Security issues in Address Autoconfiguration Protocols: An improved version of the Optimized Dynamic Address Configuration Protocol

André Langer  
Department of Computer Science  
University of California, Santa Barbara  
langer@cs.ucsb.edu

Tom Kühnert  
Department of Computer Science  
University of California, Santa Barbara  
kuehnert@cs.ucsb.edu

ABSTRACT

Dynamic address assignment is one of the most important features in wireless ad hoc networks if nodes should be enabled to join and to work in the network by automatically configuring all necessary settings. Different approaches have been developed throughout the last years to achieve this objective of Dynamic Address Autoconfiguration but research primarily focused on efficiency and correctness, less on security issues. Whereas Duplicate Address Detection has become reliable in commonplace scenarios, it is still relatively easy to suspend the whole network functionality in extraordinary situations within the boundaries of a Dynamic Address Configuration Protocol. In this paper, we therefore want to point out shortcomings and weaknesses in existing protocol solutions which address dynamic IP address assignment. We concentrate on a leader-based approach called ODACP and want to propose several solutions which improve the original protocol in such a way that it is safer against malicious host activities. Finally, we will demonstrate the improvements of our solution in a separate test scenario.

Keywords: 
Ad hoc networks, Address Autoconfiguration, ODACP, malicious nodes, Denial of Service attacks, security aspects

I. Introduction

Throughout recent years, so called mobile ad hoc networks have become more and more important because they allow building up wireless networks of mobile hosts and wireless routers. The main advantage of ad hoc networks is the fact that no further details about the network topology or infrastructure must be known, in fact it is not even necessary that any infrastructure exists. Ad hoc networks shall have the ability to work autonomously and spontaneously without any centralized instance. All necessary network adjustments shall be made in a self-configurable manner.

For this reason, many new routing and configuration protocols have been developed. One basic requirement for all network routing protocols (based on TCP/IP) is the existence of IP addresses to specify the source and the destination of a transmission. In wired networks, the distribution of IP addresses can be automated via DHCP, BOOTP or other services. Unfortunately, this technique cannot be easily used in the wireless case without having any central instance or server that supervises the distribution of IP addresses.

The need that every node must be configured with a unique network layer address to permit unicast traffic led to the development of many different address autoconfiguration protocols that use different techniques. The main problem of all these address determination and distribution protocols is to make sure that each IP address is used at most once in an ad hoc network. Beside this requirement, the protocols have to deal with other problems like a limited amount of bandwidth or energy, an increased danger of packet losses and transmission errors or different channel properties and channel behavior.

Most of the research during the last four years concentrated on the optimization of existing protocol overhead and on efficiency optimization. Nevertheless, existing address distribution protocols in wireless networks also have weaknesses dependent on the actual protocol that is being used. One of these shortcomings is the lack of security concerns. In many protocols no method to assure a secure operation of the autoconfiguration protocol exists.

Other shortcomings exist in the existing Duplicate Address Detection methods. The reason for these weaknesses can be found in the wish of a secure connection over an insecure channel along with many other additional problems that a communication over a wireless network always has.

It seems to be obvious that a secure behavior must be guaranteed within limited boundaries of Dynamic Address Autoconfiguration in wireless ad hoc networks if it should become a standard for future developments.
Especially in the business sector, a company could lose lots of money within a very short timespan if an intruder which is not even a member of the company network could have the possibility to paralyze the whole network with a simple Denial of Service attack.

In this paper, we therefore want to point out possible improvements which can add a robust behavior to existing address configuration solutions. In part 2, we want to give a brief overview of different approaches that currently exist. After this, we will concentrate our work on the Optimized Dynamic Address Configuration Protocol, proposed in [1] and want to present possible security improvements in part 3 of the paper. Part 4 contains a detailed analysis of the impact of our proposals concerning extraordinary network behaviour and different mobility scenarios. Finally, in part 5 we will discuss the results and the experiences we made and want to point out possible future developments.

