Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 73 (2015) 130 – 137

Procedia Computer Science

The International Conference on Advanced Wireless, Information, and Communication Technologies (AWICT 2015)

# A distributed and flexible architecture for Internet of Things

Ghofrane Fersi[a]

[a]*Research Laboratory of Development and Control of Distributed Applications (ReDCAD)*
*Department of Computer Science and Applied Mathematics*
*National School of Engineers of Sfax BP 1173-3038 Sfax*
*University of Sfax, Tunisia*

**Abstract**

The great breakthrough in technological devices has lead to the migration of actual Internet to the Internet of Things. There are merely trillions of smart dynamic objects that will be connected to the Internet and that interact and collaborate together independently from any physical location. This progress necessitates the proposal of an adequate architecture and routing process. In this paper, we propose a novel design and overlay architecture that fulfills the Internet of Things requirements. This approach is mainly based on Distributed Hash Table protocols to afford the required flexiblity and to handle efficiently mobility and churn cases. We present also a formal study that proves the efficiency of our proposed architecture.

*Keywords:*
Internet of Things; architecture; design; DHT

## 1. Introduction

Since ICT sector is rapidly and significantly proliferating, an increasing number of smart devices is connected to the Internet. Actually, not only computers or smart phones are connected to the Internet but also glasses, lenses, cars , areas, etc [1,2]. Internet is then invading our real lives and offering us a universal accessibility of any given thing in our life. Hence, there is a shifting from the current Internet to the Internet of Things. There are lot of approaches that have been proposed in order to present a suitable architecture for the Internet of Things. However, these solutions do not fulfill all the Internet of Things requirements. Effectively, some solutions such as [3,12,13], have given a general study for their proposed design in different cases but their proposed architecture is based on IPV6 addressing. However, this addressing type can be applicable in some devices but it is not relenvant in all devices since there are lot of existing devices that do not support it. Also, it is not feasible to embed the IP protocol to some ressource constrained things.

*E-mail address:* ghofrane.fersi@redcad.org

doi:10.1016/j.procs.2015.12.058

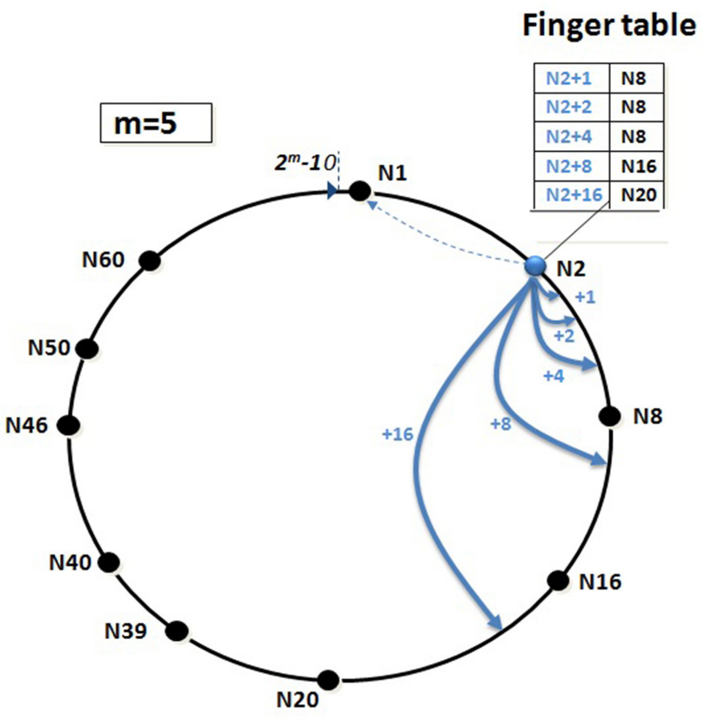

Fig. 1. Chord system

This makes the migration of the current Internet to the Internet of Things a real challenge.

