Universität Karlsruhe (TH)
Institut für Telematik

TELEMATICS TECHNICAL REPORTS

Passive Autoconfiguration of Mobile Ad hoc Networks

Kilian A. Weniger
weniger@tm.uka.de

April, 14th 2004

TM-2004-1

ISSN 1613-849X

http://doc.tm.uka.de/tr/

Institute of Telematics, University of Karlsruhe
Zirkel 2, D-76128 Karlsruhe, Germany

Passive Autoconfiguration of Mobile Ad hoc Networks

Kilian A. Weniger
University of Karlsruhe, Germany

***Abstract*— Mobile ad hoc networks (MANETs) enable communication between mobile nodes via multi-hop wireless routes without depending on an infrastructure. In contrast to infrastructure-based networks, MANETs support spontaneous networking and, thus, should be capable of self-organization and -configuration. The paper presents PACMAN, a novel approach for an efficient distributed address autoconfiguration of mobile ad hoc networks. PACMAN works almost entirely passive and thus has very low protocol overhead and high energy efficiency. This is accomplished by using cross-layer information obtained from ongoing routing-protocol traffic. A modular architecture alleviates the problem of routing protocol dependency and the application to three current routing protocols is discussed. Futhermore, PACMAN assigns IP addresses in a manner that enables the compression of IP addresses, which can reduce the routing-protocol overhead significantly.**

***Index Terms*— Mobile ad hoc networks, address autoconfiguration, PACMAN.**

I. Introduction

Most research efforts in the area of mobile ad hoc networks focus on efficient routing protocols. These protocols require the nodes to be configured with a unique address. In order to support truly spontaneous networking, the address assignment should be done automatically. Autoconfiguration protocols for traditional IP networks, however, cannot be applied to ad hoc networks for various reasons (see section III). Thus, new protocols have to be developed. Beside the dynamic multi-hop topology, MANET protocols have to deal with limited bandwidth, shared medium access on the wireless channel and limited battery power. Because control traffic, e.g., of the routing protocol, is usually transmitted in-band, the protocol overhead often limits the amount of data traffic the network can handle and the lifetime of the battery. Hence, low protocol overhead is one of the most important design goals for mobile ad hoc networking protocols. We propose a new modular architecture for the distributed address autoconfiguration in mobile ad hoc networks, which we call Passive Autoconfiguration for Mobile Ad hoc Networks (PACMAN). PACMAN consumes almost no additional bandwidth by using cross-layer information from ongoing routing-protocol traffic. A modular architecture decreases the dependency on the routing protocol.

The rest of the paper is organized as follows: the requirements on an address autoconfiguration protocol for MANETs are described in II. A brief overview of the current research efforts in the area of autoconfiguration of MANETs is given in section III. The system architecture of PACMAN is described in section IV. The main components of PACMAN are described in the following sections: an efficient addressing scheme which allows the compression of IP addresses (see section V). In section VI the address assignment function is presented, which is based on a probabilistic algorithm. Passive Duplicate Address Detection (PDAD) is introduced in section VII. In order to resolve conflicts detected by PDAD, a conflict-resolution protocol is used, which forces the involved nodes to change their address. This component is described in section IX. If a node changes its address, active transport layer connections may break. To prevent that from happening, an address change management is applied, that is described in section X. Section XI finally concludes the paper.

II. Requirements on Address Autoconfiguration of MANETs

The main task of an address autoconfiguration protocol is the management of the resource ”address space”. It must be able to select, allocate and assign a unique network address to a new node. When a node leaves the network the corresponding address must be deallocated eventually to prevent exhaustion of the address space. The configuration process should be fast, because a node cannot participate in unicast communication before it owns a unique address.

A major challenge is network partitioning and merging. Because communication across network partitions is not possible, an autoconfiguration protocol can at most guarantee unique addresses within a single network partition. If two partitions merge, address conflicts may occur. Because routing can be influenced by address conflicts, they must be resolved as quick as possible. The resolution of a conflict requires at least one node to aquire a new address. Because every address change may break transport layer connections, unnecessary address changes should be avoided. However, mechanisms known from mobility management like, e.g., tunneling can be used to hide address changes from higher layers. Autoconfiguration protocols must also consider the special properties of MANETs like, e.g., the dynamic multi-hop topology, the wireless channel and the scarcity of resources such as bandwidth and energy. Thus, control packet loss may occur frequently, but may not result in unresolved address conflicts and protocol overhead should be kept at a minimum.

One may argue, that providing unique addresses is very easy, if the address space is big enough to embed a globally unique hardware ID (as in case of IPv6). However, there is no hardware ID available which is truly globally unique. E.g., the 48-bit IEEE MAC address is a well-known hardware ID, which is used in IPv6 stateless autoconfiguration.

Such IPv6 addresses may be duplicate for various reasons: some manufacturers sell network adapters with non-registered MAC addresses or MAC addresses may get corrupted during the manufacturing process [4][5]. Furthermore, most network adapters allow users to change the MAC address to arbitrary values. One can also imagine network adapters that do not have a IEEE MAC address at all, e.g. in sensor networks. Another drawback of hardware ID-based IP addresses is the lack of location privacy. For this reason RFC 3041 [6] proposes to use a random interface identifier instead of the IEEE MAC address.

An important parameter for the autoconfiguration is the address space. Resource management can always be replaced by over-provisioning: it is simple to select a unique address from an infinite address space. But with increasing address space, the address length in bits as well as the overhead of, e.g., routing protocols increases, too. Thus, a small address space is more efficient in terms of energy and bandwidth, which represents another advantage of dynamic address assignment.

III. Related Work

Autoconfiguration protocols for traditional IP networks can be classified in

- stateless and
- stateful

approaches. In case of the stateful approach a central entity assigns addresses to new nodes. Because this entity keeps state information about addresses that have already been assigned, address conflicts do not occur. An example for such an approach is the Dynamic Host Configuration Protocol (DHCP) [1]. DHCP cannot be applied to MANETs without changes due to the requirement of a central entity, which may not always be reachable in a highly dynamic topology.

In contrast, stateless approaches are decentralized: a new node selects an address on its own and checks its uniqueness with a Duplicate Address Detection (DAD) mechanism. The node may, for instance, send a query to the chosen address. If it does not receive a response, it can assume that the address is not yet assigned to another node. In this case, the node can configure its network interface with this address and defends the address against other nodes which also want to use this address. As the nodes do not have state information about which addresses are already assigned, they can only select the address randomly or based on a globally unique hardware ID. Due to the limited address space of IPv4, the latter is usually only possible in case of IPv6. Examples of stateless approaches are the IETF Zeroconf protocol [2] and the IPv6 stateless address autoconfiguration protocol [3]. These protocols are designed for LANs and require that all nodes are reachable via a single-hop broadcast messages to query the chosen address. Because MANETs are multi-hop networks, not all nodes can be reached via a broadcast message.

