

WSN Self-address Collision Detection and Solving

Carlos Ribeiro
INESC-ID/IST

Email: Carlos.Ribeiro AT inesc-id.pt

Ivo Anastácio, André Costa, Márcia Baptista
IST

Email: {ivo.anastacio, andre.costa, marcia.baptista} AT tagus.ist.utl.pt

Abstract

The complexity and number of interconnected elements comprising today's networks have stressed the need for autonomic computing techniques. This need is even bigger in Wireless Sensor Networks, because sensor nodes should work continuously without operator intervention, i.e. have self-organization and self-configuration algorithms. The first action that a Wireless Sensor Network should do is to establish the network address of each node without collisions. However, this scenarios are dynamic which means that network partitions and rejoins are common and node addition and failure are not uncommon, which means that the network should be able to reconfigure itself without compromising availability.

In this paper we show a solution to detect address collisions in dynamic scenarios wasting a minimum level of energy (i.e. without periodic beacons or any other kind of message), and we describe a collision solving protocol which minimizes the impact on the already established routes, which also minimizes energy loss because there is no need to rerun the routing protocol.

In order to develop the present solution we have implemented several preliminary solutions that revealed some lessons about wireless communications on sensor nodes which were useful to the optimization of the final solution.

Index Terms

WSN, Autonomic, Address assignment, Security

1. Introduction

Wireless Sensor Networks (WSN) are networks composed by small and cheap devices with sensing abilities, which are able to communicate with each other by radio signals, usually running over small batteries. The combination of sensing and radio communication abilities makes these networks ideal to build distributed sensing networks where each node collaborates by sensing one or more phenomena in its neighborhood and relaying it to a central node.

Because of their properties, WSN are specially suitable for surveillance of critical infrastructures, e.g. gas pipelines, power lines, water supply systems, etc. This means that have to be reliable and be able to self-adapt to changes both in the sensed environment and in the wireless network. On

the other hand this sensors should be in place, sensing the environment, for long periods without human intervention or battery exchange, which means that reliability and self-adaptability should not be accomplished at the cost of energy consumption.

Previously, we have proposed an address self-assignment protocol, robust against a limited number of misbehaved sensor nodes [7]. The protocol is a probabilistic self-stabilizing protocol, which is able to assign 2-hop unique addresses to every node with a minimum number of messages, and yet be resilient against a limited number of misbehaved nodes that try to prevent well behaved nodes from choosing their addresses.

However, the protocol is not able to prevent address collisions from happening whenever the network changes, either because some more sensors were added or because some communication barrier was removed (e.g. a door opens) and some other sensors became visible.

Most address assignment protocols [1, 2, 3, 6, 5] do not handle address collisions resulting from incrementally deployed scenarios, others [8] solve the problem by periodically running the address assignment protocol, which is clearly inadequate for WSN because most of the time there are no collisions and the running of the protocol is a waste of energy. Furthermore, whenever a collision is detected the address of some nodes change breaking any established sessions and/or routes.

In this paper we describe two protocols: one to detect collisions with minimum energy waste and, another to solve them without breaking the established sessions and routes. We have implemented both protocols in Mica Z motes running TinyOS and show some measurements in different environments (i.e. noisy in terms of wireless spectrum and non-noisy).

The next section describes some previous work in address assignment in several contexts (WSN, MANETS, etc.). Section 3 describes briefly the RSI protocol which is the basic address assignment protocol extended by the proposed protocol. Section 4.1 describes the protocol to detect address collisions and shows some experimental results. Section 4.2 describes the protocol that solves address collisions without breaking established routes and finally in Section 5 we conclude the paper.

2. Related work

The address assignment problem was already studied before both for WSNs and for Mobile Ad-hoc Networks (MANET). Because in most situations addresses do not need to be global unique (see [7]) the address assignment problem turns up to be the neighborhood unique naming (NUN) problem, and is similar to the classical coloring graph problem with conditions at distance 2.

