article 11, 1-27.

ARORA, S. AND BARAK, B. 2009. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009, 38-61.

BALDAUF, M., DUSTDAR, S., AND ROSENBERG, F. 2007. A Survey on Context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*. 2, 4 (2007), 263-277.

BALDONI, R., BERALDI, R., CUGOLA, G., MIGLIAVACCA, M., AND QUERZONI, L. 2005. Structure-less Content-Based Routing in Mobile Ad Hoc Networks. In *Proceeeding of the IEEE International Conference on Pervasive Services (ICPS'05)*, 37-46.

BALDONI, R., QUERZONI, L., TARKOMA, S., AND VIRGILLITO, A. 2009. Distributed Event Routing in Publish/Subsribe Systems. In *Middleware for Network Eccentric and Mobile Applications*, B. Garbinato, H. Miranda, and L. Rodrigues, Eds. Springer Press, 2009, Chapter 10, 219-244.

BARTOLINI, A., RUGGIERO, M., AND BENINI, L. 2009. Visual Quality Analysis For Dynamic Backlight Scaling In LCD Systems. In *Proceedings of the Design, Automation and Test in Europe (DATE'09)*, 1428-1433.

BELLAVISTA, P., CORRADI, A., MONTANARI, R., AND STEFANELLI, C. 2003. Context-aware Middleware for Resource Management in the Wireless Internet. *IEEE Transactions on Software Engineering*. 29, 12 (December 2003), 1086-1099.

BELLAVISTA, P., CORRADI, A., AND MAGISTRETTI, E. 2005. REDMAN: An optimistic replication middleware for read-only resources in dense MANETs. *Elsevier Pervasive and Mobile Computing.* 1, 3 (September 2005), 279-310.

BETTINI, C., BRDICZKA, O., HENRICKSEN, K., INDULSKA, J., NICKLAS, D., RANGANATHAN, A., AND RIBONI, D. 2010. A survey of context modelling and reasoning techniques. *Elsevier Pervasive and Mobile Computing*. 6, 2 (April 2010), 161-180.

BLOOM, B. H. 1970. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*. 13, 7 (July 1970), 422-426.

BOLCHINI, C., CURINO, C. A., QUINTARELLI, E., SCHREIBER, F. A., AND TANCA, L. 2007. A data-oriented survey of context models. *SIGMOD Record*. 36, 4 (December 2007), 19-26.

BOLCHINI, C., CURINO, C. A., ORSI, G., QUINTARELLI, E., ROSSATO, R., SCHREIBER, F. A., AND TANCA, L. 2009. And What Can Context do for Data?. *Communications of the ACM*. 52, 11 (November 2009), 136-140.

BOLDRINI, C., CONTI, M., AND PASSARELLA, A. 2008. Exploiting users' social relations to forward data in opportunistic networks: The HiBOp solution. *Elsevier Pervasive and Mobile Computing*. 4, 5 (October

2008), 633-657.

BRODER, A., AND MITZENMACHER, M. 2005. Network Applications of Bloom Filters: A Survey. *Internet Mathematics*. 1, 4 (2005), 485-509.

BUCHHOLZ, T., KÜPPER, A., AND SCHIFFERS, M. 2003. Quality of Context: What It Is and Why We Need It. In *Proceedings of the Workshop HP OpenView*, 1-14.

CAPRA, L., EMMERICH, W., AND MASCOLO, C. 2003. CARISMA: context-aware reflective middleware system for mobile applications. *IEEE Transactions on Software Engineering.* 29, 10 (2003), 929-945.

CARTIGNY, A., AND SIMPLOT, D. 2003. Borden Node Retransmission Based Probabilistic Broadcast Protocols in Ad-Hoc Networks, In *Proceedings of Telecommunication System*, 189-204.