II. Related work

A variety of protocol implementations already exists to deal with the problem of automatic address configuration in mobile ad hoc networks. The approaches have many things in common but also differ in some important aspects. Generally, they can be divided into three groups: Decentralized approaches (MANETconf, Buddy, AAA), Best effort methods (e.g. Weak DAD, Prophet), Passive methods, or Leader Based (DACP, DAAP, …) approaches. Decentralized approaches were the first means to implement dynamic address assignment strategies in wireless ad hoc networks. The original idea was that every node in the network is equal to any other node which means that no single, dedicated server instance should be used to supervise the address assignment procedure. A node which joins the network chooses a randomly created temporary IP address and a candidate IP address it wants to obtain (out of the 169.254/16 address space according to the AutoNet specification in [7], where 0-2047 for the host ID represents a temporary address, and 2048 - (2^{16}-1) a candidate address postfix) The new node sends an Address Request (AREQ) message to all the other nodes to inform them that it wants to use the new candidate IP address. If this IP address is already in use, the specific node will send an Address Reply (AREP) message to the new node (to its temporary IP address) that the IP is already in use and the new node has to use another one. If the new node does not receive an AREP message after a certain amount of time, it will change its temporary IP address to the candidate IP address and starts to work in the network. (cmp. [3])

This is also one of the main weaknesses of the decentralized approach because a malicious node could simply pretend to own every IP address when it receives an AREQ message from another node. Another problem decentralized approaches have to face is the complexity of a network because network partitions and merges are (in the original design) not taken into consideration.

Weak DAD (a Best effort approach) wants to address this problem by preventing that a packet is being routed to a wrong destination – even if duplicate addresses exist (e.g. after a network merge). This means, the original objective of strong Duplicate Address Detection is given up in this approach which is again an opportunity for an intruder to perform some harmful activities.

Passive DAD (e.g. PACMAN [8]) tries to minimize extra protocol overhead caused by the address configuration protocol by monitoring routing protocol control traffic. The performance might be - with respect to its passive nature - slower in comparison to other approaches and it is heavily dependent from the underlying routing protocol.

In [2], a new leader-based approach is proposed. Beside other approaches like DAAP (Dynamic Address Allocation Protocol [9]), the so called Dynamic Address Configuration Protocol introduces again some kind of central instance administering the IP addresses which are used in the mobile network. As a difference to the commonly used DHCP server design in wired network environments, in wireless networks every node could become the so-called Primary Address Authority (PAA). This approach has the advantage that network merges can be handled more easily and also a Duplicate Address Detection can be performed directly at the PAA. As a consequence, this approach is suited best for implementing additional security checks for malicious node activities because a centralized database structure can be used in comparison to – for instance - the decentralized approach.

III. Design and Implementation

In this section we will discuss shortcomings of leader-based approaches and weaknesses in the protocol which allow malicious nodes to influence the whole network availability and behavior.

The research is based on the Optimized Address Configuration Protocol, first presented in [1], because this approach simplifies the problem of the decentralized approaches that a new node must validate its new network address against the IP addresses of every other node, which means reduced overhead (less broadcasts), more reliability (in the decentralized approach it could happen that a single node is temporarily not connected to the network and cannot send an AREP message), and finally a reduction of the risk that a single node pretends to have every requested IP address and covers the whole network address space.

First of all, we will start with a discussion, if the example that a single node requests multiple IP addresses or pretends to have multiple addresses is also possible in a leader-based approach like ODACP.

The second problem we want to face is the possibility, that an arbitrary node can receive all the messages sent to a destination, even if it is not the node on its own.
This opens the possibility that a malicious node could modify a reply that is sent by the PAA to a new node to confirm a registration request. This is a general problem nearly every protocol has to deal with, because normally every packet is transmitted without any encryption or other security mechanism over an insecure channel.

