Multimed Tools Appl (2016) 75:13349–13374
DOI 10.1007/s11042-015-2846-4

WhatsUpNow: urban social application with real-time peer-to-peer ambient and sensory data exchanges

Marcel Karam [1] · **Haidar Safa** [1] · **Mehedi Masud** [2]

Received: 20 September 2014 / Revised: 15 July 2015 / Accepted: 30 July 2015 /
Published online: 12 August 2015

Abstract Having the ability to generate, share, and use ambient multimedia and sensory data in real-time using both traditional sensors as well as non-traditional ones (such as smart device users) is a pioneering practice that requires specialized network capabilities and visualization metaphors. The network must support both service discovery and cache sharing to allow users to generate real-time sensory data, upload them or share them with end-users searching for the same data. Visualization and coloring schemes must support both streaming and stored sensory data to allow users to interact with either recent or up-to-the-minute ambient sensory data on either smart devices or the server. This article describes the design and reports on the simulation performance of a social network application that allows a group of users on an ad-hoc network to share real-time multimedia and ambient data with respect to venues of potential interest. At the graphical interface level, we present an intuitive interface that allows users to capture and share, often with a single hand, an array of sensory data comfortably and efficiently using touch screen smart devices. At the network level, we describe an architectural model that is supported by a specific design strategy for service discovery and caching to facilitate data sharing. The performance of the architectural model is then evaluated to show that it can efficiently handle bulk of sensory data, when accessed using smart devices in a peer-to-peer environment.

Keywords Multimedia and sensory ambient data · Visual ambient representation · Caching · Service discovery · Social urban applications

✉ Marcel Karam
mk62@aub.edu.lb

Haidar Safa
hs33@aub.edu.lb

Mehedi Masud
mmasud@tu.edu.sa

[1] Department of Computer Science, American University of Beirut, Beirut, Lebanon

[2] Department of Computer Science, Taif University, Taif, Saudi Arabia

1 Introduction

There is something quite catchy about being able to receive instant feedback about trendy places that offer day/night recreational activities in big cosmopolitan cities such as Montreal or New York. Current feedback venues come in the form of ratings through social networking services such as those provided by *foursquare .com* which relies statistically on a five-star scale system [17]. This scale may satisfy potential clients who are content with static historical data only. For astute and tech savvy clients the scale lacks most importantly dynamic up-to-the-minute information on the target place of visit. This information includes data generated from smart devices with built-in sensors such as barometer, altimeter, temperature, location, and hygrometer. The information also includes ambient data such as: light, imagery, and sound artifacts that can be captured and optionally annotated by the user. Annotations [16] can include augmenting ambient multimedia [33] with impressions that people feel when visiting a recreational place. They can also be used to keep memories about visited places and more interestingly, relate to the feelings that people have, even when they do not know each other.

Ambient and sensory data are not simply artifacts that capture reality; they are artifacts that have an emotional impact on potential visitors that look at them. For example, one can be emotionally drawn to a crowd of a certain stature in a certain recreational place by looking at pictures or videos depicting the crowd. When the recreational visitors have this kind of data available to them, they may override decisions reached exclusively based on the five-star rating system, simply because it is coming almost instantaneously from other recreational visitors in the field. This method of peer-to-peer real-time sharing of ambient-based annotated multimedia artifacts and device-generated sensory data has the potential of creating an emotionally-rich form of social interaction. This new idea has inspired us to design an urban social application we call *WhatsUpNow* or simply, the *application*. The *application* allows smart device users to discover ambient data from other users in places they want to visit. This promotes the creation of multimedia and sensory data grounded in an almost tangible environment to help people foster interpersonal communication around this data. Our *application* provides means to create, share, and disseminate ambient-based multimedia like visual or audio snippets, as well as device generated sensory data. It also allows for service discovery [11] and connections between users, based on the places they are at, and those who are searching for the same places. This allows users to generate, upload, and share, in a peer-to-peer setting, ambient and sensory data.

There are many design issues associated with the creation of our *application*. First, at the network level, users with smart devices move randomly, organize themselves capriciously, and may have very little or no knowledge with respect to identities and capabilities of other smart devices. As such, service discovery in mobile ad hoc networks [1] consists of a set of mobile devices operating without the aid of established infrastructure of the centralized administration. Therefore, service discovery offered to users in the field must be discovered effortlessly to facilitate data sharing. Many emerging applications such as emergency services, disaster recovery, environment monitoring, law enforcement, search and rescue, plus commercial and educational applications rely on mobile ad hoc networks. Service sharing and discovery play a pivotal role in a mobile ad hoc network environment (MAHNE). When joining a self-organizing MAHNE, mobile phones should be able to explore the environment to learn about, locate, and share available. Our *application* needs to use the latest technology for service discovery, especially when one might want to discover several instances at the same time. For example, a user wants to discover two other users in two different places. Second, at the

caching level [20], once services are discovered, users might want to avoid downloading data directly from the server, and opt to do so from cached copies in mobile devices within their service discovery circle. Therefore, our caching technique should use mobile devices as caching nodes to cache the ambient and sensory data. It should also query directory nodes responsible of caching the queries and associate them with their corresponding caching nodes, and a service manager node that takes managerial decisions. Our strategy allows several copies of ambient data and a query to be cached in MAHNE (logged in to the application), allowing a user to access the nearest copy of an ambient data item. Third, at the application level, smart device users should be able to automatically authenticate the alignment of the place of interest with the Global Positioning System (GPS) before they are able to generate and upload device-generated sensory and ambient data. Also, care must be taken to design a user interface that takes into consideration the context of users, especially when interacting with the application in crowded dark places using often one hand. This means generating and uploading ambient data with annotations must be done, to an extent possible with one hand. Service discovery and request should also be supported at the interface level with the same approach. Fourth, at the multimedia level, ambient data visualization must be temporally oriented (short-time intervals) with meaningful and easily understood color that are based on commonly and better yet internationally learned coloring system, such as glowing red or glowing yellow. Such visualization must help users interpret the intent of data by simply looking at it.

The rest of this article is divided as follows: in Section II, we describe work related to three essential categories for our *application*: service discovery, cashed strategies, and ambient concepts and visualization. After citing a related work for a category, we discuss design alternatives for the *application*, relevant to each category. It is important to know that the thorough coverage of all three areas is essential to reach successful design strategy for our *application*. Service discovery and cached techniques are two essential design choices that can determine the practicality and scalability of the *application*. In Section III, we describe the design choices of our application with respect to the three aforementioned categories. We also present the interface design of the a*pplication*. In Section IV, we present results of a network-based experimentation. The experiment was conducted to mimic the deployment of mobile phones or nodes with the *application*, and better understand our ambient data cached strategy. In Section V, we conclude with observations about current results and future directions.

2 Related work and general design alternatives

This section presents work related to the design issues we previously mentioned for our *application*. We cite works related to service discovery in MAHNE, caching techniques, and visualization and color schemes for ambient and device generated data. In each related work, we also describe some design challenges and offer recommendations.

2.1 Service discovery in MAHNE

In wireless ad hoc networks, service discovery approaches can be divided into broadcast-based, advertisement-based, application-based, network-supported, probability-based, and semantic routing-based, directory-based, and directory-less schemes [30]. In the broadcast-based solution, a service discovery request is broadcasted throughout the network. Mobile devices receiving the service request apply the service matching process to find out whether the

requested service description matches one of the services that they offer. If it matches, a reply containing the service description that can be used to invoke the service is generated [34]. Upon receiving a positive reply, the requesting node launches the service invocation process in which it establishes a session with the service provider to invoke the service. The requesting node may receive several replies from different service provider nodes that offer the same service. In such case, it selects the most optimal service provider through a process referred to as service selection as shown in Fig. 1.

In the advertisement-based solution the services advertise themselves to all nodes and each node interested in discovering services will cache the advertisements. When a node wants to request a service, it first runs the service matching locally and the service selection to identify the service provider, then the service invocation process to invoke the service from the appropriate service provider [19].

In the application-based solution, the service discovery protocol finds the address of the service provider node and the service description and access mechanisms that should be used to invoke the service; but a routing protocol is needed to find the route between requesting client and the service provider. This motivated, in addition to other intrinsic characteristic of ad hoc networks, the combination of routing-based and application-based service discovery in one approach [35]. Indeed, frequent mobility in ad hoc networks implies that the network topology may vary; consequently routes in the network can change, network partitioning can occur, and previously available services may become unavailable. Thus, one can say service discovery and routing are more tightly coupled than in fixed networks.