CAVENEY, D. 2010. Cooperative Vehicular Safety Applications. *In VANET Vehicular Applications and Inter-Networking Technologies*, H. Hartenstein, and K. Laberteaux, Eds. John Wiley & Sons, 2010, Chapter 2, 21-48.

CERI, S., DANIEL, F., MATERA, M., AND FACCA, M. 2007. Model-driven development of context-aware Web applications. ACM Transactions on Internet Technologies. 7, 1 (February 2007), Article 2, 1-33.

CHAN, A. T. S., AND CHUANG, S. N. 2003. Mobipads: A reflective middleware for context-aware mobile computing. *IEEE Transactions on Software Engineering*. 29, 12 (2003), 1072-1085.

CHANG, H., SHIN, S., AND CHUNG, C. 2007. Context Life Cycle Management Scheme in Ubiquitous Computing Environments. In *Proceedings of the International Conference on Mobile Data Management (MDM'07)*, 315-319.

CHAPPELL, D. A., AND MONSON-HAEFEL, R. 2000. *Java Message Service*. O'Reilly Media. December 2000

CHEN, G., AND KOTZ, D. 2000. A Survey of Context-Aware Mobile Computing Research. Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College.

CHEN, G., LI, M., AND KOTZ, D. 2008. Data-centric middleware for context-aware pervasive computing. *Elsevier Pervasive and Mobile Computing*. 4, 2 (April 2008), 216-253.

CHEN, H., FININ, T., AND JOSHI, A. 2003. An Intelligent Broker for Context-Aware Systems, In *Adjunct Proceedings of Ubicomp'03*, 183-184.

CHETAN, S., AL-MUHTADI, J., CAMPBELL, R., AND MICKUNAS, M.D. 2005. Mobile Gaia: a middleware for ad-hoc pervasive computing. In *Proceedings of the IEEE Consumer Communications and Networking Conference (CCNC'05)*, 223-228.

CHO, K., HWANG, I., KANG, S., KIM, B., LEE, J., LEE, S., PARK, S., SONG, J., AND RHEE, Y. 2008. HiCon: a hierarchical context monitoring and composition framework for next-generation context-aware services. *IEEE Network*. 22, 4 (July-August 2008), 34-42.

CHOW, C.-Y., VA LEONG, H., AND CHAN, A. T.S. 2007. GroCoca: Group-Based Peer-To-Peer Cooperative Caching In Mobile Environment. *IEEE Journal on Selected Areas in Communications*. 25, 1 (January 2007), 179-191.

CONAN, D., ROUVOY, R., AND SEINTURIER., L. 2007. Scalable processing of context information with COSMOS. In *Proceedings of the 7th IFIP WG 6.1 International Conference on Distributed applications and interoperable systems (DAIS'07)*, Springer Press, 210-224.

CONTI, M., AND GIORDANO, S. 2007a. Multihop Ad Hoc Networking: The Theory. *IEEE Communications Magazine*. 45, 4 (April 2007), 78-86.

CONTI, M., AND GIORDANO, S. 2007b. Multihop Ad Hoc Networking: The Reality. *IEEE Communications Magazine*. 45, 4 (April 2007), 88-95.

CORRADI, A., FANELLI, M., AND FOSCHINI, L. 2010a. Towards Adaptive and Scalable Context-Aware Middleware. *Invited paper in the IGI International Journal of Adaptive, Resilient and Autonomic Systems (IJARAS)*, 1, 1 (January 2010), 58-74.

CORRADI, A., FANELLI, M., AND FOSCHINI, L. 2010b. Adaptive Context Data Distribution with Guaranteed Quality for Mobile Environments. In *Proceedings of the IEEE International Symposium on Wireless Pervasive Computing (ISWPC'10)*, 373-380.

COSTA, P., MOTTOLA, L., MURPHY, A. L., AND PICCO, G. P. 2009. Tuple Space Middleware for Wireless Networks. In *Middleware for Network Eccentric and Mobile Applications*, B. Garbinato, H. Miranda, and L. Rodrigues, Eds., Springer Press, 2009, Chapter 11, 245-264.

