Dynamic address allocation for self-organised management and control in sensor networks

Falko Dressler* and Feng Chen

Autonomic Networking Group,
Department of Computer Science,
University of Erlangen-Nuremberg,
Martensstr. 3, 91058 Erlangen, Germany
E-mail: dressler@informatik.uni-erlangen.de
E-mail: feng.chen@informatik.uni-erlangen.de
*Corresponding author

Abstract: Several data-centric communication paradigms have been proposed in the domain of Wireless Sensor Networks (WSN). Therefore, the principles of operation and maintenance in such networks are changing to control massively distributed systems. Previous addressing schemes fail or produce too much overhead if only locally unique addresses of sensor nodes are required. In this paper, we present a dynamic address allocation scheme for localised address assignments in WSN. We developed a round-based address assignment with subsequent duplicate address detection that operates in a self-organised manner. It inherently allows busy-sleep periods and does not assume always awake nodes. To verify the approach, we implemented the algorithm on Mica2 sensor motes and tested it in a WSN maintenance scenario. The results demonstrate that our method works well for operation and maintenance of WSN without prior address assignments.

Keywords: duplicate address detection; dynamic address allocation; self-organization; on-demand sensor management; wireless sensor networks; WSN.

Reference to this paper should be made as follows: Dressler, F. and Chen, F. (2007) 'Dynamic address allocation for self-organised management and control in sensor networks', *Int. J. Mobile Network Design and Innovation*, Vol. 2, No. 2, pp.116–124.

Biographical notes: Falko Dressler is an Assistant Professor leading the Autonomic Networking Group at the Department of Computer Sciences, University of Erlangen-Nuremberg. He teaches on self-organising sensor/actuator networks, network security and communication systems. He has co-authored about 70 reviewed research papers. He was a Co-Chair and PC Member for various International Conferences (ACM, IEEE, GI). He is a Member of ACM, IEEE, IEEE Computer Society and GI (Gesellschaft fr Informatik). He is actively participating in several working groups of the IETF. His research activities are focused on (but not limited to) autonomic networking addressing issues in wireless ad hoc and sensor networks, self-organisation, bio-inspired mechanisms, network security, network monitoring and measurements and robotics.

Feng Chen is pursuing his PhD in the Autonomic Networking Group at the Department of Computer Sciences, University of Erlangen-Nuremberg. His research interests include the performance and security aspects in ad hoc and sensor networks.

1 Introduction

In this paper, we propose a novel scheme for dynamic address allocation in Wireless Sensor Networks (WSN). This allocation scheme was specifically developed for maintenance operations in large-scale networks, that is it works on a local set of nodes instead of building network-wide unique address tables. Therefore, it is more scalable and efficient than competitive techniques.

Ad hoc communication has become a major subject in the networking community. Especially, WSN have become a research target due to their specific requirements and restrictions (Akyildiz et al., 2002). Research challenges and further directions include energy-aware operation, scalability and optimised resource management (Chong and Kumar, 2003). Besides physical resources such as memory, processing power and energy, logical resources in terms of naming, addressing and topology control must be organised and controlled.

Self-organisation is regarded to be the new paradigm for operation and control in ad hoc networks and WSN (Dressler, 2006). Without the need for global state maintenance, the major objective scalability can be easily addressed. While methodologies for generic self-organisation are still future vision, communication and routing aspects are well-covered so far.

Usually, current network layer protocols require a unique addressing for all nodes in the network. Various routing protocols have been proposed (Akkaya and Younis, 2005). Nevertheless, in ad hoc and sensor networks, the data communication, routing strategies and topology management is very scenario dependent, that is different solutions are optimal for example, agriculture (Baggio, 2005) and habitat monitoring (Mainwaring et al., 2002) scenarios. A survey of ad hoc routing protocols is for example given by Boukerche and Nikoletseas (2004) and Akkaya and Younis (2005). The overall objective for all these solutions is to develop scalable routing protocols for application in WSN scenarios consisting of a huge number of communicating nodes (Hong et al., 2002; Iwata et al., 1999).

Interestingly, many scenarios in the domain of WSN do not depend on fixed node addresses due to various reasons:

- The application itself does not require globally unique addresses. For example, measurement results need to be transmitted and analysed in an sensor/actuator network. The application only needs to know about the region or position of an identified event but the address of a particular node.
- The deployment and maintenance becomes much easier if address-less or data-centric operations are enabled. Nodes can be deployed (or replaced) without changing the node programme.
- Even if dynamic addressing on a global base is considered, the overhead due to the address assignment process, either during development or replacement can be too high. Such overhead is caused by address assignment protocols to find network-wide unique addresses.

These findings lead to many proposals for data-centric routing approaches. Basically, most of them are based on flooding schemes (Liu et al., 2006; Pleisch et al., 2006; Kwon and Gerla, 2002). Probabilistic solutions using the idea of stochastically reducing the overhead caused by flooding approaches are very successful. The most prominent solution is gossiping (Barrett et al., 2003; Haas et al., 2002). Similarly, gossiping is used for resource location protocols (Kempe et al., 2001). Finally, geographical routing is an example of address-less operation even if this is not an example for data-centric routing (Yu et al., 2001).

The data-centric operation principles allow an efficient data communication. Nevertheless, during *operation and control*, specific nodes need to be addressed to update software modules, to calibrate sensors, to perform localisation tasks and others.

In this paper, we propose and evaluate an algorithm for localised address allocation and management. This scheme is a major building block in an overall operation and maintenance scenario. The scheme is round-based and inherently allows busy-sleep periods and does not assume always awake nodes. We also implemented the method in a testbed consisting of Mica2 sensor motes and several mobile robot systems maintaining the sensor network. These robots use the address allocation scheme to identify specific nodes that need to be calibrated or reprogrammed.