The IETF Zeroconf working group proposed a solution for MANETs [1] which rely on the discovering abilities of the underlying routing protocol. In their proposal each node independently chooses an address and then sends a routing request packet for that address, if a route is found within a timeout period, the address is already in use, otherwise the address is not used and the protocol ends. This protocol does not handle dynamic scenarios where a node may enter in the vicinity of another that has another neighbor with the same address.

The same problem is shared by other solutions [2, 3] that have a different approach to the address establishment problem but do not contemplate network changes. Both use a probabilistic self-stabilizing algorithm that must be run by all the nodes at the same time. The algorithm used in is very simple [2]. Each node keeps two variables, one with its ID and one with the ID of two colliding nodes in its neighborhood. If there are no collisions in the neighborhood the variable is empty. Each node starts by asking every neighbor their ID to calculate the second variable. It then asks its neighbors for their variables values. If any of these values is equal to its ID the node randomly chooses another. The algorithm was proven to self-stabilize, however no protocol was given to implement it. In particular it is not clear how messages from two distinct nodes with the same ID can not be confused with a common replay of the same message.

The approach followed in [6, 5] is different but also probabilistic. They leverage on the wireless nodes' ability to detect media access collisions to know if there are other nodes contending for an ID or not. If a node discovers that no one else is broadcasting at the same time it takes the ID from himself and every one else knows that that ID is taken. If several nodes broadcast at the same time, they all flip a coin to decide if they will participate in the next round. On average only half of the contenders transmit in the next round. After several rounds only one node will transmit, and gets the ID. As before this solution does not handle dynamic scenarios.

The solution proposed in [8] is able to handle dynamic scenarios by periodically repeating the protocol. In this proposal each node send a periodical message with its address. Each node keeps a log with every message seen by it, if it detects a duplicate address it sends a warning message to both nodes. Upon receiving the warning message both nodes choose another address and announce it again. With this protocol nodes may change address several times during the life-time of the network which may not be acceptable by

every application or routing protocol. Moreover, the periodic broadcasting of IDs may be too energy expensive, and it is not clear how the periodic address announcement is not mistaken with an address collision.

The ZigBeeTM communication protocol uses two types of addresses: 64 bit global unique addresses and 16 bit network unique short addresses. The 64 bit addresses are used at the beginning of the network to establish the 16-bit addresses, which are used from there after. The protocol which establishes the 16-bit addresses is similar to DR-CP/DAAP, however instead of using two distinct steps for assigning an address and for assigning a pool of addresses, ZigBeeTM only uses one; and instead of giving up half of its address space to each children node, a node divides equally its pool of addresses by all its neighbors. Both DRCP/DAAP and ZigBeeTM address assignment protocol do not scale well when the number of nodes is too large or the address space is too short.

3. RSI description

The basic protocol objective is twofold; i) ensuring a unique local identification on the WSN over a distance 2 neighborhood with an arbitrary large probability $p < 1$ and, ii) minimize the energy loss by minimizing the number of messages sent and received.

The basic protocol is very simple, each node chooses a random address for himself and asks its neighbors if they have chosen the same address. If at least one of them have chosen the same address, it replies saying that there were an address collision, otherwise each receiving node rebroadcast the query to its own neighborhood. The second hop nodes check the receiving packet for a collision with their own addresses. If they find a collision they reply in much of the same way that first hop nodes does, otherwise they do nothing. There are no negative replies only positive ones. This is because the probability of finding a collision in a 2-hop neighborhood is very low, thus in the usual scenario only query messages are sent.

The first problem that the protocol overcomes is how to distinguish the rebroadcast messages originated by the receptor node from the ones originated by other nodes. If a node trying to establish an address receives a query for that same address he should answer declaring that that address was taken, even if that action result in neither of the nodes stick with the ID. However, if the node is hearing an echo of its own query it should do nothing. In the basic version of the protocol, the messages sent by each node are stamped with a collision free 64 bit node address (extended address). This extended address can be a manufacturer unique number when available or a random number generated whenever a node starts. Note that extended addresses are used only in the context of the initialization protocol. Afterwards, only 16 bit addresses area used. In fact, the protocol can be seen as a recoloring protocol with a smaller color space.