CUGOLA, G., AND DI NITTO, E. 2001.Using a Publish/Subscribe Middleware to Support Mobile Computing. In *Proceedings of the Workshop on Middleware for Mobile Computing (MMC'01) within Middleware'01*, 1-5.

CUGOLA, G., DI NITTO, E., AND FUGGETTA, A. 2001. The JEDI event-based infrastructure and its application to the development of the OPSS WFMS. *IEEE Transactions on Software Engineering*. 27, 9 (September 2001), 827-850.

DEBATY, P., GODDI, P., AND VORBAU, A. 2005. Integrating the physical world with the web to enable context-enhanced services. *Springer Mobile Networks and Applications*. 10, 4 (August 2005), 385–394.

DERHAB, A., AND BADACHE, N. 2009. Data replication protocols for mobile ad-hoc networks: a survey and taxonomy. *IEEE Communications Surveys & Tutorials*. 11, 2 (Second Quarter 2009), 35-51.

DEY, A. K., AND ABOWD, G. D. 2000a. Towards a Better Understanding of Context and Context-Awareness. In *Proceedings of the Workshop on The What, Who, Where, When, and How of Context-Awareness within CHI'00*, 1-12.

DEY, A. K., AND ABOWD, G. D. 2000b. The Context Toolkit: Aiding the Development of Context-Aware Applications, In *Proceedings of the Workshop on Software Engineering for Wearable and Pervasive Computing*, 1-3.

DRABKIN, V., FRIEDMAN, R., KLIOT, G., AND SEGAL, M. 2007. RAPID: Reliable Probabilistic Dissemination in Wireless Ad-Hoc Networks. In *Proceedings of the 26th IEEE Symposium on Reliable Distributed Systems*, 13-22.

DURAN-LIMON, H. A., BLAIR, G. S., FRIDAY, A., GRACE, P., SAMARTZIDIS, G., SIVAHARAN, T., AND WU, M. 2003. Context-aware Middleware for Pervasive and Ad-Hoc Environments. Computing Department, Lancaster University, Tech. Rep.

ENDLER, M., AND DA ROCHA, R.C.A. 2006. Supporting Context-Aware Applications: Scenarios, Models and Architecture, Tech. Rep. of Computer Science, no. 12/06, University of Rio De Janeiro.

EUGSTER, P. T., FELBER, P. A., GUERRAOUI, R., AND KERMARREC, A.-M. 2003. The many facets of publish/subscribe. *ACM Computing Surveys*. 35, 2 (June 2003), 114-131.

EUGSTER, P., GARBINATO, B., AND HOLZER, A. 2008. Design and Implementation of the Pervaho Middleware for Mobile Context-Aware Applications. In *Proceedings of the International MCETECH Conference on e-Technologies*, 125-135.

EUGSTER, P., GARBINATO, B., AND HOLZER, A. 2009. Middleware Support for Context-Aware Applications. *In Middleware for Network Eccentric and Mobile Applications*, B. Garbinato, H. Miranda, and L. Rodrigues, Eds. Springer Press, 2009, Chapter 14, 305-322.

FAHY, P., AND CLARKE, S. 2004. CASS – Middleware for Mobile Context-Aware Applications. In *Proceedings of the Workshop on Context Awareness within MobiSys'04*, 1-6.

FALL, K. 2003. A delay-tolerant network architecture for challenged internets. In *Proceedings of the conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM'03)*, 27-34.

FRIEDMAN, R., KERMARREC, A.-M., MIRANDA, H., AND RODRIGUES, L. 2009. Gossip-Based Dissemination. *In Middleware for Network Eccentric and Mobile Applications*, B. Garbinato, H. Miranda, and L. Rodrigues, Eds. Springer Press, 2009, Chapter 8, 169-190.

