Distributed Contextual Information Storage using Content-Centric Hash Tables

Ignacio Nieto, Juan A. Botía, Pedro M. Ruiz, Antonio F. Gómez-Skarmeta

Departamento de Ingeniería de la Información y las Comunicaciones. Universidad de Murcia, Spain*

Abstract. We analyze the problem of storing contextual information across a set of distributed heterogeneous mobile devices. These devices form a Mobile Ad Hoc Network (MANET) interconnected to Internet, in which partitions may eventually happen due to mobility. We present a new mechanism called Contextual Hash Table (CHT), which uses the semantic of the contextual information to select which nodes will store that information. Unlike previous general-purpose data replication algoritms, CHT manages to store information in the area of the network in which that information is most likely needed by neighboring nodes.

1 Introduction

Future mobile applications and services are expected to adapt and interact with their environment, providing a richer set of services to end users. This kind of applications are usually known as context-aware applications, or even ubiquitous applications. These applications use information about their environment (known as *contextual information*) such us user preferences, location, device capabilities, etc. Based on this information, they can provide a wide range of services like adapting contents to the capabilities of the user's device, providing information about nearby objects, etc.

The management, adquisition, representation and storage of contextual information, defined as any information about a user and its sorrounding, becomes a central aspect of any ubiquitous computing platform. For the representation and adquisition of context, we already developed an Open Context Platform (OCP) which uses an ontology represented using Ontology Web Language (OWL). In this paper, we focus on how to store an access that contextual information in a distributed network consisting of mobile heterogeneous devices.

Mobile Ad Hoc Networks (MANETs), consist of a number of mobile devices equipped with a wireless interface, making them very flexible and easy to deploy. These networks are distributed in nature, and do not rely on any centralized entity. Nodes are free to move, and they use neighboring nodes as relays, to send data to nodes which are not within their

* This work has been partially funded by the ENCUENTRO (00511/PI/04) research project of the Seneca Foundation and the CICYT research project FUZZY-KIM (TIC-2002-04021-C02-02) of the Spanish Government.

radio range. All these properties, make ad hoc networks an ideal candidate to support ubiquitous computing environments. MANETs can also connect to fixed networks like the Internet, creating a so-called hybrid MANET. Howerver, the network may be partitioned and merged at any time due to mobility. Thus, every service within the MANET must work in a distributed way whether there is Internet connectivity or not.
Throughout the rest of the paper, we assume an ubiquitous computing scenario consisting of a set of users and its corresponding devices. Devices are heterogeneous and may have different computing capabilities. In addition, devices will form an hybrid MANET, providing connectivity among devices. As in any MANET, the network may get temporarily partitioned, preventing the communications among some set of nodes. Devices will be responsible of obtaining contextual information, and store it within the network (eventually at different nodes). In addition, a device will be able to locate and access contextual information stored in any other node. In this paper we introduce CHT, a novel approach based on hash tables toefficiently deal with contextual information in such an scenario.

2 Related work

Our concern here is related to information dissemination however, context information is not needed to be known by everyone in the system. Instead, this kind of information has to be available on demand (i.e. context information refers to dynamical situtation of users and neither a node needs to know the context fo all users nor all nodes need to know the context of a single user). Hence, here we are not referring to dissemination to all nodes but to a few of them, appropiately selected. Moreover, the kind of network we have is a mobile ad hot network (MANET). Both of these facts (i.e. ad hoc and device mobility) introduce additional complexity to the problem. However, there are examples in the scientific literature which study gossiping [1] and multicast [5] in MANET. In our context, data dissemination consists on storaging and accessing data by name. This data tokens will be located in a concrete node (and perhaps in a number of additional ones). These nodel will usually be mobile, hence location of data tokens depend on location of hosting nodes. One interesting approach using geographical information for dissemination of information is the GLS (Grid Location Service) [10]. The GLS system uses GPS information for location and information dissemination. The approach we present in this article uses indoor location information, based on the Wifi network of the building [14]. Recent work which store and access data tokens by name can be found in [12, 13]. Storing and accessing data by name implies using a mechanism based on a hash table [11]. The scenario that the authors are focused on consists on a sensor network and they use a distributed has table inside a technique called *Data Centric Storage* or *DCS*. In DCS, each node is target of some data identified by means of a hash function. We will call this node the *home node* of the data. Whenever an event is generated, the sensor that detected it (we can think of this sensor as the "source" of the event)