Recently, some autoconfiguration protocols for MANETs were proposed. The first approaches were adaptations of autoconfiguration protocols for traditional IP networks. E.g. C. Perkins et. al. [7] propose an adaptation of the stateless approach to MANETs. A node selects an address randomly and verifies its uniqueness with DAD by flooding the network with an Address Request (AREQ) message, which contains the chosen address. A node with the same address defends it by replying with an Address Reply (AREP) message, which is sent per unicast over the reverse path established by the AREQ message. If there is no other node in the network with this address, a timer at the originator node expires and the address is considered valid. As from now, we refer to this mechanism as the Query-based DAD. A serious drawback of this approach is that network partitioning and merging is not supported. In [8] a similar approach is discussed, which supports network partitioning and merging by periodically repeating the DAD. The addressing scheme is hierarchical to reduce the protocol overhead of the DAD.

Another DAD mechanism called Weak DAD (WDAD) is proposed in [9]. Other than Query-based DAD, WDAD is integrated with the routing protocol and can continuously detect duplicate addresses with information added to routing-protocol packets. Thus, the routing-protocol packet format has to be modified. The main idea is to add a key to each address that is distributed by the routing protocol. The key can be of arbitrary length and is chosen once by each node either randomly or based on a Universal Unique ID (UUID). A node detects a conflict if it receives two address-key-pairs with the same address, but different keys. Hence, a conflict cannot be detected, if two nodes have chosen the same address and the same key. In case of random keys, the probability that a conflict cannot be detected decreases with increasing key length. However, with increasing key length the protocol overhead of the routing protocol increases, too. Thus, there is a trade-off between protocol overhead and the reliability of WDAD.

Autoconfiguration Protocols for MANETs following a stateful approach can be classified according to the way they maintain the address allocation table. Three major approaches can be distinguished:

- centralized maintenance of the allocation table,
- distributed maintenance of a common allocation table and
- distributed maintenance of multiple, disjoint allocation tables.

In case of a centralized allocation table, only one node in the network is allowed to select addresses for new nodes. Hence, this node must be permanently reachable. To achieve this goal the node representing the central entity is usually elected dynamically and may change. Because the transfer of the allocation table from the old to the new entity is not always possible, the allocation table of the new entity may be empty and all nodes have be notified reliably to either re-register or aquire a new address. The latter results in a number of unnecessary address changes. A backup entity might alleviate this problem. Anyhow, the central entity may represent a bottleneck in large networks. In contrast, a distributed allocation table allows every node to select addresses for new nodes, but synchronization between all nodes is necessary to avoid the assignment of duplicate addresses. In case of a common allocation table consistency, in case of multiple disjoint tables

the disjointness of the tables must be guaranteed. Efficient and reliable synchronization is the major challenge with this approach and may consume a considerable amount of bandwidth since this may require reliable or periodic flooding. In any case, inconsistencies may result in unresolved address conflicts. Network merging must be handled specifically to synchronize the allocation tables of both partitions after the merger and identify address conflicts.

MANETconf [10] and Boleng's protocol [11] are examples for protocols utilizing the stateless approach with distributed maintenance of a common allocation table, [14] is an example of a protocol utilizing the stateless approach with distributed maintenance of multiple, disjoint allocation tables. In [11] the authors propose a new network-layer addressing scheme with variable-length addresses. Such a scheme was only proposed for MAC layer addresses before [12][13]. An advantage of a variable address length is that a significant amount of overhead can be saved for protocols that send a lot of addressing information, e.g., if only 4 bits are used instead of 32 bits to address a network of 15 nodes. In [11], each node maintains two parameters: the highest address and the address length used in the network. A node can then assign the next higher address to a new node. Potentially, the address length must be increased to accommodate the new network size. Drawbacks of the proposed addressing scheme are that the parameters are maintained as global states. As already mentioned, a reliable global state synchronization can be very costly in MANETs. Furthermore, Internet applications and routing protocols use IP addresses and, thus, cannot be used with the new addressing architecture. The authors propose to use an Internet gateway to translate MANET addresses to IPv6 addresses for the communication with Internet nodes, but they do not propose a mechanism to enable IP-based communication within the MANET.

IV. System Architecture of PACMAN

PACMAN is a modular architecture for the distributed address autoconfiguration protocol of MANETs. It uses cross-layer information from ongoing routing protocol traffic to provide an efficient DAD and address assignment. PACMAN utilizes elements of both, the stateful and the stateless approach and can, thus, be classified as a hybrid approach. State information about assigned addresses in the network is collected in a lazy manner to save bandwidth. As stateless protocols, a node self-assigns an address.

The overall system architecture of PACMAN is shown in figure 1. Due to the modularity, the dependency on the routing protocol is decreased and the integration of new routing protocols is eased. A *routing protocol packet parser* extracts information from incoming routing protocol packets and gives them to the *PACMAN manager* in a generic format, which in turn delegates the information to the respective components. The protocol parser uses modules to support different routing protocol packet formats. The *address assignment* component calculates the address space to use and selects an address using a probabilistic algorithm. It also maintains the allocation table (see section VI). Potential address conflicts, e.g. occurring after two networks merge are detected permanently by the *Passive Duplicate Address Detection (PDAD)* component. PDAD does not send any control packets, instead it analyzes incoming routing protocol packets to derive hints about conflicts (see section VII). Again, modules are used to support different routing protocols. In case a node detects a conflict of another node's address, the *conflict-resolution* component notifies the respective node (see section IX), which then changes its address to resolve the conflict. An *address change management* component can additionally notify communication partners about that address change to prevent transport layer connections from being broken (see section X). The allocation table as well as the address space can be different at each node. Thus, no reliable state synchronization is required. To reduce the size of protocols packets with a high fraction of addressing information, like, e.g., routing protocol packets, IP addresses in such packets are compressed to variable-sized addresses by the *address encoding* component before they are given to the MAC layer. Encoded IP addresses in incoming routing protocol packets are decoded back to fixed-sized IP addresses. Hence, the compression is transparent to the network layer and above (see section V).

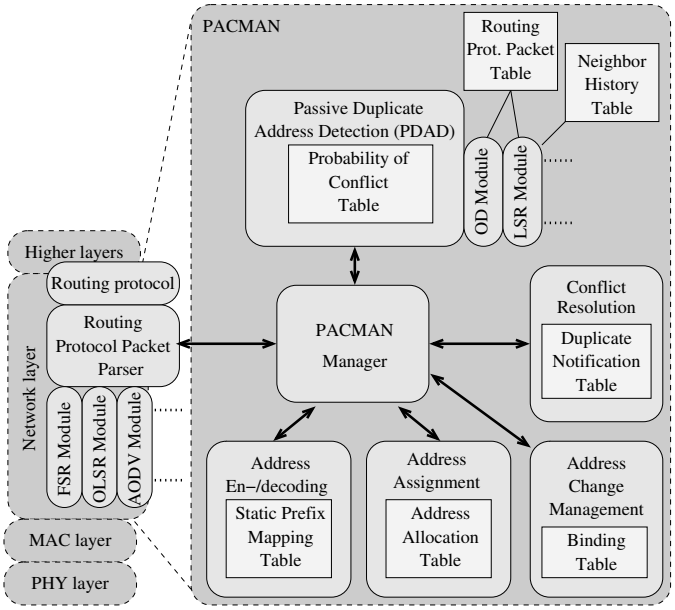

Fig. 1. System architecture of PACMAN

V. IP Address Encoding

The maximum number of nodes that can be uniquely identified in a network is determined by the address space from which the addresses are picked. The address space can also affect the size of the address representation and, therefore, also the size of protocol packets that contain addressing information. Subsequently, an heavily oversized address space means a wastage of bandwidth. For communication within a MANET, IP addresses with the MANET-local prefix (IPv4: 169.254/16, IPv6: fec0:0:0:ffff::/64) are usually used [7]. Since the MANET-local addresses are not globally routable, they only have to be unique within the connected portion of a