FRIEDMAN, R., GAVIDIA, D., RODRIGUES, L., VIANA, A. C., AND VOULGARIS, S. 2007. Gossiping on MANETs: the beauty and the beast, *SIGOPS Operating Systems Review*, 41, 5 (Oct. 2007), 67-74.

GADDAH, A., AND KUNZ, T. 2003. A Survey of Middleware Paradigms for Mobile Computing. Technical Report SCE-03-16, Dept. of Systems and Computing Engineering, Carleton University.

GERSHENFELD, N., KRIKORIAN, R., AND COHEN, D. 2004. The Internet of Things. *Scientific American Magazine*. 10, 6 (October 2004), 76-81.

GORGORIN, C., GRADINESCU, V., DIACONESCU, R., CRISTEA, V., AND IFTODE, L. 2007. Adaptive Traffic Lights using Car-to-Car Communication. In *Proceedings of the IEEE Vehicular Technology Conference (VTC'07-Spring)*, 21-25.

GU, T., PUNG, H. K., AND ZHANG, D. Q. 2005. A service-oriented middleware for building context-aware services. *Journal of Network and Computer Applications*, 28, 1 (January 2005), 1-18.

GUPTA, A., PAUL, S., JONES, Q., AND BORCEA, C. 2007. Automatic identification of informal social groups and places for geo-social recommendations. *International Journal of Mobile Network Design and Innovation (IJMNDI)*. 2, 3/4 (2007), 159-171.

GUPTA, A., KALRA, A., BOSTON, D., AND BORCEA, C. 2009. MobiSoC: a middleware for mobile social computing applications. *Mobile Networks and Applications*, 14, 1 (February 2009), 35-52.

HAAS, Z., HALPERN, J., AND LI, L. 2002. Gossip-based Ad Hoc Routing. In *Proceedings of the 21st Joint Conference of the IEEE Computer and Communication Societies (INFOCOM'02)*, 1707-1716.

HARA, T. 2001. Effective replica allocation in ad hoc networks for improving data accessibility", In *Proceedings of the 20th Joint Conference of the IEEE Computer and Communication Societies (INFOCOM'01)*, 1568–1576.

HENGARTNER, U., AND STEENKISTE, P. 2005. Access control to people location information. *ACM Transactions on Information and System Security (TISSEC)*. 8, 4 (2005), 424-456.

HENRICKSEN, K., INDULSKA, J., MCFADDEN, T., AND BALASUBRAMANIAM, S. 2005. Middleware for Distributed Context-Aware Systems. In *Proceedings of the International Symposium on Distributed Objects and Applications (DOA'05)*, 846-863.

HIGHTOWER, J., AND BORIELLO, G. 2001. A Survey and Taxonomy of Location Systems for Ubiquitous Computing. *IEEE Computer*, 34, 8 (August 2001), 57-66.

HOFER, T., SCHWINGER, W., PICHLER, M., LEONHARTSBERGER, G., ALTMANN, J., AND RETSCHITZEGGER, W. 2003. Context-Awareness on Mobile Devices - the Hydrogen Approach, In *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, 10-19.

IFTODE, L., SMALDONE, S., GERLA, M., AND MISENER, J. 2008. Active Highways. In *Proceedings of the IEEE Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'08)*, 1-5.

JAIN, S., FALL, K., AND PATRA, R. 2004. Routing in a delay tolerant network. In *Proceedings of the International Conference on applications, technologies, architectures, and protocols for computer communications (SIGCOMM'04)*, 145-158.

JONES, Q., AND GRANDHI, S. A. 2005. P3 Systems: Putting the place back into social networks. *IEEE Internet Computing*. 9, 5 (Sept.-Oct. 2005), 38-46.

JULIEN, C., AND ROMAN, G.-C. 2006. EgoSpaces: facilitating rapid development of context-aware mobile applications. *IEEE Transactions on Software Engineering*, 32, 5 (May 2006), 281-298.