sends this event to its *home node*, or the nearest node in case the *home node* is not available. If a concrete contextual information needs to be accessed, the hash function has to be applied to obtain its *home node*. Both operations can be done in $O(\sqrt{n})$.
We may also find several studies about data distribution and management in Ad Hoc networks. Data caching in an Ad Hoc environment has been treated in [3], where data transmitted in the network and the routes used to transmit it are both cached in the path of the information. In [4], an integrated data lookup and replication scheme is presented where a predictive mechanism is used to predict likely group partitioning of nodes. ADS [2] is a middleware service for Ad Hoc environments to share and receive information from the neighbourhood applied to CARLA, a M-Learning scenario. This approach relies on *information markets*, well-known places where different kinds of information from multiple applications can be pooled and exchanged. Regarding to data replication, in [6], a peer-to-peer service model for managing, searching, and streaming partitioned media objects is described, paying special attention to replication strategies for media segments. Data replication has also been treated in works such as [8] as the right approach for managing data avaliability in Ad Hoc networks, where the topology gets modified unpredictably mainly due to the mobile nature of the nodes.

3 Problem statement

We are interested in ubiquitous computing scenarios in which users, user different heterogeneous devices, and communicate among them wirelessly without the need of any centralized authority. This scenario, which is the most typical case, is characterized by its intrinsic mobility. Users will use a MANET providing connectivity among them, and eventually to applications and services over the Internet. Although Internet connectivity cannot be assumed to be always present. Our interest is in designing efficent contextual information distribution algorithms for these kind of scenarios.

3.1 Assumptions and requirements

The particular problem of information distribution in ubiquitous computing scenarios based on MANETs has some distinguishing features from traditional scenarios. Hence, we work under the following assumptions. (1) We assume that user devices are heterogeneous in terms of capabilities and computing capacity. They can be laptops, PDAs, mobile phones, etc. (2) The environment is dynamic. Devices can show up and leave the system at any time, even without prior notice (e.g. due to network disconections, battery exhaustion, etc.). (3) Users are mobile. They can freely move throughout the ubuiquitous scenario. This places an important requirement onto our system, which should be able to deal with that mobility of the users and the eventual partitions of the underlying network. (4) The underlying ad hoc network has limited resources. The proposed solution shall try to minimize the consumption of network

resources by placing the information in appropriate locations. (5) Every node in the system has an unique identifier.
Within the rest of this section we describe design alternatives for our solution.

