Computing (2014) 96:777–791
DOI 10.1007/s00607-013-0307-3

E-D2HCP: enhanced distributed dynamic host configuration protocol

L. J. García Villalba · A. L. Sandoval Orozco · J. García Matesanz · T.-H. Kim

Received: 22 January 2013 / Accepted: 5 February 2013 / Published online: 19 February 2013

Abstract Mobile Ad Hoc Networks (MANETs) consist of mobile nodes equipped with wireless devices. They do not need any kind of pre-existent infrastructure and are about self-managed networks. MANETs enable communication between mobile nodes without direct links and across multihop paths. To ensure correct operation of the routing protocols, MANETs, have to assign unique IP addresses to the MANET devices. Furthermore, the address assignment is an important issue when dealing with MANET networks because the traditional approaches are not applicable without some changes, having to provide new protocols for the address auto-configuration. These

L. J. García Villalba (✉) · A. L. Sandoval Orozco
Group of Analysis, Security and Systems (GASS), Department of Software Engineering and Artificial Intelligence (DISIA), School of Computer Science, Office 431, Universidad Complutense de Madrid (UCM), Calle Profesor José García Santesmases s/n, Ciudad Universitaria, 28040 Madrid, Spain
e-mail: javiergv@fdi.ucm.es

A. L. Sandoval Orozco
e-mail: asandoval@fdi.ucm.es

J. García Matesanz
Group of Analysis, Security and Systems (GASS), Departmental Section of Computer Systems and Computing, Faculty of Mathematics, Despacho 310-F, Universidad Complutense de Madrid (UCM), Plaza de Ciencias, 3, Ciudad Universitaria, 28040 Madrid, Spain
e-mail: julian@sip.ucm.es

T.-H. Kim
Department of Multimedia Engineering, Hannam University,
133 Ojeong-dong, Daedeok-gu, Daejeon, Korea
e-mail: taihoonn@hannam.ac.kr

T.-H. Kim
Department of Information Technologies, Global Vision School Australia (GVSA), 20 Virgina Court, Sandy Bay, TAS, Australia
e-mail: taihoonn@gvsa.asia

schemes must take into account the properties of MANETs such as dynamic topology, limited resources or lack of infrastructure. In this paper, we propose a stateful scheme for dynamic allocation of IP addresses in MANETs entitled Extended Distributed Dynamic Host Configuration Protocol because it is based on a previous piece of work (D2CHP). This extension includes the network merging not covered by its predecessor. Simulation results show that the new protocol also improves D2HCP functionality in areas such as fault tolerance, concurrency and latency.

Keywords Mobile Ad Hoc Network (MANETs) · IP address assignment · Autoconfiguration · Dynamic host configuration · Stateful protocol · Synchronization · OLSR · Proactive routing protocol · D2HCP · E-D2HCP

Mathematics Subject Classification 68M10 Network design and communication · 68M12 Network protocols

1 Introduction

A Mobile Ad Hoc Network (MANET) consists of a set of mobile nodes equipped with wireless devices. The nodes establish direct links with every node that is within its transmission range and thereby enable communication between nodes without direct links and across multihop paths. This kind of computer network is characteristic of its dynamic network topology, lack of any fixed infrastructure or administration, limited bandwidth and energy saving capacity. They are important features within the field of computer networks. One of the most important issues related to this type of network is the routing [7]. In order to provide routing in MANETs, it is necessary to design efficient mechanisms for address allocation and therefore the schemes for IP address allocation must meet certain requirements [1].

This work proposes a stateful auto-configuration protocol that guarantees the uniqueness of IP addresses under a wide variety of network conditions such as missing messages and merging and partitioning of networks. This work is structured into six sections; the first one is the present introduction. Section 2 shows the obligated references in the auto-configuration protocol scope of MANETs. Section 3 contains an itemized specification of the so-called Extended Distributed Dynamic Host Configuration Protocol (E-D2HCP), a proposal concerning IP addresses auto-configuration for MANETs. A comparison between ED2HCP and its predecessor Distributed Dynamic Host Configuration Protocol (D2HCP) is presented in Sect. 4. Section 5 presents the E-D2HCP protocol simulations carried out in NS-3 [12]. Finally, Sect. 6 discusses the main advantages of the newly developed protocol as well as potential future extensions to the study.

2 Related works

Address auto-configuration can be classified in two types; stateful and stateless. Stateful auto-configuration mechanisms assign a unique priori address to the nodes by maintaining a common address pool. DHCP [8] uses a central entity to maintain this pool and thus, cannot be used in a peer-to-peer style mobile ad hoc network. The

MANETconf [10] protocol tries to adapt this scheme to mobile ad hoc networks by using a mutual exclusion algorithm to maintain a distributed pool of addresses. Depending on the mobility scenario, the maintenance of a common address pool in all nodes of the network may be a complex and bandwidth consuming process, especially in the presence of frequent partitioning and merging of the network.