MANET and may be reused in other network partitions. For this purpose, the address space of IPv4 or even IPv6 is clearly oversized: In case of IPv4, 65536 nodes and, in case of IPv6, $\approx 1.8 \cdot 10^{19}$ nodes can be addressed. A variable or even adaptive address space (with respect to network size) would be eligible. As already mentioned in section III, the authors of [11] propose a new network-layer addressing architecture which allows a variable-sized address space. A drawback of this architecture is that it is not compatible to the IP addressing architecture. PACMAN assigns IP addresses in a manner that enables IP address compression. The length of the encoded address is variable and scales with the address space. Addresses in outgoing routing protocol packets are encoded below the network layer to lower the protocol overhead on the wireless channel. Addresses in incoming routing protocol packets are decoded back to common IP addresses. Thus, compatibility to the IP addressing architecture is preserved.

We introduce a variable address space, which we call the virtual address space. The virtual address space can be as small as 1 bit and has a maximum size of the fixed address space of the IP address minus the prefix length (16 bits in case of IPv4 and 64 bits in case of IPv6 with MANET-local prefixes). E.g., in a network of 60 nodes, a virtual address space of 6 bits would be enough to uniquely address all nodes. A possible address encoding scheme is the well-known Huffman coding, as done in [12]. Addresses are then replaced with codewords. The drawback of this approach is that the mapping of network addresses to codewords has global meaning, and some kind of global and reliable synchronization would be required. As already mentioned, such a synchronization may be costly. Our approach does not require this: Only a part of an IP address, the virtual address space, is used as the node identifier. All other bits are not used and set to zero. Subsequently, the address can easily be compressed, e.g., by the well-known run-length coding (RLC). The size of the encoded address decreases with a decreasing virtual address space, and the encoded address is self-contained: it can be decoded without any additional information, and the virtual address space is implicitly given by the run-length code. The size of the encoded address can further be reduced by replacing frequently occurring prefixes with short codewords. As already mentioned, a global mapping of addresses or prefixes to codewords is not desired. Thus, use a static mapping table and replace only well-known prefixes as the MANET-local prefix.

To clarify the encoding procedure, figures 2(a) and 2(b) show an example, where the 32-bit IPv4 address 169.254.0.13 is encoded to an 8-bit representation by using RLC and prefix replacement. The virtual address space is 4 bit in this example. In case of IPv4, the size of the encoded address is 8 bit, which is a reduction of 24 bits. An IPv6 address with the same virtual address space could be reduced to about the same size, which corresponds to a reduction of 120 bits.

Note that, in this example, the RLC is not bit-wise but block-wise. Therefore, the address space is divided in blocks (here: 4-bit blocks). Doing so can increase the compression rate, because the number of blocks of zeros is smaller than the number of zeros. On the other hand, some un-encoded bits may not be used as node identifier, if the virtual address space is not an integer multiple of the block size.

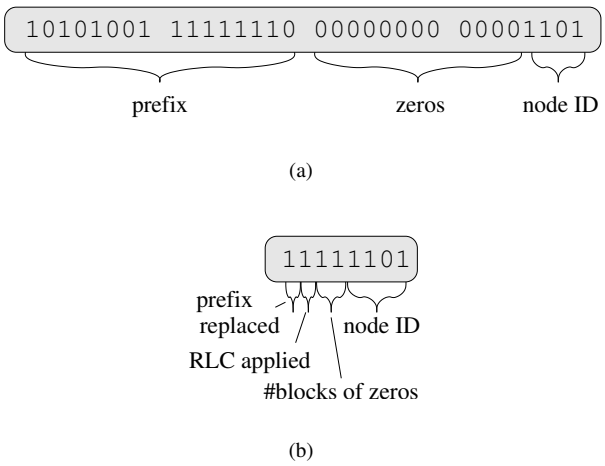

Fig. 2. Binary representation of decoded (a) and encoded (b) IPv4 address

VI. Address Assignment

Nodes running PACMAN self-assign addresses using a probabilistic algorithm. This algorithm has three parameters:

- a pre-defined probability of conflict, which depends on the type of network
- an estimation of the maximum number of nodes and
- the allocation table of the node.

Based on these parameters, the algorithm calculates the virtual address space, selects an address randomly from this space and ensures that it is not part of the allocation table.

But how likely is the occurrence of an address conflict in case of random assignment? The probability of a conflict only depends on the address space and the number of nodes in the network: the bigger the address space, the lower the probability of a conflict. This probability can be calculated using equation 1 with n being the number of nodes and r the address space (see appendix I for a derivation).

$$p_c(r,n) \approx 1 - e^{-n}\left(1 - \frac{n}{r}\right)^{n-r-\frac{1}{2}} \tag{1}$$

Figure 3 shows a plot of this equation in case of an address space of 16 bit, which is e.g. used in case of IPv4 MANET-local addresses. It can be seen that, in a network of 300 nodes, the probability of a conflict is as high as $\approx 50\%$. Figure 4 shows a 3d plot that in addition illustrates the dependence on the address space.

With respect to address encoding, the choice of the address space is a trade-off between compression rate and the probability that a conflict occurs. The bigger the address space, the lower the probability of a conflict and the lower the compression rate. Regarding the desired probability of conflict p_c as a pre-defined Quality-of-Service parameter of the network and given that the number of nodes in the network is known, the optimum virtual address space can be calculated at each node using equation 1. E.g. in a sensor network, nodes

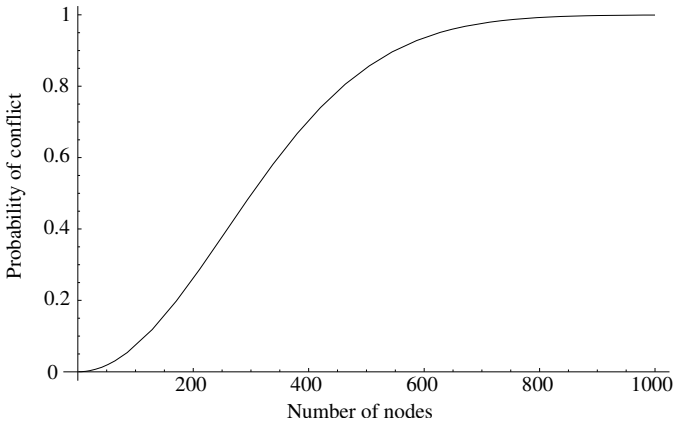

Fig. 3. Probability of conflict in case of random address assignment

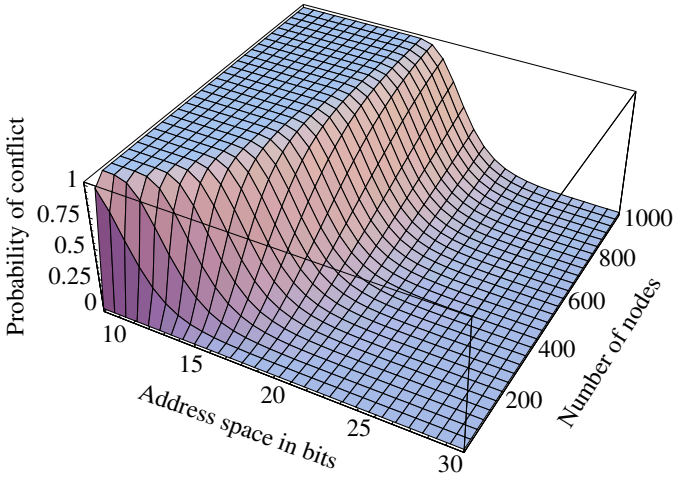

Fig. 4. Probability of a conflict in case of random address assignment

may tolerate a higher probability of conflict if energy can be saved due to a higher compression rate.