JUSZCZYK, L., PSAIER, H., MANZOOR, A., AND DUSTDAR, S. 2009. Adaptive Query Routing on Distributed Context - The COSINE Framework. In *Proceedings of the tenth International Conference on Mobile Data Management: Systems, Services and Middleware (MDM '09)*, 588-593.

KERMARREC, A.-M., AND VAN STEEN, M. 2007. Gossiping in distributed systems. *ACM SIGOPS Operating Systems Review*. 41, 5 (October 2007), 2-7.

KIM, J.-M., SON, C.-H., LEE, C.-H., AND HA, Y.-H. 2006. Illuminant Adaptive Color Reproduction Based on Lightness Adaptation and Flare for Mobile Phone In *Proceedings of the IEEE International Conference on Image Processing*, 1513-1516.

KNAPPMEYER, M., BAKER, N., LIAQUAT, S., AND TÖNJES, R. 2009. A context provisioning framework to support pervasive and ubiquitous applications. In *Proceedings of the 4th European conference on Smart sensing and context (EuroSSC'09)*, 93-106.

KRAUSE, M., AND HOCHSTATTER, I. 2005. Challenges in Modelling and Using Quality of Context (QoC). In *Proceedings of the International Conference on Mobility Aware Technologies and Applications (MATA'05)*, Springer Press, 324-333.

KJÆR, K. E. 2007. A survey of context-aware middleware. In *Proceedings of the 25th conference on IASTED International Multi-Conference: Software Engineering*, 148-155.

LEE, U., MAGISTRETTI, E., GERLA, M., BELLAVISTA, P., LIÓ, P., AND LEE, K.-W. 2009. Bio-inspired multi-agent data harvesting in a proactive urban monitoring environment. *Elsevier Ad Hoc Networks*. 7, 4 (June 2009), 725-741.

LI, F., SANJIN, S., AND SCHAHRAM, S. 2010. COPAL: An adaptive approach to context provisioning. In *Proceedings of the IEEE 6th International Conference on Wireless and Mobile Computing, Networking, and Communications (WiMob'10),* 286-293.

LOCHERT, C., SCHEUERMANN, B., AND MAUVE, M. 2010. Information Dissemination in VANETs. *In VANET Vehicular Applications and Inter-Networking Technologies*, H. Hartenstein, and K. Laberteaux, Eds. John Wiley & Sons, 2010, Chapter 3, 49-80.

MACEDO, D. F., DOS SANTOS, A. L., NOGUEIRA, J. M. S., AND PUJOLLE, G. 2009. A Distributed Information Repository for Autonomic Context-Aware MANETs. *IEEE Transactions on Network and Service Management*. 6, 1 (March 2009).

MAHAMBRE, S.P., KUMAR, M., AND BELLUR, U. 2007. A Taxonomy of QoS-Aware, Adaptive Event-Dissemination Middleware. *IEEE Internet Computing*. 11, 4 (July-August 2007), 35-44.

MANZOOR, A., TRUONG, H.-L., AND DUSTDAR, S. 2008. On the Evaluation of Quality of Context. In *Proceedings of the 3rd European Conference on Smart Sensing and Context*, 140-153.

MANZOOR, A., TRUONG, H.-L., AND DUSTDAR, S. 2009a. Using quality of context to resolve conflicts in context-aware systems. In *Proceedings of the 1st International Conference on Quality of Context (QuaCon'09)*, 144-155.

MANZOOR, A., TRUONG, H.-L., AND DUSTDAR, S. 2009b. Quality Aware Context Information Aggregation System for Pervasive Environments, In *Proceedings of the 1st International Conference on Advanced Information Networking and Applications Workshops*, 266-271

MASHHADI, A. J., BEN MOKHTAR, S., AND CAPRA, L. 2009. Habit: Leveraging Human Mobility and Social Network for Efficient Content Dissemination in MANETs, In *Proceedings of the 10th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM'09)*, 1-6.