Instead of the assignment of addresses by a second entity, stateless auto-configuration allows the nodes to construct addresses, which are usually based on a hardware ID or a random number. A Duplicate Address Detection (DAD) mechanism is used to assure the uniqueness of the address. The IETF zeroconf working group is currently developing such a mechanism for IPv4 [5] while the IP Version 6 working group has already standardized a stateless auto-configuration mechanism for IPv6 [13]. Both protocols were not designed for MANETs, but adaptations exist [11,18]. Perkins et al. [11] propose that a node floods the network with an address request message which is directed to the constructed address. If no reply is received before the timer expires, it is assumed that the address is not occupied. Due to the fact that network merging is not considered, duplicate addresses can still occur. On the other hand, Weniger and Zitterbart [18] propose a protocol that supports network merging, but although a hierarchical approach is used, a considerable amount of bandwidth is needed solely for the detection of duplicate addresses.

Weak Duplicate Address Detection [15] aims to reduce the overhead needed for the DAD by integrating it with the routing protocol. Nodes in the network are identified with a "Virtual IP address", which is the combination of an "IP address" and a "Key" that can be based on a hardware ID or a random number. This concept is similar to the scheme of embedding the MAC address in an IPv6 address; however it differs in that the key is not used for routing decisions. If a node receives a packet containing an IP address that is stored in its routing table, but that has a different key, a confliction of addresses is detected.

In brief, there are numerous works that present proposals for address configuration in a mobile ad hoc network using the Stateful [9,10,14] and Stateless [11,15,18] mechanism.

Bernardos et al. [2–4] carried out a rigorous study of the problems of the auto-configuration in MNETs, presenting an itemized review of the more representative auto-configuration protocols. Without doubt, the most representative are those described in García Villalba et al. [16].

Distributed Dynamic Host Configuration Protocol [17] reduces the control overhead by promoting cooperation between routing and auto configuration protocols. However, aspects of network merging are not covered. This paper presents an extension of D2HCP that allows network merging.

3 E-D2HCP functional specification

We considered monitoring the routing protocol as a method of synchronization because the proactive routing protocols keep updated information about the network state. The nodes send information periodically about other nodes in the network such as their links. This information allows E-D2HCP to determine the free addresses in the network and reduces the consumption of bandwidth. The selected routing protocol is OLSR [6].

This routing protocol ensures that all nodes, have sufficient topological information at all times to construct routes to all the other destinations in the network and optimizes the delivery of broadcast messages using MPRs. Here we describe the operation of the E-D2HCP protocol.

The E-D2HCP protocol is a stateful approach and implements a state machine. Each of the network devices using E-D2HCP as autoconfiguration protocol keeps information about the device configuration process. There are five possible states: *Searching Server*, *Waiting Allocation*, *Synchronizing*, *Suspended*, and *Configured*.

3.1 IP assignment

The autoconfiguration process initiates when a node wishes to join the network and needs to obtain an IP address. This node is so-called "client node".

The "client node" sends a broadcast message so-called Server_Discovery and passes on SearchingServer state to wait for a Server_Offer message from one or more nodes in the network. The "client node" remains in this state up to a time interval of SEARCH_SERV_TIME milliseconds elapses.

Once this time interval passes, the client node performs the following analysis:

- If the client node has not received any message, it increases *nRetr* counter and reawaits a time interval SEARCH_SERV_TIME in milliseconds to receive more Server_Offer messages.
  This process is carried out until *nRetr* value gets MAX_DISC_RETRIES. Whether *nRetr* has got MAX_DISC_RETRIES value, the received answers are performed.
- If some received answers proceed from a node with WaitingAllocation state, it means such a node has already found a server and has requested an address for its network interface. Hence, the client node goes to Suspended state whilst waiting for the node selected as server to complete its configuration process.
- Even if any node does not exist in WaitingAllocation state, but nodes in Suspended state exist, the client node also goes to Suspended state. It means that either the neighbour node has found another one that is performing the configuration of its interface, or a chain of nodes exists in Suspended state from neighbour node, until a node that has already begun the process. For this reason, the client node suspends the process of address configuration from network interface for a period of time, so that the neighbour nodes can end their owner process. When the waiting time expires, the client node restarts the process by going to SearchingServer state.
- If the answers come from nodes in SearchingServer state, it means that the message sender has not found any node that began or ended the configuration process and this node has initiated the process of creating a new network and it sends a ClientSynchronization message and Synchronizing is passed on. In this state, the nodes which have priority are identified.
- Whether *nRetr* counter gets MAX_DISC_RETRIES value and the node does not receive any Server_Offer message, it is assumed that a network exists in the environment, thus the client node initiates the generation of a new network.

