On the Evaluation of Self-addressing Strategies for Ad-Hoc Networks

Ricardo de O. Schmidt[1], Aiko Pras[1], and Reinaldo Gomes[2]

[1] Design and Analysis of Communication Systems
University of Twente, The Netherlands
{r.schmidt,a.pras}@utwente.nl
[2] Systems and Computing Department
Federal University of Campina Grande - Brazil
reinaldo@ufcg.edu.br

Abstract. Ad-hoc networks are supposed to operate autonomously and, therefore, self-* technologies are fundamental to their deployment. Several of these solutions have been proposed during the last few years, covering most layers and functionalities of networking systems. Addressing is one of the critical network operations on supporting others such as routing and even security of IP-based communcations. This paper has the goal of putting together different strategies for self-addressing problem by evaluating five self-addressing protocols and pointing out their strenghs and drawbacks. At this time, we did not intend to come up with a novel approach for the problem, but to evaluate the existing ones in a non-isolated manner and also considering critical ad-hoc networking situations of partition and merging. Conclusions about the evaluated approaches and their applicability are drawn at the end of this paper.

1 Introduction

Autonomous networking systems represent a new paradigm mostly based on an innovative cooperation model of network devices to create and manage communication environments automatically. The idea of creating autonomous networking systems relies heavily on the concept of autoconfiguration. The autoconfiguration problem, also self-management, can be seen as the main reason for the emergence of self-* technologies. Due to its importance for the proper network operation, addressing can be seen as one of the most important (and challenging) issues in self-* solutions.

According to documents, like [1], published within the IETF working group AUTOCONF [2], among the goals of autoconfiguration in dynamic networks, like ad hoc networks, we must consider the configuration of unique address for nodes. The well known DHCP (Dynamic Host Configuration Protocol) has a very limited applicability when considering dynamic characteristics found in these networks, such as mobility, unpredictable number of nodes, and undefined topology. Addressing is a critical functionality in IP networks given that it provides basic configuration for other network functionalities such as routing and security.

R. Lehnert (Ed.): EUNICE 2011, LNCS 6955, pp. 31–42, 2011.

Attempting to solve the addressing problem in dynamic networking systems, several self-addressing solutions were already proposed, like the ones surveyed in [3] and [4]. These solutions implement different methodologies to allow self-configuration of nodes interfaces with tentative of valid and unique addresses within the network. Such methodologies range from simple random selection of addresses from predefined addressing spaces, to mathematical efforts where equations are defined to enable nodes to calculate their own interface(s) address(es). These solutions are roughly classified as stateless, stateful or hybrid approaches, discussed in more details in the next section.

The main goal of this paper is to contribute to the specific area of addressing by comparing completely different self-addressing approaches in critical ad hoc networking situations, like networks merging and partition. Past works have evaluated several strategies for self-addressing. However, most of them only submit the proposed solutions to simple network behaviors, what may not be the case in dynamic ad hoc networks. It is important to state that we do not intent to propose a novel approach for self-addressing, but to evaluate those we consider important methodologies for approaching the problem. With the results in this paper and also further experiments with specific functionalities (like merging/partition management), we plan to come up with a stable, scalable and reliable self-addressing solution, developing new ideas or using already existing ones that proved to be efficient.

The rest of this paper is organized as follow. The concepts of self-addressing are introduced in the next section. Section 3 presents the methodology used in the experiments as well as a brief description of the implemented and simulated self-addressing protocols. Results obtained from the simulations are presented in section 4. Finally, in section 5, conclusions about the evaluated strategies are drawn.

2 Self-addressing

Self-addressing is tightly related to the concepts of autoconfiguration. It can be part of a set of technologies that enable a network to operate autonomously. Basically, a self-addressing protocol must provide a node the ability of or generating its own addressing configuration or retriving such configuration from another network entity. Self-addressing protocols can roughly be classified in one of the following categories: stateless, stateful or hybrid.