Another similar problem happens when a rebroadcast node needs to rebroadcast a reply back to the original query

node. The rebroadcast of the reply should be done independently of the current node address. The node is not acting for itself, and may also be in a process of finding its own address. To solve this problem we have added the extended address of the rebroadcast node to each message. Therefore, each message is composed of one tentative address, two extended addresses and one message type. The usage of this extended addresses are crucial to ensure the self-stability of the protocol [7].

4. Handling incrementally deployed scenarios

One important feature of address assignment protocols, which is often forgotten, is its ability to handle late deployed sensors and merging of network partitions. The deployment of additional sensors may be necessary either to improve the sensor coverage or to improve the network lifetime, the sensors in place may be at the end of its battery. The merging of network partitions may happen either because there was an obstacle dividing nodes at the time of deployment which is now removed, or because the addition of new nodes made two or more networks reachable to each other.

In such scenarios address collisions may happen, because at the time of address assignment not every node knew about each other. Most address assignment protocols do not consider these scenarios and the ones that do, choose to rerun the assignment protocol in the colliding nodes [8]. This strategy may have a negative impact on routing, because every route established through those nodes need to be rebuilt.

Another problem that these protocols need to handle is how to detect the existence of colliding addresses. In [8], address collisions are detected during the periodically neighborhood query which is done for this purpose. However, given that, the addition of new nodes, and the merging of networks are rare, such a scheme is too energy expensive. In [9] (a protocol designed for MANETs) each packet has an additional 64 bit unique number which is used to detect address collisions, but that it is not an option in WSNs given the size of each packet.

Whenever a node is added or a barrier is lifted between two or more network partitions it is possible that two or more nodes with the same address became reachable by a single node. A k -way collision happens whenever k nodes with the same address are reachable by one node, a.k.a the detector.

Both 2-way and 3-way collisions are only relevant if within a 2-hop vicinity because as stated in [7] many protocols only require local unique addresses, e.g. direct diffusion. The probability of a k -way collision in such scenario is given by equation 1, where k is the number of colliding addresses, N is the total number of nodes, n is the number of neighbors of each node (i.e. reachable in 1-hop) and $|A|$ is the address space size. If $k = 2$ (2-way collisions) and $n = N - 1$ (every node can reach every other node) the equation degenerates in the probability of the birthday paradox. In order to derive equation (1) we

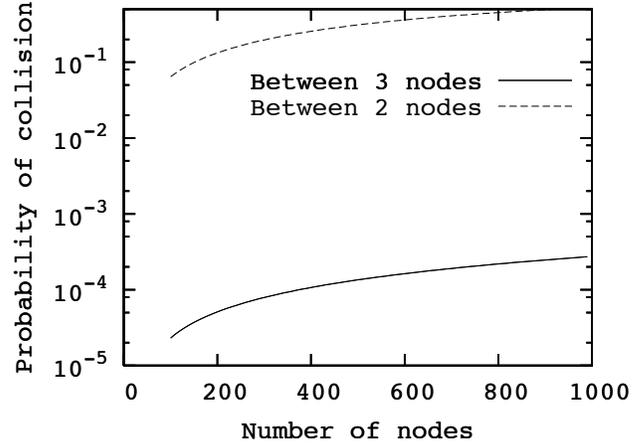


Figure 1. Collision probability between 2 nodes and between 3 nodes for a field with a density of 25 neighbors.

notice that $P(k\text{-way}) = 1 - \overline{P(k\text{-way})}$, where $\overline{P(k\text{-way})}$ is the probability of not having a k -way collision, and that $\overline{P(k\text{-way})} = 1 - p(k)^{C(k,n,N)}$, where $p(k) = \left(\frac{1}{|A|}\right)^{k-1}$ is the probability of some set of k nodes not having all the same address and $C(k,n,N)$ is the number of sets of k nodes, in radio reach of each other, that can be formed in a field of N nodes with a n neighborhood. $C(k,n,N) = C_i(k,n) + (N-n-1) \times C_o(k,n)$ where $C_i(k,n) = \binom{n+1}{k}$ is the number of sets of k nodes that can be formed within a n node neighborhood, of a chosen node in the center of the field, and $C_o(k,n) = \binom{n}{k-1}$ is the number of sets of $k-1$ nodes that can be formed using the neighborhood of each of the $N-n-1$ nodes not in the original center neighborhood.

