Received April 2, 2019, accepted April 29, 2019, date of publication May 10, 2019, date of current version May 22, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2916175

Neighbor Discovery ++– a Scalable and Robust Address Auto-Configuration for Future Internet of Things Networks

MONIKA GRAJZER[1,2] AND MARIUSZ GŁĄBOWSKI[1]
[1]Faculty of Electronics and Telecommunications, Poznań University of Technology, 60-965 Poznan, Poland
[2]Gido Labs sp. z o.o., 61-371 Poznan, Poland

Corresponding author: Monika Grajzer (monika.grajzer@gidolabs.eu)

This work was supported by the Polish Ministry of Science and Higher Education Within the Status Activity Task ''Struktura, analiza i projektowanie nowoczesnych systemów komutacyjnych i sieci telekomunikacyjnych'' in 2019 under Grant 08/82/SBAD/8229. The work of M. Grajzer was supported by the European Union under the European Social Fund within the project ''Scholarship Support for Ph.D. Students Specializing in Majors Strategic for Wielkopolska's Development'', Sub-measure 8.2.2 Human Capital Operational Programme.

ABSTRACT The growing complexity of the Internet of Things (IoT) solutions along with highly increased application needs call for more complex and robust networking solutions. Especially the importance of autonomic, self-configuration capabilities becomes particularly significant. The IPv6-based mobile ad hoc networks are envisioned to be a good candidate technology for the IoT networks. However, they lack efficient address auto-configuration mechanisms, which could extend the IPv6 to cover the requirements of such a demanding networking environment. To address this challenge, we have proposed the Neighbor Discovery ++ (ND++) solution for the enhanced stateless address auto-configuration. The ND++ incorporates a well-performing flooding control mechanism to the basic IPv6 ND design, which results in a very low protocol overhead in the order of few messages per node. The presented simulation-based evaluation, performed under diversified mobile scenarios, confirms the scalability and robustness of this approach with regard to the key identified metrics—reliability, overhead, and latency.

INDEX TERMS Neighbor Discovery++, ND++, address auto-configuration, stateless address auto-configuration, mobile ad hoc networks, IPv6, Neighbor Discovery, flooding optimization, multipoint relays.

I. INTRODUCTION

By observing current trends in the Internet-based technologies as well as predicted changes in the digital landscape, the growth of the Internet of Things (IoT) sector cannot be overlooked. Not only the prognosticated number of IoT connected devices is tremendous, reaching the estimated 50-70 billions by the year 2020 [1], [2], but also the variety of IoT devices is being increased. With the recent incorporation of beacons, wearables, smart watches, etc. into the former ecosystem of smart phones, tablets and sensors, the upcoming IoT offers more diversified capabilities. They are envisioned to support new solutions in advertising, entertainment, health and well-being monitoring systems, smart cities, smart homes and many more [3].

While the IoT applications become more and more sophisticated, they call for more complex and robust solutions from the network layer perspective [4]–[6]. Simple, hub-like, scenarios are often not enough to address their requirements. In such a scheme the core networks could be overloaded while dealing with billions of interconnected devices [3], [7]. In order to interconnect the devices in a flexible manner, IoT requires not only robust infrastructures, but also more autonomic networking functionalities that would take advantage of self-configuration, self-adaptation and self-management capabilities [4], [5], [8]. Autonomic networking should help achieve the ''plug-and-play'' functionality and enable a simplified, unsupervised network configuration and operation. This is considered particularly important for future IoT network designs [3], [5], [9], [10].

By taking into consideration the demanding requirements towards the IoT networks, we envision IPv6-based mobile ad hoc networks (MANETs) to be a good candidate technology. In addition, IPv6 and related protocols are going to form the basis for the Future Internet, part of which will be the Internet of Things. There are, however, still many challenges

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Alam.

that need to be addressed, since IPv6 is designed mainly for fixed networks environments. Mobile, ad hoc set-ups bring additional, often unresolved issues. They result from the lack of a pre-established infrastructure, dynamic network changes and strong reconfiguration needs. Moreover, autonomic (the so called self-*) behaviors are also mostly at the early stage of research and unsupervised network management still poses many challenges [3].