If the client node has received Server_Offer messages, analyzes the status of the nodes that have responded and verifies that there are nodes in Configured state:

– It analyzes the networks that have responded checking the NetworkID of the server node and it selects a network to join.
– It analyzes the nodes that responded from the chosen network and chooses one as "server node" with priority going to those nodes that dispose free addresses, have an owner address range and do not need to perform a remote request.
– After selecting a server node, the notification of its selection sends a Server_Poll message to it and therefore the client node passes it to WaitingAllocation state while it waits to receive an IP_Assigned message during an ADDR_ALL_ TIME period of time. If after this period of time the client node has not received any answer the *nRetr* counter value is increased from WaitingAllocation state. If this counter value is lower than MAX_ALL_RETRIES, Server_Poll message is then forwarded and the same state is maintained. In the case of getting MAX_ALL_RETRIES value it is passed on SearchingServer state and the process is restarted. If the node selected as server offers an address range that they are not its own, it sends the IP_Range_Request message to owner node of range ("remote node") requesting an offered address range. The "remote node" reply to server node with the IP_Range_Return message delivering requested range.
– The server node sends the IP_Assigned message with which it confers the administration of an address range to client node and it provides an address to set the network interface identified by hardware address, Therefore the client node passes onto Configured state.

3.2 Network initialization

The network initialization can be performed by a single node or group of nodes. If a client node does not find any neighbour, it chooses an address range and generates a NetworkId. The node configures the network interface with this information. If the number of nodes is bigger than one in the initialization process, the group of nodes set their state to Synchronizing. Then, the nodes select a subgroup, called the network precursors. These precursors choose an address range and NetworkId. Following this, the precursor nodes distribute the address to the other nodes.

3.3 Synchronization

The synchronization is done by monitoring the routing table of routing protocol OLSR [18]. The joining or departure of a node in the network is detected when OLSR adds a new route to its routing table, or deletes an existing one. By detecting the joining or departure of a node in the network, it is updated locally, and without exchanging any message, Free_IP_Blocks table.

For this reason, the following rules are obeyed:

– The responsibility of recovering the IP addresses that a node which leaving the network makes available is one that can be attached to the right of the free block. This will not be possible when the block to be collected contains the lowest address

of the network. In that case, the node that picks the block up is one that can add to it by the left.
– By dividing the free addresses in two blocks to deliver one of them to a new node that joins the network, the node that acts as server delivers the sub-block, which does not contain its own IP address, to the client.

By detecting the joining of a new node, a new entry in the table for it is created, and the free address block of the node which supplied its IP address will be updated. To know who this node which acted as server was, it is simply necessary to find out which node has the IP address of the new node in its free address block.

When the node departure is detected, its entry must be removed, and an update of the corresponding nodes, now available IP address must be recorded.

Above is similar in D2HCP. However, when OLSR loses a node, E-D2HCP examines its entry. If the lost flag from this entry is true, the entry is deleted. In this case, the address block assigned to node A is handed over to node B. This node B has the preceding address block to the block of node A. If there is no preceding block, the node B will be the node with the next block. If the lost flag value is false, then it is set to true. This technique prevents OLSR failures, and route changes.

3.4 Protocol messages

Message header sending by E-D2HCP protocol has the following fields:

– *Type*: The kind of contained message. It can have the following values: 1 (*Server_Discovery*), 2 (*Server_Offer*), 3 (*Server_Poll*), 4 (*IP_Assigned*), 5 (*IP_Range_Request*), 6 (*IP_Range_Return*), 7 (*Hello*), 8 (*Merge_Network*), 9 (*IP_Address_Invalidate*).
– *Reserved*: Reserved bits for future usage. It contains the value 0.
– *Length*: The message length.

Then the different messages utilized in the protocol are detailed.

3.4.1 Server_Discovery

This message is sent by a node which wishes to set a network interface. The fields are the following:

– *Hwtype*: The type of hardware, for example: Ethernet (1) or IEEE 802 Networks (6). To refer STD 2 - Assigned number of Internet for a completed list.
– *Hwlength*: Hardware address length in bytes. Ethernet and token-ring use 6, for example.
– *G*: Flag which indicates whether the client node disposes of a link to another network.
– *N*: This field indicates the type of request. The value 0 refers that it does not exist any constraint for networks. The value 1 denotes that a network with a determinate group of identifiers is being looked for. The value 2 indicates it is looking for a network created by a device with a determinate hardware address.