Our third scenario is specific for ODACP. The Primary Address Authority sends so called PAA advertisement messages in a fixed time interval to every other node in the network. If a new node joins the ad hoc network, it picks up the PAA advertisement message. The packet contains the IP address of the PAA and the new node is able to register a new IP address at the PAA because the packet contains information about the source address of the PAA which the new node uses as the destination address for its address request. ODACP allows network merges and partitions. If two independent networks merge, according to the protocol specification the PAA with the highest MAC address will automatically become the new PAA for the whole network after the two Address Authorities exchange the IP address tables, check them for duplicate addresses, and if this case occurs, force the specific nodes to reregister a new network address. If a client receives a different – or more than one – PAA advertisement message, it will automatically update its server information without validating the source of the new PAA advertisement. This lack of security offers a big point of attack for every intruder because a malicious host could simply send its own, faked PAA advertisement message with a bigger network identifier and every new node in the network will lose the contact to the actual PAA. It is even worse because the malicious host could send a reregister command to every node in order to paralyze the whole network (because in fact there will not exist any PAA thread running on the malicious host or at least no appropriate registration response).

1. **Registration of multiple IP addresses at PAA**

In the original design of ODACP, the PAA confirms every Address Registration Request, if the network address is not already used by another (registered) node. It therefore depends on the concrete implementation of the IP address table of the PAA if a node can hold multiple IP addresses or only a single address at a single time. Whereas the second case is not so serious because the malicious node could only react immediately to another node request (that is the advantage of the centralized approach, because there is no “objection” functionality for other nodes as it can be found in the decentralized approach), the first possibility could become difficult when a malicious node registers several IP addresses with a very high lifetime.

To solve this problem, we introduced a request counter as an additional data field in the IP address table of the PAA together with a timestamp field which indicates the last request time. This enables the PAA to evaluate the request behavior of a specific node. Two matters have to be taken into consideration:

On one hand, it must be still possible for a node to change its current IP address for several reasons; on the other hand, it could happen that a node is temporarily disconnected from the network and re-requests its old network address after a certain time. A modification of the protocol has to make sure that this node is in neither of the two cases identified as a malicious node. Test runs showed that a counter value of 6 is probably a good threshold to accomplish this goal. Additionally, an alter function is used to decrease the counter values again after a certain time in the same way as it is for example done in computer memory management. A node, whose IP request counter exceeds the threshold, is being blacklisted by the PAA and identified as a malicious node. We leave it open to detailed implementations in which way a blacklisted node could lose its blacklist status again or if its requests will be ignored to infinity.

2. **Faked PAA registration reply packets**

Due to the fact that in wireless ad hoc networks every node can receive and read the information of a packet that is sent between two nodes (promiscuous mode), it would be also possible for every node to modify the packet content in such a way that the destination host is not able to distinguish the injected data from the original data. A simple checksum field in this case is not sufficient and the identity of the source and destination node must be validated somehow.

This is especially important for ODACP because the PAA has got a particular status that is important for the entire network functionality. A malicious node could send arbitrary messages to other nodes which seem to have its origin at the PAA or it could send modified PAA responses again to the destination node and force it to do a specific action. The impact of the latter case could be limited with a proper protocol implementation, for example that a new node does ignore a registration denial (that forces it to register a new IP address), when it has already received a confirmation for a previously sent registration request.

The general case cannot be solved in a trivial way. A simple identification string sent with the request / reply messages respectively is obviously not sufficient because another node could simply copy this identity string. On the other hand, an entire encryption of a package content would be time consuming and does not necessarily guarantee that the origin of a message is really the assumed source node, i.e. the PAA.

Thus, we decided to apply a well-known, but in this research sector totally new idea for this test scenario. To make sure that no data field in a registration packet has been changed during its way to the destination host, a checksum field is added to the address registration reply message (AREQ) from the PAA. This checksum field contains a numeric value in which the values of the other message fields (operation code, requested candidate ip address, destination mac address, lifetime) are encoded.
For the encoding, a session key is used which can be generated by the PAA as well as by the destination node, but not by any other host in the network. To minimize calculation overhead, we decided to use a procedure which is known as the Diffie-Hellmann key exchange method. It is based on the idea that every communication partner creates a random private key ahead of the communication. The private key has to be in the range \{1, \ldots, p-2\} where p is a prime number. We will refer to the private key of the PAA with \(a\) and to the private key of the new node which wants to request a new network address with \(b\). Additionally, a number \(g\) is needed which satisfies the condition \(2 \leq g \leq p-2\). \(g\) has to be a so called primitive root of \(p\), for an exact definition we refer to [4].