Especially the self-configuration functionality, along with network address management and address auto-configuration (AAC), calls for future research [11]. In the demanding environment of future networks, each node has to possess a unique IPv6 address, since duplications can be very harmful for the whole network configuration and operation. Moreover, the assignment of a new address should be possible without the need for manual configuration [12], [13] and, in particular, without access to a DHCP server or any other related technology [12]. Substantially, the Internet Engineering Task Force (IETF) [14], which standardizes IPv6, requires that each newly assigned address have to be verified [12]. This is strongly related to the recent common duplications between MAC interface cards addresses being traditionally used as a basis for creating unique node identifiers [15]–[17]. The reliability of the Duplicate Address Detection (DAD) procedure is thus of significant importance [3].

AAC in MANET networks is much more challenging than in the fixed networks environment due to possible node mobility, changing network topology being ad hoc in nature, lack of pre-defined topology and network-wide initial configuration information. For fixed networks, for which IPv6 is mainly targeted, it is sufficient to verify address uniqueness only among the nodes' direct neighbors. Broadcasting in MANETs poses more challenging problems as possible duplications between more distant nodes also need to be detected and DAD should span across the whole network domain [11]. Hence, there is a need for an AAC methodology which would allow to assign unique IPv6 addresses to the nodes in the presence of very limited information about network configuration state or even without it. Under such conditions stateless AAC solutions, similar to the one proposed for fixed networks as a part of the IPv6 Neighbor Discovery (ND) protocol [12], [18], are of increased importance. Therefore, new control and configuration solutions are needed to extend IPv6 [19] in this regard.

Current AAC solutions for MANETs, especially the stateless AAC approaches [16], [20]–[23], to some extent rely on unrestricted flooding of protocol messages and, therefore, lack adequate robustness and efficiency. Their limitations are particularly visible for large-scale networks, where the protocol overhead issues become a substantial factor influencing the whole network operation.

In order to overcome these issues, we propose an extension to ND protocol – the Neighbor Discovery ++ (ND++) [24]–[27]. Through this approach we have managed to incorporate one of the well-performing flooding control mechanisms based on the Multipoint Relay (MPR) concept [28] into the DAD process. This has made it possible to restrict the number of necessary message forwarding events during flooding and, as a result, to come up with a new, low-overhead stateless AAC mechanism. To the best of our knowledge, this is the only solution proposed so far which includes flooding control method available for use before addresses are fully assigned and verified in the network. Substantially, it is also independent of a routing protocol and available before routing is fully operational.

In our previous works on this topic [24]–[27], [29]–[31], we proposed initial ND++ set-up and performed baseline evaluation of ND++. We presented a proof-of-concept evaluation of ND++ in the real-world testbed platform [25] and followed with an initial simulation-based evaluation [26], [27], [29]–[31]. All scenarios investigated to date considered a situation where a single node was starting DAD for a new IPv6 address in static MANET network set-ups (without mobility). In this article, we present the most recent novelties introduced to the ND++ protocol design that allow fully mobile environments to be properly addressed – in particular, the updated algorithmic procedures of the flooding control mechanism, which are highlighted in Section IV-C, and the updated format of some of the proposed ND++ messages (as depicted in Section IV-A). We also describe in detail the final protocol functionality, which was not presented in such a comprehensive manner in our previous works. Moreover, we present the newest evaluation results of the proposed ND++ solution in the scenarios with node mobility, which was not performed before. It was executed under diversified MANET environment conditions, including those of variable load in terms of a number of simultaneous address requests. The focus of this most recent research, that we present in this paper, was to evaluate protocol scalability, reliability (expressed as a probability of successful detection of address duplications) and performance identified as the message overhead as well as protocol latency. The results reveal that ND++ is a scalable and robust solution reaching top reliability performance not only in the ideal, but also in realistic channel conditions, with very low protocol overhead in the order of just a few messages per node.

