# Self-Organising Node Address Management in Ad Hoc Networks[*]

Stephen Toner and Donal O'Mahony

Networks & Telecommunications Research Group (NTRG) Trinity College,
Dublin 2, Ireland
Tel:+353 6082336, Fax: +353-1-6772204
{stephen.toner,donal.omahony}@cs.tcd.ie

**Abstract.** Whilst much effort has been put into the creation of routing algorithms to handle all sorts of mobility scenarios in ad-hoc networks, other fundamental issues, such as the addresses used by nodes, haven't been dealt with adequately. This addressing problem has recently attracted increased attention and a few proposals have been made, though often these schemes only work in limited scenarios. In this paper we present an autoconfiguration protocol designed to work in a mobile ad hoc network (MANET). The scheme allows joining nodes to dynamically obtain addresses, and has been designed to efficiently manage addressing, and to handle such scenarios as the merging and partitioning of networks. We discuss an implementation used within an emulated environment and a real self-organising ad-hoc network.

## 1 Introduction

An ad-hoc network is one where nodes collaborate to allow communication without the required presence of network infrastructure. Their dynamically changing membership and topology means that specialised routing protocols are required. The lack of manual management means that auto-configuration is a highly desirable goal. Whilst research in ad-hoc routing protocols has been strong in recent years, the same intensity has not been applied to other important related areas, such as node addressing. However these routing protocols typically rely on nodes having a unique address, and ignore this vital issue. Often nodes are assumed to have addresses configured a priori, but this is impractical and not easily accomplished. Instead we propose a dynamic scheme for assigning and managing addresses within ad-hoc networks. Nodes require a unique address for packets to be delivered to the "correct" destination, and due to the routing side effects that may arise from nodes using duplicate addresses.

In fixed IP-based networks hosts use IPv4 or IPv6 addresses, which have a hierarchical element to their structure. An IP address typically has 2 purposes: It identifies the node, and encodes routing information. This second point is vitally

[*] This material is based upon work supported, in part, by the European Office of Aerospace Research and Development, Air Force Office of Scientific Research, Air Force Research Laboratory, under Contract No. F61775-01-WE052.

important when nodes are mobile. Traditionally this mobility has been handled in two ways. DHCP uses servers to auto-configure nodes with a topologically correct address as they move, so normal IP routing can be used. Also Mobile IP allows nodes to maintain a static identity based on the "home address", so a node is always contactable via a static address (albeit inefficiently).

Unfortunately the use of the same combination of DHCP and Mobile IP within an ad-hoc network may prove impractical, and may be even impossible. Because there may be no infrastructure available, DHCP, which relies on an address server, is unable to provide a solution to the addressing problem. In fact it is actually unnecessary to configure a topologically correct address as, unlike in the fixed network, the address used in an ad-hoc network has no routing purposes (for most routing protocols). Addresses are simply used as a means of identification within the ad hoc network, and so must simply be unique within the network. Instead the ad-hoc routing protocol handles the routing, typically using a flat address space.

An obvious solution to this problem is to provide each node with a permanent unique identifier that could be used to identify the node within the MANET. *A hardware-based solution,* where every piece of hardware has a permanent unique identifier introduces a number of problems:

- Whilst 48-bit IEEE MAC addresses are the closest thing we have to this, and are designed to be unique, existing MAC addresses aren't guaranteed to be unique [7, 3]. There are known instances where network interface cards (NICs) have been issued with duplicate MAC addresses. Also not all hardware has an IEEE MAC address, as it may be using different wireless technologies.
- It would require a change in existing software, as often 32-bit addresses are used.
- Interworking between fixed and ad-hoc networking would be complicated
- The fact that the identity of the node can be determined from its address raises privacy concerns.
- The larger the address used, the greater the per-packet overhead. This becomes more of an issue in source routing protocols where a list of addresses is placed in each packet.