In short, stateless approaches like Strong DAD [5] are those where nodes do not keep track of addresses in use in the network (i.e., the state of addresses is unknon). For instance, a stateless protocol can randomly select an address from a predefined range and test it within the network to ensure that no other node is configured with such address. This testing procedure is generally named Duplicate Address Detection (DAD). Stateful approaches like SooA [6], on the other hand, allow nodes to be aware of addresses state. Usually, in a stateful approach, the nodes that keep track of addresses are responsible for performing addressing tasks on assignment and management of resources. For instance, information

about addresses can be stored in tables. However, alternative strategies like Prophet Allocation [7] define mathematical approaches which allow nodes to track addresses that may be in use, thus characterizing a stateful solution. Finally, hybrid approaches mix properties of both stateless and stateful protocols. Hybrid solutions usually implement random address selection, DAD testing, and the address registration within one or more addressing authorities (like in HCQA [8]) or within tables spread through the network (like in MANETconf [9]).

3 Experiments

3.1 Evaluated Protocols

Five self-addressing protocols were compared in our experiments. Three stateless protocols: Strong DAD, AIPAC and AROD; one stateful: Prophet Allocation; and one hybrid: MANETconf. The protocols AROD and AIPAC were fully evaluated. That is, we also considered their mechanisms for handling networks merging and partition. All protocols were implemented in C++ and added to the well known network simulator NS2 (Network Simulator 2) [10].

We opted for these protocols due to their different proposals on addressing assignment, DAD procedures and partition/merging management. In our experiments we aimed a fair comparison among the five protocols, following AUTOCONF's [2] goal of proposing a common and general self-addressing solution for ad-hoc networks. The protocols were submitted to the same network scenarios, disregarding their implementation (or not) of advanced operations for handling complex networking situations. Next, a brief description of each evaluated protocol is given.

Strong DAD [5] is a stateless addressing protocol which implements random selection of addresses followed by DAD procedures. Each starting node is responsible for its own address configuration. Upon starting a node selects randomly, from a predefined range, two addresses: temporary and tentative addresses. The former is used as the node's address during the DAD procedure on behalf of the latter. On succeding in the DAD procedure, the tentative address is used to configure the node's interface. Otherwise, the tentative addresse selection and testing is done until no conflicts are identified. Strong DAD does not deal with network partition and merging. Consequently, its applicability to more critical networking scenarios is restricted.

AIPAC [11] implements the concept of requester and initiator nodes. The former is a starting node and the latter an already configured node which negotiates an address within the network on behalf of the former. As well as in Strong DAD, a starting node randomly selects temporary address and uses it to communicate with its initiator node during the configuration procedure. This temporary address is used until a negotiated one is received from the initiator. The negotiation procedure is executed by the initiator node. It randomly selects an address from a predefined range and tests it with the network through a DAD procedure. AIPAC also defines functions for dealing with partition and

merging of networks. A network identification, defined by the first node (i.e., network's founder), is added to the protocol's messages, enabling nodes to identify different networks that come into each others range. AIPAC's main drawback is the dependancy on routing messages and tables. It requires modifications to the routing protocol operating in the network, making AIPAC unsuitable for dynamic networks that may even negotiate and change the routing protocol.

AROD [12], although being also based on DAD procedure and requester-initiator scheme, focuses on reducing configuration time and control overhead. To do so, it implements address reservation and optimistic DAD. The main difference is that an initiator node may have reserved addresses, previously validated within the network, which can be allocated to a requester node without the DAD delay, reducing the configuration time. A DAD procedure is usually executed for more than a single tentative address at the same time, which allow nodes to obtain spare addresses for future configurations of requester nodes.

Prophet Allocation [7] is the only evaluated approach in this paper that does not implement DAD procedure (at least not during the procedures specified by the authors). Unlikely the other described solutions, Prophet Allocation does not select addresses randomly, but it obtain addresses inside a predefined range from a formula. This formula is based on the fundamental theorem of arithmetic, which says that: every positive integer may be expressed uniquely as a product of primes. Each node has a 2-tuple identifier, composed by their address and the formula's state. Everytime an address is calculated through the formula, its state is incremented by one. However, as stated by the authors, this formula might generate duplicated addresses within a network. Therefore, when a node starts (requester) it receives from an already configured node (initiator) an address, generated by the formula, and a network identifier. This identifier is added to the protocol's messages enabling nodes to identify messages from different networks (merging situation). Unfortunately, no additional information on how the protocol deals with the merging problem was provided by the authors in [7]. Consequently, due to this the lack of information, we did not considered management of network partition and merging as Prophet Allocation function.