- *I*: If it contains the value 1, this indicates that a network with Internet access is looked for.
- *S*: Required service number by the network (DNS).
- *It*: Current iteration of server discovery process.
- *Client Hardware Address*: Hardware address of network interface that wishes to configure.
- *Gateway Address*: Optional field. This field is taken into account if the flag G is activated.
- *Network id*: Reference identifier when the value of field N is 1 or 2.
- *Serv id i*: Optional field. Each field contains the identifier of a service. There will be as many fields as the value of field S indicates.

3.4.2 Server_Offer

This message is sent in response to Server_Discovery message. This message may be sent as nodes which have ended the configuration process or as nodes which have not completed the process yet. The client node is notified of the node state and, if it is configured, relative information is sent to network. The fields are the following:

- *Hwtype*: The kind of hardware, for example: Ethernet (1) or IEEE 802 Networks (6). To refer STD 2 - Assigned number of Internet for a completed list.
- *Hwlength*: Hardware address length in bytes. Ethernet and token-ring use 6, for example.
- *S*: It indicates the node state which has replied to the Server_Discovery message.
- *NG*: This field contains the gateway number that such a network has.
- *access*: It indicates the access constraints to network.
- *Client Hardware Address*: Hardware address of network interface that wishes to configure the client node.
- *Server Address*: Server IP address.
- *Network id*: Network identifier.

3.4.3 Server_Poll

This message is sent by a client node to server node. Once a client node has received a Server_Offer message from one or more nodes, it sends a Server_Poll message to one of them. This message indicates to the recipient that it has been selected by the client node as the server. The fields are the following:

- *Hwtype*: The kind of hardware, for example: Ethernet (1) or IEEE 802 Networks (6). To refer STD 2 - Assigned number of Internet for a completed list.
- *Hwlength*: Hardware address length in bytes. Ethernet and token-ring use 6, for example.
- *Reserved*: Reserved bits for future usage. It contains the value 0.
- *Client Hardware Address*: Hardware address of network interface that wishes to configure the client node.

3.4.4 IP_Assigned

Message sent to client node by a selected server node. This message provides the management with an address range and an address to configure the network interface identified by a hardware address. The fields are the following:

- *Hwtype*: The kind of hardware, for example: Ethernet (1) or IEEE 802 Networks (6). To refer STD 2 - Assigned number of Internet for a completed list.
- *Hwlength*: Hardware address length in bytes. Ethernet and token-ring use 6, for example.
- *NG*: Available Gateway number in the network.
- *Reserved*: Reserved bits for future usage. It contains the value 0.
- *Client Hardware Address*: Hardware address of network interface that wishes to configure the client node.
- *First IP Address*: First address of granted address range.
- *Last IP Address*: Last address of granted address range.
- *Client IP Address*: Assigned address to network interface.
- *Network mask*: Network mask.
- *Network id*: Network identifier.
- *Gateway i Address*: Address of gateway number i.
- *Network i Address*: Network address whose gateway i gives access to it.

3.4.5 IP_Range_Request

Message sent to owner node of range ("remote node") requesting an offered address range. The fields are the following:

- *Server Address*: Address of selected node as server. It is the node which initiates the request for an address range to a remote server.
- *Remote Server Address*: Address of the remote server which an address range is requested from.

3.4.6 IP_Range_Return

When a configured node receives the request of an address range from another node in the network, in the case of disposing of free addresses, it replies with an IP_Range_Return message.

The fields are the following:

- *Server Address*: Address of selected node as server. It is the node which initiates the request of an address range to a remote server.
- *Remote Server Address*: Address of the remote server which an address range is requested from it.
- *First IP Address*: First address from address range.
- *Last IP Address*: Last address from address range.
- *Client IP Address*: IP address granted to network interface.
- *Network Mask*: Network mask.

3.4.7 Client_Synchronization

When two adjacent nodes start the auto-configuration process and there is no other node that has progressed in this task, they send a synchronization message to set which of them will have priority to initiate the network. The fields are the following:

- *Hwtype*: The kind of hardware, for example: Ethernet (1) or IEEE 802 Networks (6). To refer STD 2 - Assigned number of Internet for a completed list.
- *Hwlength*: Hardware address length in bytes. Ethernet and token-ring use 6, for example.
- *Reserved*: Reserved bits for future usage. It contains the value 0.
- *Client Hardware Address*: Hardware address of network interface that wishes to configure the client node.
- *sHwtype*: Type of hardware, for example, Ethernet (1) or Networks IEEE 802 (6). With reference to STD 2 - Assigned numbers from the internet for a complete list.
- *sHwlength*: Hardware address length in bytes. Ethernet and token-ring use 6, for example.
- *Reserved*: Reserved bits for future usage. It contains the value 0.
- *Server Hardware Address*: Hardware address of network interface that wishes to realize the synchronization.
- *Delay*: It indicates the time in milliseconds that it has elapsed from when the sender node started the configuration process until it received the Server_Offer message from the other node.