*Mobile-IP* based solutions (e.g. MIPMANET [6]) typically require nodes to have a permanent home address, which is used as the unique address within the ad-hoc network. All existing hardware then requires a permanent address, and the increased address size that would be required (probably requiring the migration to IPv6 due to the shortage in IPv4 addresses) would require software changes, and create extra overheads. It would also create new issues if the ad-hoc network was to become connected to the global network, with the selection of a care-of address either through the selection of an appropriate foreign agent (as would be the case if IPv4 were used), or the configuration of a local care-of address. For example with MIPv6 the mechanism for auto-configuring a care-of address [7] still requires Duplicate Address Detection (DAD), even though the address has the MAC address embedded in it. Performing this DAD in an ad-hoc network is not a trivial matter, and a new scheme is required to manage this autoconfiguration of addresses.

If we discount the use of a static address then a scheme is required to allow nodes to autoconfigure an address within an ad-hoc network. Current solutions typically have limitations, working only in certain scenarios. The scheme we present provides an ad-hoc autoconfiguration mechanism, that isn't reliant on the MAC address and works in an ad-hoc environment. It is based on the use of IPv4 addresses, but this could easily be extended to use IPv6 addresses. It aims to provide a solution that minimises both traffic and time requirements for configuration and management. We also briefly examine how these networks could be connected to the fixed network.

## 2 Related Work

Perkins et al [1] have proposed a simple Duplicate Address Detection (DAD) based scheme whereby nodes choose a tentative address and send a request to this address. The absence of a reply is assumed to indicate that this address is free for use. However there are a few limitations to this approach.

- In the event of a partitioned network, or the coalescing of two networks, the address would not be guaranteed to be unique and the procedure would have to be repeated again after merging.
- No mechanism is suggested to detect the merging of networks, and even if such a mechanism were in place, every single node in both networks would have to redo this DAD test, potentially causing a disruptive broadcast storm. Whilst this problem is recognised, no solution is suggested.
- Timeout-based approaches suggest that the interval required for a packet to successfully traverse the entire network can be calculated, whereas the nature of ad-hoc networks makes this difficult.
- Joining the network takes a long period, as a number of retries are necessary to ensure that all nodes received the broadcast, and time must be allowed for any node to respond.

Vaidya's proposal [2] aims to ensure packets are delivered to the "correct" node even if two nodes are using the same address. [2] argues that "Strong" duplicate address detection (i.e. showing "correct" behaviour where the existence of duplicates can be detected within a bounded interval) is impossible. The basis of this argument is that an ad-hoc network may become partitioned and so the detection of duplicates within these two partitions is infeasible. Whilst this may be true, we argue that an ad-hoc network that has two partitions can be treated as two distinct ad-hoc networks. A node's address is only required to be unique within the same partition as only these nodes are reachable. This proposal requires the modification of existing routing protocols to incorporate their scheme, something that we would rather avoid.

In MANETconf [3] joining nodes select an existing member (the "initiator") of the network to obtain an unused address. This floods the network with a request for a specific address and all nodes reply with positive or negative responses, if this address is in use. This scheme displays a few limitations:

- If 2 nodes were to try to concurrently join the network, in the same region, their requests would be indistinguishable. The "initiator" node has no means to distinguish between the nodes when communicating, and so both nodes could select the new address.
- The requirement for the nodes to reply positively (suggesting the address is free) creates extra traffic and is superfluous as regards the particular address allocation at hand (a single negative response will override all these responses), and is just used as a means to detect nodes having left.

- To detect merging and resolve duplicate addresses it is proposed that two previously distant nodes should exchange information about their partition when they come together (suggesting some sort of proactive beaconing). In the event of conflict the node with the least TCP connections is expected to acquire a new address, though the scheme provides no details on how this would be determined.

Patchipulusu [4] uses a leader to identify the group, and nodes joining the network receive sequential addresses, with the newest member taking over the role as leader. Each node periodically sends an update beacon message to the nodes with the next and previous addresses so that node losses can be detected. Any node that becomes separated for a period must acquire a new address. Similarly when two networks merge, all the nodes in the network with the least number of nodes must acquire a new address. This scheme potentially requires a lot of unnecessary address changes, causing loss in routing information, loss in connectivity of connections, and disrupts communication.