Finally, MANETconf [9] is a completely distributed and hybrid addressing protocol, which implements allocation tables and DAD procedure. Its functionality is simple but it requires a lot of broadcasting for DAD and tables synchronization. MANETconf also implements requester and initiator nodes. The difference is that after successfully negotiating an address to a requester node, the initiator floods the network announcing that such address has been allocated. With this flooding, all network nodes update their respective allocation tables. The allocation tables are used to increase the reliability of the DAD procedure. Despite the fact of flooding the network with broadcasts, MANETconf has the advantage of not depending on any other technology.

3.2 Scenarios and Evaluation Metrics

The experiments were planned to evaluate the implemented protocols under critical situations of ad hoc networks. The protocols were set to operate with very

limited addressing resources (256 available addresses) and submitted to conditions of random deployment and mobility of nodes, leading to situations of networks isolation, partition and merging. Two different scenarios were simulated, as described below.

In the first scenario, named *Scenario A*, 100 nodes ramdonly positioned in a predefined area, forcing situations of nodes isolation in the beginning of the simulation, due to geographical distance between nodes. In this scenario nodes are static and merging was resulted from the deployment of intermediate nodes between the isolated networks. In the second scenario, named *Scenario B*, two initially isolated 10-node networks were deployed and later merged, resulting in a 20-node network. The area of these networks was defined so that no isolated nodes were deployed in the initial formation. Simulation time and nodes arrival rate did not play an important role in our scenarios. In addition, from previous experiences, like [6] and [12], we learned that variations on addressing space have a very low impact on protocol's operation.

We made use of a methodology described in [13] to determine the necessary number of simulation runs so that our results would fall sufficiently close to the mean, in a confidence interval of 95%. This guarantees that we perform a fair comparison between two or more solutions mainly due to the use of random variables (i.e., nodes positioning and random mobility).

To evaluate the protocols, we considered three basic metrics: (a) control overhead, which quantifies the data generated and transmitted during the protocols operation; (b) configuration delay, which represents the time between the node deployment and its final configuration; and (c) address uniqueness, which measures the efficiency of protocols on avoiding problems with duplicated configurations (i.e., conflicting configurations).

4 Results Analysis

4.1 Scenario A: Highly Populated Network

The graphics in Fig. 1 present the traffic generated and transmitted by the protocols while configuring all 100 nodes. However, the lines do not represent all the traffic generated during the entire simulation time, which was 2000 seconds, but the traffic from the beginning of the simulation until the configuration of the last node.

Regarding the traffic in number of packets, as presented in Fig. 1(a) and 1(b), and the traffic i nnumber of bytes, as presented in Fig. 1(c) and 1(d), one can conclude in a first moment that the best performances were reached by the protocols Strong DAD and AROD. However, we must consider that the former implements only allocation of addresses and no further maintenance operations.

When the last node is configured with Strong DAD, the protocol concludes its operation. Therefore, all the traffic generated by the protocol is related to address assignment and DAD procedure. Thus, if after configuring all nodes, n nodes happen to be configured with duplicated addresses, such problem will remain unsolved until the end of the simulation or, in another possible scenario,