$$P(k\text{-way}) = 1 - \left(1 - \left(\frac{1}{|A|}\right)^{k-1}\right)^{\left(\binom{n+1}{k} + (N-n-1) \times \binom{n}{k-1}\right)} \quad (1)$$

Figure 1 shows the probability of collision for 3-way and 2-way collisions for several number of nodes, giving an address space $|A| = 2^{15}$ (15 bit addresses, we will use the extra bit for collision solving) and a neighborhood of 25 nodes. The collision probability is most of the time two orders below for 3-way collisions in comparison with 2-way collisions. Moreover, since we are considering only the nodes that are deployed after address assignment, because during address assignment the collision solving protocol is not used, what is used is the RSI protocol, the probability of detecting and having to solve a 3-way collision is even lower.

Given the above results we make the hypothesis that k -way collisions with $k > 2$ are extremely unlikely and we are going to focus our efforts in detecting and solving 2-way collisions.

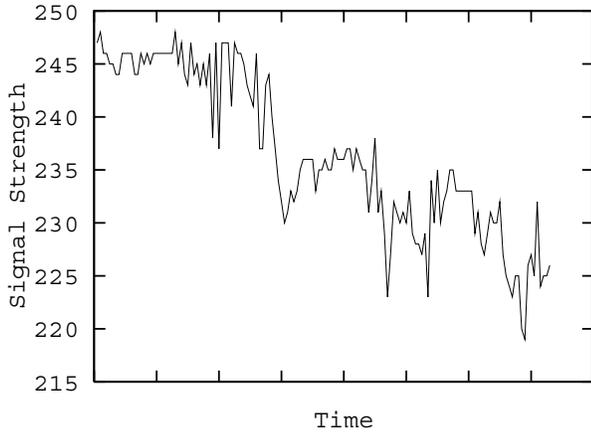


Figure 2. Signal strength of messages received by the same node over time.

4.1. Detecting address collisions

Our approach to detect address collisions is motivated by the way that people distinguish two voices in a crowd. If one of the voices is loud and the other is soft then there are probably, two persons talking. If the heard sentences do not make sense because they seem garbled, then it is possible that they are produced by more than one person. Neither of these heuristics give a precise information about the existence of colliding addresses but they may be used as triggers for a collision solving protocol.

The former solution is independent on the transport protocol while the latter is not. In order to detect out-of-order messages the transport protocol must have the notion of order which is not the case for many transport protocols in WSNs, this is why we have chosen the former solution.

Given the hypothesis that only *2-way* collisions may happen whenever the network changes only two scenarios are possible:

- The address of the added node is the same of one of the nodes already in the network and both are reachable by a third node (merge of partitions).
- The address of two of the nodes in the network are the same and they are reachable by the new node (node addition).

The first scenario is simpler than the second, although, as we will see, they will be handled the same way. If the nodes in the network knew each other they are able to know the signal strength (SS) average and standard deviation of messages sent by each other. If one of the nodes detects a message with a SS much different from the usual it may suspect of an address collision. To be sure of the collision it will have to run the collision solving protocol described in the next section.