The rest of this paper is organised as follows: related work is discussed in Section 2. The algorithm and operation details are presented in Section 3. Then, we discuss an application scenario in Section 4. Section 5 concludes this paper.

2 Related work

Solutions for dynamic address allocation have been proposed in various contexts. The best known example is Dynamic Host Configuration Protocol (DHCP) for IP and IPv6 networks, respectively (Droms, 1997, 2004). Similarly, techniques for operation in much more dynamic environments such as mobile ad hoc networks have been proposed by Bernardos and Calderon (2005). A detailed summary can be found in a comprehensive study of dynamic addressing schemes (Sun and Belding-Royer, 2004) and a paper providing an overview and future directions for such address allocation solutions (Weniger and Zitterbart, 2004). A self-organising address allocation scheme was proposed by Toner and O'Mahony (2003), which is based on the original Duplicate Address Detection (DAD) algorithm as used by Weniger (2004). In the context of ad hoc and sensor networks, Passive Autoconfiguration for Mobile Ad hoc Networks (PACMAN) need to be mentioned as an optimised solution for efficient dynamic address allocation (Weniger, 2005). This approach is directly based on the lessons learnt from the DHCP development.

In the following, we shortly discuss two solutions that we used as a starting point for developing our novel localised address allocation scheme: DHCP and Passive Duplicate Address Detection (PDAD).

2.1 *Dynamic host configuration protocol*

DHCP (Droms, 1997, 2004) is a client-server-based network protocol. It consists of two major building blocks: a protocol for delivering specific parameters to the client and a mechanism for selection and suggesting IP addresses. Each DHCP server maintains one or more address pools. Such a pool describes available and currently used addresses, respectively. The address management is server-oriented. This working principle ensures the uniqueness of assigned IP addresses. In consequence, no detection scheme for duplicate addresses is necessary.

DHCP supports three different mechanisms for IP address allocation: automated, operator-controlled and dynamic. In the first case, an IP address is permanently assigned to a client after its first registration. Similarly, an operator can manually assign an address to a particular client. Finally, dynamic address allocation provides the possibility to temporally bind an IP address to a client. This assignment is limited to time and must be renewed after a given lease time.

The dynamic assignment is the only possibility to reuse addresses after the first allocation. Thus, dynamic address allocation is usually used if only short-term connections of clients are envisioned. Focusing on the management and control in ad hoc and sensor networks, this seems to be an adequate solution. Unfortunately, all the assignments are based on the MAC address of each client that is worldwide unique. Therefore, DHCP only maintains an IP address to MAC address binding.

2.2 *Passive duplicate address detection*

PDAD (Weniger, 2004) was specifically developed for mobile ad hoc networks. Basically, PDAD is not an algorithm for choosing an address but for detecting duplicates. The primary objective during the development of PDAD was its application in networks with high node mobility. Therefore, the result was a lightweight scheme for address allocation with passive duplicate detection.

The behaviour of the algorithm is as follows: each node randomly choses an address by itself. There is no need for a particular server or any other required network infrastructure. In a second step, the node performs the DAD algorithm to verify the uniqueness of the selected address by passively observing the network traffic. PDAD continuously checks routing information for bandwidth-efficient DAD, for example, sequence numbers are verified.

In summary, PDAD provides an efficient address allocation algorithm that depends on particular ad hoc routing information. In case of data-centric data communication, additional messages must be created to enable PDAD in such networks. Obviously, this results in unnecessarily high overhead. Additionally, PDAD's objective is to maintain globally unique addresses. In many scenarios, there is no such requirement.

3 Algorithm and methodology

The proposed algorithm for dynamic address allocation benefits from many ideas learned from DHCP and PDAD. A selected device, for example, a server performing management and control operations, initiates and controls the address assignment process. This server maintains a list of previously allocated addresses. While this behaviour looks similar to DHCP, there is no binding to any kind of hardware address or other unique identification of a participating sensor node. Therefore, an extension, DAD, is necessary. After such an address allocation step, the server can continue in maintaining the surrounding sensor nodes by contacting each of them individually.

The envisioned scenario allows multiple server nodes performing management and control actions concurrently. Additionally, we assume mobility and spontaneously emerging or failing nodes. Therefore, the address assignment cannot last for an unlimited period of time. Nevertheless, we do not enforce periodic reassignments to prevent the disruption of running maintenance operations such as node reprogramming. Instead, a round-based system is used that creates a logical ordering of time information in a local environment. If a new round is started, all nodes must update their addresses while subrounds are used to discover new nodes that appeared in the surroundings.

Initially, the server cannot communicate with particular nodes using unicast communications because the server cannot assume a previous address-assignment step. Therefore, broadcast messages are employed to discover neighbouring nodes and to initiate the address allocation. In ad hoc and sensor networks, such a broadcast is also the most efficient way to reach all surrounding nodes with a single radio packet.