In this article, we present a new approach for use in the Internet of Things. It includes a complete architecture that integrates things identifiers assignment, routing, churn and mobility management. Our paper is organized as follows. Section 2 presents our research background. We present our proposed approach in section 3. We present in section 4 a formal study in order to prove the efficiency of our proposed approach. Section 5 concludes the paper.

## 2. Background

Since our proposed architecture is mainly related to Chord[4,5] and Virtual Cord Protocol (VCP)[8], we give in this section a brief overview of these two protocols.

### 2.1. Chord

Chord[4,5,6] is one of the most well-known and most used Distributed Hash Table system. Chord maps nodes identifiers and keys onto the same virtual ring. This is ensured by a hash function (for example SHA-1) that generates m-bit unique identifiers for both nodes and keys. Keys are obtained by hashing data identifiers. Identifiers can be obtained by hashing the node's IP or MAC address. Keys are spread onto nodes in a balanced way. The key k is assigned to the node having the identifier that is equal or follows k. This node is the successor of k. Nodes identifiers are organized in an identifier modulo $2^m$ as depicted in figure 1. Successor(k) is the first node clockwise from k. To ensure lookup efficiency, each node maintains a list of m entries in a table called finger table. The ith entry of a peer n contains the successor $n + 2^{i-1}$ where $1 =< i <= m$.

To find the successor of a given key, the current node picks from its finger table, the node having the closest identifier to that key and forwards the query to it. Recursively, the successor of the searched key is reached in O(log N) hops, where N is the total number of nodes.

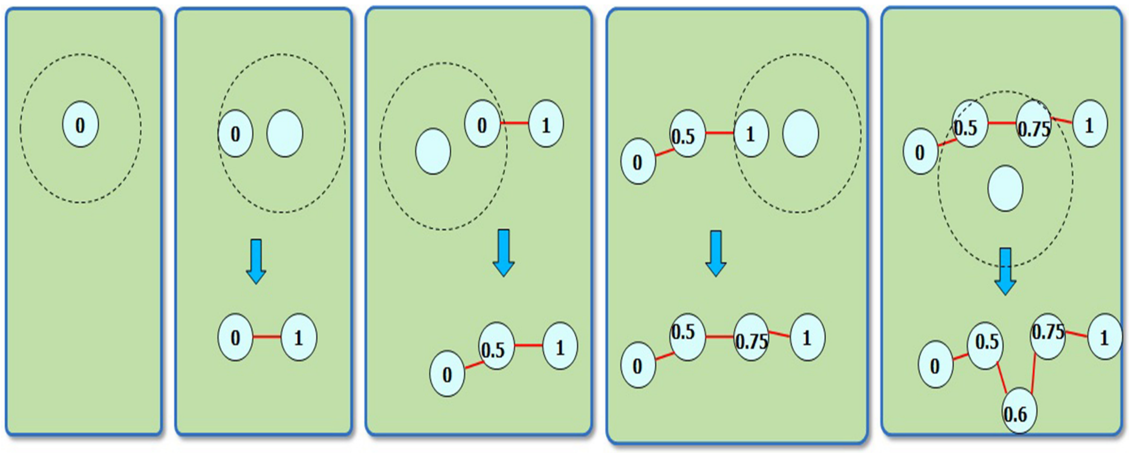

Fig. 2. VCP nodes joining

Chord takes into account nodes joining and departure. When a node joins the network, it initializes its finger table as well as its predecessor set. The Finger tables and predecessors of affected node are also updated to take into account this node joining

### 2.2. Virtual Cord Protocol (VCP)

Virtual Cord Protocol[8,9,10] is a virtual location based protocol. The main specificity of VCP is that virtual and physical neighbors in VCP are the same. When a node joins the network, it starts by broadcasting HELLO messages. Other nodes broadcast periodically HELLO messages. Based on the information sent in these HELLO messages by neighboring nodes (neighbors identifiers and their position in the cord), the joining node computes its relative position in the cord. This relative position is also the node identifier.