For example, the following numbers could be used:

<table>
<thead>
<tr>
<th>(p)</th>
<th>(\text{Primitive root } g)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>2, 3</td>
</tr>
<tr>
<td>7</td>
<td>3, 5</td>
</tr>
<tr>
<td>11</td>
<td>2, 6, 7, 8</td>
</tr>
<tr>
<td>13</td>
<td>2, 6, 7, 11</td>
</tr>
<tr>
<td>17</td>
<td>3, 5, 6, 7, 10, 11, 12, 14</td>
</tr>
<tr>
<td>19</td>
<td>2, 3, 10, 13, 14, 15</td>
</tr>
<tr>
<td>23</td>
<td>5, 7, 10, 11, 14, 15, 17, 19, 20, 21</td>
</tr>
<tr>
<td>29</td>
<td>2, 3, 8, 10, 11, 14, 15, 18, 19, 21, 26, 27</td>
</tr>
</tbody>
</table>

Table 1: Prime numbers and their corresponding primitive roots

\(p\) and \(g\) can be fixed and known by every node. For example, in the simulations we ran, we used \(p=7\) and \(g=3\). The reason for this can be seen in an attempt to minimize the calculation time to create the session key which is described further below.

The two randomly created private keys must be kept confidential. Every node then creates a public key out of the private key number, the prime number \(p\) and the well-known primitive square \(g\) in the following manner:

\[
A = g^a \mod p \\
B = g^b \mod p
\]

where \(A\) represents the public key of the PAA and \(B\) the public key of the client node.

The PAA now continuously sends its private key as an additional field in the PAA advertisement message to the other nodes in the network.

If a single node wants to request a new network address, it sends its public key embedded in the registration request message (AREQ) to the PAA, so that each of the hosts has each of the two public keys in the end.

The session key to encrypt the checksum of the response can then be calculated with the following equation:

\[
PAA: K = B^a \mod p \\
Client: K = A^b \mod p
\]

To create the checksum, every appropriate formula could be used dependent from the actual implementation; we simply used

\[
\text{checksum} = (\text{operation} + \text{candidateip} + \text{macaddress} + \text{lifetime}) \mod K
\]

because this calculation has to be performed every time and should not be too time-consuming.

Due to the fact that a malicious node cannot know the private key that was chosen by the PAA, it nevertheless could modify packet information, but it cannot easily calculate the correct checksum for the modified data by using the appropriate session key. The node which receives the packet and requested the IP address on the other hand creates the same checksum out of the packet data like the PAA did. If there is a mismatch between the checksum that was transmitted within the reply and the checksum calculated by the node on itself, the client node will drop the packet as a faked message.

It has to be pointed out that this method works fine with a very high probability. Nevertheless, there is the possibility that a malicious node incidentally uses the right private key for creating a new checksum for a modified (faked) packet. In this case, the destination node would not be able to recognize the faked message.

Although computation time increases, we therefore recommend that a client node uses alternating private (and as a consequence public) keys with every new registration request, so the probability could be reduced to a minimum that the client node is not able to recognize a modified registration reply packet.

3. Faked PAA registration reply packets

The third weak point analyzed, also includes the problem of sending faked messages to other network devices. This time, not the registration messages are faked but the PAA advertisement message itself.
The original Optimized Dynamic Address Configuration Protocol does not distinguish between the reasons why multiple Address Authorities could be found in the same network at a particular time. The normal case is that after a network merge two different PAA advertisement messages with two different network identifiers are sent periodically. The two Address Authorities will exchange their node address tables and the PAA with the biggest value for the network identifier (i.e. its MAC address) will become the new PAA for the whole network whereas the other Address Authority gives up its status. All other nodes in the network will simultaneously update their PAA contact data and communicate with the new PAA in the nearby future.

This could be catastrophic if a malicious node uses this security leak by sending own PAA messages. It only has to make sure that it uses a bigger value for the PAA advertisement. If the new PAA is still alive in this network partition, the other PAA will be accepted after the timer expiration. When a node finally receives a new PAA advertisement which differs in its network identifier than the current PAA does (and this is easy because the malicious node knows the current network id). The result would be that the actual PAA will give up its Address Authority status, the nodes in the network will update their saved PAA IP address and will send new requests to the malicious node which pretends to be a PAA. Unfortunately, the network behavior after this step cannot be predicted because now the malicious node could do everything it wants to do. Sending wrong information, forcing the other nodes to continuously register new IP addresses and finally deny all registration requests. The other nodes would never elect a new PAA as it is done in a network partitioning scenario because every node still gets a response from the pretended PAA in form of faked advertisement messages.