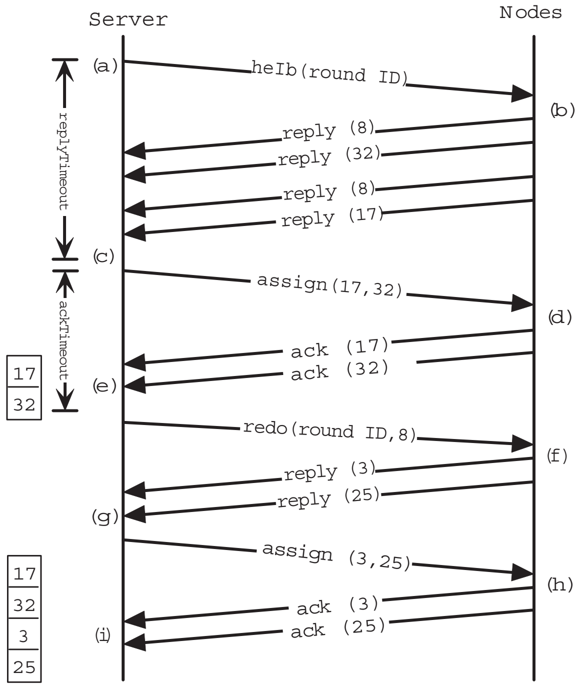
After the first allocation step, the server can seamlessly continue with its management and control operation using unicast communication targeting specific nodes. For example, nodes might be calibrated, reconfigured or reprogrammed. To prevent disruptions, measures have to be taken in order to ensure the uniqueness of the selected addresses in the local environment. This also includes the possibility for spontaneously arriving nodes and even busy-sleep cycles without always demanding awake nodes.

In the following, we depict the involved processes during the address selection, duplicate prevention and address management. Exemplary, we implemented the scheme on Mica2 sensor motes running TinyOS (see also Section 4). In the WSN community, this system is a quasi standard in the academic world.

3.1 Original DAA algorithm

In this section, we propose the original DAA algorithm and assume that only one server exists and no collision occurs throughout the DAA process. A DAA process is always started by a dedicated node. We call this node *server* because we assume that this node will provide management and control operations for the neighbouring sensor nodes (which is the only reason for maintaining unique addresses). The algorithm is illustrated by an example with one server and four nodes as shown in Figure 1.

Figure 1 The original DAA algorithm for single server without collision



(a) First, the server initiates a new round of DAA process by broadcasting a HELLO message with a new *round ID*. After sending, the server sets a *replyTimeout* and waits for the REPLY from its neighbouring nodes.

(b) Upon reception of the HELLO message, all neighbouring nodes first store the new *round ID* and the server address. Then, each of them randomly chooses an address and submits it to the server in a REPLY message.

(c) Upon reception of one REPLY message, the server stores the received address into a local list called

ReplyList. When *replyTimeout* expires, the server verifies the uniqueness among all addresses stored in the *ReplyList* and put all unique addresses into a list called *AssignList* and all duplicate addresses into another list called *RedoList*. If the *AssignList* is not empty, the server will broadcast an ASSIGN message with this list. In this example, the *AssignList* contains 17 and 32. After sending out the ASSIGN message, the server sets an *ackTimeout* and waits for the acknowledgments from those nodes notified in the ASSIGN message.

(d) Upon reception of the ASSIGN message, each node seeing its preselected address listed in the ASSIGN message, should register its radio interface with this address, which will be used in all future unicast communications. Each registered node should send its address in an ACK message to the server and will ignore all future messages received from the server with the current *roundID*.

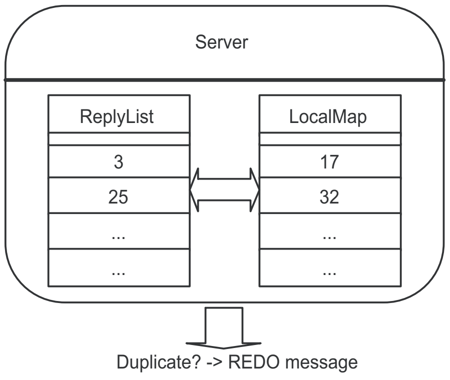
(e) Upon reception of one ACK message, the server stores the address in the ACK message to a list called *Ack-List*. When *ackTimeout* expires, the server registers the addresses in the *AckList* by moving them into a address map called *LocalMap*. Till now, each correctly acknowledged address has been registered by both the server and the node generating this address. If the server does not receive all expected ACKs listed in the *AssignList*, it will add those unacknowledged addresses to the *RedoList*. If the *RedoList* is not empty, the server will broadcast a REDO message with this list. All nodes seeing its own address contained in the REDO message have to regenerate a random address and send it back to the server in a REPLY message. For purpose of discovering new nodes, we let REDO messages carry the current *roundID*. In this way, the nodes that have missed the previous HELLO message can also join in the current DAA process upon reception of the REDO message. After sending out the REDO message, the server should clear all the local lists and set a *replyTimeout* to wait for coming REPLY messages.

(f) Upon reception of the REDO message, all nodes that have not yet obtained a unique address from the server in the current round, have to randomly choose an address and send it back to the server in a REPLY message.

(g) The server will repeat the same operations described at step (c). The difference is that, when performing uniqueness verification, the server should compare the *ReplyList* containing all newly received addresses with the *LocalMap* containing all registered addresses, to identify the duplication, as shown in Figure 2.

The following steps (h) and (i) will repeat the steps (d) and (e). As shown in Figure 1, the server terminates the current round of DAA process at the end of step (i), because it has found no more address in the *RedoList*. If the server wants to discover more neighbouring nodes, it can send one more REDO message with only the current *roundID*. It can be easily imagined that the following process will repeat the one between step (e) and (i). Since the REDO message does not change the current *roundID*, we call such a circle between two consecutive REDO messages as subround. The server should follow certain rules to take control of the number of subrounds needed for the current DAA round. We will discuss this problem in the later section.

Figure 2 Uniqueness verification



3.2 Possibilities for duplicate addresses

We now analyse the probability of producing duplicate addresses in a single subround. Such duplicates can appear if two or more nodes choose the same address upon reception of a HELLO or REDO message from the server.