Figure 2 explains the joining procedure of five nodes. The two first nodes present the cord limits i.e. all nodes identifiers are included in [0,1]. The five steps are presented to include the five nodes in the cord.

Routing tables in VCP only store information about physical neighbors. Their updates requires only HELLO messages of neighboring nodes.Greedy routing is applied to route messages in VCP. In each routing step, the physical neighbor having the closest identifier to the final destination is elected as next hop. The path length in VCP is $O(\sqrt{N}-1)$.

## 3. Proposed approach

We present in this section our proposed architecture for the Internet of Things and we specify protocols that should be used to suite this architecture. Since smart objects are placed massively in every place in the world, it is beneficial to partition the network into two main parts: the first part is made up of access points and the second part is made up of smart objects. Access points have an important memory size and important processing capabilities. They are static and do not have any energy constraints since they have access to power. This enables them to have an extended coverage. Each access point is responsible of smart objects in its coverage. It stores important information related to these smart objects.

### 3.1. Identifier assignment

Each node should have a location-independent identifier in order to be reachable wherever it is in the world. The identifier can be the hash of the MAC address. We call it *global identifier*. When a node belongs to a local network and it interacts with other nodes in the same network it needs a *local relative identifier* so that nodes belonging to

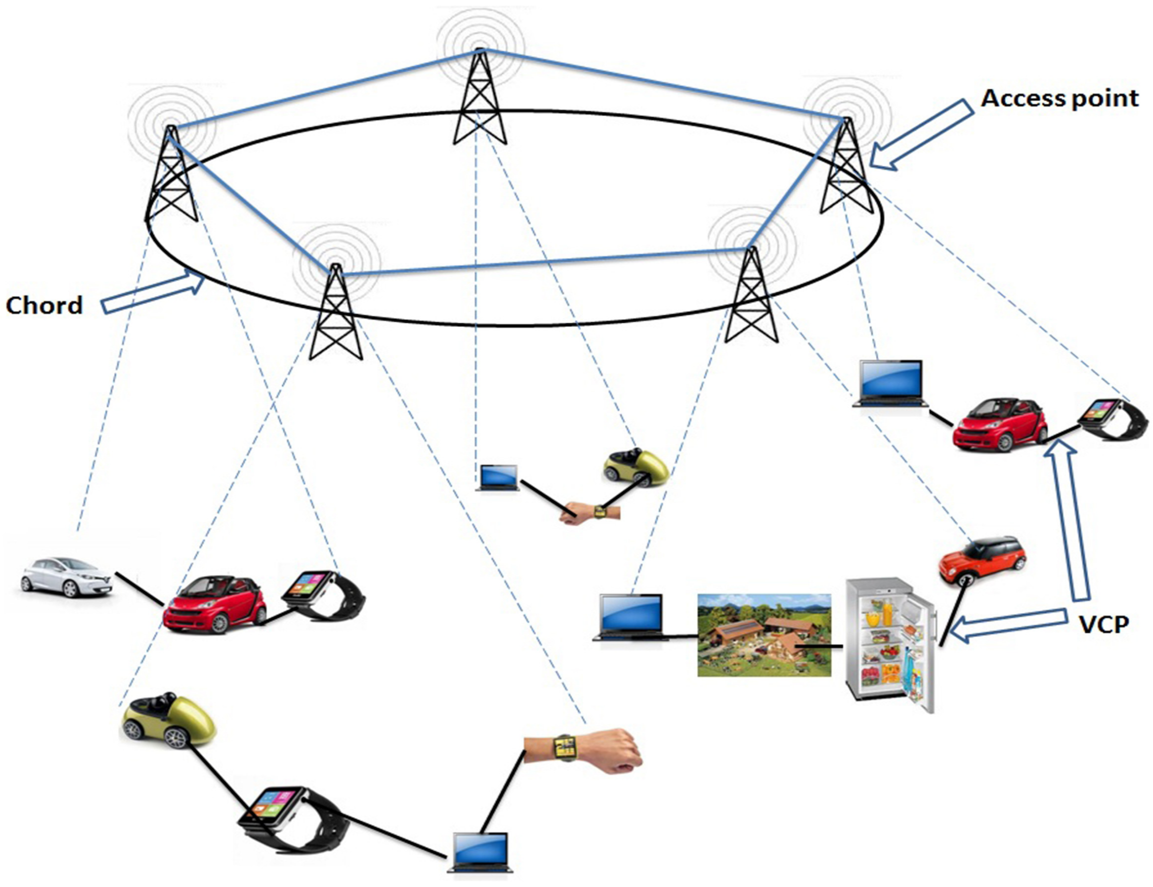

Fig. 3. Internet of Things proposed architecture

the same network can communicate with it easily and without the need to other external networks. When this node changes its physical location, this local identifier should be updated.

Local identifiers should not be related to the geographical coordinates of the node because this requires that nodes are equipped with positioning system such as GPS. However, such systems are costly and cannot be applied in all nodes. A suitable and economic solution is to assign *virtual position based identifiers*. This means that each node when it joins the local network computes its identifier in a way that it will be at the same time physical and virtual neighbor for the nodes in its range. VCP is a suitable solution since it computes relative identifiers in a distributed way. The entire procedure can be found in[8].

## 3.2. *Routing*

Each node in our architecture has a VCP routing table that maintains information about its physical/virtual neighbors. Each access point maintains two additional tables: addresses table containing the global identifiers of smart objects that belong to it and their corresponding local identifiers, and a Chord finger table containing the global addresses of m other access points.

### 3.2.1. *Routing in local networks*

If the source and destination nodes belong to the same local network i.e. they have the same access point, a virtual position-based routing protocol is applied to route messages in this local network. We use VCP because of its scalability and its ability to route efficiently messages between nodes placed on the vicinity of each others and because it obviates the growing size of routing tables since each node in VCP stores in its routing table only information about its physical neighbors.