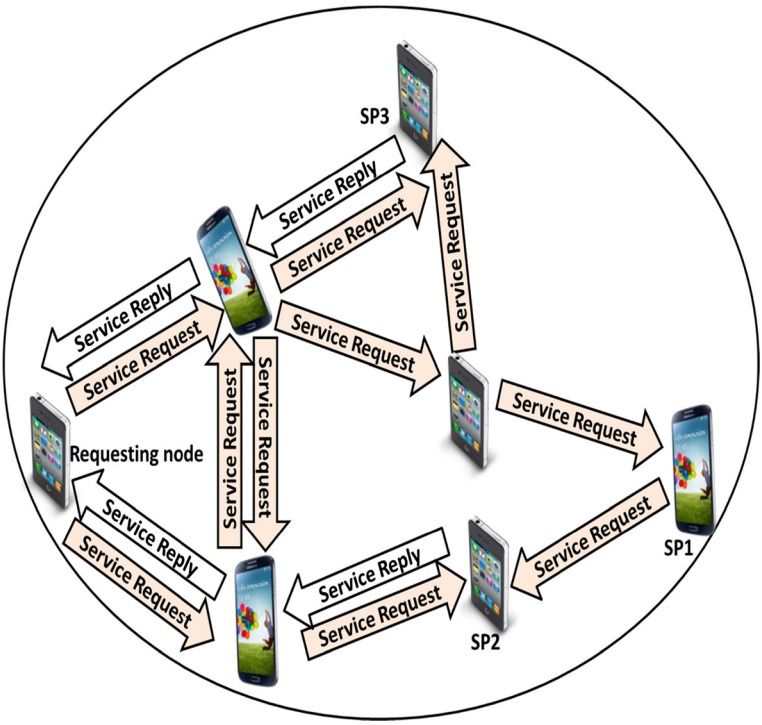


Fig. 1 Packet forwarding in the case of both cache hit and miss

In probability-based service discovery schemes, when receiving a service request, nodes that fail the service matching process forward the packet with probability P [14, 15]. When the value of P is high, most if not all nodes will be forwarding any request they receive, causing flooding and forwarding redundancy, serious contention, and frequent collisions. When the value of P is low, the redundancy of request forwarding will be reduced. Meanwhile the coverage of service request will also be reduced, which may affect the ability of finding matched services. In semantic routing based schemes such as [8], nodes can intelligently select the next-hop nodes for service request packets by inspecting service description semantics as well as current network topology. When receiving a service request, only selected nodes will forward the packet.

In directory-less solutions, which are based on client/server architecture, the service requester (i.e. clients) reactively send out service request messages. Service providers (i.e., servers) listen to such messages at a well-determined network interface and port, and apply the service matching process on the received requests. If the requested service is matched, then a reply message is generated and sent to the clients. Service requesters can also passively learn about the available services by listening to the service advertisements that are proactively generated by the servers. Directory-based solutions involve service directories that reside between service requestors and the service provider logical entity [4, 28] as shown in Fig. 2. Service requesters direct their requests to well-known service directories with which service providers register their services. In return, service directories send back service reply messages

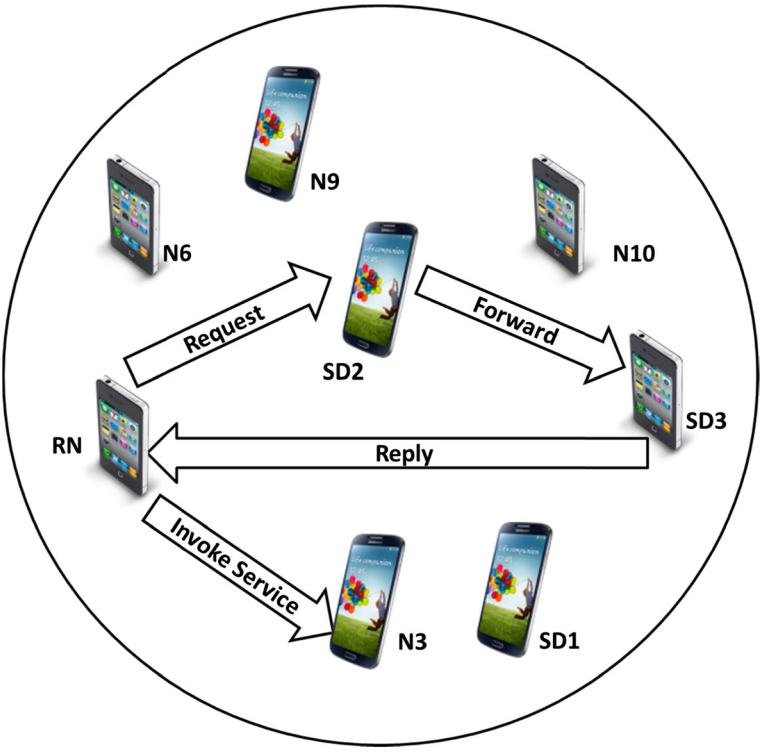


Fig. 2 Packet forwarding in the case of both cache hit and miss

to the clients and registration acknowledgments to the service providers. Upon receiving a reply message, the requestor starts the service selection and service invocation processes [30].

2.2 Cached techniques

Many caching techniques were recently proposed [5, 9, 10, 12, 29]. The cooperative-based database caching system (*COACS*) that was proposed in [5] caches database data and SQL queries in separate nodes in MANETs. In COACS, nodes can play the roles of Query Directory (QD) nodes, Caching Nodes (CN), and/or Requesting Nodes (RN). CN nodes are responsible of caching the data and associate them with their corresponding queries in QD nodes. QD nodes are responsible of caching the queries and associate them with their corresponding CN nodes. *COACS* distributes the network management responsibilities to several nodes. If a node receives data items for a certain query from the database, the node will become a CN and the query will be cached by the nearest QD. The role of QD nodes is essential in *COACS* since they represent a distributed indexing system used to access data items in CN nodes. In COACS the first node needing to cache a data item sends a special packet to its neighbors to specify the list of QD nodes, then this packet is forwarded sequentially until all the nodes are explored. The last explored node assigns the node with the highest score to become a QD. When a node requests a data item, it sends a message to the nearest QD. If the QD does not cache the query, it forwards the message to its nearest QD. The process continues until a QD that caches the data is found or all QD nodes are checked. In case of a cache miss, the data is fetched from the database server. In case of a cache hit, a QD will ask the CN that caches the response to respond to the requesting node. The problem with *COACS* is that it allows only one copy of a data item and a query to be cached in the network. Therefore, it suffers from a latency problem because the QD may be several hops away from the RNs and also a CN may be far away from the QDs indexing its data items.

2.3 Ambient visualization

Information visualization can be generally defined as using visual displays to offer users visual representations of abstract data in an attempt to amplify their cognition about the data. Unlike traditional information visualization, ambient information visualizations reside in the environment of the user rather than on the screen of a desktop, smart device or phone. Ambient data normally reside in the user's peripheral environment. Dynamic ambient information visualization updates were presented in computer-less environments. For example, the work in [18] introduced a lamp that uses different intensity to indicate variations in an information source. Figure 3 depicts a glass lamp that is showing stock market trends, weather forecast, and local traffic via color encoding.

Figure 4 depicts an umbrella handle that glows when rain is in the forecast. This is intended to provide users with contextual visual clues to help them remember to take the umbrella when going out.

Figure 5 depicts a commercial ambient device [2] that provides two movable ears and means for audio/speech output.

Digital display-less approach has imposed limitations with respect to the type and complexity of information that can be shown. With the rapid advancement of display technologies, and the progressive landslide in costs, researchers have resorted to exploring more and more the real possibilities of using digital graphics for ambient displays. The work in [24] uses

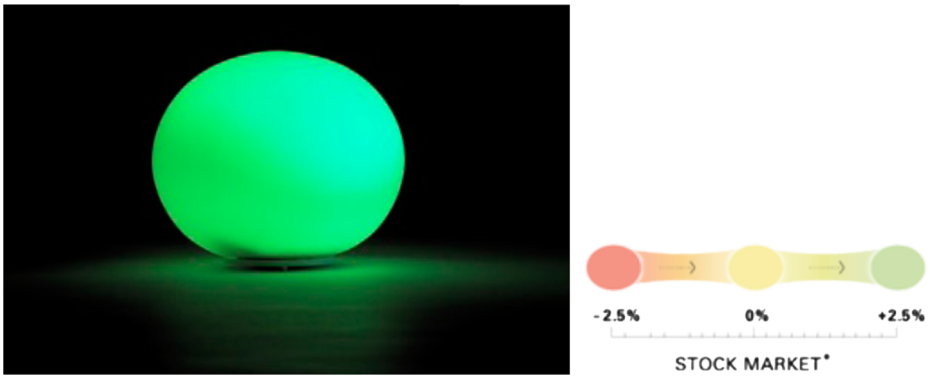


Fig. 3 Physical ambient visulization. Adapted from [18]

specialized computer displays to create virtual paintings from traditional wall-hung art to inform the design and use of such displays. The work in [13] uses digital picture frame of a family and augment border of the photo with information about the health of an elderly family member. Another example is depicted in Fig. 6. It presents a clock that integrates with Google calendar. It shows appointments of the day. The background-color changes from blue over yellow to orange to alert people to the beginning & end of events.