## 3 Our Approach

Our proposal is based on the election of a "leader" within each network partition. This leader:

- provides identification for the network so that the merging of networks can be detected.
- acts in a manner similar to a DHCP server, whereby it hands out addresses to new nodes.
- allows for an efficient merging of networks.

This leader maintains a list of all addresses in use, and new nodes must apply to it for an appropriate address. One of the fundamental considerations of an approach like this is that the mechanism should cope easily with the loss of the leader node.

Each node has a 32-bit address (for compatibility with the existing IPv4 network), an extra 32-bit unique identifier (UID), and a network identifier (NID). In the event of the ad-hoc network becoming joined to the fixed network, the IP network will not use this auto-configured address to identify the node (Later in this section we outline our global connectivity scheme). Only this 32-bit address is used elsewhere, for routing etc., with the other identifiers (UID and NID) only used within the addressing protocol to uniquely identify nodes, and network merges.

*Joining*

A node wishing to join chooses a temporary random address from a reserved range. Whilst using this address the node does not participate in the routing protocol. Due to the limited number of addresses available in this range the possibility of duplicates being chosen is reasonably high, but the address is only used in forming a link-local address with an existing member of the network, and must only be unique within the physical range of this existing member. To supplement this, a node also chooses the UID. Any identifier could have been chosen, for example the MAC address would have been ideal, but a random 32-bit value was selected for simplicity. The address management scheme uses this tuple {address, UID} to determine if a node is actually a different node using the same address, or just the same node rejoining the network (possibly due to the temporary segmentation of the network).

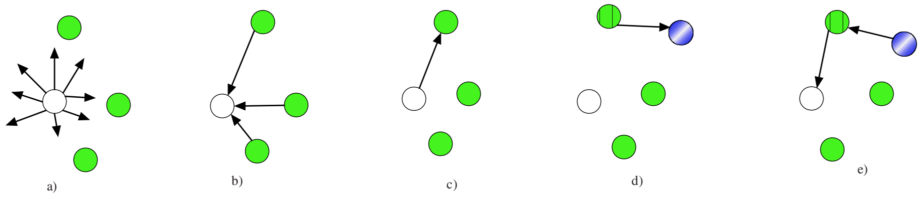


**Fig. 1.** Joining the network

The node broadcasts the *join* message, containing the temporary address and UID. All nodes in range reply, and the joining node selects one to act as an "agent" (this is a node already in the network and so can participate in the routing protocol) to obtain an address (preference is given to leader nodes if they are directly within range as this reduces the necessary traffic). This agent node stores the address details and sends a request to the leader node, containing a unique reference number for this request. The leader selects an address that isn't currently in use (this can be done immediately as the leader has a list of all addresses in use and so doesn't need to flood the network to determine if the address is free), and returns this to the agent, along with a copy of the current table of addresses in use. The agent then returns this to the joining node, after adding the appropriate destination, by translating from the reference number in the response to the temporary address the corresponding node was using.

*Leader Election*

A fundamental principle of ad-hoc networking is that any node is liable to fail, so we distribute a copy of the table to each joining node, along with a corresponding version number. When a node joins it is added to the table, and the version number is incremented. The election of a new leader is then reduced to the task of simply locating the holder of the latest version. Whilst this may not contain all the entries that the original master table held it is guaranteed to hold all the addresses of the currently connected nodes. The last node to join will hold the latest version of the table.

The leader proactively floods a beacon periodically throughout the network. This addition of a proactive element is realistically unavoidable as nodes must become aware that adjacent nodes do not belong to the same network. This could be accomplished through one of two mechanisms: either every node periodically broadcasts a message that isn't repeated, or a single node (the leader) broadcasts a message that is to be flooded throughout the entire network. However, if the leader floods the network, then the loss of this node can also be detected readily. To cope with network losses a threshold number of flooded messages may be missed before an election is initiated.