The probability of a conflict can further be reduced by incorporating the address-allocation table, which contains addresses already allocated in the network. The allocation table is again maintained using cross-layer information from ongoing routing protocol traffic. To speed up the configuration time a node can request the address-allocation table from already configured neighbors. If the selected address is not part of the allocation table, the node assigns the address to its network interface. However, there can be a considerable amount of unknown or hidden allocated addresses in the network, e.g. if nodes join the network, but the routing information about them is not yet distributed in the network. This is especially an issue for routing protocols that do not distribute routing information by flooding. A lot of hidden allocated addresses also occur after a lot of nodes simultaneously start the autoconfiguration, after a network merger occurred or in case of on demand routing protocols.

Considering the allocation table, equation 1 does not correctly express the probability of a conflict anymore. The number of known allocated addresses in the address allocation table just reduces the address space, from which the addresses are chosen randomly. Equation 2 gives an estimation of the probability of conflict with the allocation table and is derived in appendix II. j is the number of unknown allocated addresses, r the virtual address space and n the number of nodes in the network.

$$p_c(r,n,j) \approx 1-(r\cdot e)^{-j}\frac{(r-n+j)^{r-n+j+\frac{1}{2}}}{(r-n)^{r-n+\frac{1}{2}}} \quad (2)$$

For $j = n$ the allocation table is empty. In this case, equation 2 equals equation 1. For $j = 0$ all allocated addresses are known and the probability of a conflict is zero. If a maximum allowed probability of conflict p_c and an estimation of the maximum number of nodes in the network n can be given, each node can calculate the optimal virtual address space r using this formula.

VII. Passive Duplicate Address Detection

Duplicate addresses can occur, e.g., after the merging of two already configured ad hoc networks. To ensure proper unicast routing these conflicts must be detected and resolved in a timely manner. PACMAN uses Passive Duplicate Address Detection (PDAD) [15] to detect duplicate addresses. With PDAD, a node analyzes incoming routing protocol packets to derive hints about address conflicts. Thus, PDAD does not consume additional bandwidth. The basic idea is to apply various PDAD algorithms to incoming routing protocol packets, which are based on protocol events that either

1) never occur in case of a unique address, but always do occur in case of a duplicate address or
2) occur seldom in case of a unique address, but often in case of a duplicate address.

In the first case, a conflict is detected if the event occurs once, whereas in the second case a conflict is present only to a certain probability. In this case, a long-term monitoring may be necessary. Such PDAD algorithms are probabilistic. Each node maintains a table, which contains the probability of conflict for each address that is monitored (see figure 1). Each time a PDAD algorithm derives a hint about a conflict, it modifies the probability for the corresponding address in the table. If a certain threshold is reached, PDAD assumes that the address is indeed duplicate and triggers the conflict resolution. The incoming routing protocols give information about the routing protocol state of a specific sender. This state can then be compared to the state of the node to detect a conflict of the own address or to the state of another node obtained from a recently received routing protocol to detect a conflict of another node. Therefore, each node has to store information obtained from received routing protocol packets in a routing protocol packet information table (see figure 1). To limit the storage overhead, we only store the lastly received protocol packet for each originator address.

The receiver must have information about the time a received routing protocol packet was sent to reason about conflicts. Without synchronized clocks and additional information in the packets, this time can only be estimated: if the time a specific routing protocol packet is distributed in the network is bounded to t_d, a worst case estimation is possible. Usually

this assumption is given, because the protocols utilize a duplicate message cache and soft-state maintenance of routing information. Otherwise bandwidth would be wasted and the routing protocol would have difficulties to distinguish old from new routing information, e.g. after a sequence number wrap-around.

A. PDAD algorithms for proactive link-state routing protocols

In [15], we propose and discuss three algorithms to detect duplicate addresses for proactive link-state routing protocol. The algorithms were first developed for a classic (proactive) link-state routing protocol model and were then applied to existing link-state routing protocols for MANETs. With the classic link-state routing model we used, each node periodically issues link-state packets. These packets contain the originator address, a sequence number and the link states consisting of all neighbor addresses. The link-state packets distributed in the network by flooding. Each node forwards the packets on application-layer. A duplicate cache is used to prevent unnecessary forwarding. In the following the above mentioned and some additional algorithms are presented.

1) PDAD-SN: The Sequence Number (SN) algorithm exploits the sequence numbers in the routing protocol packets to detect duplicate addresses in the network. The assumptions for this algorithm are, that each node uses a sequence number only once within the time t_d and that each node increments its own internal sequence number counter periodically and only its own counter.

The algorithm is best explained by means of an example: Figure 5 shows a scenario, where node A and E have the same address 1. If node E now receives a routing protocol packet with an originator address that is equal to its own (1) and a slightly lower sequence number (1) than its internal sequence number (9), it cannot decide if this packet was recently sent by itself or by another node with the same address.

In contrast, if node A receives a packet with originator address 1 and a sequence number (9) that is higher than its internal sequence number (1), this packet can only be send by another node with the same address[1]. Subsequently, node A has detected a conflict of its own address.

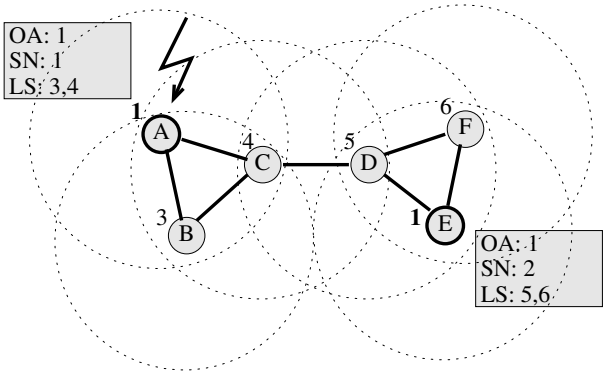

Fig. 5. Example of conflict detection using PDAD-SN

[1]The only exception is a sequence number wrap-around. See [15] for details.

2) PDAD-SND: In certain situations, it is also possible that an intermediate node detects the conflict using the difference of the sequence numbers (SND). In the scenario illustrated in figure 6, node A and E again have the same address, but this time both nodes have considerably different internal sequence numbers. "Considerably different" means, that the difference is higher than the maximum possible incrementation within t_d. In this case, an intermediate node receives two consecutive packets, one from node A and E, respectively, with the same originator address and a sequence number difference that is higher than the above mentioned maximum incrementation. Subsequently, these packets cannot be sent by the same node and intermediate nodes can conclude, that the originator address is duplicate.