3.4.8 Hello

Message periodically sent to detect possible network fusions. The fields are the following:

- *Node Address*: IP address of sender node.
- *Network Id*: Network identifier.

3.4.9 Merge_Network

Unicast message sent by Co-MergeAgent to MergeAgent node to detect a merge. The fields are the following:

- *Owner address*: node address.
- *First address*: the first address of the block from node with address “Owner Address”.
- *Last address*: the last address of the block from node with address “Owner Address”.
  The previous triplet (Owner address i, First Address i, Last Address i) repeats for all network nodes.
- *Network id*: Network identifier.

3.4.10 IP_Address_Invalidate

Message sent by Co-MergeAgent to MergeAgent node to detect a merge. The fields are the following:

Broadcast message sent by MergeAgent nodo to all nodes in network to invalidate their IP address in your network and and instruct each node to start the process of setting up your IP address on the network that will to merge. The field are the following:

– *Network id*: Network identifier.

4 E-D2HCP versus D2HCP

E-D2HCP is an extension of D2HCP. The following changes are made to the protocol D2HCP:

Two new messages have been added:

– *ClientSynchronization*: When two adjacent nodes begin the auto-configuration process and another node that has progressed on this task, doesn't exists, they send a synchronization message to establish which of them will have priority to initiate the network.
– *Hello*: Message sent periodically to detect possible network merging.

The following states have been modified:

– *SearchingServer*: State indicating the node has sent the Server_Discovery message.
– *WaitingAllocation*: It enters this state when the client node has found a server that can provide a valid address.
– *Suspended*: A client node enters this state when none of the nodes that have responded to the Server_Discovery message have completed the configuration process.
– *Synchronizing*: When two adjacent nodes have not found any node that has started or ended the configuration setup, they go to start a new network. In this state the decision is made as to which nodes have priority in this process.

5 Simulation and results

For the performance evaluation of auto-configuration protocol E-D2HCP we have taken into account the metrics of latency in assignment, control message number emitted in each address configuration and overhead as well as packet number in Bytes.

E-D2HCP was implemented and evaluated with Network Simulator 3. In this section, we analyze the impact of the number of nodes in the protocol overhead and latency. Table 1 summarizes the main parameters used during simulations.

In Figs. 1 and 2, we verified the scalability of E-D2HCP. We have evaluated the latency and overhead in the address assignment process.

Figure 1 shows the evolution of the latency value depending on the node number in the network, showing values of IPv4 addresses from Class C, observing that the E-D2HCP latency decreased by 50 % respect to D2HCP.

Table 1 Simulation parameters

Parameter	Values
Simulation area	750 × 750 m^2
Routing protocols	OLSR
OLSR update/hello interval	5 s/2 s
Node density	40 nodes/km^2
Simulation time	600 s
Mobility model	Random waypoint
MAC protocol	802.11
Transmission range	276 m
Network address class	B and C