3.2 Design alternatives

As we stated before, our goal is to efficiently distribute contextual information. That is, information about entities, objects and locations of the environmentin which the users are moving around. Every device in the system will have an unique identifier. Contextual information will be generated in a concrete node, and the system must assign a *home node* where to store it.
There is a wide range of alternatives for distributing information in such an ad hoc scenario with our requirements. One extreme is storing every contextual information in the node that generated it, taking order O(1) in terms of network messages. In this case, queries from the rest of nodes to obtain this information will require a broadcast in the entire network (we don't know in which node the data is a priori). So the cost of accessing the information requires O(n), where n is the number of nodes in the network. That is, n messages are needed for the broadcast plus $\sqrt{n}$ for sending the answer back to the querier, being $\sqrt{n}$ the mean path length.
Another extreme in the range of solutions is storing all the contextual information generated by any of the nodes replicated in every node in the network. In that case, performing the storage of each data requires a broadcast, which has a cost of O(n) in terms of the number of messages. In this case, queries can be resolved locally in order O(1).
There are a number of other intermediate solutions based on the idea of replicating the contextual data only in a set of nodes, but reaching a good tradeoff is not an easy task. In addition, almost in all these cases, the mechanisms require a broadcast operation, which tends to be costly for these environments, even if broadcast reduction techniques like common dominating sets (CDS) are used. Thus, we target a different approach based on hash tables and a hierarchical structure to achieve a better solution. In addition, by distributing the date based on their content, we can better select *home nodes* to reduce the cost in most common operation scenario.
As we mentioned, we will use a different approach based on a hash function to distribute the information across the nodes. Figure 1 shows a typical layout of a set of devices in the MANET, grouped by their location. In our example, we assume we are inside a building, and nodes are distributed all over the building. Nodes form an Ad Hoc network that may or may not have connectivity with external networks like the Internet. If there exists such connectivity, then we may have also one SCPP taking control of a group of areas (e.g. nodes in certain rooms). We will denote by n_i the diameter (expressed in number of nodes in the longest path) of the area i. We also denote by N the diameter of the whole area controlled by the SCPP. Every time we need to know the *home node* of a given contextual information (in case we need to update this information or we need to access it) we will call a hash function returning the *home*

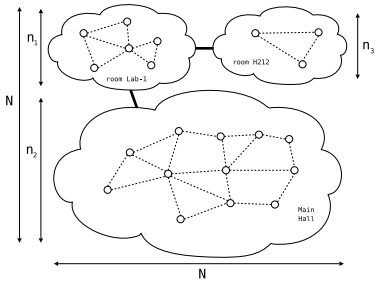


Fig. 1. Typical layout of our environment using CHT.

node of this information. This operation can be performed in order O(1). If the *home node* of that information lies in the same room or area we can obtain the information with two messages (answer and response). Which means in $O(\sqrt{n_i})$ transmissions of each intermediate node because $\sqrt{n_i}$ is the mean path length for nodes within the area n_i. In the worse case in which the route to the *home node* is not kwnon, a limited broadcast can be used, having a cost of $O(n_i)$ messages. Our hash function will be designed so that contextual information will be placed close to the area in which there are more nodes interested in. Thus, the access cost will be reduced. In the unlikely case in which a node would be interested in contextual information of other areas, the access time would be $O(\sqrt{N})$ in the average case and $O(N)$ in the worst case,in which we don't know a route to the destination and we need to flood the whole area.

4 CHT: Contextual Hash Table

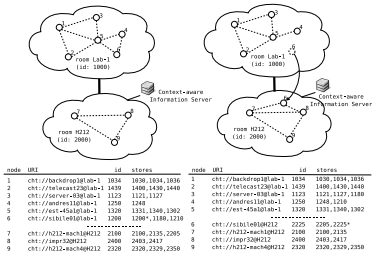
This section describes in detail the CHT algorithm for distributing contextual information represented according to our Ontology Context Platform (OCP). In section 4.1 we describe the algorithm in detail, showing concrete examples. In adition, in section 4.2, we discuss the issue of replication and we introduce some replication mechanisms for CHT to guarantee the avaliability of data it handles even when mobility and node failure rates increase.

4.1 The CHT algorithm

As we stated before, each node producing, consuming or storing contextual information in the system will have one unique identifier. Provided that our OCP works with OWL-represented onlogies, in which identiers are URIs, our identifiers will have the following format:

```
URI = cht://resourceDescription@physicalLocation
```