The obvious but less understood advantage of digitally displaying contextual complex ambient information, to boost cognition and share information to help make better decisions, has not been fully researched. Recent research efforts have focused rather on overcoming challenges associated with ambient information visualizations. The work [26] categorized data sources, distinguished ambient, peripheral, and notification systems, and provided comprehensive overview of many research and commercial ambient systems. The comparative study used, design dimensions, information capacity notification level, representational fidelity, and aesthetic emphasis. The work in [32] showed how information can be dynamically displayed

Fig. 4 Physical ambient visulization. Adapted from [18]

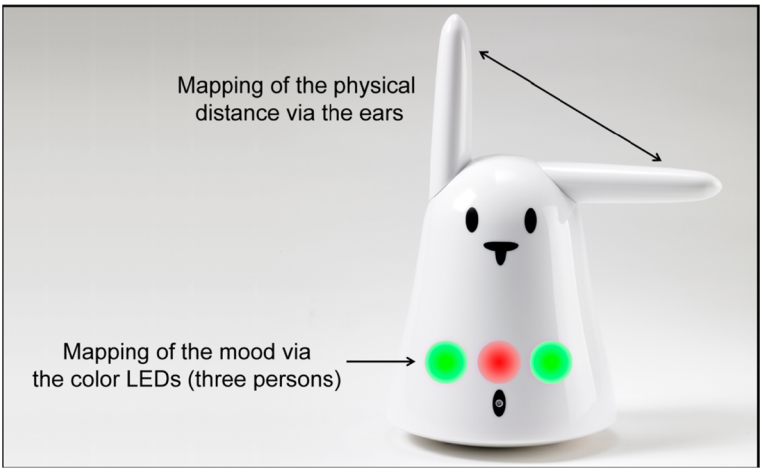


Fig. 5 Visualization of distance and mood. Adapted from [2]

using ideas learned from [26]. The work in [23] presented a service-oriented framework supporting the implementation of ambient awareness systems based on the Awareness Mark-up Language (AML). Only few researchers looked into the evaluation of ambient displays in general. The work in [22] provided a set of generic evaluation criteria. In contrast to studies that evaluate a concrete tool like the carnet display by [27], this enables evaluating a concept abstracted from an actual implementation. In fact, most projects presenting ambient systems use simple information sources such as weather forecasts [31] or bus arrival times [21]. Hence, they cover mostly public rather than personal information. Even projects that consider shared (working) environment displaying content created by [7], do not consider context information of those people themselves. The work in [2] tried to close the gap between the large amount of contextual information available and the opportunities of presenting them in an ambient way. It investigated how ambient displays can be used to share contextual

Fig. 6 Ambient clock and calendar. Adapted fom [13]

information by experimenting with different designs and conducting an online survey to find out which context information and representations are considered to be useful by users.

3 Design of our application

3.1 The service discovery strategy

Due to the wide variety of service discovery applications in MAHNE, this topic is still considered as a very important area of research. The research community is headed towards improving the existing protocols or creating new protocols in order to face the challenges of MAHNE. The most important of these challenges are the scalability, the limited power resources, and the device heterogeneity of the network [30]. The scalability and the limited power resources are tightly related in that minimizing the energy spent by a mobile node will result in longer network lifetime and will allow the handling of large scale networks. The cross layer approaches are mainly being studied and adopted in order to minimize communication by combining service discovery and routing which will reduce the amount of messages processed by each mobile node and will result in a lower energy consumption. In addition, new technologies are constantly emerging with new devices that have wireless support, with different hardware capabilities. Therefore, the need for protocols that can bridge the gap seamlessly between the different devices is growing. In this work, we use a scalable and decentralized service discovery protocol similar to the one proposed in [35], which can be easily customized to our architecture. Clustering promotes a more efficient use of resources in MAHNE, and as such, the cluster heads (CHs) are used as service lookup directories that know which nodes hold the requested services. A node that is interested in a particular service contacts its CH, which is expected to hold a current list of service providers (SPs) in the network. For service advertisement, an SP will only advertise its services to its CH, and the latter will disseminate it to the other CHs. Furthermore, by integrating quality of service (QoS) into its functionality, the service discovery approach allows a requesting node (RN) to choose an SP that is expected to offer the highest quality service as compared with others.

3.2 The caching strategy

Our strategy aims at minimizing the data accessing delay and maximizing the availability of data cached in the network without inducing excessive large traffic. Unlike COACS which employs three types of nodes, CN, QD, and RN, our strategy defines a fourth type called the Service Manager (SM) which is an elected entity that oversees the resources of nodes in the network and makes “managerial” decisions accordingly. These decisions include electing QD nodes, calculating the maximum number of QD nodes, updating the list of QDs when needed, and informing all nodes in the network about the current list of QD nodes as will be described shortly. A QD is assigned and supervised by a SM and it is responsible for caching the queries requested by mobile nodes without their corresponding data. The CN caches the query responses returned by the database server or by another CN. The CN keeps indices to the QD nodes that index its cached data. These indices can be used later to inform a QD that the cached data in the CN has been removed or replaced. A RN is any node that requests a query. Figure 7 shows a conceptual overview of the proposed approach. The service discovery protocol that we have adopted can be easily customized to work with this architecture by

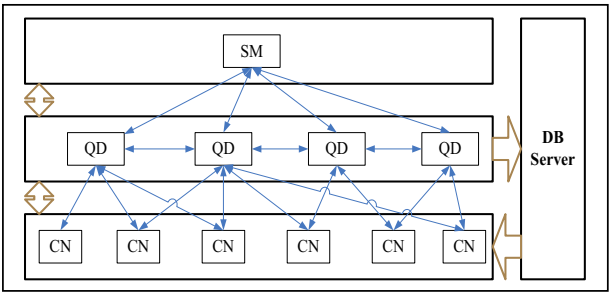


Fig. 7 Conceptual and hierarchical view of the proposed architecture

considering the QDs node as cluster heads for service discovery purpose. We present next its network setup process, search algorithm, and cache management policy.

i. *Network Setup Process*

The SM and QD nodes are assigned responsibilities that require considerable power and processing resources. Therefore, powerful nodes with high capabilities should be designated as SM and QD nodes. To measure the capability whether a node should become a SM or QD, each node is designated a SM score and a QD score. A node with high SM score is elected as a SM. Nodes with high QD scores are considered as Potential QD (PQD) nodes. A PQD is a candidate to become a QD anytime a new QD is needed. We have used a scoring mechanism similar to the one used in [12], which considers five node parameters:

1) Cache size of the node at time *t* which is calculated as $cache\ (t) = 1 - loadOfMem\ \frac{(t)}{MAXMem}$ where MAXMem is the node maximum cache size and loadOfMem(t) is the used portion of cache capacity in MByte at time *t*.
2) Node battery level at time *t* which is calculated as $Bat(t) = 1 - loadOfBat(t)$ where loadOfBat(t) is the actual used battery capacity at time *t*.
3) Node CPU capacity at time *t* which is calculated as $CPU(t) = 1 - loadOfCPU(t)$ where loadOfCPU(t) is the actual used fraction of the node's CPU at time *t*.
4) Transmission rate which specifies the device's transmission rate.
5) Class that specifies the type of the mobile node (cell phones, PDAs, or Laptops).

Each parameter is associated with a weight. A weight of a certain parameter will differ depending on the SM or QD score. The node computes a weighted score, W(t), for SM and QD that varies between [0..1] where 0 means that a node is not powerful:

$$W(t) = \frac{Wcpu \times CPU(t) + Wcache \times cache(t) + Wbat \times Bat(t) + Wtyp \times TYP(t) + Wtr \times TR(t)}{NUM}$$

where *NUM* is the total number of parameters used.

To elect a service manager node, we use the following algorithm:

1) A node starts the SM election by sending a S*core Packet* (SP) sequentially to the nodes in the network. The SP is composed of an address list, a score list, and an exhausted node list. The address list contains addresses of all nodes that have already received the SP

message. The score list contains the 3 highest SM scores. If a SM score is added to the score list, it references the corresponding node address in address list. A node address is added to the exhausted node list if all nodes from its routing table are included in the address list.

2) Upon receiving a SP packet, a node checks if its score is larger than the scores included in the SP. If it is, it replaces the lowest score with its score. Then the node checks if the information about all other nodes in the network are included in the packet by checking the address list. If they are not included, the node sends an SP to the nearest node that is not included in the address list and exhausted list. If they are included, the node adds its address to the exhausted list and sends a SP to a node included in the address list but not in the exhausted list. If such node does not exist, the node becomes a SM Assigner (SMA).
3) SM Assigner sends a SM Assignment Packet (SMAP) to the node with the highest score in the SP. This node becomes the new SM and broadcasts a *Network Information Packet* (NIP) packet to inform all other nodes in the network about the new SM.
4) Upon receiving the NIP packet, the nodes in the network will send a *Credentials packet* (CP) containing their SM and QD scores to the SM.