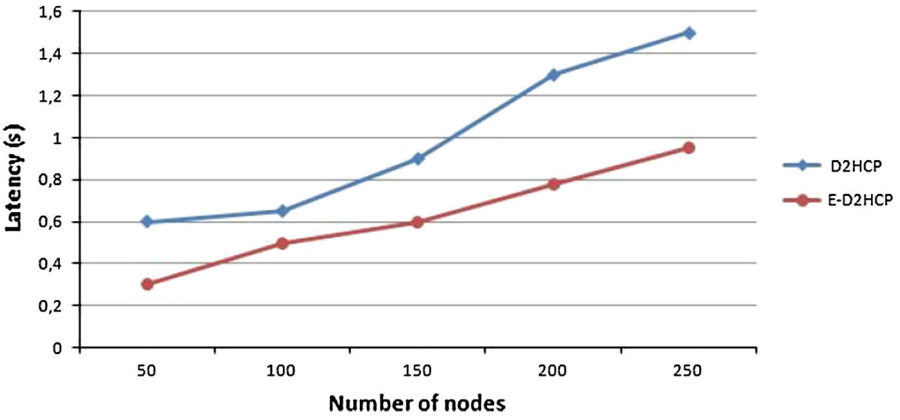

Fig. 1 Latency graph

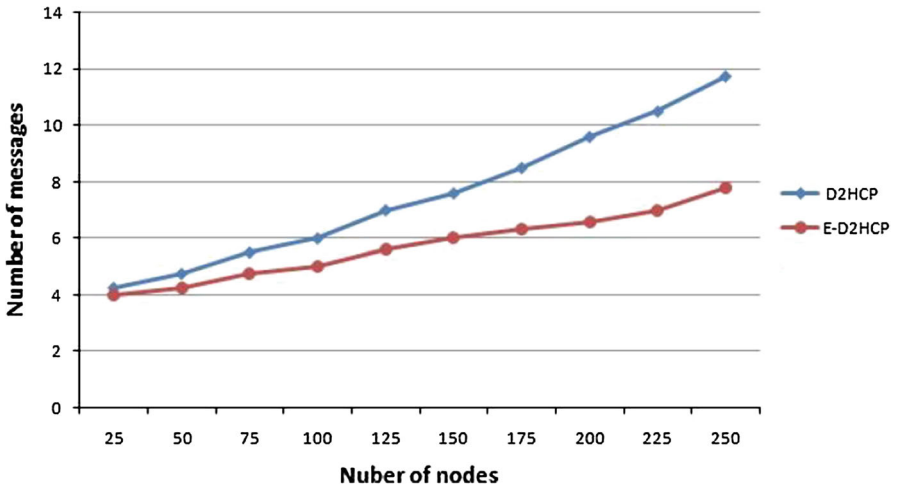

Fig. 2 Overhead graph

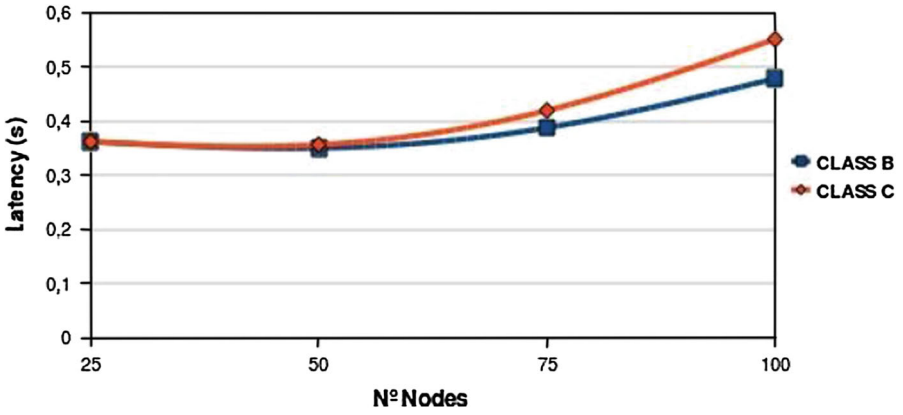

Fig. 3 Latency comparative

Figure 2 shows the message number exchanged during IP address configuration process. This one is reduced substantially in E-D2HCP according to the node number increases in the network. With the improvements in E-D2HCP a 25 % reduction in overhead from 150 nodes with respect to overhead presented in D2HCP is achieved.

The protocol performance is worse when the percentage of assigned addresses increases. However, when the network consists of 240 nodes (95 %), the latency time is less than 1 s (see Fig. 1).

In Fig. 3 we analyzed the latency for two types of IPv4 address classes: Class B and Class C. We see that the latency for the assignment of class C address grows quicker than class B addresses. It is observed from 75 nodes.

These results indicate that the parameter which determines the latency, in general, is the occupied address percentage.

In the case of class C addresses, by having fewer addresses than class B, it cannot carry out a local assignment being necessary to request the address to another node, increasing the necessary time to complete the process.

In general, the average latency in address assignment process is low for both address classes in this scenario.

In Fig. 4 we show the average number of emitted control messages in each address configuration.

The E-D2HCP protocol presents a large reduction in the control packet overhead in the network.

In fact, in most cases, the configuration is realized in a local way, i.e., the neighbour will assign the address to a new node. In the occasion where it couldn't assign addresses locally, a unicast transmission with chosen server is realised.

The message average number exchanged is very similar for both IPv4 address classes. From 60 nodes class C has a slight increase in the average number of messages with respect to class B.

Finally, in Figs. 5 and 6 we show the overhead in the packet and Bytes number.

Figure 5 shows that the overhead in packet number is very similar up to 75 nodes for both address classes. From 75 nodes it increases the received packets without appreciate difference in the two classes of analyzed networks.