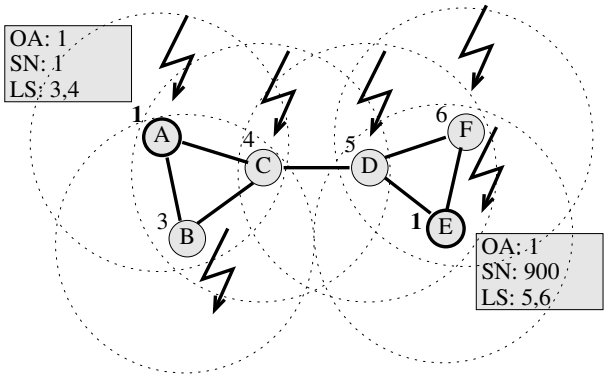

Fig. 6. Example of conflict detection at intermediate nodes using PDAD-SND

3) PDAD-SNE: In the scenario shown in figure 7, node A and E have the same sequence number. If an intermediate node receives packets from both nodes, the originator address as well as the sequence numbers are equal (SNE). Subsequently, the packets should be the same forwarded by different nodes in case of a unique originator address. But if the link states in the packets are different, the packets are obviously not the same. Intermediate nodes can then conclude that the originator address is duplicate in the network.

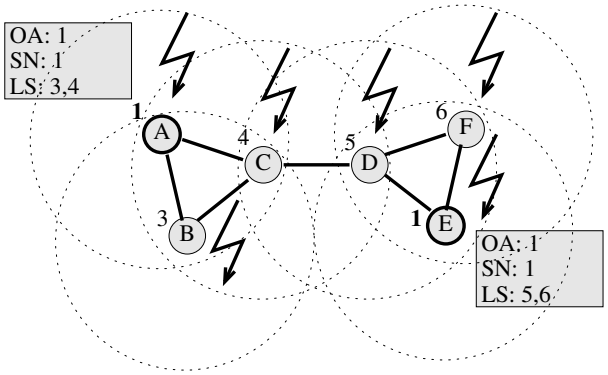

Fig. 7. Example of conflict detection at intermediate nodes using PDAD-SNE

With the model defined above, a combination of PDAD-SN, PDAD-SND and PDAD-SNE is already able to detect all conflicts.

4) PDAD-NH: The basic idea of the Neighborhood History (NH) algorithm is to exploit the bidirectional link states in the

packets. If a node receives a link-state packet with its own address as a link state of the originator, the originator must have been a neighbor of this node during the last period of time t_d, if the corresponding link state is bidirectional. Thus, a node has to save its recent neighborhood history. Figure 8 shows an example, where both, node A and E again have address 1. Node A receives a link-state packet from node F, which is a neighbor of node E. Subsequently, node F's link-state packet contains address 1. Because node A has recorded all recent neighbor addresses during the last t_d, it notices that the originator address (6) has not been a neighbor recently. Thus, it can conclude that its own address is duplicate.

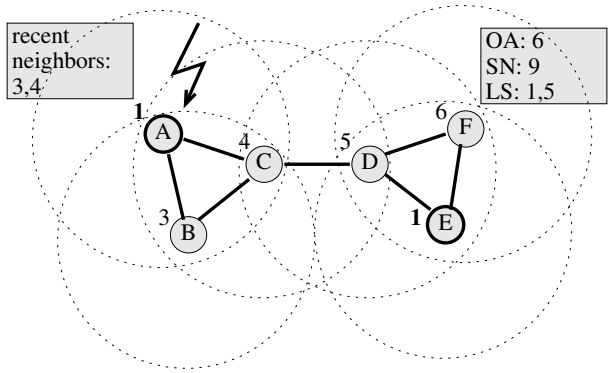

Fig. 8. Example of conflict detection using PDAD-SN

5) PDAD-LP: The basis for the Locality Principle (LP) algorithm is that the neighbor addresses or link states of a node usually do not change completely within one update interval of the routing protocol. This only holds to a certain ratio of the node's speed (relative to the neighboring nodes) and the update interval. Other than PDAD-SN or PDAD-NH, PDAD-LP is a probabilistic scheme. Hence, PDAD-LP can only give hints about the probability of a conflict. Of course, at some point of time a decision must be made that a conflict is present or not. If a node receives a packet, it compares the link states in the packet with the link states of the last packet received from this originator address.

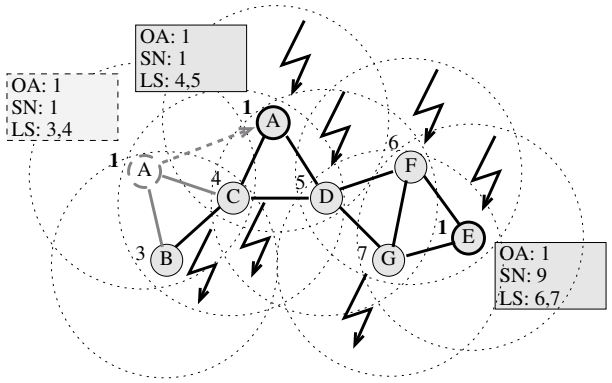

Fig. 9. Example of conflict detection at intermediate nodes using PDAD-LP

To aquire a normalized probability of a conflict p_i that is based on the difference of the link states ls_1 in packet 1 and ls_2 in packet 2 with the same originator address, equation 3 can be used. The lower the number of link states that occur in both packets divided by the number of overall link states, the higher the probability of a conflict.

$$p_i = 1 - \frac{|ls_1 \bigcap ls_2|}{|ls_1| + |ls_2|} \tag{3}$$

Due to potential packet reordering during the distribution, this probability can also be high if no conflict is present, especially in heavily loaded networks. To alleviate this problem, the value p_i can be smoothed, e.g., using equation 4. The conflict resolution is triggered if p_i exceeds a certain threshold p_t. Furthermore, only packets arriving within a certain inter-arrival time should be considered.

$$p_i = \alpha \cdot p_i + (1 - \alpha) \cdot p_{i-1} \tag{4}$$

6) PDAD-SA: The Source Address (SA) algorithm considers information in the IP header of the routing protocol packets. If the routing protocol forwards packets on application layer, the IP source address is always the address of the last forwarder. Thus, a conflict with a neighboring node can be detected, if the IP source address of a received routing protocol packet is equal to the node's own address.

7) PDAD-DC: The Duplicate Cache (DC) algorithm is also based on the IP header of the routing protocol packets. Because each routing protocol packet with the same originator address and the same sequence number is only forwarded once and the forwarding takes place on application layer, a conflict between two neighboring nodes can be detected, if such a packet is received more than once from the same IP source address.