Assuming *n* surrounding nodes and *a* available addresses, the probability of an address collision is

$$P_{\text{duplicate}} \sim 1 - \frac{a!/(a-n)!}{a^n} \tag{1}$$

Proof: Because there is no synchronisation between the nodes, more than one node can assign the same address in each subround. Therefore, the number of permutations without one or more duplicates is $P_a^n = \binom{a}{n} \times n! = a!/(n!(a-n)!)$. Thus, the probability to find a combination of addresses without duplicates is $P_{\text{noduplicate}} = P_a^n / a^n$. This leads to the equation for the probability of having duplicate addresses $P_{\text{duplicate}} = 1 - P_{\text{noduplicate}}$ as shown in Equation (1).

For validation of the algorithm, we assume a server with *a* surrounding nodes and an address length of 16 bit (unit 32 *t*) that is, a total number $a = 2^{16} = 65,536$ possible addresses. Then, the probability of duplicates in a round will be about 0.00068 for 10 nodes, 0.0028 for 20 nodes and 0.072 for 100 nodes (which are usual numbers in sensor networks with an address length of only 16 bit).

The obvious conclusion is the more sensor nodes are in the surrounding, that is, the larger area the server radio covers, the higher is the probability to find duplicate addresses in a subround. Such duplicates cannot be prevented. Nevertheless, the probability can be controlled by verifying the application scenario and adapting the address space. Additionally, the method for choosing the address can be changed from a random process to one that maintains history information. Such a process would decrease the probability of duplicate addresses in a subround (not preventing it). However, it would also add a remarkable overhead to sensor nodes.

3.3 Collision analysis

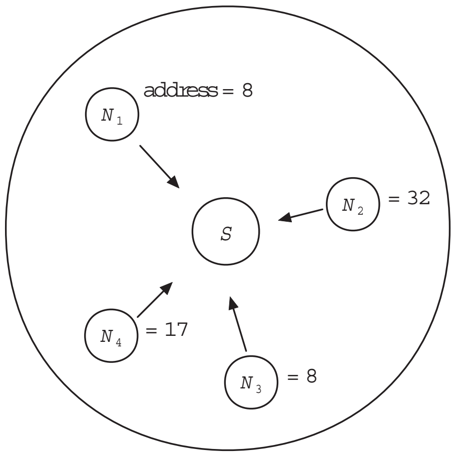
In the previous section, the original DAA algorithm was proposed under the assumption of no collision. Now we introduce collision problem into our algorithm. In contention-based wireless communications, collision happens when two or more packets arrive at a certain node at the same time. In this section, we reuse the example already shown in Figure 1 to show what will happen, when collision occurs during the DAA process.

We should first identify which messages may collide. For all broadcasting messages sent by the server, including HELLO, ASSIGN and REDO messages, they will not collide with other messages, as long as the server carefully controls the timing of sending these messages. Such timing control has been considered in the original algorithm by setting a timeout after broadcasting a message in the server, for example, *replyTimeout* and *ackTimeout* as Shown in Figure 1.

For all messages sent by the nodes, including REPLY and ACK are prone to collisions, because there is no time control for them so far in the original algorithm and they are sent in a contention way. Our DAA algorithm cannot rely on those usual collision avoidance measures like RTS/CTS mechanism used by IEEE 802.11 protocols for solving hidden station problem, because unique addresses are prerequisite to these mechanisms.

We have listed below all possible situations that REPLY or ACK messages collide during the DAA process. The Figure 3 shows the step (b) for the example shown in Figure 1. At this step, all four nodes randomly choose an address and perfect it back to the server in a REPLY message.

Figure 3 Possible collisions in a subround with one server



Case A: no duplicate in collision, for example, node2's REPLY collides with node4's. At step (c), the server detect the duplicate between node1 and node3 but do not send an ASSIGN due to the collision. Afterward it sends a REDO with the duplicate address eight. Therefore, no address has been allocated in this subround and all nodes have to rechoose addresses in the next subround. No error occurs in this case.

Case B: all duplicates in collision, for example two REPLY messages with address eight collide. At step (c) the server is not aware of the duplication due to collision and only sends an ASSIGN message with address 17 and 32. There are two possible subcases.

- *Case B1*: If the server receives two ACKs from node2 and node4, it may wrongly consider that all nodes have been allocated and then terminate the current DAA round. The consequence is that node1 and node3 are abandoned by the server in this round.
- *Case B2*: If two ACKs sent by node2 and node4 collide, the server will find out the lost ACK when *ackTimeout* expires. Therefore, the server will send a REDO message to initiate a new subround. No error occurs in this subcase.

Case C: not all duplicates in collision, for example, node1's reply collides with node2's. At step (c), the server thinks that the address eight is unique and adds it into the ASSIGN message. At step (d), upon reception of the ASSIGN message, both node1 and node3 register the same address and then send back a ACK. There are three possible subcases.

- *Case C1*: the server receives two ACKs with address 8 and detects the duplication. Then it will nod add this address into its *LocalMap* and later add the duplicate address to the REDO message. According to the original algorithm, this is of no help, because all allocated nodes will ignore all further messages in the current round. However, it is possible that this address will be registered by the server and some other nodes in a certain subround later on.
- *Case C2*: both ACKs with the duplicate address 8 collide. The server receives no ACK from the nodes with the address 8. Then it will not add this address into its *LocalMap* and later add the unacknowledged address 8 to the REDO message. The consequence will be the same with that in Case C1.
- *Cases C3*: only one ACK with the duplicate address 8, for example from node1, is received by the server. The other ACK with the duplicate address 8 collides with the ACK from some other nodes, for example, from node4. The consequence is that the address 8 will be registered by the server and by both node1 and node3.

Sum up the above three subcases, if case C happens, the duplicate address will be allocated, which is in most cases strictly forbidden.