MIRANDA, H., LEGGIO, S., RODRIGUEZ, L., AND RAATIKAINEN, K. 2009. An Algorithm for Dissemination and Retrieval of Information in Wireless Ad Hoc Networks. *Concurrency and Computation: Practice & Experience*. John Wiley and Sons. 21, 7 (May 2009), 889-904.

MOTTOLA, L., CUGOLA, G., AND PICCO, G. P. 2008. A Self-Repairing Tree Topology Enabling Content-Based Routing in Mobile Ad Hoc Networks. *IEEE Transactions on Mobile Computing*. 7, 8 (August 2008), 946-960.

MUHL, G., ULBRICH, A., AND HERRMAN, K. 2004. Disseminating information to mobile clients using publish-subscribe. *IEEE Internet Computing*. 8, 3 (May-June 2004), 46- 53.

MUSOLESI, M., AND MASCOLO, C. 2009. CAR: Context-aware Adaptive Routing for Delay Tolerant Mobile Networks. *IEEE Transactions on Mobile Computing*. 8, 2 (February 2009), 246-260.

NEISSE, R., WEGDAM, M., AND VAN SINDEREN, M. 2008. Trustworthiness and Quality of Context Information.

In *Proceedings of the 9th International Conference for Young Computer Scientists (ICYCS'08)*, 1925-1931.

PADMANABHAN, P., GRUENWALD, L., VALLUR, A., AND ATIQUZZAMAN, M. 2008. A survey of data replication techniques for mobile ad hoc network databases. *The VLDB Journal*. 17, 5, (August 2008), 1143-1164.

PELUSI, L., PASSARELLA, A., AND CONTI, M. 2006. Opportunistic networking: Data forwarding in disconnected mobile ad hoc networks. *IEEE Communications Magazine*. 44, 11 (November 2006), 134-141.

POWERS, S. 2003. *Practical RDF*. O'Reilly Media, 2003.

RANGANATHAN, A., AND CAMPBELL, R. H. 2003. A middleware for context-aware agents in ubiquitous computing environments. In *Proceedings of the ACM/IFIP/USENIX International Conference on Middleware (Middleware'03)*, 143-161.

RANGANATHAN, A., MUHTADI, J. A., CHETAN, S., CAMPBELL, R., AND MICKUNAS, M. D. 2004. MiddleWhere: a middleware for location awareness in ubiquitous computing applications. In *Proceedings of the 5th ACM/IFIP/USENIX International Conference on Middleware (Middleware'05)*, 397-416.

REZENDE, C., BOUKERCHE, A., ROCHA, B., AND LOUREIRO, A. 2008. Understanding and using mobility on Publish/Subscribe based architectures for MANETs. In *Proceedings of the IEEE Conference on Local Computer Networks (LCN'08)*, 813-820.

RIVA, O., NADEEM, T., BORCEA, C., AND IFTODE, L. 2007. Context-Aware Migratory Services in Ad Hoc Networks. *IEEE Transactions on Mobile Computing*. 6, 12 (December 2007), 1313-1328.

SACRAMENTO, V., ET AL., 2004. MoCA: A middleware for developing collaborative applications for mobile users. *IEEE Distributed Systems Online*. 5, 10 (October 2004).

SASSON, Y., CAVIN, D., AND SCHIPER, A. 2003. Probabilistic Broadcast for Flooding in Wireless Mobile Ad hoc Networks. *In Proc. of the IEEE Wireless Communications and Networking Conference (WCNC'03)*, 1124-1130.

SCHILIT, B. N., ADAMS, N., AND WANT, R. 1994. Context-Aware Computing Applications. In *Proceedings of the Workshop on Mobile Computing Systems and Applications (WMCSA'94)*, 85-90.