We therefore propose a different behavior when multiple PAA advertisement messages occur in one network. Unless a node does not receive more than one PAA advertisement message, everything seems to be correct. When a node finally receives a new PAA advertisement which differs in its network identifier from the ones previously received, the node will wait for another time span twaitforoldPAA for a response of the old, known PAA. If the real PAA also sends a PAA advertisement within this time (that means, the PAA is still alive in this network partition), the other PAA advertisement is simply ignored and dropped. Otherwise, the other node will be accepted after the timer expiration as the new PAA.

The question is now, what happens, if there are really two PAAs within one network, for example after a network merge. In our proposal, the only two nodes which have to become active and have to communicate with each other, are the two PAAs themselves. The communication has to be done anyway to exchange the IP address tables, so a short validation process could also be performed to make sure that the partner on the other hand was really a Primary Address Authority in its old network partition (for example by sending a short check message to some of the nodes network addresses out of the other IP address table to make sure they really exist and are still alive, or by using some kind of trusted domain system).

The validation process seems to be necessary for a correct network behavior in future times but should not be too time-consuming. Therefore, it might be interesting for future research activities.

Finally, one of the two address servers could agree to give up its status or could send a broadcast message to every node in the network that the other PAA should be ignored because it might be a malicious device. If each of the two Address authorities sends an ignorePAA – broadcast, it is recommended to ignore both of the two nodes because at least one of them seems to lie and a totally new PAA is elected by the rest of the nodes in the network. If this action is not performed, an evil node could achieve that every competing PAA will be ignored easily by the client nodes.

IV. Testing and Characterization

In this section, we will present test results which we gained in several test runs. The tests were intended to evaluate the operational reliability of our solution. We will refer to the original protocol with ODACP and to the extended version with sODACP to indicate the improved security behavior against malicious node activities.

We used the QualNet Simulator (version 3.9) to simulate the behavior of sODACP under different test scenarios. Therefore, we developed four characteristic test scenarios which are the basis for the following test runs and analysis.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6 nodes in a single line, node 1 acts as PAA, nodes spaced 10m apart, no mobility</td>
</tr>
<tr>
<td>2</td>
<td>16 nodes in a grid, horizontal and vertical space 100m, one node as PAA, no mobility</td>
</tr>
<tr>
<td>3</td>
<td>A random network of 20 nodes in a 100m x 100m network area, one PAA in the center</td>
</tr>
<tr>
<td>4</td>
<td>A random network of 20 nodes in a 100m x 100m network area including mobility, min speed 0 m/s, max speed 10 m/s, one PAA in the center</td>
</tr>
</tbody>
</table>

Table 2: Test scenarios

The protocol behavior was simulated five times in every test scenario (seed values 1-5), and an average of the values measured is used for the following discussion. We tested the original protocol (ODACP) as well as sODACP with the network settings mentioned above, and refer to the results with no_mod and all_mod, respectively to indicate the usage of our modifications (which were discussed in part III of the paper).

On the other hand, it was necessary to implement conclusive tests with an appropriate malicious node activity simulation. We also tested each of the two protocols in every scenario with these modifications; that either no malicious node activity occurs in the network (no_mal) or a malicious node tries disrupt the normal network functionality (all_mal) in the following way:
one node in the ad hoc network tries to register multiple IP addresses at the PAA although it has already its IP address and wants to block IP address space in comparison to other nodes; number of IP addresses to create and register: 2 per second

one malicious node tries to send faked PAA registration reply messages to another node that seem to have its origin at the actual PAA

the malicious node sends a faked PAA advertisement to its neighborhood although a working PAA exists in the current network partition. There is no need to change the PAA and the malicious host does not have a valid client table nor the right to act as a new Address Authority.