When the elected SM receives the SM and QD scores of all nodes, it calculates the maximum number of QD nodes $N_{QD}^{\max}$ using the following notations:

1) The expected number of hops between any two nodes in a rectangular network topology with area $a \times b$ where one destination choice is available is given as $E[H]=0.521a/r_0$ where r_0 is the maximum node transmission range.
2) The expected number of hops, from RN to the nearest QD, within a system that comprises N_{QD} QD is $E\left[H_i|N_{QD}\right] = \sum_{j=N_{QD}+1-i}^{N_{QD}} E[H|n=j]$.
3) The expected number of hops to get to the QD with the requested data is $E\left[H_{QD_{Data}}\right] = \sum_{i=1}^{NQD} P_i E\left[H_i|N_{QD}\right]$ where $P_i=1/N_{QD}$ is the probability that the data is located in QD_i.
4) The expected number of hops to get to the last QD in the system is

$$E\left[H_{QD_{Last}}\right] = E\left[H_{N_{QD}}\middle| n = N_{QD}\right] = \sum_{j=1}^{N_{QD}} E\left[H\middle| n = j\right]$$

5) The expected number of hops to the external network is $E[H_{AP}]=0.7652/r_0$.
6) The expected response time of the system (query directory access and delay) is calculated in function of the hit ratio R_{hit}, the delay for transmitting packets between nodes inside the network T_{in}, and the delay for accessing a node outside the network T_{out}. In case of a hit, the delay is the summation of $T_{in}E\left[H_{QD_{Data}}\right]$ (to get the QD with the data) and $2T_{in}E[H]$ (to access CN and reply to RN), then the expected response time of the system is $E[\tau_{hit}] = T_{in}\left(E\left[H_{QD_{Data}}\right] + 2E[H]\right)$. In case of a miss, the delay is the summation of T_{in} $E\left[H_{QD_{Last}}\right]$ (after traversing all QD nodes), $T_{in}E[H_{AP}]$ (when forwarding the request to

AP), $2T_{out}$ (delay for accessing the database and getting its reply), and $T_{in}E[H_{AP}]$ (to send the reply to RN), then the expected response time of the system is $E[\tau_{miss}] = T_{in}\left(E\left[H_{QD_{Last}}\right] + 2E[H_{AP}]\right) + 2T_{out}$. Hence, the general expected response time of the system is

$$E[\tau] = R_{hit}T_{in}\left(E\left[H_{QD_{Data}}\right] + 2E[H]\right) + (1-R_{hit})\left(T_{in}\left(E\left[H_{QD_{Last}}\right] + 2E[H_{AP}]\right) + 2T_{out}\right)$$

The maximum number of QD nodes is

$$N_{QD}^{max} = max\left(N_{QD}\right)\Big|E[\tau] < E\left[\tau_{NoCaching}\right]$$

where $E[\tau_{NoCaching}]=T_{in}(2E[H_{AP}])+2T_{out}$.

Using the above equations, we can obtain the following inequality:

$$E\left[H_{QD_{Data}}\right] + (1/R_{hit}-1)E\left[H_{QD_{Last}}\right]-0.4884a/r_0-2T_{out}/T_{in} < 0$$

This inequality is evaluated for different values of R_{hit} and N_{QD}. Table 1 shows the maximum N_{QD} for different R_{hit} values.

When a new node joins the network, it gets the list of QD nodes and potential QD nodes from its neighbors in case a SM already exists. If no SM exists the node sends a SP packet and the SM election starts.

After electing an SM, the SM selects the first N_{QD} nodes with the highest QD scores as PQD nodes and broadcast a NIP packet to inform other nodes in the network about the list of PQD nodes. Then, it waits a certain period of time (i.e., until RN nodes start requesting queries) before broadcasting for the first time the list of QD nodes. QD nodes will be chosen from the list of PQD nodes progressively by different RN nodes. A RN chooses a PQD from the list of PQD nodes, and then it sends a request to SM to assign this PQD as QD. After assigning it as QD, the SM sends a NIP packet to inform all nodes about the new list of QD nodes. Because some QD nodes can be assigned simultaneously as soon as RN nodes start requesting data, allowing the SM to wait shortly before broadcasting the first list of QD nodes will reduce the number of times a NIP is broadcast.

ii. *Information Search Algorithm*

When a RN needs to access QD nodes to search for a cached item, it generates a query then checks if it caches the requested data. If this is the case, the RN can directly access the data in its cache. Otherwise, it sends a *Data Request Packet* (DREQ) to the nearest QD. If this QD caches the query, the request is forwarded to the CN that caches the corresponding data. If the CN caches the data, it will send a *Data Reply Packet* (DREP) directly to RN as shown in Fig. 4 and a CN Acknowledgement Packet (CN_ACK) to the QD. If the nearest QD does not cache the requested query, it adds its address to the list of checked QD nodes to indicate that it has

Table 1 Maximum values of N_{QD} to avoid excessive delays

$R_{hit}\rightarrow$	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
$N_{QD}^{max}\rightarrow$	2	3	5	9	12	20	32	55

been checked for this request, appends this list to the requested query, and sends the query to its nearest QD that has not been checked yet. The process continues until a QD that caches the data is found or all QD nodes have been checked. If all the QD nodes are checked and no one caches the query, the data is fetched from the database server as shown in Fig. 8.

If the CN does not cache the requested data, it sends a *Negative Acknowledgement Packet* (NACK) to the QD which deletes the corresponding query to prevent further misdirected requests for the data then sends the DREQ to the database server. On the other hand, if no acknowledgement is received from the CN, the QD will conclude that the CN has gone offline and therefore, it deletes the query and forwards the request to the database server.

When a RN receives the data corresponding to a requested query from the database server directly, then it caches them and sends a *Query Add Packet* (QAP) or a *Cache Add Request* (CAREQ) to a QD as explained in the next sub-section. Whenever a query is cached in a QD, the latter sends a *QD Acknowledgement Packet* (QD_ACK) to the corresponding CN to create and maintain a link to the QD. In case the data is deleted or replaced in CN, this link will be used to inform the QD that the data has been deleted.

iii. *Cache Management Policy*

The cache management policy is composed of both a cache admission control and a cache replacement policy. The cache admission control is triggered when a RN receives a requested data to decide if it should cache it and become a CN. It is composed of two parts. The first part describes how a RN reacts in case of a cache miss (i.e., the QD does not cache the requested query and the database replies to RN's request). The second part presents how a RN reacts in case of a cache hit. The cache admission control algorithm uses three thresholds. The *threshold_RN* indicates when a RN can adopt or ignore the cache admission control algorithm in case of a cache hit. If the number of hops between the RN and the QD caching the query is greater than *threshold_RN*, RN triggers cache admission control algorithm; otherwise RN

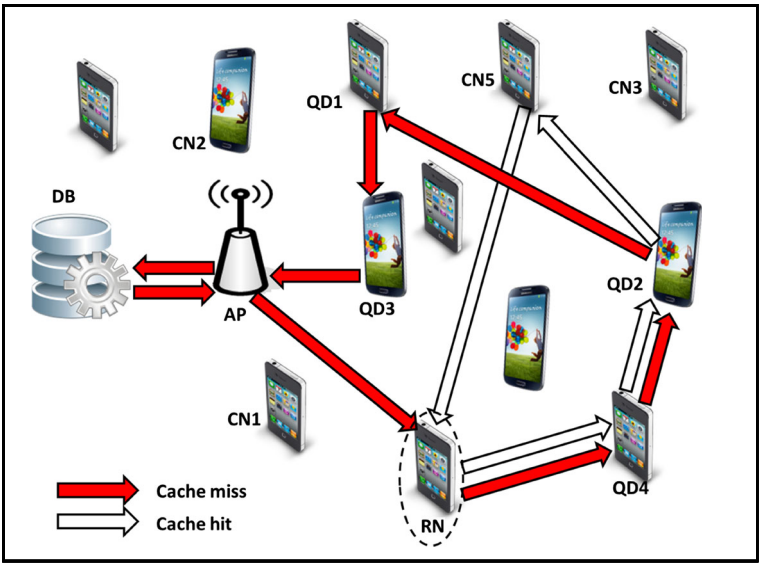


Fig. 8 Packet forwarding in the case of both cache hit and miss

ignores it and does not cache the requested data. The two other thresholds, called *threshold_QD* and *threshold_PQD*, used on both cache miss and hit, are the maximum number of hops a RN should be away from a QD or PQD respectively to cache the requested query on it. Note that *threshold_QD* and *threshold_PQD* are usually equal, but they can be different from *threshold_RN*.