B. PDAD algorithms for on-demand routing protocols

Passive Duplicate Address Detection can also be applied to on-demand routing protocols. Due to the passive nature, DAD is also on-demand and can only detect conflicts between nodes involved in a route discovery or maintenance procedure. In the following, we define a model of an on-demand routing protocol and present some PDAD algorithms for this model. The model can be regarded as a simplified version of AODV [19]: A node floods the network with a Route Request (RREQ) message to discover a route to a specific destination. This message is forwarded by intermediate nodes on application layer. Unlike nodes running AODV, only the destination may answer with a Route Reply (RREP) message, which is routed over the reverse path of the RREQ. The RREQ contains a RREQ ID to identify a specific message and apply a duplicate message cache. Thus, a single RREQ is only processed once by each node. Both messages contain sequence numbers to assure loop freedom. The route is maintained as long as it is needed by periodically sending RREQ messages.

PDAD-LP and PDAD-NH, of course, cannot be applied, because there is no link-state information in the packets. PDAD-SA and PDAD-DC can be applied. PDAD-SN can be applied to the RREQ ID and the sequence numbers in RREQ messages. However, there are other protocol events, that do occur in case of a duplicate address, but do not occur in case

of unique addresses and that are special to on-demand routing protocols:

1) PDAD-RNS: The RREQ-Never-Sent (RNS) algorithm detects a conflict, if a RREQ is received with the receiver's originator address, but a RREQ was never sent for this destination. This means that another node with the same address sent a RREQ.

2) PDAD-RwR: The RREP without RREQ (RwR) algorithm detects a conflict, if a RREP is received with an originator address equal to the own address, but a RREQ was never sent for this destination.

3) PDAD-2RoR: Due to the duplicate message cache, a singe destination node only replies once to a specific RREQ. The 2RREPs-on-RREQ (2RoR) algorithm detects a conflict, if more than one RREP is received from the destination address on a RREQ.

A combination of PDAD-RNS, PDAD-2RoR and PDAD-SN, PDAD-SND and PDAD-SNE is able to detect a conflict at the time the next RREQ containing this address is sent. If node A and node B both have the same address, there are two possible cases:

1) Neither node A nor node B issues a RREQ. Instead a third node C issues a RREQ with the address of node A and B as destination address. In this case, node C can detect the conflict using PDAD-2RoR.
2) Either node A or node B issue a RREQ. In this case either node A or node B can detect the conflict using PDAD-RNS, PDAD-SN, PDAD-SND or PDAD-SNE.

VIII. Applicability to current AD HOC routing protocols

The PDAD algorithms as presented above only apply to the defined models. In this section, the applicability to current routing protocols such as FSR, OLSR and AODV is discussed.

A. PDAD and FSR

The Fisheye State Routing (FSR) [16] protocol is a proactive link-state routing protocol with two special properties: the aggregation of link-state packets and the application of the fisheye technique. Other than most link-state routing protocols, FSR does not flood the link-state packets in the network. Instead, only one aggregated routing protocol packet per update interval is sent by each node, similar to the technique used in proactive distance-vector routing protocols like, e.g., DSDV [17]. In order to prevent unlimited distribution, topology information is hold in a soft-state manner and only distributed if it is fresh or if a neighbor has outdated information. The link state packets also serve the purpose of neighbor sensing. Thus, no hello messages are sent by FSR. The fisheye technique corresponds to the update frequency of routing information and was introduced to reduce the protocol overhead: Routing information of far-away node are sent less frequently than information of nearby nodes. This is based on the assumption that the angle of direction varies less for far away nodes than for nearby nodes. Because a route gets more accurate the closer the packet approaches the destination, the packet can still reach the destination.

Although the maximum distribution time of routing information t_d is bounded, it is much higher than in case of routing protocols that distribute routing information by flooding. Hence, PDAD can be applied, but requires more time to detect a conflict. If PDAD-LP is used in conjunction with FSR, false alarms may occur due to a too slow update frequency for far away nodes. Because the link states in FSR packets are uni-directional, PDAD-NH cannot be used. In contrast, PDAD-SN can be used and is able to detect all conflicts. Note that with FSR a node only forwards new routing information about an address, i.e. with a higher sequence number than previously forwarded routing information. Three cases can be distinguished:

- Two nodes having the same address have considerably different sequence numbers. In this case, the conflict can be detected by an intermediate node with PDAD-SND or by the node with the lower sequence number with PDAD-SN.
- The sequence numbers of two nodes having the same address are very close. The conflict can then not be detected by an intermediate, but by the node with the lower sequence number with PDAD-SN.
- Two nodes with the same address have the same sequence numbers. The conflict can be detected by an intermediate node with PDAD-SNE.

PDAD-SA can accelerate the detection in some cases.

B. PDAD and OLSR

The Optimized Link-State Routing protocol (OLSR) [18] is similar to the classic link-state routing model. The link states in OLSR packets are bidirectional and the packets are flooded in the network. Other than FSR, OLSR uses hello messages for neighbor sensing. A special feature of OLSR is the employment of Multi-point Relays (MPR): Every node selects certain neighbors as its MPR nodes. A node selects at least those one-hop neighbors, that allow the node to reach all two-hop neighbors over these MPR nodes. Only the nodes that are chosen as MPRs are allowed to forward a node's OLSR packets. A node that is no other node's MPR does not issue OLSR packets at all. In contrast, an MPR node issues OLSR packets, but only declares its MPR selectors. Using the MPR technique, a considerable amount of protocol overhead can be saved.

In case of OLSR, PDAD-LP may trigger false alarms, because the link states are only a subset of the neighborhood, which can change completely due to a change in the MPR selection. OLSR packets also contain bidirectional link states and, thus, PDAD-NH can be used. PDAD-SN can also be used. In fact, both schemes should be used in conjunction. As already mentioned, non-MPR nodes do not issue link-state packets, but PDAD-SN only detects conflicts of nodes that do issue link-state packets. PDAD-NH on the other hand requires a neighbor of a node to issue a link-state packet. To accelerate the conflict-detection time, PDAD-SN and PDAD-NH can also be applied to hello messages. However, the MPR selection is affected by duplicate addresses. Thus, it can happen that too few nodes are MPRs and that routing protocol messages

are not propagated in the entire network. Subsequently, the network gets partitioned in terms of routing. However, this only happens in special constellations. An example is shown in figure 10. Node A and E and node C and G have the same address. As a consequence, both, node C and node E only know of one 2-hop neighbor and, thus, select only one of their neighbors as their MPR. If node C selects node B and node E selects node F, the network is partitioned. Thus, only node D receives routing information from both partitions and could detect the conflict, but only if both conflict partners send routing protocol packets. However, node A and node G are not selected as MPRs and, thus, do not send routing protocol packets: the conflict cannot be resolved.

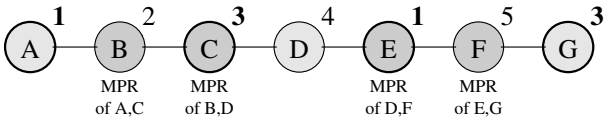

Fig. 10. Example of influence of address conflict on MPR selection (1)

A solution to this problem is, that node D executes PDAD-NH on behalf of node E or C: if node D receives a routing protocol packet from node B containing the address of a neighbor of node D (here: node E with address 1), node D checks whether the originator address (here: 2) is contained in the hello packets of node E. If this is not the case, it may conclude that there is a conflict of address 1. However, because node D does not know the neighborhood history of node E, it cannot be sure. Thus, it sends a special message to node E, which can then make decision based on the neighborhood history.