### *3.2.2. Inter-access point routing*

To route messages between access points, we use Chord since it offers efficient data packets routing without the need of any location information. When an access point receives a message, it checks if the local destination address belongs to its own network. If this is the case, it uses VCP routing protocol to deliver the message to its final destination.

Otherwise, the access point responsible of this destination node should be searched. To achieve this, the current access point picks from its finger table the access point having the closest identifier to the global identifier of the destination node and forwards the message to it using Chord. When the current access point picks from its addresses table an entry having the information about the local identifier of the final destination, it applies VCP in its local network in order to deliver the message to its final destination.

## *3.3. Churn handling*

The churn is the continuous and massive nodes joining and departure. This phenomenon is frequent in Internet of Things. That's why an adequate protocol should be employed to handle churns efficiently. When a node joins the network, it starts by broadcasting HELLO messages containing its global identifier. Each node that receives this message, responds it by a Hello response message containing its local and global identifiers as well as the identifier of the access point at which it belongs. If the joining node receives only messages from nodes belonging to the same network (having the same access point identifier), it joins this network using VCP joining procedure and computes its local identifier. Then it sends a notification message containing its local and global identifiers, to the access point responsible of this network using VCP. When the access point receives the notification message, it adds the pair (global address, destination address) of the joining node to the addresses table.

If the joining node receives HELLO response messages from nodes having different access point identifiers, it chooses to join the local network having the minimal number of connected nodes, in order to ensure load balancing between access points. If there are lot of nodes that join the local network simultaneously, the recursive DHT-based bootstrapping protocol[11 7] can be employed. It orchestrates the nodes joining in a sequential and totally distributed way to avoid concurrent joining. This is achieved by organizing nodes in a bootstrapping tree and allowing nodes joining from parent to children in a distributed way. This protocol should be slightly modified to be applicable with VCP protocol. Parent nodes when they receive HELLO messages from their children for the first time, assign a local identifier to each of them and place them to their waiting list, then send permission message to the first node in this waiting list to allow it to join the network as described above. When a parent does not receive any HELLO response message, it sends a reallocation message to its parent to allow other children nodes to join the network.