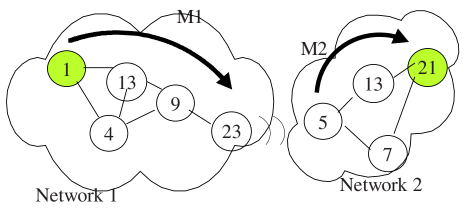
The election should cause a minimal amount of traffic. If a node realises that the leader is absent, it waits a random period before flooding an *election* message through the rest of the network stating its intention to take over as the new leader. This message contains the version number of the table held by the node, the node's address, and the last beacon message received. Nodes hearing this message will react accordingly: It compares the version of the table it holds, and the table indicated by the prospective leader. If the node's own version is greater it becomes the new provisional leader and starts a new election. If the initiating node had a more recent version, then it is marked as the provisional leader and this election message is flooded onwards. If both versions are the same, the conflict is resolved by choosing the node with the highest address as the provisional leader. In this way every node should independently and correctly determine the new leader.

When a node initiates an election it initiates a timer. If no other messages are received from more "eligible" nodes then the node assumes that it is has been elected as the new leader and floods the network with a *leader_elected* beacon containing the new information. All other nodes will be waiting for this message, as its election message should have been flooded throughout the network. If any node doesn't receive this message within a threshold period then it will reinitiate the election.

*Network merging*

The merging of the two networks can be detected through the presence of more than one leader. A node receiving a beacon from a foreign leader (containing a different NID and leader address) message must inform his current leader. Only the two leaders of the respective networks need to communicate, comparing their lists of currently used addresses to discover if any duplicate addresses are in use. These duplicate nodes will then have to change their address by re-applying for a new address.

Communication between the leaders is complicated by the fact that the path through which they communicate cannot be guaranteed to be free of duplicate addresses, as it passes through two address spaces. For example Figure 2 shows a path where two nodes along the path are using the same address. Therefore the leaders communicate through the nodes that received the foreign *leader_beacon.* For example, in Figure 2 node 23 is used so that two distinct partial paths are generated (M1 and M2) with the addresses within each guaranteed to be unique.



- Node 23 heard a leader_beacon from the 2nd network, and delivered this to its leader (Node 1)
- This encapsulates its packets, and sends to this node (i.e. Node 23, along path M1), which in turn injects the inner packet into the other network
- This is then delivered (along M2) to the leader of the second network.

**Fig. 2.** Merging Networks

*Address Reuse:* To prevent "address leakage", whereby nodes leave the network without freeing the address for reuse, a management mechanism, whereby infrequent periodic requests test for the presence of nodes, was introduced. Although this address leakage wouldn't be a problem where the address space is large enough relative to the number of nodes, it does create a problem whereby the address table could become prohibitively large.

*Routing Traffic Isolation:* When two networks merge, there will be a short period of vulnerability during which adjacent nodes belong to different networks, and so should not participate in routing with the other network until addresses have been checked for duplicates. By adding the NID to each packet this can be easily detected, and prevented. It also means that network merges can be detected more readily than having to wait on the periodic network beacon.

*Table Size:* One potential problem with the scheme is the fact that in large networks the size of the table containing the used addresses could become prohibitively large. Mechanisms for limiting the size are being examined whereby addresses may not be completely random, but may follow some sequence.

*Global Connectivity:* So far we have considered the ad-network as an isolated entity. However it would be desirable for this network to be able to interoperate with the fixed network should a possible connection be available. We are currently using a NAT-based solution to accomplish this. Related work, based on the TRIAD [8] scheme, aims to identify nodes globally using a name rather than an address. The operation of this scheme is outside of the scope of this paper.