We have started by using an algorithm from Knuth [4] to incrementally calculate the average (\bar{ss}) and standard deviation σ_{ss} of the SS without having to keep all samples, i.e. the calculus is incremental. In order to get a four nines confidence level in the collision detection we

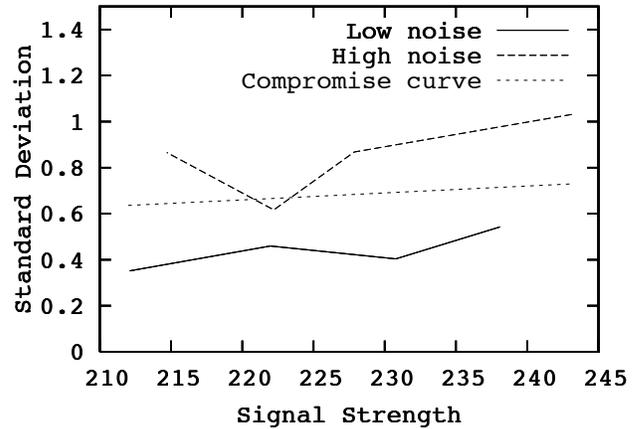


Figure 3. Standard deviation behavior with the signal strength ($SS_{max} = 255$) of messages.

check if the SS of each message is within four times the standard deviation of the average ($|\bar{ss} - ss_i| < 4\sigma_{ss}$), otherwise we signal a collision. However, we have realized that the SS average varies over time due to battery drain and environment changes. Figure 2 shows the signal strength over time, of messages received in a field of MicaZ Motes using the TinyOS 802.15 stack. The first approach to solve this problem was to calculate the SS average and standard deviation using only the last few messages, however that proved ineffective because the standard deviation with too few messages lacked the necessary precision.

Instead of computing both the average and standard deviation we chose to compute the average over the last few messages and predict the standard deviation based on the average. As expected the standard deviation over a few messages is very small in low noise scenarios and is almost twice in high noise scenarios (Figure 3). A good compromise is given in Figure 3, in which the predicted standard deviation is given by $\hat{\sigma}_{ss} = 0.003ss_i$. With this information the protocol signals a collision whenever $|\bar{ss} - ss_i| > 4\hat{\sigma}_{ss}$. Our measurements showed that this equation is suitable both for low and high noise scenarios, however if for extremely high noise scenarios the number of false collisions detected by this protocol is too high we incorporate a self regulatory mechanism which increases the standard deviation every time that a false positive is detected.

This solution has another advantage it is able to handle the second collision scenario. If a node arrives to a network and starts to communicate with two nodes with the same address at the same time, it will not have previous information about average and standard deviation therefore it will not be able to find discrepancies with past history. However, with this solution, although the average will still be wrongly calculated because it will be something in between the two signals of the two communicating nodes, the standard deviation will not change much, which will allow the detection of the collision.

The Listing 1 shows that whenever a message is received with a signal strength above or below a predefined confidence

```

...
typedef struct {
    uint_16 add;
    byte ss_avg[2]; // signal strength average.
    byte n[2]; // messages received.
    boolean solving;
} NodeRecord;
const int maxMsgCount = 15;

void recMsgPwr(int add, int msgtype,
    int isB, // 1-broadcast, 0-otherwise.
    int ss) { // signal strength.

    NodeRecord nrec = getAddRecord(add);
    if(nrec.solving) //if its already solving
        return; //for that address, returns.
    if(nrec.n[isB] > 0) {
        int ci = 4 + falsePositives;
        byte diff = abs(ss - nrec.ss_avg[isB]);
        if(diff > ci*0.003*ss)
            startColSolving(add);
    }
    if(nrec.n[isB] > maxMsgCount) {
        nrec.n[isB] = 1;
        nrec.ss_avg[isB] = ss;
    } else {
        nrec.n[isB]++;
        int delta = ss - nrec.ss_avg[isB];
        nrec.ss_avg[isB] += delta/nrec.n[isB];
    }
}

```

Listing 1. Detecting an address collision using previous information on signal strength.

interval the solving protocol (see section 4.2) is started.

Notice that the set of values kept for each address is comprised by the average of the signal strength of every message received (*ss_avg*), the number of messages received from that address and a value indicating that a conflict is being solved. Notice, also, that the record structure uses two sets of values for each address, because it is expected that broadcast communications be done with a different transmission power than unicast communications. The transmission power of unicast communications is usually adapted to the distance between peer nodes, while the broadcast communications do not have this kind of adaptation.