Each node in our architecture broadcasts HELLO messages every period of time T. Neighboring nodes maintain the set of their physical neighbors. When a node fails or changes its physical position, its physical neighbors do not receive HELLO messages from it. After a given period, the neighboring nodes consider that node as failed and remove it from their routing tables. The immediate successor of the failed node sends a notification message to its corresponding access point to remove the failed node from the addresses table.

## *3.4. Mobility*

Our proposed approach uses a novel mobility management protocol called MOBYDHT. MOBYDHT differentiates between two mobility cases:
1.When the mobile node still belongs to the same local network (still belongs to the same access point), MOBYDHT uses the same procedure defined in VCP to update the local identifier of the mobile node. Once the identifier is updated, the mobile node sends a notification message to its access point in order to update the local identifier corresponding to the mobile node.
2. When the mobile node does not receive HELLO messages from its local network nodes, this means that it should change its corresponding local network and its corresponding access point. To realize this, it hears HELLO messages from its neighboring nodes. These HELLO messages contain information about the corresponding access point and

the number of nodes that belong to the local network of the source node . The moved node joins the local network having the minimal number of nodes, computes its new local identifier, then sends an association message to its new access point.

The old neighboring node detect the departure of the mobile node when they do not receive any HELLO message from it during a given period T. They consider it as failed and repeat the same procedure at the case of node failure.

## 4. Formal study

We evaluate in this section the performance of our proposed architecture. We start by specifying the total number of hops needed to deliver a message from a source to a destination when these nodes belong to the same local network, then when these latters belong to different local networks. After that, we estimate the time needed to deliver these messages.

### 4.1. Hops number

When the source and the destination nodes belong to the same local network, they only apply VCP to route messages, this means that the mean hops number to reach the final destination is $\sqrt(M) - 1$ where M is the number of smart objects in the local network.
If the source and destination nodes belong to two different local networks, they should use Chord to search the access point that is responsible of the final destination, then, when this access point is reached, VCP is applied to reach this final destination. That is the average hops number needed to reach the final destination is :
log (N)+ $\sqrt(M) - 1$ where N is the total number of access points and M is the total number of smart objects in the local network

### 4.2. Time to deliver messages

The required time needed to deliver correctly a message is given by this equation:

$$MessageDeliveryTime = MessageTransmissionTime + PropagationDelay \quad (1)$$

We ignore in this equation the time of awaiting and processing. Message transmission time and propagation delay are as follows:

$$MessageTransmissionTime = Messagesize/Bitrate \quad (2)$$

$$PropagationDelay = Distance/Propagationspeed \quad (3)$$

### 4.3. Case study

In our proposed architecture, smart things are connected using Blootooth smart. In this standard, the bit rate is 1Mbit/s . The average smart objects communication range is 100 m. The distance between access points is set to 200 m. Since smart objects use wireless networks to communicate, their propagation speed is equal to the light celerity= $3.10^8 m/s$ . Access points in our proposed architecture are connected through a wired connection using copper wires. The propagation speed in these wires is $2.10^8 m/s$
The message size is 124 bits (destination address (20 bits)+ source address (20 bits)+ global identifier (30 bits)+ next hop (30 bits)+ message type (8 bits) + Cyclic Redundancy Check (16 bits)).
The message transmission time is then $0.124 * 10^{-3} s$.
The propagation delay in local networks is $3.33 * 10^{-7} s$ and in the case of routing in two different networks is $10^{-6}$.
Hence we have:

$$Time\,needed\,to\,deliver\,a\,message\,in\,a\,local\,network = (\sqrt(M) - 1) * (0.124 * 10^{-3} + 3.33 * 10^{-7}) \quad (4)$$