3.4 Improved DAA algorithm

The collision analysis in the previous section has shown that collisions may lead to malfunction of the original DAA algorithm. In this section, we propose the solutions to those problems to improve our algorithm. The improvements are mainly focused on two aspects. One is Collision Avoidance (CA), that is, to add some CA mechanisms to reduce collision probability. The other is collision processing, that is how to deal with collisions.

We consider first collision avoidance. Collisions are prone to happen while sensor nodes are sending REPLY or ACK messages back to the server, because they contend for the channel at almost the same time. Carrier Sense Multiple Access (CSMA) with exponential backoff algorithm is the most usual solution. However, due to the fixed and very small length of REPLY and ACK messages, we prefer applying a simpler method. For example, after receiving a HELLO or a REDO we let each node randomly choose a slot within the reply window to send its REPLY without performing carrier sense. The reply window size and the length of one slot must be negotiated between the sever and sensor nodes beforehand or be carried by HELLO or REDO messages. Furthermore, the reply window size must be smaller than *replyTimeout*. We apply this method only to sending REPLAY messages. For ACK messages, another algorithm will be given later in this section.

We now discuss how to deal with possible errors caused by collisions. The improvements are proposed based on the collision analysis in the previous section. The cases with errors include B1, C1, C2 and C3.

For case B1: in this case, the server terminates the current round of DAA process, when it gets an empty *RedoList* after processing the *AckList*. To prevent those nodes, whose REPLYs collide, from being abandoned by the server, we add a rule to the server. We call such a subround, in which the server receives no REPLYs after sending a HELLO or a REDO, a null subround. The new rule is described as follows.

When the server finds its *RedoList* empty at the end of the current subround, it must start a new subround by broadcasting a REDO with only the current *roundID*. The server can terminate the current round of DAA process only after it has experienced a certain number of null subrounds consecutively.

This rule is described in Figure 5. The parameter *maxNullSubround* is application specific and defines the number of consecutive null subrounds needed to be executed before the server ends current DAA process.

For Case C1 and C2: in both cases, the duplicate address is registered by the two nodes first but not registered by the server afterward. To correct this, we need to use the following algorithm to register a unique address at a sensor node.

If the node receives an ASSIGN message from the server and finds its address in the message, it will register its address only when it receives a REDO message in the next subround and does not see its address in the message. The algorithm is depicted in Figure 6.

For Case C3: in this case, the duplicate address is first registered by the two nodes and then by the server. Even if the improved algorithm proposed above is applied, it will not change the result. A straightforward solution to this problem is to prevent this case from happening. More specifically, we need to find a way to prevent all ACK messages with different addresses from colliding. To achieve this goal, we propose a schedule-based algorithm for sending ACK messages and explain it as follows.

The server should first rearrange all addresses in the *RedoList* in a certain order, for example, in increasing order, and then sends an ASSIGN message with the ordered *RedoList*. Upon reception of the ASSIGN message, each node listed in the message has to send its ACK at a specific time point according to the position of its address in the *RedoList*. To achieve this, we divide the acknowledgement window – defined as the period for which the server should wait for all expected ACK messages – into a certain number of slots. The number of slots is equal to the number of addresses in the *RedoList*. The length of a single slot is fixed and should be negotiated between the server and sensor nodes beforehand. The parameter *ackTimeout* must be chosen larger than the acknowledgement window. In this way, only those ACK messages with the same address will be sent by the nodes in the same slot and they will never collide with those with different addresses.

A possible subcase with the improved algorithm under case C that we have discussed in the previous section is shown in Figure 4. We can see that, the ACKs with the duplicate address 8 are sent by node1 and node3 in the first slot of the acknowledgement window and will have no chance to collide with node4's ACK, which is sent late in the second slot by node4. This means that the possibility of case C3 is eliminated and case C has only two possible subcases, case C1 and case C2. No matter which of the two subcases happens, the improved algorithm will not lead to errors any more.

Till now, the complete DAA algorithm for single server with collisions has been given. For a better understanding of our algorithm, we have designed two UML diagrams for both server and sensor nodes, as shown in Figure 5 and Figure 6, respectively.

4 Application scenario

To validate the algorithm, we implemented the dynamic address allocation scheme in the context of a project that focuses on dynamic reconfiguration and reprogramming of