where *resourceDescription* is the description name of the node (a.k.a. resource) that must be unique in the system. Additionally, *physicalLocation* is the name of the physical area in which the node is currently located. In case a node does not know its physical location, it will use "unknownLocation" as default. Some examples of CHT URIs are `cht://backdrop1@room1` and `cht://printer2@labH44`. Besides its URI, each node in the system will generate a unique identifier expressed as a long integer. This identifier will be temporal, and will be regenerated if the user changes from one location to another (i.e: the user changes to another room in the building). Given that users are most interested in the contextual information related to their current location, our CHT will mostly consider the location as the most important piece of contextual data, when selecting *home nodes*. Thus, contextual information is expected to be stored in the same area in which the user is. In fact, location is the most important contextual information in mobile context-aware scenarios[9, 7]. Hence, the location of the user will be the dominating factor when selecting its temporal identifier. So, the higher order bits of the identifier will be taken from the location of the node (e.g. identifier of the room obtained from the ontology). The rest of the identifier will be formed from the context of the user or node, its URI and/or the arriving order of the nodes to the room. This identifier (a long integer, such as 25000) will be used as a reference to discriminate the contextual information that must be stored at the user or node. That's to say, given the identifier returned from the hash function, the *home node* for that contextual information will be the one whose hash value equals or is closer to that identifier than any other node. According to our requirements, the identifiers of two nodes that are in the same area will always be closer than the identifiers of two nodes in different areas.



node	URI	id	stores
1	cht://backdrop1@lab-1	1034	1030,1034,1036
2	cht://telecast2@lab-1	1439	1400,1430,1440
3	cht://server-03@lab-1	1123	1121,1127
4	cht://andres1@lab-1	1258	1248
5	cht://est-45a1@lab-1	1320	1331,1340,1302
6	cht://sibille01@h212	2225	2205,2225*
7	cht://h212-mach1@H212	2100	2100,2135,2205
8	cht://impr32@H212	2400	2403,2417
9	cht://h212-mach4@H212	2320	2320,2329,2350

node	URI	id	stores
1	cht://backdrop1@lab-1	1034	1030,1034,1036
2	cht://telecast2@lab-1	1439	1400,1430,1440
3	cht://server-03@lab-1	1123	1121,1127,1180
4	cht://andres1@lab-1	1258	1248,1210
5	cht://est-45a1@lab-1	1320	1331,1340,1302
6	cht://sibille01@H212	2225	2205,2225*
7	cht://h212-mach1@H212	2100	2100,2135
8	cht://impr32@H212	2400	2403,2417
9	cht://h212-mach4@H212	2320	2320,2329,2350

Fig. 2. CHT operation.

Invoking the hash function with a cocrete contextual information will give us a hash value dependent, as said before, on a concrete contextual information and a source URI. We will use this value to identify the *home node* of that information by means of a simple check of integer proximity, computing the minimum:

$$\min_{1 \leq i \leq n} abs(id - id_i)$$

where id_i is the identifier obtained from node i, id is the hash value obtained for the contextual information and N the total number of nodes. Using an ordered distributed identifiers table, this could be done in small time.
Figure 2 shows a concrete example of the CHT algorithm operation. Our example consits of two interconnected rooms: "Lab-1" and "room H212". Each room has several nodes we have labelled from 1 to 9. Each node has a unique URI and an identifier it generated when it entered the room. Each node stores contextual information we have identified by its hash value. Both identifiers and hash values are represented by long integers. This way, node i stores the information with hash value v if there is no other node j with a nearer identifier to v. That is to say:

$$home - node(context) = i \Longleftrightarrow \nexists j | abs(j - v) < abs(i - v)$$

Notice that the six nodes of the room "Lab-1" have their identifier closer to 1000 (the identifier value associated to that location) because the location is the most important factor when assigning node identifiers. Similarly, the three nodes of the room "room H212" have their identifiers close to 2000. Down in the same figure we can see the hash values of the information they currently store. In the situation depicted on the upper part in the figure, node 6 leaves room "Lab-1" to enter room "room H212". Before leaving the room, node 6 announces their neighbours of its intentions, and transmits to them the contextual information that must stay in the room. The node has at that moment three bits of information: local information concerning its own preferences and the device characteristics, data that has no more relevance for the nodes of the room "Lab-1", and two contexts of interest for the nodes of the room (identified with hash values 1180 and 1210). These two bits of information must be kept in the room, so node 6 distributes them to their nearby nodes, nodes 3 and 4 respectively.