We have conducted two sets of experiments, using MicaZ sensor nodes running TinyOS 2.0, in three different environments: two small non-noisy environments and one open and noisy environment, i.e. a large student hall with many students moving, each one with its own laptop device with Wifi, and many Wifi antennas in the vicinity. The first set of experiments were designed to detect false positives, and the second to detect false negatives.

The false positive ratio is an important metric because it impacts the energy consumed by each node in the collision solving protocol. As expected the number of *false positives* in the non-noisy scenarios was close to 0% which is consistent with the four nines accuracy specification. In the noisy environment the system showed a rate of almost 11% of false positives if no adaptation is taken place, however, with the self regulatory mechanism after 3 false positives the system reaches a stability situation without (0%) false positives. Another interesting result is the distribution of false negatives with the distance. Most false positives (10%)

were experienced when the colliding nodes were 5 meters apart from the arriving node. For larger distances, within the transmission ratio of our nodes (in noisy environments 15 meters), the false negative rate is much lower, which may be related with the saturation of the signal (it is difficult to distinguish two persons shouting very near to us).

Although, we have defined two collision scenarios: the arriving node connecting two networks previously disconnected and the arriving node having the same address of another node, we have only measured the false negatives in the former, because it is more general than the latter. Notice that after *maxMsgCount* messages received from the same address, the detecting node will be in the same situation than the arriving node that connects two networks, because the SS history will be cleaned.

Assuming that all nodes have an equal radio range, the arriving one will be in the position of connecting two previously disconnected nodes with potentially the same address if it is place within the intersection of the two radio coverages, and the trouble of detecting a collision will be higher when the node is place at the exact some distance of the tow colliding nodes. The protocol shows 0% of *false positives* if it is closer to one of the colliding nodes, or if it is run in a low noise environment, however in a high noise environment with the two colliding nodes at exactly the same distance we have experience a false negative rate of 4%. A false negative does not mean that the arriving node will never detect the collision, it just means that it is not able to detect the collision within the frame period of 2 times *maxMsgCount*. Fast detection is important to minimize the impact of garbled communications.

4.2. Collision solving protocol

The collision detection protocol described in the previous section does not provide a definitive answer on the existence of an address collision. It ends by sending a query to every node with a specific address. Only if several nodes reply (with different extend addresses) the collision is confirmed.

When a node detects a collision it gives to every node with the same address a nickname, and informs the node of that nickname. The situation is similar to having two students in the same class named John, and refer to one as “Little John” and to the other as “Big John”. Notice that they will still be named John for every one else, and we cannot just name them “Little” and “Big”, because we would create other collisions.

The solution is to reserve one bit from the 16 bit addresses for nicknames. Therefore, only 15 bit of the 16 bit addresses are assigned by the address assignment protocol, the remaining bit is originally set to zero. When a node detects a collision it informs each of the colliding nodes that one of their address will have to set the bit to one. Each of the colliding nodes stores in a table the nickname for which it is known by that node. Whenever the node that changed its address receives a message from the node that detected the collision it will only accept it if the bit on the

destination address is set to one. Notice that other nodes continue to communicate with the node that changed the address with the old address, the change is only relevant for the communication with nodes that detect the collision.

This solution is only able to solve a single collision. If the address of a node collides with two other nodes the protocol does not work because, it would be possible for a node to end-up being known by two nicknames by two different nodes, which would have a negative impact in broadcast communications. In fact, unicast communications would not be affected because the node could choose the nickname to use depending on the message destination, however for broadcast communications the node would not know the nickname to choose. Nevertheless, this is not a big problem because 3-way collisions are much less probable than 2-way collisions.

The proposed algorithm is shown in Listing 2. The algorithm assumes that the node detecting the collision (the initiator) has sent a “collision query” message (`ColQuery`) to all the nodes with the colliding address. After receiving that message a node replies with its extend address. When the initiator receives a “collision reply” (`ColReply`) message it schedules a “collision solving” message (`ColSolve`) to be sent after a predefined timeout. If the node receives another “collision reply” message with a different extended address it confirms the existence of a collision. If that happens it marks the address has a collision and modifies the “collision solving” message waiting to be sent by adding the new extended address. Finally, upon receiving the “collision solving” message the colliding nodes choose independently the one that is going to adopt a nickname by comparing the extended addresses. The one with the smallest extended address adds a nickname to its address. Note that whenever there is no collision (i.e. the collision detection protocol had a false positive) none of the nodes adopts a nickname, because the smallest extended address in the message is 0 which is an invalid extended address.