Figure 4 The improved DAA algorithm for single server with collision

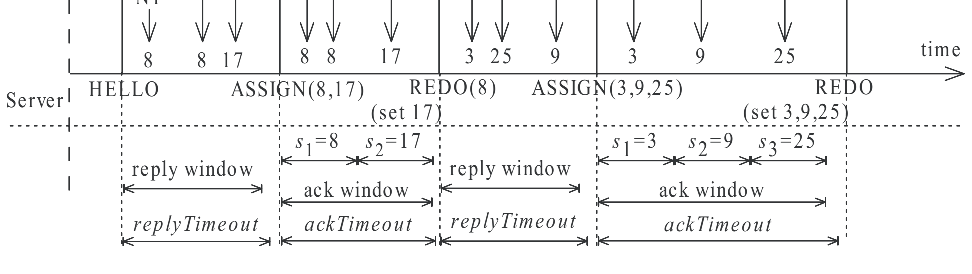
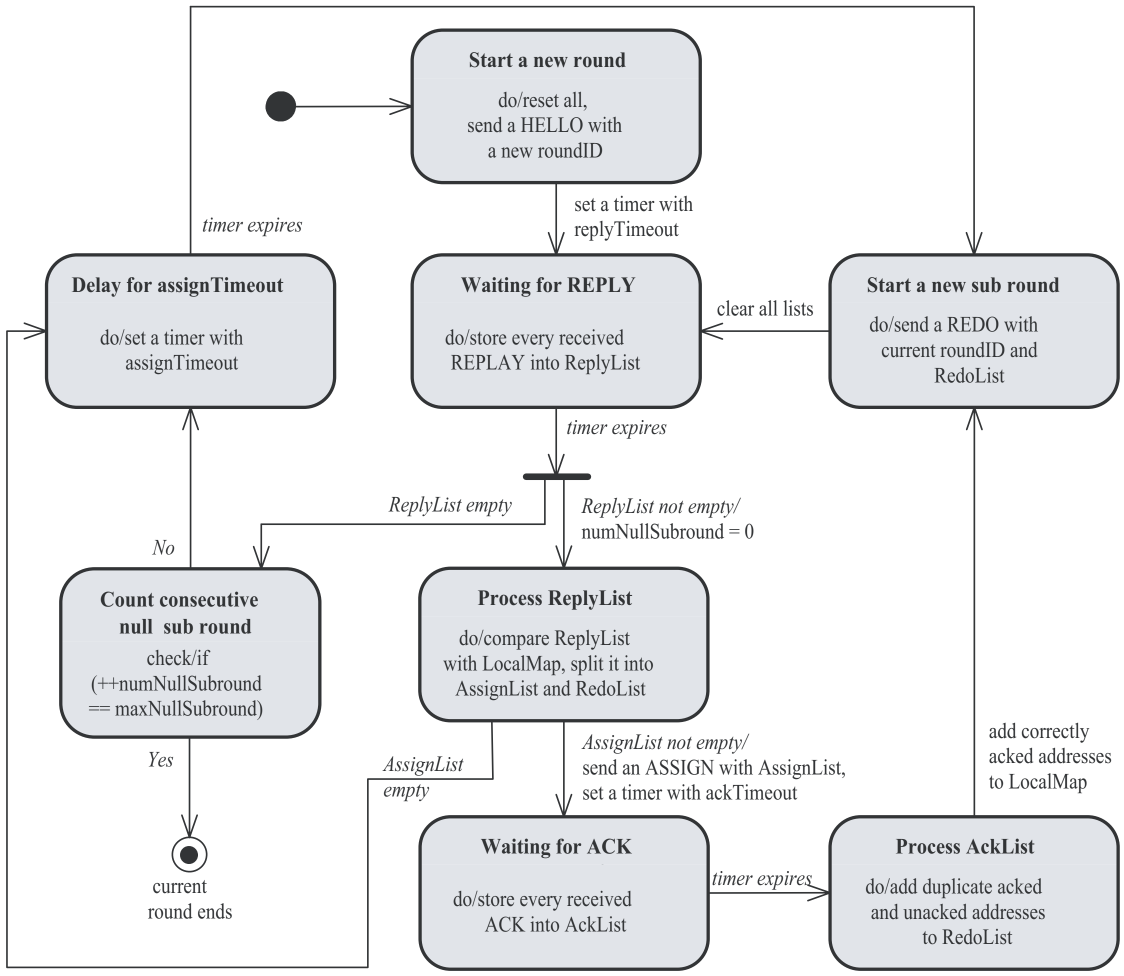


Figure 5 UML diagram of the DAA algorithm for server nodes


Start a new round
do/reset all, send a HELLO with a new roundID
set a timer with replyTimeout
timer expires
Delay for assignTimeout
do/set a timer with assignTimeout
Waiting for REPLY
do/store every received REPLAY into ReplyList
clear all lists
Start a new sub round
do/send a REDO with current roundID and RedoList
timer expires
ReplyList empty
ReplyList not empty/ numNullSubround = 0
No
Count consecutive null sub round
check/if (++numNullSubround == maxNullSubround)
Process ReplyList
do/compare ReplyList with LocalMap, split it into AssignList and RedoList
add correctly acked addresses to LocalMap
Yes
AssignList empty
AssignList not empty/ send an ASSIGN with AssignList, set a timer with ackTimeout
Waiting for ACK
do/store every received ACK into AckList
timer expires
Process AckList
do/add duplicate acked and unacked addresses to RedoList
current round ends

sensor nodes using mobile robot systems (Yao et al., 2006). Due to the steadily increasing heterogeneity and dynamics in terms of hardware and software configurations in sensor networks, software management is becoming one of the most prominent challenges in this domain. We developed a profile-based software management scheme (Truchat et al., 2006) that consists of a dynamic profile-matching algorithm to identify current SW/HW configurations, an on-demand code generation module and mechanisms for dynamic network-centric reprogramming of sensor nodes. A mobile robot system is employed for decision processes and to store the source code repository. Our proposed address allocation scheme is used to prevent global preconfiguration of all network nodes.

Figure 7 shows the principal concept of sensor node software reconfiguration (Fuchs et al., 2006).

1 Depending on the goal, the robot drives to the position in the sensor network where reconfiguration is necessary (we do not assume a particular navigation scheme, various mobility models can be applied).

2 The robot collects information about the environment, builds the context and explores its neighbourhood. At this step, additional actions can be initiated such as preparing the sensor calibration or starting an algorithm for dynamic addressing.

3 All sensor nodes respond to the exploration message by sending their current profiles (HW/SW descriptions).

4 The robot uses the information gathered at steps (b) and (c) for profile matching and to assign the roles of the sensor notes (depending on the current goal). Finally, it creates the new binaries of the sensor notes.

5 The robot reprogrammes selected sensor notes over the air.

In our lab, we use the Robertino robot platform developed at the Fraunhofer Institute AIS running embedded Linux and Mica2 sensor motes developed at UCB running TinyOS. For direct robot-sensor communication, we installed a single Mica2 mote on the robot.

We successfully used the dynamic address allocation scheme for management issues in WSN as described in this paper to provide a communication infrastructure for communication between the robot system and surrounding sensors. The implementation allows to choose 16-bit addresses as used in the algorithm description.