Figure 9 shows that in case of a cache miss, a RN becomes a CN and caches the requested data. To cache the requested query, RN first checks if there is any QD where the number of hops between RN and this QD is less than *threshold_QD*. If such QD exists, RN sends to it a *Query Add Packet* (QAP) to cache the query. If the QD is full, the cache replacement policy is invoked. If such QD does not exist, RN checks the list of PQD nodes for a PQD where the number of hops between it and RN is less than *threshold_PQD*. If such PQD exists, RN sends an *Allocate QD Packet* (ALLQDP) to SM to assign this PQD as QD. Then, RN sends a QAP to the new QD to cache the query. If such PQD does not exist, RN sends a CAREQ to cache the query in one of the existing QD nodes. Although CAREQ and QAP are both sent by a RN to cache a query, QAP is sent to only a specific QD and does not contain a list of visited QD nodes. CAREQ is sent to several QD nodes till one of them accepts to cache query and has to contain a list of visited QD nodes.

Figure 10 shows that, in case of a cache hit, the RN checks if the number of hops between it and the QD caching the requested query is greater than *threshold_RN*. If this is the case, the RN may become a CN by caching another copy of the requested data in its cache. Therefore,

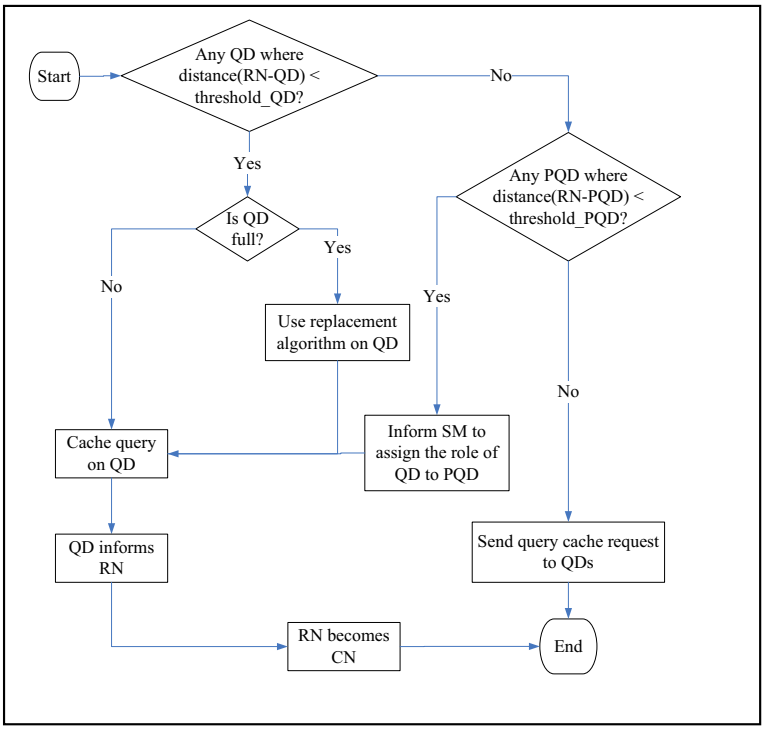


Fig. 9 Cache Admission Control Algorithm (miss)

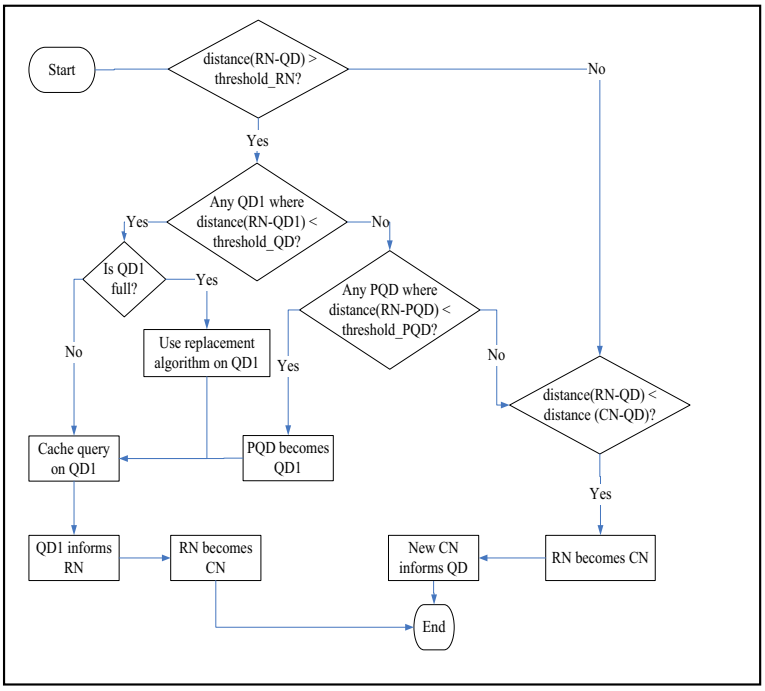


Fig. 10 Cache Admission Control Algorithm (hit)

several copies of a data item or a query can be cached in the network. In this case, the RN triggers the following steps:

1) RN checks if there is any QD, QD1, where the number of hops between RN and QD1 is less than *threshold_QD*. If this is case, RN becomes CN and sends to QD1 a QAP packet to cache the query. If QD1 is full, the cache replacement policy is used to cache the query in QD1
2) If there is no near QD1, RN checks the list of PQD nodes for a PQD, PQD1, where the number of hops between RN and PQD1 is less than *threshold_PQD*. RN becomes a CN and caches the query on PQD1 after requesting from SM to assign the role of QD to PQD1.
3) Even when distance between RN and QD caching requested query is larger than *threshold_RN* and no QD1 or PQD1 are found in step 1 and 2, RN checks if it is closer to QD than the CN caching the requested data. If this is the case, RN becomes a CN for the requesting data and sends an ACK packet to QD to delete its link with the old CN and establish a new link with the new one.

If the number of hops between RN and the QD caching the requested query is less than *threshold_RN*, RN checks if it is closer to QD than the CN caching the requested data. If this is the case, RN becomes a CN for the requesting data and sends an ACK packet to QD to delete its link with the old CN and establish a new link with the new one. A cache replacement policy is used in both QD and CN to decide which data item should be replaced when the cache is full. An item can be either a query for a QD or a data item for a CN. We use a two-phase cache replacement algorithm that is based on both Least Frequently Used (LFU) and Least Recently Used (LRU) replacement algorithms. In this algorithm, we use first the LFU replacement algorithm which bases its decision

on the frequency of references: a data item which is used the least often is selected. If several items that have the same least number of accesses are selected, then we apply on them the LRU replacement algorithm which bases its decision on the heuristic reference of data items. Therefore, among the several selected data items, the least recently used one will be selected to be replaced. When a data item is replaced, the CN informs the corresponding QD that holds the corresponding query to invalidate this query to prevent any misdirected requests for the data.

3.3 The GUI design strategy

There are many design issues with respect to the graphical user interface (metaphors to use), the ambient data visualization, and the feedback a user gets when the mobile device is not in plain sight.

i. The Graphical User Interface

a) The Serach View

To get to the Search View, the user normally chooses the Search View from the dropdown menu that is built into the slider/button called: "Search for a Location". This action takes the user to the view shown in Fig. 11(b). In this view, the user sees the search result based on the GPS location of the smart phone. The search results (various clubs) are displayed on a map as icons that the user can interact with. There are two modes of

Fig. 11 The Top view (**a**), and the Search view (**b**)

interactions with these icons. The first mode requires a quick press, and will pop up a multi-function widget that allows the user to choose the date and request the desired ambient and sensory data. Figure 11(b) shows the image of a guitar player as the data received and displayed in the time-based rollover media widget for ambient data. This data can come from the server or from a cached copy (services discovery, and cached data as previously mentioned) on other smart devices in the field. The second mode requires the user to press longer on an icon which causes the icon to be selected. Once, the desired number of icons (i.e. icons representing the search result on the map) the user can click the "Get Selected" button and this will take the user to the view in Fig. 12. The user can also choose the "Search tab" and perform a manual search, after which the results will be displayed as previously described.