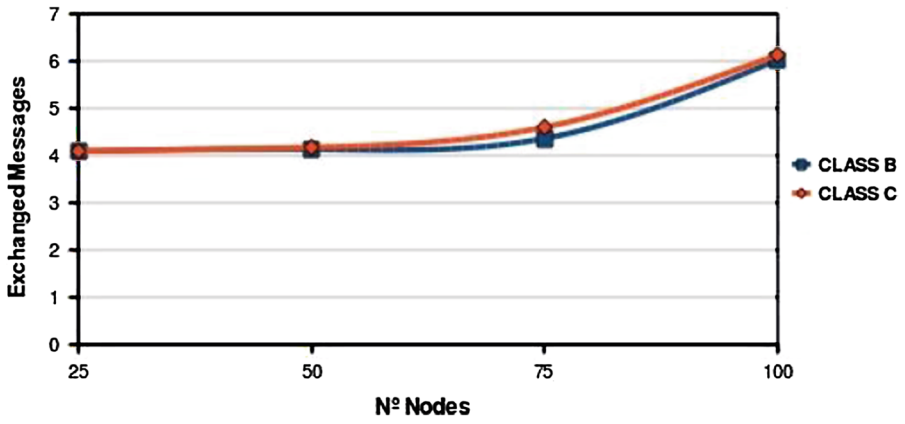

Fig. 4 Exchanged messages

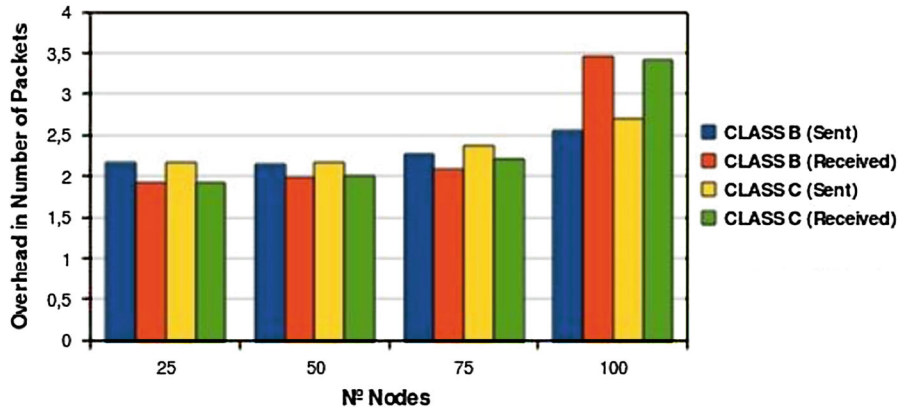

Fig. 5 Overhead in number of packets

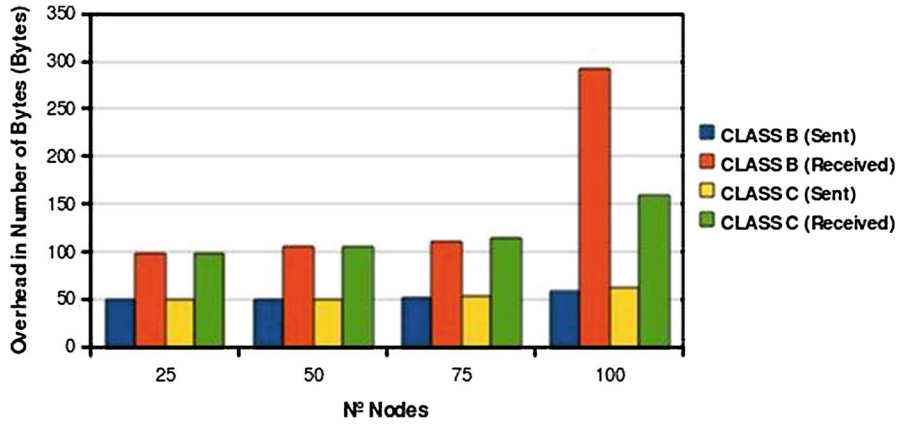

Fig. 6 Overhead in number of bytes

In Fig. 6 we have the overhead in number of Bytes. The graph shows a similar pattern for both classes except at 100 nodes where the received bytes are double for class B than for class C because there's more possibility of addressing.

6 Conclusions and future work

The E-D2HCP protocol is a stateful approach and therefore it does not require DAD, which reduces the overheads. Additionally, monitoring the routing protocol simplifies the synchronization process. The latency depends on the address space size. If the percentage of allocated addresses is small, the chances of finding a server with available addresses increase. In this case, the latency is small.

E-D2HCP is an extension of D2HCP protocol. Among the improvements which it contemplates, the following exist:

- Network Merging.
- Support for the failure of client-server links during the configuration process.
- Improvements in the network initiation process achieving node synchronization which they wish to initiate the network concurrently.
- Improvement in latency.