However, some scenarios exist where this algorithms fails. Figure 11 shows such a scenario. Because a node only considers real 2-hop neighbors for the MPR selection, i.e. nodes that are not 1-hop neighbors at the same time, node D and node B both have no 2-hop neighbors and, thus, select none of their neighbors as MPR. As a consequence the algorithm described above cannot detect the conflicts, because the neighbors' addresses of the conflicting nodes are the same. But the conflict can be detected using the link codes in OLSR hello messages if a node additionally stores the link codes in transmitted hello messages in the neighborhood history table. The TC message that node C receives from node B has originator address 2 and contains address 1 as a link state. Thus, node C can conclude that a node with address 1 must has selected a node with address 2 as its MPR. But the hello messages received from node D indicate that a node with address 1 has not selected a node with address 2 as MPR. Thus, node C can assume a conflict. Because the TC message can be delayed by t_d and node C is usually not aware of the complete neighborhood history of node D, it sends a special message to node D, which can then make the final decision based on its neighborhood history table. With this algorithm, from now on referred to as *PDAD-MPR*, all conflicts can be detected in a passive manner.

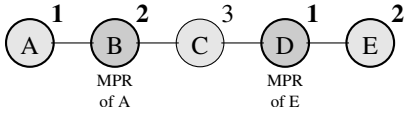

Fig. 11. Example of influence of address conflict on MPR selection (2)

C. PDAD and AODV

The Ad hoc On-demand Distance-Vector (AODV) routing protocol [19] has some differences to the model defined in section VII-B:

1) Intermediate nodes are allowed to reply to a RREQ if they have a fresh-enough route to the destination. Thus, PDAD-2RoR may only be applied if RREPs sent by intermediate nodes can be distinguished from RREPs sent by the destination. As this is not the case for AODV, a new flag must be introduced to allow the application of PDAD-2RoR.
2) The route is reestablished only if a link on the route breaks. In this case, an intermediate node either performs local repair or sends a Route Error (RERR) message back to the source, which then can issues a new RREQ. Thus, if a route is active for a long time and does not break, the sequence numbers at a node about a certain destination may differ considerably from the sequence number at the destination and PDAD-SN, PDAD-SND or PDAD-SNE may trigger false alarms. To prevent that, PDAD-SN, PDAD-SND and PDAD-SNE may only be applied to the RREQ ID or AODV must be modified to enable the synchronization of sequence numbers on long-term active routes from time to time.
3) Because RREPs are also used as hello messages, the PDAD-2RoR algorithm should only be applied to multi-hop RREPs or hello messages must be marked as such with a special flag.

A major problem for PDAD is the expanding-ring-search technique of AODV. Using this technique, a node first searches in the immediate vicinity for a node with a route to the destination address and incrementally expands the search area. It aborts the search procedure, if a node with the desired route information is found. If the destination address is duplicate and a node having a route to this address is in the initial ring-search area while a second node with routing information about the other node with the same address is not, the conflict cannot be resolved and it cannot be determined whether the route found leads to the intended destination node. Thus, PDAD is only able to detect all conflicts if the expanding-ring-search technique is not used (at least the first time a route is established).

IX. Conflict resolution

As already discussed in section VII, a node can either detect a conflict of its own address or of another node's address. In any case, at least one node involved in the conflict must be notified, so that it can change its address and the conflict can be resolved. This is done by sending so-called Address Conflict Notification (ACN) messages to one of the nodes with

the duplicate address. There are numerous possible strategies for the decision which node should change its address, e.g., the number of open transport layer connections, the time a node is part of the network etc. To prevent the distribution of additional information in the network PACMAN unicasts the message in the direction from which the corresponding routing protocol packet was received. This way, the node that has joined the network most recently tends to change its address more often than the other node and no additional information is needed.

In some scenarios a lot of intermediate nodes detect a conflict at almost the same time. In this case, a lot of nodes unicast notifications to one node: the node with the duplicate address. This many-to-one communication is known as concast. Concast is a problem especially in large ad hoc networks, because it can lead to increasing media-access delays in the area of the destination node. Figure 12a illustrates the problem. Each unicast packet with a corresponding medium access is represented as an arrow. To alleviate this problem, a technique similar to the duplicate-message cache for flooded RREQs in AODV [19] is applied: If a node has already forwarded a notification to an address during the last t seconds, it does not forward incoming notification messages to this address anymore. Using such a mechanism, a lot of media accesses can be saved (see figure 12b).

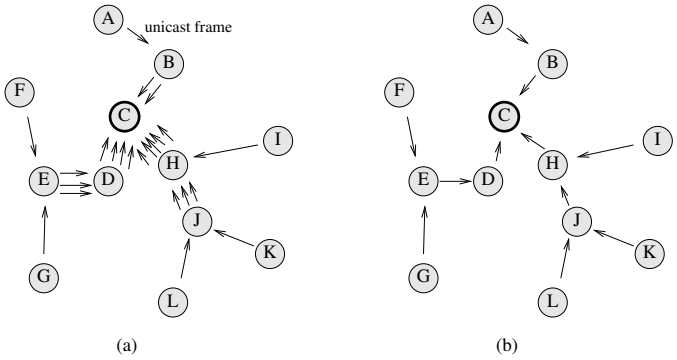

Fig. 12. Example of a concast communication in MANETs without duplicate-message cache (a) and with duplicate-message cache (b)

If PDAD detects a conflict, the packet is not given to the routing protocol. This way, the routing protocol table can be held clean of faulty routing information. However, this only works with a non-probabilistic scheme. In case of a probabilistic scheme, faulty routing information will leak to the routing protocol before PDAD is certain that a conflict exists.

X. Address Change Management

If two nodes are involved in an address conflict, at least one of them has to change its address to resolve the conflict and to bring the network back to a properly configured state. On an address change all active transport-layer connections of the node may break. This problem is similar to the situation of network-layer mobility, where a node moves to another subnet and has to change its address to a topologically correct one. For these scenarios, solutions like Mobile IP [20] exists. But unlike Mobile IP, there are no home or foreign agents in mobile ad hoc networks. We borrow the idea of route optimization from Mobile IPv6 [21]: The node that changes its address notifies the corresponding node about the new address using a Binding Update message. PACMAN uses its old address as home address. The corresponding node can then use the routing header in case of IPv6 or an IP tunnel in case of IPv4 to send packets to the new address. These packets can then be de-capsulated and arrive at the transport layer with the old IP destination address. If a location management or name service exists it should also be notified.

XI. Conclusion

This paper presents PACMAN, a modular architecture for the distributed autoconfiguration of mobile ad hoc networks. PACMAN follows a hybrid approach and massively uses cross-layer information from on ongoing routing protocol traffic to provide an efficient address assignment based on a probabilistic algorithm and a Passive Duplicate Address Detection (PDAD). Various PDAD algorithms for proactive and reactive routing protocols are proposed and the applicability to current routing protocols is discussed. For OLSR and FSR, a configuration of PDAD is proposed that allows the passive detection of all conflicts without modification of the routing protocol. For AODV slight modifications are necessary. The problem of routing protocol dependency is alleviated by a modular architecture. Due to its passive nature, PACMAN requires almost no protocol overhead. It can even lower the routing protocol overhead by assigning address in a manner that enables IP address encoding.