In our experiments, we never observed the special case of address duplication. To test this case, we manually configured specific nodes with a fixed address. Using this kind of experimentation, we verified the correct behaviour of the round based DAD.

Figure 6 UML diagram of the DAA algorithm for sensor nodes

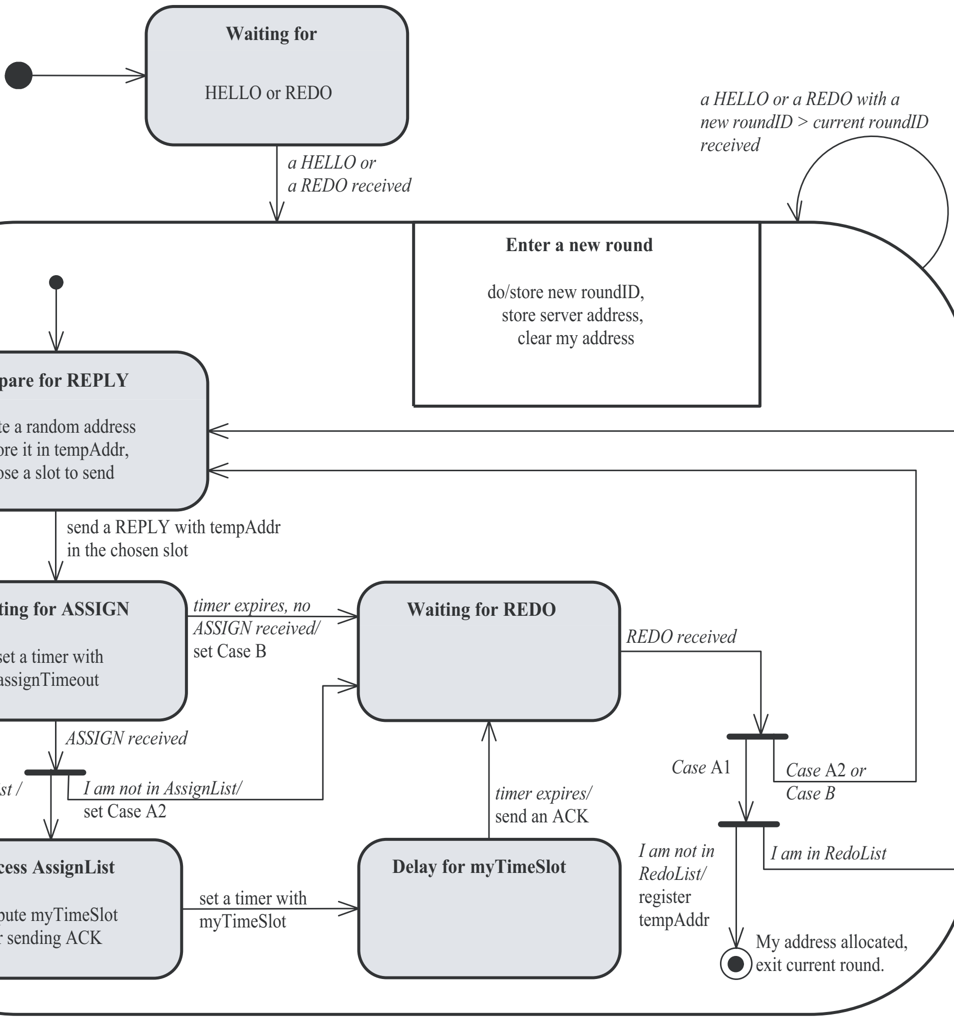
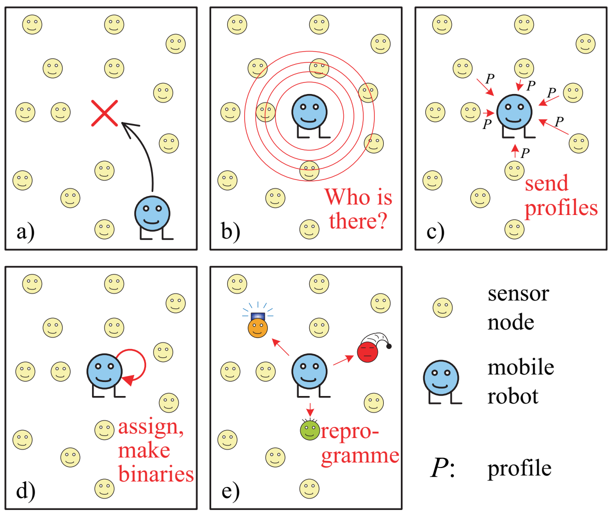


Figure 7 Application scenario for reconfiguration



Source: Fuchs et al. (2006).

5 Conclusion

In conclusion, it can be said that we developed an address assignment algorithm that works in a localised environment. Therefore, the overhead due to management of topology and uniqueness of the addresses becomes very low. Additionally, the method profits from the single-hop communication that is usually more reliable compared to a multi-hop approach. Basically, we selected particular solutions from PDAD and DHCP to create an efficient and robust dynamic address allocation scheme for management and control in WSNs.

Compared to PDAD, our solution is more reliable, has less overhead and is independent of the employed routing algorithm. Similarly to DHCP, the algorithm is server-centric, that is, the allocation is initiated by a particular system. The DAD is performed by the same system. Therefore, this verification is very simple. The communication overhead increases linearly with the number of nodes. New nodes do not influence previous allocations. This is also true for waking up nodes performing busy-sleep cycles. They are processed individually in a new subround.

References

Akkaya, K. and Younis, M. (2005) 'A survey of routing protocols in wireless sensor networks', *Elsevier Ad Hoc Network Journal*, Vol. 3, No. 3, pp.325–349.