The `cleanMsg` method is used whenever a message is received. If the message comes from someone with a nickname and the receiver have not detected that collision (some other node did) the nickname bit is clean. This ensures that a node that adopted a nickname may use it in broadcast communications.

5. Conclusion

We have described an algorithm for detection of address collisions which thus not require the transmission of extra messages and, a collision solving protocol which thus not hinder routing tables. The overall collision solving system is very energy efficient because: the collision solving protocol only takes 4 messages; it does not invalidate existing routing tables, which means that routing tables do not need to be reconstructed; and the collision detection algorithm is very precise in most scenarios. Overall the amount of energy wasted to solve address collisions is very small. We have tested the collision detection algorithm and proved that

```

...
void receiveMsg(msg_t msg) {
    switch(msg.type) {
        ...
        case ColQuery:
            msg = {msg.s_id, myId, ColReply, myXId};
            send(msg, replyPower);
            break;
        case ColReply:
            if(isMsgScheduledTo(msg.s_id, msg.xid[0])) {
                addXIdToMsg(msg.s_id, msg.xid[0]);
                markAdd(msg.s_id);
                falsePositives--;
            } else {
                msg = {msg.s_id, myId, ColSolve, msg.xid[0]};
                scheduleMsgToSend(msg, timeout);
                falsePositives++;
            }
            break;
        case ColSolve:
            if(myXId == min(msg.xid[0], msg.xid[1])
                myId != 0x80;
            break;
    }
}
void cleanMsg(msg_t msg) {
    if(isMarked(msg.s_id))
        msg.s_id &= 0x7f;
}

```

Listing 2. Solving an address collision in late deployed nodes.

radio-print measurements are quite effective in distinguishing two non-byzantine nodes.

6. Acknowledgments

This work was partially supported by the European Community’s FP7/2007-2013 under grant agreement 224621.

References

- [1] Mobile Ad, Charles E. Perkins, and Samir R. Das. IP address autoconfiguration for ad hoc networks. Internet Draft draft-ietfmanet-autoconf-01.txt, Internet Engineering Task Force, MANET WG, July 2000.
- [2] Maria Gradinariu and Colette Johnen. Self-stabilizing neighborhood unique naming under unfair scheduler. In *Euro-Par*, pages 458–465, 2001.
- [3] Ted Herman and SÉbastien Tixeuil. A distributed tdma slot assignment algorithm for wireless sensor networks. In *ALGOSENSORS*, pages 45–58, 2004.
- [4] D. E. Knuth. *The art of computer programming. Vol.2: Seminumerical algorithms*. Atmospheric Chemistry & Physics, 1981.
- [5] Aleksandar Micic and Ivan Stojmenovic. A hybrid randomized initialization protocol for tdma in single-hop wireless networks. In *IPDPS*, 2002.
- [6] Koji Nakano and Stephan Olariu. Randomized initialization protocols for ad hoc networks. *IEEE Transactions on Parallel and Distributed Systems*, 11(7):749–759, 2000.
- [7] Carlos Ribeiro. Robust sensor self-initialization: Whispering to avoid intruders. In *SECUREWARE 2007, The International Conference on Emerging Security Information, Systems, and Technologies*, pages 101–107. IEEE Computer Society, October 2007.
- [8] Curt Schurgers, Gautam Kulkarni, and Mani B. Srivastava. Distributed assignment of encoded MAC addresses in sensor networks. In *MobiHoc*, pages 295–298. ACM, 2001.
- [9] Nitin H. Vaidya. Weak duplicate address detection in mobile ad hoc networks. In *MobiHoc*, pages 206–216. ACM, 2002.