This article is organized as follows: Section II presents the related work regarding stateful, stateless and hybrid AAC mechanisms, Section III specifies ND++ design objectives, describes key protocol mechanisms and features and presents a qualitative comparison of ND++ with the most relevant related works, while Section IV provides a detailed protocol description. The details of the simulation-based evaluation are given in Section V, including the specified simulation set-up, parameters and evaluation criteria. Section VI presents the obtained results and the ND++ performance in diversified set-ups with regard to the identified metrics as well as provides a quantitative comparison with other solutions. Finally, Section VII concludes the paper.

II. RELATED WORK

Address auto-configuration is an essential part of the node configuration process that is typically performed at the initial stage of its operation. Substantially, unique addresses have to be assigned in a MANET network before routing is fully operational. This process has to be performed even without access to external configuration information. In order to deal with such circumstances, several approaches can be identified. In general, AAC can be organized in the stateful or stateless manner. The proposed ND++ solution is a stateless AAC approach. Hence, the stateless methods are the key reference works, though some of the stateful and hybrid mechanisms should also be taken into consideration, since they attempt to achieve the same goal, i.e., provide address auto-configuration to a MANET network without external configuration servers, in a distributed manner. Below, we present an overview of the key AAC solutions [3].

A. STATEFUL ADDRESS AUTO-CONFIGURATION

Stateful AAC requires the presence of designated servers being capable of configuring other network nodes. It strongly relies on a pre-shared knowledge about network state – in particular, on previously assigned addresses. In IPv6-based fixed networks, the DHCPv6 protocol [13] is an example of such an approach. Its applicability to MANETs is, however, very limited – typically, the existence of a connection from each node to the central server cannot be assumed. There have been several attempts to overcome this drawback and apply stateful mechanisms to MANET networks [15], [32]–[35]. These attempts mostly rely on managing and distributing a unique, non-overlapping address pools (or addresses) to the designated nodes in a decentralized manner. In these solutions, the new node usually sends an address query to its direct neighbors. Usually one of them is capable of assigning an address or providing information about the available resources [3].

One of the key reference solutions from this group is MANETConf [15]. It requires that each network node keeps the list of all addresses assigned in the network thus far. A new node joining the network obtains the address from one of its neighbors, which becomes a kind of a ''proxy'' and manages communication with the other nodes with regard to the new address of interest. This neighbor node floods the address request to the other nodes in order to identify if it indeed can be used. Only after a positive reply is received from all other known network nodes, the address can be assigned to the new node. After the address assignment phase is finished, the information about newly utilized IP address is again flooded through the network. The disadvantages of this approach are the need for storing in each node possibly high amount of data about the network state and very high protocol overhead, which results from the active participation of all the nodes not only in message relaying but also in generating address replies [3].

The solution proposed by Fernandes *et al.* [34] overcomes some drawbacks of MANETConf and instead of sharing full information about allocated addresses proposes to share a filter which represents the current set of allocated addresses. However, flooding is still necessary to announce the newly utilized address and to detect potential duplications resulting from simultaneous queries for similar address (although reduced). Moreover, in this approach flooding the entire network is necessary also when the node leaves the network – to release the address resources it was using [3].

Gammar *et al.* [35] proposed to choose in a distributed manner the specialized ''agent'' nodes which would be allowed to assign addresses to the other nodes from the disjoint address pools – an initial address pool of an agent is split when a new agent is nominated. The agents advertise themselves in the link-local scope, while the non-agent nodes store information about nearby agents and periodically check their availability. This approach requires additional actions dedicated to address space management, including dealing with exhausted address pools and address recovery. All those additional operations can introduce high overhead, since many of them require flooding the whole network. Substantially, in the worst case flooding-based communication can be initiated even by all the agents in the network in response to a single address recovery query [3].

In general, stateful AAC methods often require that the messages are exchanged only in the link-local scope (i.e., within 1-hop) in order to assign an address to the new network node. However, they still demand flooding-based communication among all network nodes to establish node hierarchy or update information about allocated addresses (additionally, storing this information may imply large memory usage). There are no flooding control methods used, except for limiting the number of copies of each message during message forwarding. Moreover, in case of statefulAAC, the issues arise related to address space management – there might exist nodes with exhausted address pools which cannot configure the newly established network nodes any more. Furthermore, it is necessary to additionally manage addresses when the nodes are leaving the network in order to ensure that the freed address resources could be reused in the future. Such an operation usually requires network-wide flooding [3].