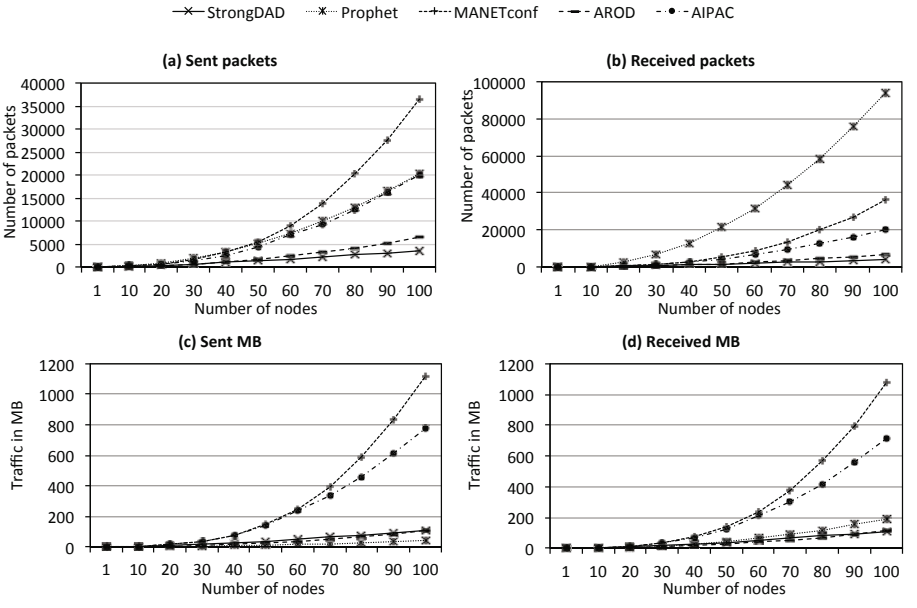

Fig. 1. Protocols performance in *Scenario A*

only one of the conflicting nodes remains active in the network. Even being a flood-based approach, Strong DAD generated significantly less traffic than the other approaches. It is due to the fragmentation of the network resulted from the random positioning of nodes in the simulation area, where the less neighbors a node has the smaller is the impact caused by its flooding procedure. In addition, the number of bytes transmitted by Strong DAD grows in different rates than the number of packets. It occurs because in the DAD procedure a list of previous hops (reverse path) must be kept within a request message for allowing a node, which identifies a conflict, to reply correctly to the request generator. It is also applicable to other protocols that implement DAD procedure with reverse path.

As one can observe in Fig 1, regarding traffic overhead, AROD performed better that Strong DAD, even executing DAD procedures. The main reason for that is the address reservation implemented by AROD, where for testing several addresses with DAD, the protocol generates pratically the same overhead than Strong DAD would generate for testing a single address. With the address reservation, AROD avoids further DAD executions when configured nodes provide new ones with already tested addresses. It is important to inform that in our experiment nodes operating AROD were able to reserve only one address in addition to the address for their own configuration. One can infer that the protocol performance may improve if, in cases that the addressing space is larger, the nodes are allowed to reserve more than one address for further allocations.

Prophet Allocation, which does not implement DAD procedure on address allocation, did not generated high overhead to configure nodes. The simple handshake procedure for allocating an address allows the protocol to quickly configure new nodes. Given that in this experiments we did not considered cooperation between addressing and routing protocols, the strategy for address maintenance implemented by Prophet Allocation was the responsible for the most overhead. The periodically ack broadcasted by all nodes was the reason for the high values achieved by Prophet Allocation regarding the control overhead. The difference between the sent and received values achieved by the protocol is due to the number of neighbors each node has, where the higher the number of neighbors, the higher is the number of received packets/bytes. In addition, due to the small payload of ack messages, the numbers of packets is much higher than the number of bytes, as observed in Fig. 1.

The traffic generated by AIPAC when configuring nodes is similar to Strong DAD. However,the exceeding traffic generated by AIPAC, as observed in Fig. 1, is due to the execution of merging management procedure called "gradual merging". As one can observe in Fig. 1, in general MANETconf generated most traffic when configuring all nodes. It is due to the excessive flooding during the procedures of DAD and tables synchronization. The need of all nodes replying a DAD request received generates a very high control overhead, which is summed with the broadcasted message used to update allocation tables on every new address that is allocated.

Due to post-allocation address maintenance, Prophet Allocation, AROD and AIPAC continued to generate traffic even after concluding the configuration of all nodes. However, simulation parameters were set (e.g., nodes arrival rate) so that the configuration of the last node happened close to the 2000 seconds. This way, the additional traffic did not impact significantly in the final results.