Table 3: Malicious node activity overview

The simulation time for every simulation was set to 600s, in scenarios with node mobility, the random waypoint model was used. In the end, we obtained 200 different simulation files, so that we want to focus the analysis primarily on a general comparison between ODACP (no_mod) and sODACP (all_mod) on one hand, and the behavior of this address autoconfiguration protocol in the no malicious host (no_mal) case and the worst malicious node case (all_mal, including attacks I, II and III) on the other hand.

The first chart shows a general traffic analysis. The first two bars on each side indicate the amount of bytes received and sent by the client when no malicious node activity occurs in the network (no_mal), the other two bars on each side the case with all simulated malicious node activities (all_mal). The results shown in the chart correspond to test scenario 1 (6 nodes in one line). The y-axis is logarithmically scaled.

It is possible to see in chart 1, that the additional overhead which is necessary for our protocol extensions can be disregarded (20 versus 24 bytes for a register reply message, 1442 versus 1920 for all bytes received by the client; especially PAA advertisements) in the no_mal case whereas the impact of our modifications is tremendously reducing the network traffic in the case of malicious node activities (875861 bytes received in the no_mod (ODACP) case versus 7087 bytes received in the all_mod (sODACP) case.) The packet overhead has its root in the additional packet fields to exchange the public keys to provide checksum functionality whereas the tremendous traffic reduction is a result of the ignorance of the major amount of packets sent by the malicious node.

Chart 2 shows this result in a more sophisticated way with an analysis of the registration reply type sent by the PAA:

It first, it can also be seen that the load on the PAA can be reduced drastically. In the all_mal case, all of the registration requests from the malicious node in attack case I have to be confirmed whereas the malicious node is simply blacklisted in the all_mal case (on the right of the chart) and all other incoming requests from this MAC address are ignored for future times. Due to the fact, that this chart also represents test scenario 1 (6 nodes in a single line) it is unlikely that one of the six nodes chooses a network address which is already in use. This is an explanation why the value of the “denials sent” bar in chart 2 equals 0.

Regarding the influence of mobility with respect to the time a new node needs to obtain a valid IP address that is not already in use in the ad hoc network, chart 3 gives some further information.

The first interesting thing in chart 3 is that the time necessary to compute the checksum (and therefore the session key transmitted with a register reply message) causes nearly no recognizable timing overhead although power operations have to be calculated. This can be seen because the graph for no_mod, no_mal and all_mod, no_mal nearly overlap. Furthermore, chart 3 provides evidence that it is possible for a new node to obtain a new IP address more quickly in most of the cases as in the standard ODACP. Additionally, we want to point out, that it is dependent from the actual implementation how a node reacts when it receives a faked registration denied message for a non-confirmed registration request.
Normally it would create a new random IP address and re-request it at the PAA. We implemented this behavior of the original ODACP in such a manner, that if the node receives a registration confirmation message after it already requested a new IP address, the new node will use the IP address contained in the reply message and skip the waiting for the new registration request (that means, the new node ignores all new incoming registration responses). Otherwise, the graph for the no_mod, all_mal case would even be worse and the QualNet simulation for this case already needed one minute for 5 seconds to simulate which might be an indicator that the entire network functionality could be destroyed with such an attack and an insufficient implementation with no security extensions.

Chart 4: Total network traffic

Chart 4 sums up the results that we already discussed in the analysis of the previous charts. Although sODACP requires the extension of ODACP messages by one field, it can reduce malicious network traffic tremendously in order to provide network functionality even in extraordinary situations that were not considered during the original protocol design process.

The last chart shows the behavior of sODACP if more than one malicious node is present in one network. Although the probability for this scenario is not very high, chart 5 demonstrates that even in this case the entire network functionality can be guaranteed and – especially in malicious activity test case 3 – it is not possible for any of the malicious nodes to become a new PAA in this network without any legitimation.

To understand chart 5, additional data must be provided. The actual PAA sends PAA advertisement messages every 5s, a malicious node sends a faked PAA adv message every 2 seconds (simulation time: 600s). Due to the fact that actually no PAA service is running on a malicious node, the number of adv_sent in the chart represents the number of real advertisement messages in the network. Although the number is expected to be constant (120 adv messages), it jitters a little bit in the presence of malicious nodes because the PAA might be not able to send the last messages due to a time delay and busy work.