Upon entering the room "room H212", node 6 modifies its URI from `sibile01@lab-1` to `sibile01@H212`, and its identifier also gets modified from 1200 to 2225. Its contextual information, now adapted to show the change in location (besides any other contextual change occurred in the process), has now been labeled with hash value 2225. The node then performs a broadcast to any other node in the new room announcing its presence. The nodes then add its URI and identifier to their tables and, checking their own contextual informations, node 7 notices the contextual information with hash value 2205 it was storing is now nearer to the new node, so it transmits this information to node 6.

With CHT integrated with OCP, the contextual information is relocated as the users move throughout the building. By definition, neighboring nodes will also have close identifiers. When a user leaves a room and en-

ters another (see Fig.2), it passes the room-dependent information (contextual information about objets, people or devices in the room) to other nodes in the room, dropping this information if is not of its interest now. When the node enters the new room it receives a new dynamic identifier, and its new neighbours determine wich information should the new node store. The node is now the new "nearby" node to some other nodes and the *home node* of some contextual information. Similarly, the contextual information of the room has changed with the entrance of the new node, and this newly generated contextual information will be distributed and replicated over the nodes of the network as exposed previously. With CHT, the contextual information moves through the building continuously as a distributed informational entity.

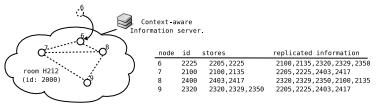


Fig. 3. Data replication in CHT.

4.2 Replication of Contextual Information

To enhance the basic operation described so far, we use a replication mechanism to guarantee that contextual information is not lost due to node failures or network partitions. The CHT replication mechanism consists of maintaining copies of the contextual information of every node in their upper and lower nearby nodes (in terms of their identifiers). As an example, consider the figure 3, where node 9 will store apart from its own contextual information, that of nodes 6 and 8. This information will be stored as "replicated information", and kept apart from the contextual information that has node 9 as *home node*. The "border nodes" (i.e: nodes that have a minimum or maximum hash value so are missing a upper or lower nearby node, like node 8) will replicate their information on their only nearby node and the other only node with only one neighbour (the node with maximum identifier will store the contextual information from the node with minimum identifier and viceversa). As an example, figure 3 shows that node 8 has no upper nearby node, so it will replicate its contextual information on nodes 6 (its lower nearby node) and 7 (the node with minimum identifier). This mechanism works remarkably well and shows great scalability as number of nodes in the system increases. The replication schema must be taken into account specially when a node enters or leaves the system. In the example shown in figure 3, when node 6 enters room H212, it stores its own contextual information now identified with hash value 2225, receives the contextual information identified with hash value 2205 from node 7 as *home node* contextual information,

and stores the contextual information of nodes 7 and 9 as "replicated information". Notice that node 6 will only receive the target information of nodes 7 and 9 (specifically contextual informations identified with values 2100, 2135, 2320 , 2329 and 2350), not their replicated information (like the replicated information node 9 stores from node 8, its upper nearby node). Similarly, when a node exits a room, the nodes must drop or redistribute both their target contextual informations and their replicated contextual information properly.

With this mechanism, if a node fails or leaves unexpectedly, its contextual information is not lost, because their nearby nodes already possess that information, and they only must set that replicated information as target information if they are now the *home nodes* of that information. They will detect it because the previous *home node* is no longer reachable. As an example, we assume that node 9 in figure 3 dies unexpectedly due to an energy failure. When the node failure is detected, the remaining nodes inmediatly adjust their contextual identifier tables to reflect the missing node, and nodes 7 and 8 (the nearby nodes of node 9) re-evaluate the replicated information they are storing from node 9 (i.e: contextual information labeled with hash values 2320, 2329 and 2350). All this contextual informations are now nearer to node 8 than 7, so node 8 happens to be the new *home node* of them, so it sets them as target information, dropping them from their replicated information area. Node 8 also sends this three contextual informations to node 6, its lower nearby node, that node 6 stores as replicated information.