The view in Fig. 12 shows the four tabs that were created for the specific category-based search. These tabs provide the user with a quick read and flips between the chosen places and their ambient data. As can be seen from the figure, each chosen place on the left vertical tabs has four horizontal tabs associated with it. Each one of which, represents a specific ambient data. For each ambient data tab, there is a clock with a *ButtonArrow* that

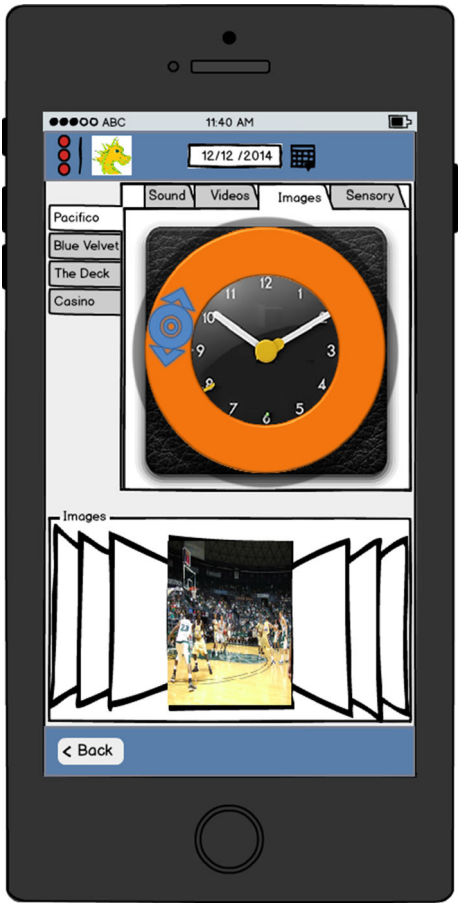


Fig. 12 The clock with the ButtonArrow to examine ambient data

slides clockwise and counter clockwise. This new kind of interface element gives contextual affordance; that is, it allows the user to intuitively recognize the possibility of examining ambient data according to time. This *buttonArrow*, when clicked, provides a gallery (relevant to the chosen time). Changing the date can simply be done by clicking on the date/time widget. Users can browse the gallery for ambient media of their choice. The images gallery shows a court side image of a basketball game.

Ambient data can be seeded on any node or phone, and a requesting node or a phone asking for the ambient data can get cached copies, as mentioned before in our caching network discussion.

b) The Post View

To get to the Post View, the user has to authenticate first. This can be done from the dropdown menu that is built into the (initially disabled) slider/button called: "Post for a Location". The Authentication ensures that the user must confirm the GPS location and place of visit pair. The tuple (G4/WiFi, GPS, Google Maps/place) is used for the authentication. A 50 m radius is permissible. This allows for free movement without jeopardizing the authentication. This feature ensures that the people posting the data are in the place of visit, or extremely close by during observational reports. Once the authentication is confirmed, the slider/button called: "Post for a Location" is enabled. Once clicked, it will take the user, as depicted in Fig. 13, to the Post View where the user has three choices: Post Ambient Sensors, Post Ambient Media, and Post Extras. These Buttons will remain enabled until a new authentication is required. This feature is designed to boost the confidence of people relying on this data to make informed decisions. The Post Ambient Sensors button, once clicked, activates all relevant sensors and allows the *application* to capture ambient sensory data such as temperature, humidity, barometer, and noise level. This option is fairly straightforward and in line with minimizing the interaction with the GUI and keeping the operation possible with one hand. The Ambient Media Post button, once clicked, takes the user into a tabbed folder with three tabs: Video, Sound, and Images. When for example the Images tab is chosen, the user sees a Record button. After recording the user is given the option to annotate the picture. Posting is then done from the same tab, analogous to Facebook. The Post Extra Button allows the user to post using predetermined GUI elements, happy hour, prices, etc. The user can also type feelings and or suggestions. The typed text is not further processed by the *application*. It is reported as is, and has no bearing on the visual representation of ambient data.

c) The History View

The History View allows the user to browse stored ambient and sensory data on the device This data can be used later for satistical purposes. At this point we are not considering such data for statistical informative purposes. We will however do so in our future work. The historical/statistical data can have many uses such as annual averages of sensory data. This data might be usefull to many agencies such as the Fire and Health Departments

d) The Settings View

The Settings View is a standard smart-device settings view with extra options to allow users to perform many settings necessary to maximize the benefits of using the *application*. For example, the user can set the phone to vibrate/make sound if many people are reporting about the same place. Other settings can adress in our future work the idea of notifications.

Fig. 13 The post view

The Graphical User Interface (GUI) design must take into consideration that users interacting with the *application* are often doing it with one hand. This condition puts design constraints on the way a user accesses different features of the *application*. Once the *application* is started and the user enters his/her credentials, the user is given the choice, as depicted in Fig. 11(a), to access four main views: the Search, Post, History, and the Network and GPS Settings views. We will describe each view next.

ii. The Visual Representation of Ambient data

The Request View is designed to metaphorically represent folders, multimedia and time. The vertical tabs representing places to visit can start to glow based on ambient data coming from that place. Also horizontal tabs can start to glow indicating activities in ambient media or sensory data reporting. The clock concept is influenced from the work in [13]. In our work though, time is associated with all activities; especially when it comes to people wanting to visit one or more place during the day/night. This watch/clock is normally in the ambient of a person when planning or executing a trip. So it is an excellent metaphor for all ambient data that stems from time based intervals. The clock is surrounded by a wide band that is initially orange, that is, an equal amount of Red/

Yellow/Green (RYG). During the operation of the application, different glowing shades RYG, according to different ambient data, can be spotted around the clock. The user can slide to a glowing area of interest, and then hold the button which causes the ambient data that associated with that time frame is shown in the gallery. The clock also has four resting corners onto which happy faces can appear to indicate up to 4 preferential groups of friend's state of overall visit to a place. Friend groups can be added in the settings.

4 Network performance evaluation

To determine the scalability of our *application*, we have evaluated it using the network simulator NS2 [25] and compared it to that of *COACS* [5] and *No Cache* strategies. Several metrics were measured used such as average query delay, average hit ratio, and bandwidth consumption per node. Each metric was measured by varying different parameters such as QD's cache size, CN's cache size, time, and zipf distribution.

In this simulation, we have used a network topology of 100 mobile nodes distributed in an area of 1000×1000 m with only one access point placed near one of the corners with a data rate and a transmission range are 2 Mb/s and 250 m respectively. We have used the Random Waypoint (RWP) mobility model. The speed of the mobile nodes is set by default to 2.00 m/s in most scenarios and the pause time is 300 s. We have used 7 as the number of starting QD nodes in *COACS* because *COACS* yields best results with this starting number of QDs. The proposed approach starts with only one QD and 7 PQD nodes. Then QD nodes are assigned as needed depending on the cache admission control policy. The cache size of CN and QD were set to 200 Kbytes and 250 Kbytes respectively. The *threshold_QD*, the *threshold_PQD*, and the *threshold_RN* were all set to 3 hops. The query size is set to 500 bytes and the query results size was set to 10 Kbytes. The used routing algorithm was DSDV. The database server has a *database* containing 1000 data items. i.e. n_q. Every 20 s, the *query generator* on RN nodes generate a read-only query following a Zipf-like access pattern [6]. In Zipf's law, the frequency of a request for the i^{th} most popular data item is proportional to 1/i. Therefore, the probability P of accessing a data item with rank i ($1 \le i \le n_q$) is $1/\left(i^{\theta}\sum_{k=1}^{n_q} 1/k^{\theta}\right)$ where θ varies between 0 and 1. If θ is 1 or 0 then we have a strict Zipf's law distribution or uniform distribution respectively. If θ is between 0 and 1, we have a Zipf-like distribution.

The number of runs for each scenario was set to 10 using the central limit theorem [3] to achieve 90 % confidence level, given the precision level for both the average hit ratio and delay as 0.2 and 10 respectively.

We first studied the effect of QD's cache size. In contrast to *COACS* where a query is cached only once in the network, the cache admission control algorithm of the proposed approach allows the possibility of query replication. Therefore, its performance depends on the cache size of QD nodes. As the QD's cache size increases its performance increases because the QD nodes have more room for replication without excessively using the cache replacement policy. The left side of Fig. 13 shows the effect of QD's cache size on the average query delay. The figure shows that the proposed approach outperforms *COACS*. It shows that as the QD's cache size increases the performance of the proposed approach increases because the QD nodes have more room for replications without excessively using the cache replacement policy. Consequently, an RN can access the nearest copy of the query while in *COACS* one

copy of a query is available in the network and a request has to traverse several QD nodes before getting to the QD that caches the requested query. Figure 14 (right) shows the effect of QD's cache size on the average bandwidth consumption per node. The performance of *No Cache* is better than the performance of *COACS* and worse than the performance of the proposed approach. Unlike *No Cache* which sends a request directly to the database server; in *COACS* a request packet traverses a list of QD nodes before sending a request to the database server in case of a cache miss. First, during a cache miss a request packet should be forwarded to all QD nodes before being forwarded to the database server. Second, in case of a cache hit, a request packet starts traversing the list of QD nodes before finding the requested query in one of the QD nodes. The proposed approach aims to minimize the average bandwidth consumption per node by minimizing the number of QD nodes traversed by a request packet. First, it minimizes as possible the number of QD nodes used in the network and therefore reduce the number of traversed QDs compared to *COACS* during a cache miss. Second, during a cache hit, because a query can be cached in several QD nodes, a request packet will get to the nearest QD caching the query and therefore traverse less QD nodes to reach the request.