Fig. 2 illustrates the average delay achieved by the the protocols for configuring the nodes in Scenario A. Strong DAD usually has an average of 15 seconds for nodes configuration, due to the three rounds of 5-second DAD procedure. However, given that several nodes were isolated in the beginning of the simulation, Strong DAD configured such nodes without executing the DAD procedure and, consequently, the average delay at the end of the simulation dropped to 9 seconds.

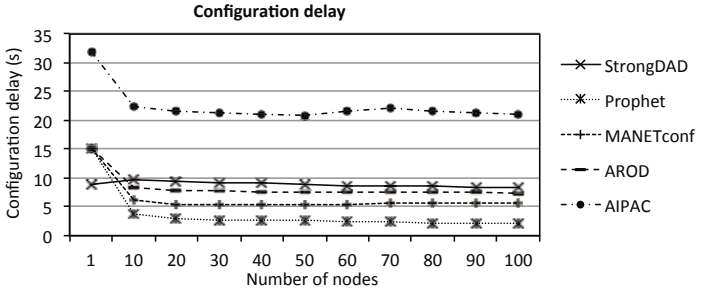

Fig. 2. Configuration delay

The configuration delay in MANETconf is strictly connected to the duration of the DAD procedure. Consequently, the average delay achieved by MANETconf was 5 seconds. In this case, the time for updating the allocation tables was not considered as part of the configuration delay because such procedure is only triggered after the new node is already configured. The merging procedure implemented by MANETconf solves conflicts locally and, consequently, it did not impacted significantly on the protocol's performance.

Unlike MANETconf, the highly fragmented network had a strong impact on AROD operation. The average configuration delay achieved by AROD was 8 seconds, even operating with the address reservation scheme. The control overhead for addressing maintenance and the merging management in AROD compromised its performance. At the end, the reliable mering aimed by AROD spent much time and generated excessive traffic.

Prophet Allocation achieved the best results reagarding configuration delay due to its simple handshake for address allocation. However, we must consider that the protocol's authors assume that the implemented mathematical equation may generate duplicate addresses in a certain moment. One can infer that by executing procedures for conflicts correction, which would be a flooding-based DAD, a much higher maintenance load would be demanded from the nodes, what would result in performance degradation on overhead and configuration delay. In this case, we believe that the averages presented as results in Fig. 2 for Prophet Allocation would be increased in 5 seconds.

AIPAC achieved the highest configuration delay. Its DADi procedure lasted 15 seconds in average. Two factors mainly contributed for such results. First, although merging problems were corrected later by the protocol, the procedures for gradual merging were unnecessarily triggered several times during nodes configuration. The second reason is that AIPAC, unlike the other protocols, do not consider networks of one node. It means that an isolated node does not configure itself if it does not have at least one neighbor. Therefore, isolated nodes in the beginning of the simulation did not configured themselves until another node was deployed in their neighborhood. This waiting time increased the average configuration delay for the first nodes, as shown in Fig. 2.

A serious problem identified on the results of this experiment was the number of nodes, at the end of the simulation, that were configured with conflicting addresses. Both netowrk fragmentation and reduced addressing space drastically impacted on the configuration consistency. As presented in Table 1, all evaluated protocols concluded their operations with conflicted nodes configuration.

The weak performance of AIPAC, if compared to the other protocols, was rewarded by the success achieved on merging a highly fragmented network and finishing the simulation with the lowest number of conflicts. When compared to the other protocols, MANETconf also achieved good results on the number of conflicts. It is due to the allocation tables implemented in all nodes. As well as with AIPAC, the costly performance of the protocol was justified by the reliability of its configurations at the end of the simulations.