Topics for future research include the development of QoS and security extensions.

Acknowledgments This work was supported by the Agencia Española de Cooperación Internacional para el Desarrollo (AECID, Spain) through Acción Integrada MAEC-AECID MEDITERRÁNEO A1/037528/11. This work was also supported by the Security Engineering Research Center, granted by the Ministry of Knowledge Economy (MKE, Korea).

References

1. Baccelli E (2008) Address autoconfiguration for MANET: terminology and problem statement. Internet Draft draft-ietf-autoconf-statement-04, Internet Engineering Task Force. http://www.tools.ietf.org/html/draft-ietf-autoconf-statement-04
2. Bernardos C, Calderon M, Moustafa H (2008) Ad-Hoc IP autoconfiguration solution space analysis. Internet Draft draft-bernardos-autoconf-solution-space-02, Internet Engineering Task Force. http://www.tools.ietf.org/html/draft-bernardos-autoconf-solution-space-02
3. Bernardos C, Calderon M, Moustafa H (2008) Evaluation considerations for IP autoconfiguration mechanisms in MANETs. Internet Draft draft-bernardos-autoconf-evaluation-considerations-03, Internet Engineering Task Force. https://www.datatracker.ietf.org/doc/draft-bernardos-autoconf-evaluation-considerations/
4. Bernardos C, Calderon M, Moustafa H (2008) Survey of IP address autoconfiguration mechanisms for MANETs. Internet Draft draft-bernardos-manet-autoconf-survey-04, Internet Engineering Task Force. http://www.tools.ietf.org/html/draft-bernardos-manet-autoconf-survey-04
5. Cheshire S, Aboba B, Guttman E (2005) Dynamic Configuration of IPv4 Link-Local Addresses. RFC 3927, Internet Engineering Task Force. http://www.ietf.org/rfc/rfc3927
6. Clausen TPJ (2003) Optimized link State routing protocol (OLSR). RFC 3626, Internet Engineering Task Force. http://www.ietf.org/rfc/rfc3626
7. Corson S, Macker J (1999) Mobile Ad Hoc Networking (MANET): routing protocol performance issues and evaluation considerations. RFC 2501, Internet Engineering Task Force. http://www.ietf.org/rfc/rfc2501
8. Droms R (1997) Dynamic host configuration protocol. RFC 2131, Internet Engineering Task Force. http://www.ietf.org/rfc/rfc2131

9. Mohsin M, Prakash R (2002) IP address assignment in a Mobile Ad Hoc Network. In: Proceedings of the Century Military Communications Conference, vol 2. Anaheim, California, USA, pp 856–861. doi:10.1109/MILCOM.2002.1179586
10. Nesargi S, Prakash R (2002) MANETconf: configuration of hosts in a Mobile Ad Hoc Network. In: Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies, vol 2. New York, USA, pp 1059–1068. doi:10.1109/INFCOM.2002.1019354
11. Perkins CE, Malinen JT, Wakikawa R, Belding-Royer EM, Sun Y (2001) IP address autoconfiguration for Ad Hoc Networks. Internet Draft draft-perkins-manet-autoconf-01, Internet Engineering Task Force. http://www.tools.ietf.org/html/draft-perkins-manet-autoconf-01
12. The NS-3 Network Simulator. http://www.nsnam.org/
13. Thomson S, Narten T, Jinmei T (2007) IPv6 Stateless Address Autoconfiguration. RFC 4862, Internet Engineering Task Force. http://www.ietf.org/rfc/rfc4862
14. Thoppian MR, Prakash R (2006) A distributed protocol for dynamic address assignment in Mobile Ad Hoc Networks. IEEE Trans Mobile Comput 5(1):4–19. doi:10.1109/TMC.2006.2
15. Vaidya NH (2002) Weak duplicate address detection in Mobile Ad Hoc Networks. In: Proceedings of the 3rd International Symposium on Mobile Ad Hoc Networking & Computing. Lausanne, Switzerland, pp 206–216. doi:10.1145/513800.513826. http://doi.acm.org/10.1145/513800.513826
16. Villalba LJG, Matesanz JG, Orozco ALS, Díaz JDM (2011) Auto-configuration protocols in Mobile Ad Hoc Networks. Sensors 11:3652–3666
17. Villalba LJG, Matesanz JG, Orozco ALS, Díaz JDM (2011) Distributed dynamic host configuration protocol (D2HCP). Sensors 11:4438–4461
18. Weniger K, Zitterbart M (2002) IPv6 autoconfiguration in large scale Mobile Ad-Hoc Networks. In: Proceedings of the European Wireless Conference. Florence, Italy, pp 142–148