Figure 14 shows the impact of CN's cache size on the average query delay, average hit ratio, and average bandwidth consumption per node. The left side of the figure shows the effect of CN's cache size on the average query delay. We can see that the proposed approach outperforms COACS because of several replications of a query. CN's cache size has no effect on *COACS* and the proposed approach in terms of average query delay because the effective query delay depends on the cache size of QD nodes as we have seen previously. The right part of Fig. 14 shows the effect of CN's cache size on the average hit ratio. The hit ratio of *No Cache* protocol is zero because no data or query is cached in the network and all requests are forwarded to the database server. The figure shows that the proposed approach hit ratio is close to that of *COACS*. The marginal difference is because the proposed approach has less QD nodes and deals with representations there is always the possibility of replacing queries that have no other copies in the network. In *COACS* when a query is cached in the network, there is

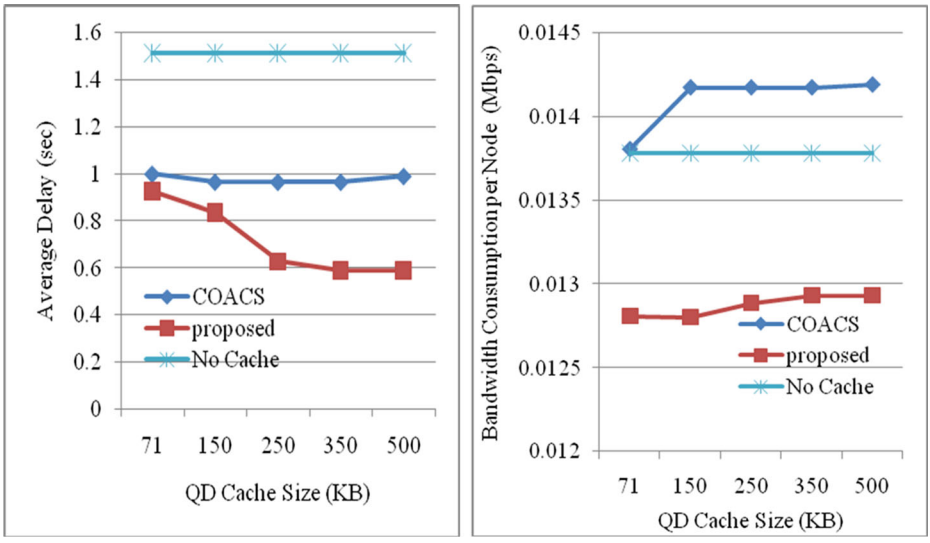


Fig. 14 QD's cache size versus average query delay (*left*) and versus average bandwidth consumption per node (*right*)

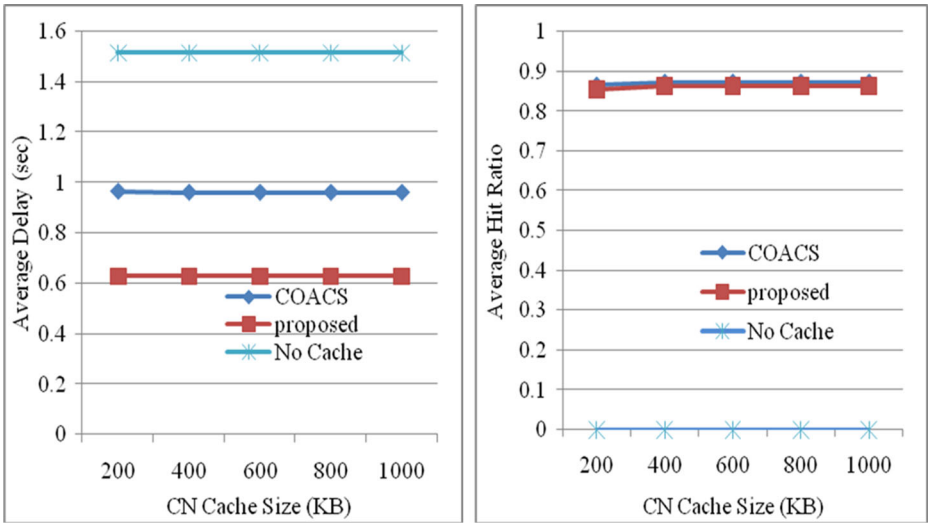


Fig. 15 CN's cache size versus (*left*) average query delay (*right*) average hit ratio

no possibility that this query is replaced unless all QD nodes are full and a query coming from the database server has to be cached. Therefore, we have always a tradeoff between average delay and average hit ratio. The average hit ratio slightly increases as CN's cache size increases for *COACS* and the proposed approach because fewer cache replacements are encountered in CN nodes and therefore fewer queries will be deleted from QD nodes.

Figures 15 and 16 show the effect of Zipf parameter θ on the average query delay. The figure shows that when Zipf parameter θ is between 0 and 0.3, the average query delay of the proposed approach and *COACS* slightly increases. Then when Zipf parameter θ starts increasing above 0.3, the average query delay decreases because, as Zipf parameter θ increases, the requesting nodes are more likely to request similar queries especially neighbouring nodes. A new request for a query is then more likely to be answered by a QD that caches it rather than

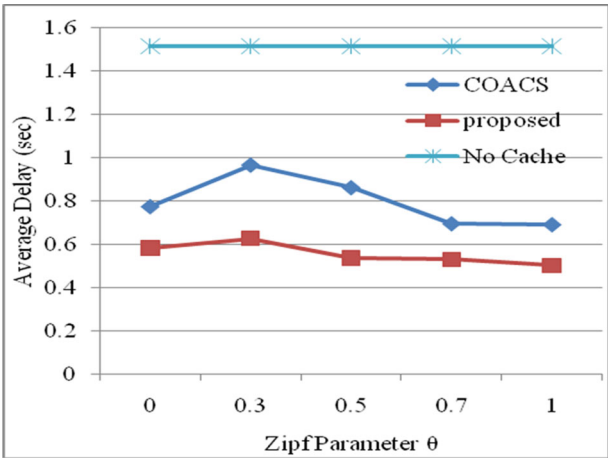


Fig. 16 Zipf parameter θ versus average query delay

the database server. The proposed approach outperforms *COACS* because an RN receiving a reply packet triggers the cache admission control algorithm and tries to cache a copy of the query in a neighboring QD (QD1) depending on *Threshold_QD* and *Threshold_PQD*. Also, the nodes within the vicinity of RN and QD1 are more likely to request the same query, and in this case, the request will be answered by QD1. In addition, the cache replacement policy of the proposed approach is based on LFU and LRU and makes use of the Zipf distribution property by replacing queries and data items in QD and CN that are the least accessed.

5 Conclusion

Our work is centered on the creation of new opportunities for people to use the latest in service discovery, caching techniques and ambient data (media/sensory) in a state of the art application for smart devices we call WhatsUpNow. This application allows for the generation of ambient data, and peer-to-peer communication of such data. The purpose it to give people the opportunities to make better informed decisions when deciding to visit different places. It is argued that a key design principle for urban social applications is to include a set of features that bring value to individual users, before they start interacting with one another. We believe that the design features of the application both in terms of its social capabilities and ambient data visualization features that ultimately have the potential of reducing the cognitive load of users should be a decisive factor in removing the challenge associated with the social urban application syndrome, especially with the young, the savvy, outgoing generation. When the application is deployed, we intend to conduct a field study that includes 30 students over a year to learn from their experiences. With respect to the networking features of the application, we have proposed a replication based caching strategy that focuses on improving the latency problem that exists in the *Cooperative and Adaptive Caching System (COACS)* proposed in [5]. The proposed approach is mainly composed of caching nodes (responsible of caching the data items, query directory nodes responsible of caching the queries, and associate them with their corresponding caching nodes) and a service manager node that takes managerial decisions. It focuses on improving the latency problem of *COACS* strategy. Indeed, *COACS* caching techniques allow only one copy of a data item and a query to be cached in the network. However, the proposed approach allows several copies of a data item and a query to be cached in the network. Therefore, a requested node accesses the nearest copy of a data item it requests. We have evaluated the performance of the proposed approach using the network simulator NS2. The results show that the proposed approach outperforms other strategies in most experiments.