SENART, A., BOUROCHE, M., CAHILL, V., AND WEBER, S. 2009. Vehicular Networks and Applications. In *Middleware for Network Eccentric and Mobile Applications*, B. Garbinato, H. Miranda, and L. Rodrigues, Eds. Springer Press, 2009, Chapter 17, 369-382.

SHAHEEN, A., AND GRUENWALD, L. 2010. Group based replication for mobile ad hoc databases (GBRMAD), Technical Report available at http://www.cs.ou.edu/~database/documents/shaheen.pdf, University of Oklahoma (accessed September 2010).

SICHITIU, M. I., AND KIHL, M. 2008. Inter-vehicle communication systems: A survey. *IEEE Communications Surveys & Tutorials*. 10, 2 (2008), 88-105.

SIEGEL, J., 2000. *CORBA 3 Fundamentals and Programming*. John Wiley & Sons. 2nd edition, April 2000.

SOUSA, J. P., AND GARLAN, D. 2002. Aura: an Architectural Framework for User Mobility in Ubiquitous Computing Environments. In *Proceedings of the IFIP 17th World Computer Congress*, 29-43.

STRANG, T., AND POPIEN, C. L. 2004. A context modeling survey, In *Proceedings of the Workshop on Advanced Context Modelling, Reasoning and Management within UbiComp'04*, 1-8.

SUTTON, P., ARKINS, R., AND SEGALL, B. 2001. Supporting Disconnectedness-Transparent Information Delivery for Mobile and Invisible Computing. In *Proceedings of the IEEE International Symposium on Cluster Computing and the Grid (CCGrid'01)*, 277-285.

SWEITZER, J. W., THOMPSON, P., WESTERINEN, A. R., WILLIAMS, R. C., AND BUMPUS, W. 1999. Common Information Model: Implementing the Object Model for Enterprise Management. John Wiley & Sons, 1999.

TAHERKORDI, A., ROUVOY, R., LE-TRUNG, Q., AND ELIASSEN, F. 2008. A self-adaptive context processing framework for wireless sensor networks. In *Proceedings of the 3rd International Workshop on Middleware for sensor networks (MidSens'08)*, 7-12.

TANENBAUM, A. S. 2002. *Computer Networks (4th Edition)*, Prentice Hall, 2002, 397-417.

TILAK, S., MURPHY, A., AND HEINZELMAN, W. 2003. Non-uniform Information Dissemination for Sensor Networks. In *Proceedings of the 11th IEEE Conference on Network Protocols (ICNP'03)*, 295-304.

VAN KRANENBURG, H., BARGH, M.S., IACOB, S., AND PEDDEMORS, A., 2006. A context management framework for supporting context-aware distributed applications. *IEEE Communications Magazine*, 44, 8 (Aug. 2006), 67-74.

VAN SINDEREN, M.J., VAN HALTEREN, A.T, WEGDAM, M., MEEUWISSEN, H.B., AND EERTINK, E.H. 2006. Supporting context-aware mobile applications: an infrastructure approach. *IEEE Communications Magazine*, 44, 9 (Sept. 2006), 96-104.

YAU, S. S., HUANG, D., GONG, H., AND SETH, S. 2004. Development and Runtime Support for Situation-Aware Application Software in Ubiquitous Computing Environments. In *Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04)*, 452-457.

YIN, L., AND CAO, G. 2006. Supporting Cooperative Caching In Ad Hoc Networks. *IEEE Transactions on Mobile Computing*. 5, 1 (January 2006), 77-89.

ZHANG, Z. 2006. Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: Overview and challenges. *IEEE Communications Surveys & Tutorials*. 8, 1 (First Quarter 2006), 24-37.

ZIMMERMANN, A., LORENZ, A., AND OPPERMANN, R. 2007. An operational definition of context. In *Proceedings of the 6th International and Interdisciplinary Conference on Modeling and using Context (CONTEXT07)*, Springer Press, 558-571.