When the source and destination do not belong to the same local network, we have:

$$Time\,needed\,to\,deliver\,a\,message = (\sqrt(M) - 1) * (0.124 * 10^{-3} + 3.33 * 10^{-7}) + log(N) * (0.124 * 10^{-3} + 10^{-6}) (5)$$

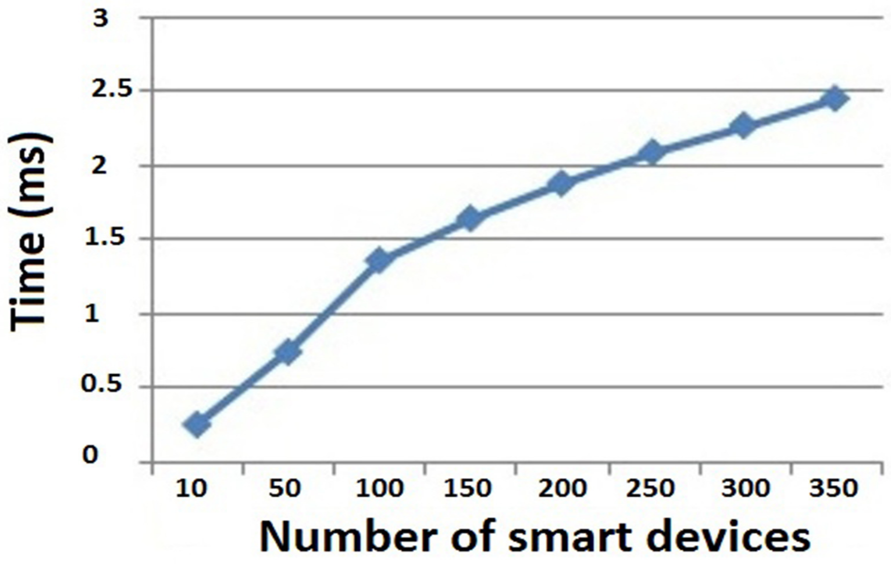

Fig. 4. Message delivery time in a local network

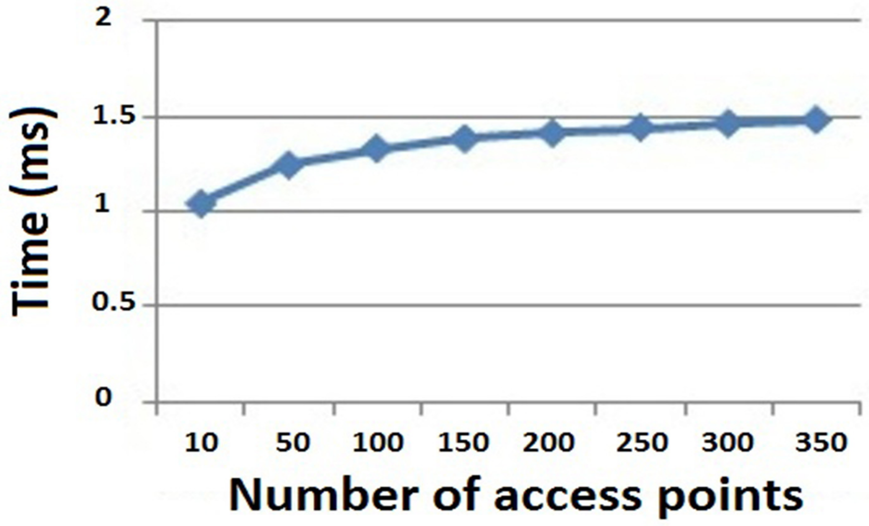

Fig. 5. Message delivery time when source and destination nodes belong to two different local networks

#### *4.3.1. Variation of the number of smart objects*

In the first study, we have varied the number of smart objects in a given local network and we have computed the average time to deliver a message. As depicted from figure 5, when the number of smart objects in a given local network increases, the time needed to deliver the message increases. This can be explained by the fact that when the number of nodes increases, the number of intermediate hops needed to reach the destination increases as well. This evidently increases the delivery time. However this figure shows that this increase is limited and the delivery time in all network sizes remains so limited and the message is received in a timely fashion which proves the scalability and the efficieny of our proposed architecture.