References

1. Abolhasan M, Wysocki T, Dutkiewicz E (2004) A review of routing protocols for mobile ad hoc networks ad hoc networks. Ad Hoc Netw 2(1):1–22
2. Alt F, Shirazi AS, Kaiser A, Pfeuffer K, Gürkan E, Schmidt A, Holleis P, Wagner M (2010) Exploring ambient visualizations of context information. In Pervasive computing and communications workshops. pp 788–791
3. Andrel T, Yasinsac A (2006) On credibility of manet simulations. Computer 39(7):48–54
4. Artail H, Safa H, Hamze H, Mershad K (2007) A cluster based service discovery model for mobile ad hoc networks. In: Proc. 3rd IEEE international conference on wireless and mobile computing, networking and communications (WiMob 2007), New York, USA
5. Artail H, Safa H, Mershad K, Abou-atme Z, Sulieman N (2008) COACS: a cooperative and adaptive caching system for MANETS. IEEE Trans Mob Comput 7(8):961–977

6. Breslau L, Cao P, Fan L, Phillips G, Shenker S (1999) Web caching and Zipf-like distributions: evidence and implications. Eighteenth annual joint conference of the IEEE computer and communications societies (IEEE INFOCOM'99), New York, USA, 21–25 March 1999, vol 1, pp 126–134
7. Cadiz J, Fussell SR, Kraut RE, Lerch FJ, Scherlis WL (1998) The awareness monitor: a coordination tool for asynchronous, distributed work teams. Unpublished manuscript
8. Chakraborty D, Joshi A, Yesha Y, Fin T (2006) Toward distributed service discovery in pervasive computing environments. IEEE Trans Mob Comput 5(2):97–112
9. Chand N, Joshi RC, Misra M (2007) Cooperative caching strategy in mobile ad hoc networks based on clusters. Wirel Pers Commun Int J 43(1):41–63
10. Chow C, Leong H, Chan A (2007) GroCoca: group-based peer-to-peer cooperative caching in mobile environment. IEEE J Sel Areas Commun (J-SAC) Spec Issue Peer-to-Peer Commun Appl 25(1):179–191
11. de Santis F, Malandrino D (2014) QoS-based web service discovery in mobile ad hoc networks using swarm strategies. J Comput Netw Commun. doi:10.1155/2014/450194
12. Elfaki M, Ibrahim H, Mamat A, Othman M, Safa H (2014) Collaborative caching priority for processing requests in MANETs. Elsevier J Netw Comput Appl 40:85–96
13. Fogarty J, Forlizzi J, Andhudson SE (2001) Aesthetic information collages:generating decorative displays that contain information. In: Proceedings of UIST 2001, ACM symposium on user interface software and technology, ACM Press. pp 141–150
14. Gao Z, Wang L, Yang M, Yang X (2006) CNPGSDP: an efficient group-based service discovery protocol for MANETs. J Comput Netw
15. Gao ZG, Yang XZ, Cai SB (2005) Flexible forward probability based service discovery protocol for manets. J Harbin Inst Technol (Chinese) 37(9):1256–1260
16. Geurts J, van Ossenbruggen J, Hardman L (2005) Requirements for practical multimedia annotation. In: Proceedings of the workshop on multimedia and the semantic web, pages 4–11
17. http://www.lifewithalacrity.com/2006/08/using_5star_rat.html. Accessed 30 Apr 2015
18. Ishii H, Ullmer B (1997) Tangible bits: towards seamless interfaces between people, bits and atoms. In: Proceedings of ACM SIGHI conference on human factors in computing systems. Addison Wesley/ACM Press, New York, pp 234–241
19. Koodli R, Perkins CE (2002) Service discovery in on demand ad hoc networks. IETF internet draft, draft-koodli-manet-servicediscovery-00.txt
20. Majd NE, Misra S, Tourani R (2014) Split-cache: a holistic caching framework for improved network performance in wireless ad hoc networks. IEEE GLOBECOM
21. Mankoff J, Dey AK, Hsieh G, Kientz J, Lederer S, Ames M (2003) Heuristic evaluation of ambient displays. In: CHI'03, pp 169–176
22. Matthews T, Rattenbury T, Carter S (2007) Defining, designing, and evaluating peripheral displays: an analysis using activity theory. Hum-Comput Interact 22(1):221–261
23. Metaxas G, Markopoulos P, Aarts E (2007) Amelie: a recombinant computing framework for ambient awareness. In: AMI'09, pp. 88–100
24. Miller T, Stasko J (2002) Artistically conveying peripheral information with the InfoCanvas. In: Proceedings of the working conference on advanced visual interfaces (AVI 2002). pp 43–50
25. NS-2 simulator. http://www.isi.edu/nsnam/ns/. Accessed 30 Apr 2015
26. Pousman Z, Stasko J (2006) A taxonomy of ambient information systems: four patterns of design. In: AVI'06, pp 67–74
27. Roessler CP, Shelton BE (2004) The care-net display: lessons learned from an in home evaluation of an ambient display. In: UbiComp'04. pp 1–17
28. Safa H, Artail H, Hamze H, Mershad K (2007) A collaborative service discovery and service sharing framework for mobile ad hoc networks. IFIP international conference on network and parallel computing (NPC 2007), published also by LNCS (Lecture Notes in Computer Science), Dalian, China
29. Safa H, Artail H, Nahhas M (2010) A cache invalidation strategy for mobile networks. Elsevier J Netw Comput Appl 33(2):168–182
30. Safa H, Koteish Z (2010) Service discovery in mobile ad hoc networks. Next generation networks and ubiquitous computing. IGI Glob 217–225. **ISBN 9781605662503**
31. Skog T, Ljungblad S, Holmquist LE (2002) Bringing computer graphics to everyday environments with informative art. In: SIGGRAPH'02 abstracts and applications, p 153
32. Sukthankar R (2005) Towards ambient projection for intelligent environments. In: CV4IIE'05, p 172
33. Teichrieb V, Neto SG, Farias T, Teixeir JM, Lima JP, Almeida G, Kelner J (2007) Augmented ambient: an interactive mobility scenario, universal access in human-computer interaction. Ambient interaction. Lect Notes Comput Sci 4555:565–574

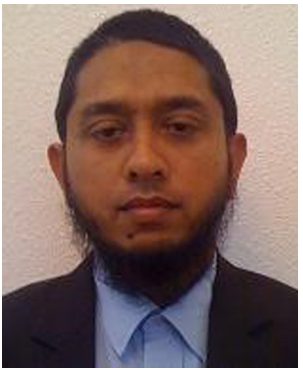
34. Toh C-K (2002) Ad hoc mobile wireless networks: protocols and systems. Prentice Hall, New Jersey
35. Torres DA, Garcia-Macias JA (2005) Performance analysis of two approaches to service discovery in mobile ad hoc networks. In: ISSADS'05 Proceedings of the 5th international conference on advanced distributed systems

Marcel Karam has received an advanced Major in Computer Science from Dalhousie University in 1995, a Master in Computer Science from the Technical University of Nova Scotia in 1997, and a PhD from Dalhousie University in 2002. Dr. Karam is currently an Associate Professor in the Department of Computer Science at the American University of Beirut. His research work aims at providing end-users with techniques and tools that are augmented with visual capabilities to improve the quality and efficiency of the many facets that comprise the software engineering process, especially software testing, interface design and multimedia. Dr. Karam has also taken some interest in applying computer architecture techniques in the area of networking protocol strategies to solve emerging problems of interest. He was also successful in securing many research and equipment grants. Dr. Karam has engaged in research activities at IBM Watson (USA), Monash University (Australia), University of Nevada, Reno (USA), and University of Ottawa (Canada). Dr. Karam has served as Associated Dean and as Chief Accreditation Officer for both ABET and NCAAA at Taif University (KSA).

Haidar Safa received a B.Sc. in Computer Science in 1991 from Lebanese university, Lebanon, M.Sc. in Computer Science in 1996 from University of Quebec at Montreal (UQAM) in which he proposed some models for storing and presenting multimedia documents, and a Ph.D. in Electrical and Computer Engineering in 2001 from Ecole Polytechnique de Montreal in which he proposed some models for reducing location update and search Costs in IS-41 based mobile wireless networks. Dr. Safa is currently an Associate Professor in the Department of Computer Science at the American University of Beirut.

Dr. Safa joined ADC Telecommunications and SS8 Networks in 2001 where he worked on designing and developing networking and system software related to surveillance and lawful interception. In 2003, he joined the

American University of Beirut where he is currently an associate professor at the Department of Computer Science. Dr. Safa is also associated with the Mobile Computing and Networking Research Laboratory (LARIM), Ecole Polytechnique de Montreal, Montreal, Canada. His research interests include mobile and wireless networks, mobility management, distributed computing, quality of service, routing, network security, web services, and cloud computing. Dr. Safa's research interest include, but are not limited to, mobile and wireless networks, mobility management, distributed computing, quality of service, routing, network security, web services, cloud computing, mobile computing, multimedia and sodtware engineering problems with networking applicability.



Mehedi Masud received his PhD in Computer Science from the University of Ottawa, Canada. He is an Assistant Professor at the Department of Computer Science, Taif University, KSA. His research interests include issues related to P2P and networked data management, query processing and optimization, eHealth, and information security. He has published several research papers at international journals and conferences. He has served as a member of the technical committees of several international conferences and workshops. He is on the editorial board of several journals including Journal of Internet and Information Systems (JIIS), Journal of Engineering and Computer Innovations, and Journal of Software (JWS). He served as a guest editor for Journal of Computer Science and Information Science (ComSIS).