Appendix I
Probability of conflict in case of random address assignment

In case of random assignment the probability of conflict only depends on the address space and the number of nodes in the network: the bigger the address space, the lower the probability of conflict. This probability can be calculated using probability theory: the first node in the network of course gets a unique address. The second node selects a unique address with the probability of $\frac{r-1}{r}$ with r being the address space. In general, the probability that no conflict occurs between a number of n nodes is given by equation 5. We approximate this equation by applying Stirling's formula (equation 6) in order to get a more compact representation (equation 7). Finally, equation 8 expresses the probability that an address conflict occurs.

$$\begin{aligned} p_n(r,n) &= \frac{r}{r}\cdot\frac{r-1}{r}\cdot\frac{r-2}{r}\cdots\frac{r-n+1}{r} \\ &= \frac{r(r-1)(r-2)\cdots(r-n+1)}{r^n}\cdot\frac{(r-n)!}{(r-n)!} \\ &= \frac{r!}{r^n(r-n)!} \end{aligned} \tag{5}$$

$$n! \approx \sqrt{2\pi n}\cdot n^n\cdot e^{-n} \tag{6}$$

$$
\begin{aligned}
p_n(r,n) &\approx \frac{r^{r+\frac{1}{2}} \cdot e^{-r}}{r^n \cdot (r-n)^{r-n+\frac{1}{2}} \cdot e^{-r+n}} \\
&= \left(\frac{r}{r-n}\right)^{r-n+\frac{1}{2}} \cdot e^{-n} \\
&= \left(1-\frac{n}{r}\right)^{n-r-\frac{1}{2}} \cdot e^{-n} \qquad (7)
\end{aligned}
$$

$$
p_c(r,n) = 1 - p_n(r,n) \approx 1 - e^{-n}\left(1-\frac{n}{r}\right)^{n-r-\frac{1}{2}} \qquad (8)
$$

APPENDIX II
PROBABILITY OF CONFLICT IN CASE OF RANDOM ADDRESS ASSIGNMENT AND CONSIDERATION OF THE ALLOCATION TABLE

We assume that the first nodes have an empty allocation table and pick an address randomly from the whole address space. Other nodes joining the network learn about allocated addresses from ongoing routing protocol traffic. They pick an address randomly, but consider the addresses in their allocation tables additionally. We assume, that a new node i does not know the last j addresses before selecting an address, e.g., because they are not yet distributed in the network. The probability that no conflict occurs can then be estimated by equation 9.

$$
\begin{aligned}
p_n(r,n,j) &= \frac{r}{r} \cdot \frac{r-1}{r} \cdots \frac{r-n+j}{r-1} \cdots \frac{r-n+1}{r-n+j+1} \\
&\quad \cdot \frac{(r-n)!}{(r-n)!} \cdot \frac{(r-n+j)!}{(r-n+j)!} \\
&= \frac{(r-n+j)!}{r^j (r-n)!} \qquad (9)
\end{aligned}
$$

After applying the Stirling's formula (equation 6), we get equation 10 for an estimation of the probability of conflict in case of random assignment and consideration of the allocation table.

$$
p_c(r,n,j) \approx 1 - (r \cdot e)^{-j} \frac{(r-n+j)^{r-n+j+\frac{1}{2}}}{(r-n)^{r-n+\frac{1}{2}}} \qquad (10)
$$

ACKNOWLEDGMENT

This work was funded in the context of the project IPonAir by the German Federal Ministry of Education and Research (BMBF).

REFERENCES

[1] R. Droms, "Dynamic host configuration protocol," RFC 2131, Mar. 1997.

[2] S. Cheshire, B. Aboba, and E. Guttman, "Dynamic configuration of IPv4 link-local addresses," IETF Draft, 2003.

[3] S. Thomson and T. Narten, "IPv6 stateless address autoconfiguration," RFC 2462, Dec. 1998.

[4] Intel. (1997) Duplicate mac address on intel stl2 server board. [Online]. Available: http://www.intel.com/support/motherboards/server/stl2/ta-503.htm

[5] Cisco. (1997) Duplicate mac addresses on cisco 3600 series. [Online]. Available: http://www.cisco.com/warp/public/770/7.html

[6] T. Narten and R. Draves, "Privacy extensions for stateless address autoconfiguration in IPv6," RFC 3041, Jan. 2001.

[7] C. Perkins, J. T. Malinen, R. Wakikawa, E. M. Belding-Royer, and Y. Sun, "IP address autoconfiguration for ad hoc networks," IETF Draft, 2001.

[8] K. Weniger and M. Zitterbart, "IPv6 autoconfiguration in large scale mobile ad-hoc networks," in *Proc. of European Wireless 2002*, vol. 1, Florence, Italy, Feb. 2002, pp. 142–148.

[9] N. H. Vaidya, "Weak duplicate address detection in mobile ad hoc networks," in *Proc. of ACM MobiHoc 2002*, Lausanne, Switzerland, June 2002, pp. 206–216.

[10] S. Nesargi and R. Prakash, "MANETconf: Configuration of hosts in a mobile ad hoc network," in *Proc. of IEEE Infocom 2002*, New York, USA, June 2002.

[11] J. Boleng, "Efficient network layer addressing for mobile ad hoc networks," in *Proc. of International Conference on Wireless Networks (ICWN'02)*, Las Vegas, USA, June 2002, pp. 271–277.

[12] C. Schurgers, G. Kulkarni, and M. B. Srivastava, "Distributed assignment of encoded mac addresses in senosr networks," in *Proc. of ACM Mobihoc 2001*, Long Beach, USA, Oct. 2001.

[13] V. Bharghavan, "A dynamic addressing scheme for wireless media access," in *Proc. of IEEE ICC 1995*, Seattle, USA, June 1995.

[14] M. Mohsin and R. Prakash, "IP address assignment in a mobile ad hoc network," in *Proc. of IEEE Milcom 2002*, Anaheim, USA, Oct. 2002.

[15] K. Weniger, "Passive duplicate address detection in mobile ad hoc networks," in *Proc. of IEEE WCNC 2003*, New Orleans, USA, Mar. 2003.

[16] M. Gerla, X. Hong, and G. Pei, "Fisheye state routing protocol (FSR) for ad hoc networks," IETF Draft, 2002.

[17] C. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers," in *Proc. of ACM SIGCOMM '94)*, London, England, Oct. 1994, pp. 271–277.

[18] T. Clausen, P. Jaqcuet, A. Laouiti, P. Minet, P. Muhlethaler, A. Qayyum, and L. Viennot, "Optimized link state routing protocol," IETF Draft, 2003.

[19] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (aodv) routing," RFC 3561, July 2003.

[20] C. Perkins, "Ip mobility support for ipv4," RFC 3344, Aug. 2002.

[21] D. Johnson, C. Perkins, and J. Arkko, "Ip mobility support in ipv6," IETF Draft, June 2003.

Kilian Weniger received his diploma degree in electrical engineering from the Technical University of Braunschweig, Germany, in 2000. He is now a research assistant at the Institute of Telematics, University of Karlsruhe. His current research interests include autoconfiguration, clustering and routing in mobile ad hoc networks.