B. STATELESS ADDRESS AUTO-CONFIGURATION

IETF RFCs assume that each node in the IPv6 network can be configured in the stateless manner [12], [18]. Stateless AAC caters to the need for fast and simplified network configuration in case there is no access to the stateful DHCP-like services. In the IPv6-based stateless AAC there is no shared network state information and no entity which could steer the configuration of the new network nodes. Hence, a node chooses and configures an IPv6 address on its own and then verifies its uniqueness via the Duplicate Address Detection procedure [12], [18]. DAD is performed by sending an address query in a pre-defined range (scope). If no reply is received within a specified timeout, the address is considered valid and can be used by a given interface [3].

In the fixed networks, stateless AAC is performed by means of the Neighbor Discovery protocol [12], [18] in the procedure of Duplicate Address Detection. However, the proposed IETF standards [12], [18] focus only on the configuration of hosts within the link-local scope, which is limited to directly reachable neighbors (forwarding is not allowed). DAD is, thus, performed by sending a Neighbor Solicitation (NS) message with a query for the new address, which is transmitted within a 1-hop scope of the sending node (i.e., among its direct neighbors, it cannot be forwarded). If this address is already used by another direct neighbor, it replies with a Neighbor Advertisement (NA) message. In the case where the originating node receives a reply within a specified timeout, the address is identified as duplicated and cannot be used. A new address is chosen and the procedure is repeated. Otherwise, the address is considered unique and AAC is finished. Essentially, by design, the basic ND specification cannot detect more distant address duplications. While this is sufficient in the fixed networks, such an approach cannot serve MANET scenarios where each node is considered to be both host and router. Hence, enhanced stateless AAC mechanisms are necessary to enable the configuration of unique IPv6 addresses in the whole MANET domain [3].

With regard to the solutions proposed to address the challenges and requirements of MANET set-ups, the baseline and initial approach is the so called *strong DAD* proposed by Perkins *et al.* [20]. It performs the DAD steps specified above while using multicast, multihop address request messages (AREQ – corresponding to NS) and address replies (AREP – corresponding to NA). These are modified ND protocol messages with extended range (scope) of n hops which can be forwarded. In this approach, each node initially assigns two addresses – a temporary one, used only during DAD, and the target address. AREQs are flooded through the network, while AREPs are to be sent to the unicast temporary address (while sending AREQs intermediate nodes store the route information which is exploited while forwarding back AREPs). The probability of a temporary address duplication is assumed to be small due to its short lifetime, though this feature opens a possibility of improper protocol functionality in the case when the address turns out to be invalid. Moreover, this approach offers no overhead control, and thus its authors propose the recommended values of protocol parameters (such as the *HopLimit* specifying protocol's range and node timeout to wait for an address). They are specified so that it would be possible to reach all network nodes, while avoiding flooding storm and protocol misconfiguration. These are, however, based on some assumptions on network size, and thus may not be applicable to diversified range of MANET scenarios [3].

A similar approach, based on the use of currently deprecated site-local addresses [36], was proposed by Park *et al.* [23]. In this proposal the range of ND messages was extended by sending multihop protocol messages to the all-nodes-multicast address instead of the solicited-node-multicast address with a link-local scope (as initially specified in [12], [18]). There is no flooding optimization, however, it is envisioned that each node would send only one copy of each address request, thus permitting broadcast storm [3].

The *weak DAD* [37] approach was proposed mainly in order to overcome the need for AAC message flooding and related protocol overhead in the *strong DAD*. It does not verify the uniqueness of each address in the way described above for DAD, but, instead, ensures that routing is correct even in the case of address duplications. It is achieved by distributing unique keys together with routing protocol messages. This implies necessary changes in the routing protocol itself. There have also been *passive DAD* approaches (e.g., [38]) which, instead of verifying each of the new addresses, infer existing duplications through the observations of network behavior. These are strongly related to a particular choice of the routing protocol and require interactions with routing information and processes [3].