It can be seen in chart 5 that only the real advertisement messages from the PAA are accepted (horizontal line in the adv_recvd bar) Additionally, the total number of advertisement messages received the network does not increase linearly with the number of malicious nodes because of message interferences during transmission so that the advertisement messages do not always reach their destination.

Due to limited space, it is not possible to analyze every protocol modification for each of the malicious node activities in detail. Additionally, we want to point out that we concentrated on security aspects of address assignment protocols, thus we left out some of the protocol details in our implementation like the existence of a BAA (backup address authority) or the behavior in case of partitions or network merges. As a consequence, we cannot deliver exact test results for the third modification and the case, what happens after a real network merge when two PAAs from different network partitions are in one network now and have to negotiate a future PAA. The detailed explanation of the network behavior in part III.3 of the paper and the good test results which were shown in the charts of this section lead us to the conclusion that also the network behavior after a network merge is determined and will stay reliable with the security modification against faked PAA messages.

V. Discussion section

In this paper, the network behavior of Dynamic Address Autoconfiguration protocols in wireless ad hoc networks in exceptional situations that could occur in the boundaries of the network protocol was presented. We focused primarily on malicious node activities like Denial of Service attacks that interfere with – or even paralyzed – the whole network availability. Analysis was based on leader-based autoconfiguration protocols, i.e. the Optimized Address Configuration Protocol. Several solutions were presented which extend the original protocol design in such a way that common network attacks could be confined or even avoided. In different test scenarios, functionality of the improvements introduced before was shown and analyzed, and the relevance of the attack scenarios was emphasized.
1. Future work

Security issues will always be a problem for the communication between multiple network devices. Many solutions and proposals have been developed throughout the last years which address different security problems, but a total robustness of networks against security attacks will not be feasible unless research focuses mainly on effect and not on cause. Especially in a wireless ad hoc network, security issues are heavily influenced by other problems like network availability and connectivity, power consumption and delivery guarantees.

The presented security improvements in this paper are based on fixed MAC addresses this means no node is able to change its own MAC address individually or even the MAC address data in transmitted packets. Dependent from the importance of this scenario, other solutions have been developed to make sure that no MAC address has been changed during packet transmission from the source to the destination node [10] which could be used in scenarios where this danger is relevant for consideration.

In this paper, another major risk scenario was left out: The question is what happens if a malicious node really becomes a PAA – for example directly after network initialisation, when the malicious host seems to be the first node in the new network, or for instance after a network merge which was discussed in section III.3. New nodes will send registration requests messages to the assumed PAA but would never get a confirmation that the request was accepted successfully. On the other hand, PAA advertisements are sent continuously, so normally there is no reason why a new Address Authority should be elected. It is imaginable that client nodes could count the number of denies which they already received from the PAA and exchange them among each other. If multiple nodes in the network were not able to get a register confirmation after several attempts, clients could blacklist the PAA on their own and handle its status as “PAA unreachable” which means that they would elect a totally new Address Authority among themselves. This is only a hypothetical proposal and would be obviously interesting for future research activities.

2. Lessons learned

This paper was developed as a final project in the CS284 Mobile Computing graduate course at University of California, Santa Barbara in spring 2006. It was the first time that we used a network simulation environment to create our own modification of an existing application protocol with a new basic idea and an entire own implementation. It was interesting to see how much research has to be done to implement a new idea until it is finally functioning as expected. Network protocols turned out to be very effective and straightforward on one hand, but on the other hand very complex, so it was necessary at the beginning to balance the improvement proposals by their main concepts; which of them are really useful, important and practicable and which of them are only hypothetical without any appliance in real world scenarios.

It was also interesting to read the papers of other researchers who currently work on the same problem and to develop an opinion concerning the research topic. There are already so many developments and proposals with respect to new internet protocols that it seems hard to be up-to-date and to know everything about new developments. Basic network protocols used today have been designed many years ago and unless there will be a real new invention concerning network communication, single research projects will only contribute very small innovations to the systems that are used nowadays.

VI. References