*4.3.2. Variation of the number of access points*

In the second case study, we have studied the case of extended networks where source and destination nodes do not belong to the same local network. This means that in order to deliver a message between these nodes, we need to use intermediate access points. Figure 6 shows clearly that even if the network is extremely extended, the delivery time remains negligible. This is because we have employed Chord to route messages between the different access points, which ensures an efficient and speed lookup leading to reach the destination quickly even if it is physically faraway from the source node.

## 5. Conclusion

In this paper, we have proposed a Distributed Hash Table-based architecture that fits well the Internet of Things specificities and challenges. The proposed design handles the main challenges facing the Internet of Things and ensures nodes and data accessibility independently from any location information in a timely fashion. In a future work, we aim to deploy our proposed architecture in an Internet of Things protoype to evaluate its performance.

## References

1. Jayavardhana G, Rajkumar B, Slaven M, Marimuthu P. Internet of Things (IoT): A vision, architectural elements, and future directions. Future Generation Computer Systems 29, 1645–1660(2013)
2. Daniele M, Sabrina S, Francesco D. P, Imrich C. Internet of things: Vision, applications and research challenges, Ad Hoc Networks 10, 1497–1516(2012)
3. Sungmin H, Daeyoung K, Minkeunha S. B, Sang J. P, Wooyoung J, Jae-eom K. SNAIL: AN IP-BASED WIRELESS SENSOR NETWORK APPROACH TO THE INTERNET OF THINGS. IEEE Wireless Communications (2010)
4. Stoica I, Morri, R, Karger D, Kaashoek M. F, Balakrishnan H. Chord: A scalable peer-to-peer lookup service for internet applications. In Proceedings of the ACM conference of the Special Interest Group on Data Communication, San Diego, CA (2001)
5. Wehrle K, Gtz S, Rieche S. Distributed Hash tables. In R. Steinmetz, K. Wehrle (Eds.): Peer-to-Peer systems and applications (Chapter 7, pp. 7993). Berlin, Heidelberg: Springer (2005)
6. Fersi G, Louati W, Ben Jemaa M. Distributed Hash table-based routing and data management in wireless sensor networks: a survey. ACM/Springer Wireless networks 19 (2), 219-236 (2013)
7. Fersi G , Louati W, Ben Jemaa M. Time estimation of a ring-based bootstrapping protocol in wireless sensor networks. LiveCity Workshop on Smart and Pervasive Communications for Enhanced Communities in conjunction with Saconet conference, Paris, France (2013)
8. Awad A, German R, Dressler F. Exploiting virtual coordinates for improved routing performance in sensor networks. IEEE Transactions on Mobile Computing, 10(9), 12141226 (2011)
9. Awad A, Sommer C, German R, Dressler, F. Virtual cord protocol (VCP): A flexible DHT-like routing service for sensor networks. In Proceedings of the 5th IEEE international conference on mobile ad hoc and sensor systems, Atlanta, Georgia (2008)
10. Awad A, Shi L, German R, Dressler F. Advantages of virtual addressing for efficient and failure tolerant routing in sensor networks. In Proceedings of the sixth IEEE/IFIP international conference on wireless on-demand network systems and services, Snowbird, UT (2009)
11. Fersi G , Louati W, Ben Jemaa M. Consistent and Efficient Bootstrapping Ring-Based Protocol in Randomly Deployed Wireless Sensor Networks Intenational Conference on telecommunications (ICT), Maroc (2013)
12. 6LoWPAN Working Group, http://tools.ietf.org/wg/6lowpan/
13. ROLL Working Group, http://tools.ietf.org/wg/roll/.