Table 1. Address Conflicts in *Scenario A*

Protocol	Conflicts
StrongDAD	29
Prophet Allocation	45
MANETconf	8
AROD	30
AIPAC	5

Strong DAD configured initially isolated nodes and later did not handle merging situations by locating and correcting possible conflicts. Consequently, the number of conflicts was very high with Strong DAD. Prophet Allocation achieved the highest number of conflicts at the end of the simulation. Given that it does not implement a procedure for handling merging, the fragmented network was also one of the reasons for its bad performance. In addition, Prophet Allocation allows an implementation of more states in its equation, which would generate more distinct sequences of addresses and, according to its authors, less conflicts. However, we decided for implementing its equation with only one state due to our goal of testing the protocols in very limited conditions.

Although AROD ended the simulation with high number of conflicts, it is important to mention that the protocol's procedure for conflict resolution was not implemented in this experiment. However, AROD successfully identified all the conflicts in the network. Such conflicts were resulted from the merging of all isolated nodes and small networks. Considering that AROD implements DAD procedure for resolving address conflicts, triggering such operation would degrade the protocol's performance by increasing control overhead and configuration delay. This leads us to conclude that it would bring AROD's results close to AIPAC's.

4.2 Scenario B: Handling Networks Merging

Past works have evaluated self-addressing protocols under more controlled networking scenarios, considering mainly the allocation procedure and not the resources maintenance and management after allocation. However, real ad-hoc network scenarios may be quite different. In these, nodes arbitrarily come and go. In some situations not only the nodes move, but entire networks may move from one place to another, temporarily or permanently merging with each other. In this second scenario, we tested the merging procedures of AROD and AIPAC in a scenario of merging between two networks. Each network was composed by 10 nodes. In a predefined moment, after the initial configuration of all nodes, network *N1* moved towards network *N2*, merging and creating a single network *N3* with 20 nodes.

Both protocols AROD and AIPAC successfully performed the network merging. The difference between the strategies implemented by the protocols for handling merging is clearly observed on the results presented in Fig. 3. The conflict

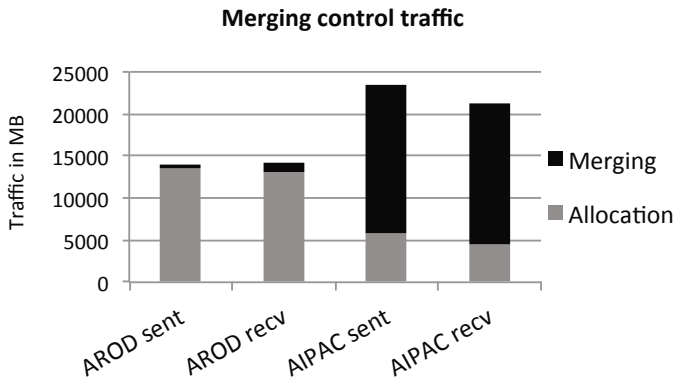

Fig. 3. Merging traffic Vs. Allocation traffic

resolution procedures were not considered in this experiment, but seem it is seen as future work.

As soon as AROD identifies that two networks start to overlap, the mechanism for merging is triggered and a single network is quickly formed. The merging procedure in AROD affects all the nodes in one of the overlapping networks. On identifying that it is necessary to join the other network, a node announce the merging to its network's leader and the latter floods the network with a reconfiguration message. It is a quick process and does not generate high control overhead. As one can observe in Fig. 3, in average only 5% of the total traffic generated by AROD was resulted from the merging procedure.

On the other hand, AIPAC needs more time for executing the merging of networks. In addition, as illustrated in Fig. 3, if compared to AROD, a much higher percentage of its traffic is related to the merging procedure. After successfully merging the two networks, the traffic related to merging accounted for 80% of the total traffic generated by the protocol. It was due to the gradual merging strategy implemented in AIPAC, where some nodes migrated to the other network and later migrated back to their original network. Therefore, such situations doubled the traffic generated by these nodes during the merging procedure.

Regarding the procedure delay, AROD detected and performed merging faster than AIPAC. AROD started the merging procedure as soon as the first nodes identified the networks overlap, what happened in averages. AROD needed in average 40 seconds for completing the merging process. With the gradual merging strategy, AIPAC took longer for detecting the merging and concluded the procedure with and average delay of 94 seconds.