Now we will calculate the chance of loosing a concrete contextual information with our replication method. We will call N the number of active nodes in some area of the system, and we may assume a probability of p of a node failure without loose of generallity. In order for a contextual information to get lost all three nodes carrying that information (its *home node* and their two nearby nodes) must fail simultaneously. The probability that this will happen is $(\frac{p}{N})^3$. As an example, with a N value of 100 nodes and a failure probability of 5% we have a probability of $1,25E^{-10}$ of a contextual information loose. This schema guarantees with a high probability that our contextual information will be always available to users, an important factor if we want to handle a trusted distributed contextual storage and management system.

5 Conclusions and future work

In this paper we present a contextual information distribution algorithm that distributes the information according its semantic content. CHT (Contextual Hash Table) approachs the problem of information distribution with a hash table strategy, where the information is distributed according to its contextual significance. A replication mechanism guarantees that the information will always be available to other users, and the hash table approach offers a balanced storage of the information and a reduced access time, both for storage and recovery of the contextual data.

References

1. N. Bauer, M. Colagrosso, and T. Camp. An efficient approach to distributed information dissemination in mobile ad hoc networks. Technical Report Technical Report MCS-04-01,, The Colorado School of Mines, February 2004.
2. Matthias Brust, Daniel Görgen, Christian Hutter, and Steffen Rothkugel. Ads as information management service in an m-learning environment. *Knowledge-Based Intelligent Information & Engineering Systems*, 2004.
3. Guohong Cao, Liangzhong Yin, and Chita R. Das. Cooperative cache-based data access in ad hoc networks. *Computer, vol. 37, no. 2*, pages 32–39, February 2004.
4. Kai Chen and Klara Nahrstedt. An integrated data lookup and replication scheme in mobile ad hoc network. 2004.
5. M. Hauspie, A. Panier, and David Simplot-Ryl. Localized probabilistic and dominating set based algorithm for efficient information dissemination in ad hoc networks. In *Proceedings of the 1st IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS'04)*, Fort Lauderdale, Florida, USA, 2004.
6. Shudong Jin. Replication of partitioned media streams in wireless ad hoc networks. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 396 – 399, New York, NY, USA, October 2004.
7. F. Bennet D. Clarke J.B. Evans A. Hopper A. Jones and D. Leask. Piconet: Embedded mobile networking. *IEEE Personal Communications, 4(5)*, pages 42–47, 1997.
8. Goutham Karumanchi, Srinivasan Muralidharan, and Ravi Prakash. Information dissemination in partitionable mobile ad hoc networks. In *Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems*, page 4, Washington, DC, USA, October 1999.
9. R. H. Katz. Adaptation and mobility in wireless information systems. *IEEE Personal Communications, 1(1)*, pages 6–17, 1994.
10. Jinyang Li, John Jannotti, Douglas S. J. De Couto, David R. Karger, and Robert Morris. A scalable location service for geographic ad hoc routing. In *Proceedings of the 6th ACM International Conference on Mobile Computing and Networking (MobiCom '00)*, pages 120–130, Boston, Massachusetts, August 2000.
11. W. W. Peterson. Addressing for random access storage. *IBM Journal of Research and Development*, 1(2):130–145, 1957.
12. Sylvia Ratnasamy, Deborah Estrin, Ramesh Govidan, Brad Karp, Scott Shenker, Li Yin, and Fang Yu. Data-centric storage in sensornets. *SIGCOMM 2002, February 1st*, 2002.
13. Sylvia Ratnasamy, Brad Karp, Scott Shenker, Deborah Estrin, Ramesh Govindan, Li Yin, and Fang Yu. Data-centric storage in sensornets with ght, a geographic hash table. *Mobile Networks and Applications*, 8:427–442, 2003.
14. Ignacio Nieto Carvajal Juan A. Botía Blaya Pedro M. Ruiz Antonio F. Gómez Skarmeta. Implementation and evaluation of a location-aware wireless multi-agent system. *Proceedings of the International Conference on Embedded and Ubiquitous Computing*, 2004.