Stateless approaches often require fewer messages to be flooded across the network in order to reach AAC goals than the stateful ones, however their limitations are in an unoptimized flooding and in protocol configuration that often requires manual set-up of the protocol's range and timeout values. These are strongly related to network size. Overestimating protocol range may lead to a broadcast storm in the case where no control is taken over message flooding (e.g., in [20]) [3].

C. HYBRID AND OTHER APPROACHES

In order to deal with a significant overhead imposed in the case of stateless methods, in particular *strong DAD*, a hierarchical solution has been proposed by Weniger and Zitterbart [21], [22]. It aims at partitioning the network into smaller sets which would be configured by the leader nodes. These are to be selected in the distributed manner. Although the flooding in such a case is limited to the smaller range of several hops (r-hops scope), there is additional cost incorporated, which is necessary to establish and maintain the leader nodes hierarchy. Hence, the periodic flooding of control traffic by each leader node is required in the scope of r hops, which has to be performed even when no new addresses are being verified. Moreover, the leader nodes assign subnet IDs that have to be unique in the entire MANET network. This demands another control traffic to be flooded across the network. The authors propose [21] that the flooding is limited to one copy of each packet to be forwarded by each node. This is done by exploiting additional random source ID. In this approach, it is preferable that nodes move in logical groups [21]. Moreover, the authors seem to assume that leaders are uniformly distributed over some network operation area [22], which cannot be ensured in many network topologies or during the network bootstrapping phase. This approach is likely to be more beneficial for very big ad hoc networks, however for networks with the size of 10,000 nodes or more the necessary network-wide, unoptimized flooding would impose significant overhead and, in fact, could even lead to a broadcast storm [3].

Wang and Qian [16] propose a hybrid methodology which combines the centralized and distributed AAC approaches. This method uses a pre-selected, hierarchical structure of cluster heads among which the address pools are assigned by a central node in the centralized manner, while the nodes within the cluster are configured using a distributed approach (in a stateful manner). The proposed algorithm allows the minimized set of cluster heads to be selected. They are chosen based on a periodical exchange of messages within a link-local scope. The cluster head selection gives priority to new nodes with a significant number of new neighbors, hence the structure is likely to become increasingly unstable during the network lifetime. The central node is elected based on a periodic flooding of control messages across the entire network. In the case where a cluster head does not receive any message from the central node within a specified timeout, it starts to advertise itself as a central node candidate. The central node is then selected through a comparison of all the advertisements from other cluster heads. During flooding only messages from the cluster heads with the highest priority received so far are forwarded further, which limits the flooding storm. Once the central node has been elected, it floods periodically its credentials. This process is strongly dependent on pre-configured timing, which relies on several factors such as physical node capabilities, expected transmission delay and, in particular, network size. Hence, it may be difficult to configure a proper timer value for diversified practical implementations.

During the address allocation process the communication with central node and among cluster heads is assumed to be unicast. However, that requires a properly established routing. The central node manages the allocation of IPv6 addresses and related address pools to the cluster heads, while cluster members query their cluster heads for an address of interest (by sending message to the 1-hop scope).

Although the presented scheme proposes an interesting approach to hierarchical and distributed AAC, it may be vulnerable to link failures and node mobility occurring before the structure of cluster heads has been established. During topological changes in the network and in the presence of more severe message delays and drops, the whole network could be reconfigured. Although this would not introduce significant additional overhead, frequent changes of IPv6 addresses may corrupt higher-level communication sessions. Moreover, since unicast routing-based communication is necessary, this protocol may not be operational during initial, abrupt, initialization of a network, when many nodes arise at the same time and routing is likely not be set-up yet (due to the lack of confirmed unique node identifiers) [3].