5 Conclusion

Several self-addressing solutions were already proposed for ad-hoc networks. From experiments, like the one presented in this paper, we can observe that

such solutions still lack on providing alternatives for handling complex networking situations. Consistency of addresses is important due to its influence on other operations like routing. However, designed strategies for dealing with this issues result in problems with excessive control overhead and degradation on performance of basic addressing. In addition, dependency on other technologies is not a good approach for self-addressing protocols when considering Future Network scenarios. For instance, addressing solutions like AIPAC and Prophet Allocation, which depend on routing protocols, even requiring modifications on routing operations, have their applicability drastically limited to scenarios with the specific routing protocol. It is even more problematic if we consider scenarios where the routing is also dynamic and two or more routing protocols may coexist and cooperate.

This paper presented the first steps on evaluation of self-addressing approaches. As future work, the authors plan to carry more extensive and complete simulations with self-addressing solutions, also covering other network scenarios. Our main goals are: (a) to contribute with the decision on the best self-addressing solution (or combination of two or more solutions) for different ad-hoc networks; (b) to develop a general evaluation framework for self-addressing solutions, based on AUTOCONF [2] documentation on requirements and guidelines; and (c) to propose an alternative self-addressing solution for SooA [6], which is going to be used on the absence of addressing servers, providing nodes with temporary configuration.

References

1. Baccelli, E.: Address Autoconfiguration for MANET: Terminology and Problem Statement. IETF Internet Draft (2008)
2. AUTOCONF: Ad-Hoc Network Autoconfiguration. IETF Working Group, `http://www.ietf.org` (accessed in April 2011)
3. Bernardos, C., Calderon, M., Moustafa, H.: Ad-Hoc IP Autoconfiguration Solution Space Analysis. IETF Internet Draft (2008)
4. Weniger, K., Zitterbart, M.: Address autoconfiguration in mobile ad hoc networks: current approaches and future directions. IEEE Network Magazine, Special Issue on Ad Hoc Networking: Data Communications & Topology Control 18(4), 6–11 (2004)
5. Perkins, C.E., Malinen, J.T., Wakikawa, R., Belding-Royer, E.M., Sun, Y.: IP Address Autoconfiguration for Ad Hoc Networks. IETF Internet Draft (2001)
6. de Schmidt, R.O., Gomes, R., Sadok, D., Kelner, J., Johnsson, M.: An Autonomous Addressing Mechanism as Support for Autoconfiguration in Dynamic Networks. In: Proceedings of the Latin American Network Operations and Management Symposium, LANOMS (2009)
7. Zhou, H., Ni, L.M., Mutka, M.W.: Prophet Address Allocation for Large Scale MANETs. In: Proceedings of the 22nd Annual Joint Conference of IEEE Computer and Communication Societies (INFOCOM), vol. 2, pp. 1304–1311 (2003)
8. Sun, Y., Belding-Royer, M.E.: Dynamic Address Configuration in Mobile Ad Hoc Networks. Technical Report 2003-11, University of California at Santa Barbara (2003)

9. Nesargi, S., Prakash, R.: MANETconf: Configuration of Hosts in a Mobile Ad Hoc Network. In: Proceedings of the 21st Annual Joint Conference of IEEE Computer and Communication Societies (INFOCOM), vol. 2, pp. 1059–1068 (2002)
10. Fall, K., Varadhan, K.: The ns Manual (formerly ns Notes and Documentation). The VINT Project (May 2010), `http://www.isi.edu`
11. Fazio, M., Villari, M., Puliafito, A.: AIPAC: Automatic IP Address Configuration in Mobile Ad Hoc Networks. Elsevier Computer Communications (COMCOM) 29(8), 1189–1200 (2006)
12. Kim, N., Ahn, S., Lee, Y.: AROD: An address autoconfiguration with address reservation and optimistic duplicated address detection for mobile ad hoc networks. Elsevier Computer Communications (COMCOM) (30), 1913–1925 (2007)
13. Jain, R.: The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modeling. Wiley-Interscience, New York (1991); ISBN 0471503361