Akyildiz, I.F., Su, W., Sankarasubramaniam, Y. and Cayirci, E. (2002) 'A survey on sensor networks', *IEEE Communications Magazine*, Vol. 40, No. 8, pp.102–116.

Baggio, A. (2005) 'Wireless sensor networks in precision agriculture', *ACM Workshop on Real-World Wireless Sensor Networks (REALWSN 2005)*, Stockholm, Sweden.

Barrett, C.L., Eidenbenz, S.J. and Kroc, L. (2003) 'Parametric probabilistic sensor network routing', *International Conference on Mobile Computing and Networking*, San Diego, CA.

Bernardos, C. and Calderon, M. (2005) 'Survey of IP address autoconfiguration mechanisms for MANETs', Internet-Draft (work in progress), draft-bernardos-manet-autoconf-survey-00.txt.

Boukerche, A. and Nikoletseas, S. (2004) 'Protocols for data propagation in wireless sensor networks: a survey', in M. Guizani, (Ed). *Wireless Communications Systems and Networks*, Kluwer Academic Publishers.

Chong, C-Y. and Kumar, S.P. (2003) 'Sensor networks: evolution, opportunities, and challenges', *Proceedings of the IEEE*, Vol. 91, No. 8, pp.1247–1256.

Dressler, F. (2006) *Self-organization in ad hoc networks: overview and classification*, Technical Report 02/06, University of Erlangen, Department of Computer Science 7.

Droms, R. (1997) *Dynamic Host Configuration Protocol*, RFC 2131.

Droms, R. (2004) *Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6*, RFC 3736.

Fuchs, G., Truchat, S. and Dressler, F. (2006) 'Distributed software management in sensor networks using profiling techniques', *1st IEEE/ACM International Conference on Communication System Software and Middleware (IEEE COMSWARE 2006): 1st International Workshop on Software for Sensor Networks (Sensor-Ware 2006)*, New Delhi, India.

Haas, Z.J., Halpern, J.Y. and Li, L. (2002) 'Gossip-based ad hoc routing', *IEEE INFOCOM 2002*, pp.1707–1716.

Hong, X., Xu, K. and Gerla, M. (2002). 'Scalable routing protocols for mobile ad hoc networks', *IEEE Network*, Vol. 16, pp.11–21.

Iwata, A., Chiang, C-C., Pei, G., Gerla, M. and Chen, T-W. (1999) 'Scalable routing strategies for ad hoc wireless networks', *IEEE Journal on Selected Areas in Communications: Special Issue on Ad-Hoc Networks*, Vol. 17, No. 8, pp.1369–1379.

Kempe, D., Kleinberg, J. and Demers, A. (2001) 'Spatial gossip and resource location protocols', *Journal of the ACM (JACM)*, Vol. 51, No. 6, pp.943–967.

Kwon, T.J. and Gerla, M. (2002) 'Efficient flooding with passive clustering (PC) in ad hoc networks', *ACM SIGCOMM Computer Communication Review*.

Liu, H., Wan, P., Jia, X., Liu, X. and Yao, F. (2006) 'Efficient flooding scheme based on 1-hop information in mobile ad hoc networks', *25th IEEE Conference on Computer Communications (IEEE INFOCOM 2006)*, Barcelona, Spain.

Mainwaring, A., Polastre, J., Szewczyk, R., Culler, D. and Anderson, J. (2002) 'Wireless sensor networks for habitat monitoring', *First ACM Workshop on Wireless Sensor Networks and Applications*, Atlanta, GA.

Pleisch, S., Balakrishnan, M., Birman, K. and van Renesse, R. (2006) 'MISTRAL: efficient flooding in mobile adhoc networks', *Seventh ACM International Symposium on Mobile Ad Hoc Networking and Computing (ACM MobiHoc 2006)*, Florence, Italy.

Sun, Y. and Belding-Royer, E.M. (2004) 'A study of dynamic addressing techniques in mobile ad hoc networks', *Wireless Communications and Mobile Computing*, Vol. 4, No. 3, pp.315–329.

Toner, S. and O'Mahony, D. (2003) 'Self-organising node address management in ad hoc networks', *8th IFIP International Conference on Personal Wireless Communications (PWC 2003)*, Vol. LNCS 2775, pp.476–483, Venice, Italy, Springer.

Truchat, S., Fuchs, G., Meyer, S. and Dressler, F. (2006) 'An adaptive model for reconfigurable autonomous services using profiling', *International Journal of Pervasive Computing and Communications (JPCC): Special Issue on Pervasive Management*.

Weniger, K. (2004) 'Passive duplicate address detection in mobile ad hoc networks', '*IEEE Wireless Communications and Networking Conference (WCNC)*, New Orleans.

Weniger, K. (2005) 'PACMAN: passive autoconfiguration for mobile ad hoc networks', *IEEE Journal on Selected Areas in Communications (JSAC)*.

Weniger, K. and Zitterbart, M. (2004) 'Address auto-configuration in mobile ad hoc networks: current approaches and future directions', *IEEE Network Magazine: Special Issue on Ad Hoc Networking: Data Communications and Topology Control*.

Yao, Z., Lu, Z., Marquardt, H., Fuchs, G., Truchat, S. and Dressler, F. (2006) 'On-demand software management in sensor networks using profiling techniques', *ACM Second International Workshop on Multi-Hop Ad Hoc Networks: From Theory to Reality 2006 (ACM REAL-MAN 2006), Demo Session*, pp.113–115, Florence, Italy.

Yu, Y., Govindan, R. and Estrin, D. (2001) *Geographical and Energy Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks*. Technical Report UCLA/CSD-TR-01-0023, UCLA Computer Science Department Technical Report.