Hussain *et al.* [39] propose SAAMAN – a location-based AAC method exploiting evenly-distributed structure of Duplicate IP address Detection Servers (DDS). The whole approach is based on the assumption that network nodes know their geographic positions (have inbuilt GPS) and thus geographic forwarding can be used to limit flooding overhead. However, in many MANET set-ups this may be impossible to be executed in practice. Moreover, the presented results [39] show that in more sparse networks the geographic forwarding may more often fail, leading to increased packet loss. All those features highly limit the practical applicability of this solution [3].

Among the reviewed previous works, there also were attempts to shift the paradigm and include stateless AAC functionality, in particular DAD, in some of the routing protocols. The works of Boudjit *et al.* [40] and Boudjit [41] address the usage of OLSR, which is particularly interesting from the perspective of the work presented in this article. The OLSR protocol chooses MPR nodes which are taking part in forwarding of multihop messages. The main challenge in using OLSR also for AAC purposes lies in handling MPR selection in the presence of inaccurate IPv6 address information – before DAD is finished, addresses cannot be assumed unique. In the earlier work of Boudjit *et al.* [40], the authors propose to resolve this issue by the specification of additional forwarding rules which diminish forwarding performance achieved by MPRs. Furthermore, the rules do not allow proper MPR selection to be executed in the occurrence of more distant address duplications between more than one pair of nodes simultaneously. This aspect decreases reliability of this solution. The later work by Boudjit [41] proposes another approach in which the above issue is able to be handled. This is, however, performed at the cost of additional message forwarding being necessary in the 1-hop scope of each node which decreases flooding optimization performance achieved in the MPR-based flooding. Furthermore, the proposed solution may likely fail in the presence of higher node mobility. In the ND++ solution proposed in this paper the above-mentioned challenge is handled in a more efficient way which ensures proper functionality during node mobility. Substantially, it is also argued [19], [21], [23], [33] that routing and AAC should be independent processes and we follow this approach. Hence, the applicability of the methods [40], [41] is very limited and they are not good candidates for a general-purpose AAC solution [3].

D. LIMITATIONS OF CURRENT PRACTICES

Looking at the initial motivation to provide MANET networks with AAC services in order to configure a network without access to DHCP servers, stateless protocols are closer to the solutions proposed for fixed networks as a part of ND protocol [12], [18]. However, the methods developed so far to address the above problem in a ''purely'' stateless manner (e.g., [20], [23]) did not manage to solve the flooding issue and, hence, were characterized by a very high protocol overhead making their practical realization doubtful. Moreover, restricting flooded area to several hops ([21], [22]) have not brought any significant improvement. On the other hand, stateful and hybrid approaches seem to avoid high amount of flooding when a new node is making an address query, but this incurs inevitable cost of maintaining the distributed "configuration agents" structure and of managing address pools. Hence, in most of the AAC solutions, flooding the entire network is necessary at some configuration step (e.g., [15], [16], [20]–[23], [34], [35]). A similar situation is observed for the solution proposed by Weniger and Zitterbart [21], [22], where stateless DAD flooding is performed within a limited scope, but the overhead imposed by the leader nodes hierarchy is significant. Surprisingly, the results presented in [34] show that the overhead of many stateful protocols is worse than in the case of *strong DAD* stateless approach [3], [20].

The methods that do not require network flooding (e.g., [37]–[39]) are often not performing DAD *per se* ([37], [38]), i.e., by triggering AAC processes, and thus their reliability in terms of the probability of successful duplication detection is questionable. Furthermore, the approach by Hussain *et al.* [39] requires additional equipment, such as a GPS system, and special routing protocols for the nodes to provide basic AAC services. Since the key assumptions of these solutions are significantly different than those employed for the other stateless and stateful AAC methods, including ND++, we will not focus on the comparison to these methods.

Notwithstanding flooding being the key aspect limiting AAC performance, flooding optimization was not deployed in most cases. This is a challenging task, since before AAC is finished in the network, network-wide node identifiers are not yet confirmed and verified for uniqueness. Hence, many flooding control mechanisms, such as, e.g., Connected Dominating Set flooding or tree-based techniques [42]), cannot be used at this stage either or become inefficient. Although the authors of [40], [43] have tried to incorporate MPR-based flooding, they have included the AAC functionality as a part of the OLSR routing protocol. This implies that only this (modified) routing protocol could be used in the network. However, it is claimed that routing and AAC should be independent [3], [19], [21], [23], [33].