## 4 Implementation

In the Networking and Telecommunications Research Group (NTRG) at Trinity College Dublin we have developed a flexible testbed, where components are assembled using a layered architecture. Layers have been developed performing a variety of networking functions, with working implementations provided for a number of ad-hoc routing protocols. Testing was performed with DSR (Dynamic Source Routing). The JEmu emulator [5] was used extensively to emulate realistic movement scenarios. This aided the development, as well as throwing up some interesting observations and design considerations. For example, when nodes were programmed to follow a random mobility pattern, it was found that the network often split for short periods and then might become reconnected. It was important therefore that merging and splitting of networks could be performed quickly and efficiently, and that nodes weren't unnecessarily made change address. The address configuration layer was implemented as a separate layer within our stack, requiring no changes to existing routing protocols.

Emulation scenarios were run with up to thirty nodes following a variety of mobility patterns for testing. This implementation provided validation that the protocol does handle mobility in a realistic dynamic environment, where merging and partitioning could occur, and where message loss was evident. By artificially constricting the available address range to between 51 and 100 for testing (with 1-50 used for the initial temporary address), the chance of duplicate addresses occurring was hugely increased, providing an adequate test for the protocol's ability to handle duplicate addresses. Initial testing using virtual nodes (using the JEmu emulator) indicated that the protocol could operate in realistic scenarios. The frequency of the beaconing by the Leader node had a large impact on performance, and had to be chosen to provide a balance between the need for rapid resolving of network partitioning and merging, and to avoid excess traffic.

Next the implementation was tested with real nodes (on iPaqs running Windows CE, and on laptops, using 802.11b as the wireless medium). The Dublin Ad-hoc Wireless Network (DAWN) network is a project underway in Trinity College aiming to create a useful operational ad-hoc network, and all mobile nodes in the network use this scheme. A number of applications have been developed to run across this network, including point-to-point telephony and instant-messaging applications. It contains both mobile nodes, and fixed machines that provide a permanent population for testing and connectivity. To avoid a centralised system for allocating addresses, this protocol has been used so nodes can enter the network and obtain an address without the requirement for any pre-configuration.

## 5 Conclusions

The proposed address autoconfiguration mechanism provides a dynamic protocol for mobile ad-hoc nodes, and allows the creation of a self-organising network. The proposed solution has a number of advantages over other proposals.

- It works with existing hardware and software.
- It allows nodes to quickly join the network, without having to wait for timers to expire to indicate that no other node is using the same address.
- Merging is extremely efficient, as only two nodes need communicate, rather than requiring every node to individually validate its address by flooding the entire network.
- Unlike some related protocols, nodes maintain the first address received wherever possible. This stability helps maintain cached routing information, and minimises network disruption.
- It recovers quickly from the failure of any node.
- It is independent of both routing protocol and the underlying wireless medium (e.g. there is no requirement for a unique MAC address).

This scheme provides a locally unique address within a MANET. Further work will look at evaluating this protocol in more detail. We are currently examining the issue of providing connectivity to the Internet through the use of a NAT-based mechanism for IPv4. We will also examine possible security considerations.

## References

[1] Perkins et al. *"IP Address Autoconfiguration for Ad Hoc Networks"* – Internet Draft (Nov. 2001).

[2] Nitin Vaidya *"Weak Duplicate Address Detection in Mobile Ad Hoc Networks"*, ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), June 2002.

[3] S. Nesargi and R. Prakash , *"MANETconf: Configuration of hosts in a mobile ad hoc network",* in INFOCOM, 2002.

[4] P. Patchipulusu, *"Dynamic Address Allocation Protocols for Mobile Ad Hoc Networks"*, M.Sc. thesis.

[5] Flynn, J. et al., *"A Real-Time Emulation System for Ad Hoc Networks"*. Communication Networks and Distributed Systems Modelling and Simulation Conference (CNDS 2002), Texas, 2002.

[6] U. Joönsson, F. Alriksson, T. Larsson, P. Johansson, and G. Q. M. Jr. "*MIPMANET - mobile IP for mobile ad hoc networks"*, MobiHoc, August 2000.

[7] S. Thomson and T. Narten, "IPv6 Stateless Address Autoconfiguration, RFC 2462," Internet Engineering Task Force, Zeroconf Working Group, December 1998.

[8] D. Cheriton and M. Gritter. "TRIAD: A new next generation Internet architecture", March 2000.