III. OVERVIEW OF KEY ND++ FUNCTIONALITY

Considering the limitations of related works, it can be observed that, to the best of our knowledge, there are no solutions which would extend the basic IPv6 Neighbor Discovery protocol and in the meantime achieve efficient AAC in MANET networks. Hence, AAC and IPv6 cannot be easily integrated. Stateless methods seem to be the most straightforward realization of such an integration. However, the key obstacle in this context is a relatively high protocol overhead. It is imposed as long as flooding is not restricted since network-wide flooding is required throughout the AAC process. Moreover, stateless mechanisms require to deal with additional challenges related to protocol configuration – they are dependent on timers and a number of other protocol parameters, such as protocol range, which need to be properly set-up in order to provide flexible operation in diversified network environments and sizes.

To address the above challenges, we have proposed Neighbor Discovery++ [24]–[27] for the IPv6-based MANET networks. This solution not only extends the range of the ND protocol, making it capable of reaching all MANET nodes during AAC, but also focuses on flooding optimization. Moreover, a broad evaluation in diversified network set-ups has allowed to propose algorithmic procedures and particular protocol parameters set-up, which make it possible to cover diversified network environments without the need for frequent reconfiguration. In particular, while designing ND++ we have addressed the following objectives:

1) Efficient and scalable design of the AAC and, in particular, DAD techniques in IPv6-based MANET networks
2) Overhead reduction – ND++ aims for a significant reduction of protocol overhead by proposing flooding optimization method
3) Very high reliability also in realistic MANET environments, in the presence of message losses, etc.
4) Latency at the levels acceptable in practical set-ups
5) Small memory consumption
6) Easy integration with core IPv6 solutions, in particular Neighbor Discovery protocol
7) Node mobility support
8) Large-scale MANETs support
9) Independent from routing protocols

The proposed Neighbor Discovery++ is an extension to the IPv6 Neighbor Discovery protocol, being an IETF Draft Standard [18]. It aims to provide efficient stateless AAC to IPv6-based MANET networks. The ND++ solution is concentrated on the enhanced DAD. Hence, it enables to verify address uniqueness in the extended range covering the whole MANET domain. This is enabled by the optimized message flooding that exploits the MPR concept [28]. The algorithmic solutions designed for ND++ allowed for the efficient usage of a flooding control method with node identifiers (IPv6 addresses) confirmed only within 1-hop (direct) neighborhood of each node. In this way, ND++ can address the requirements of SAA in MANETs, including those of large scale networks, where node mobility as well as network merging and splitting can occur. The goal was also to incorporate the existing ND mechanisms into the proposed extension as seamlessly as possible, so that the nodes that do not recognize the new protocol version can still interact in a regular manner. In this way, backward compatibility with ND is ensured.

Following the key stateless AAC assumptions [12] and the approach presented by, e.g., Weniger and Zitterbart [21], we envision the entire ND++-based AAC procedure to be organized in the below steps:

1) the node creates a link-local IPv6 address from the randomly chosen identifier
2) the address is verified for uniqueness in the MANET-wide scope (enhanced DAD of ND++)
3) a prefix (with a MANET-wide or global scope), which is used in this network is assigned to this address.

ND++ functionality, similarly to ND and directions from [12], [18], is aimed for the realization of the second AAC step, i.e., the duplicate address detection. This is the main part of the process. It usually has to be combined with the method for configuring a routable address with a MANET-wide scope at the third AAC step. For this purpose, Perkins *et al.* in his initial work on strong DAD [20], proposes to select an address on the prefix specified particularly for MANETs – the *MANET_INITIAL_PREFIX*. However, the prefix can also be obtained by receiving a Router Advertisement message [18] from a designated router. Assignment of prefixes to this router is out of the scope of AAC (e.g., a Prefix Delegation mechanism [44], [45] could be used). Alternatively, the available prefix could be combined with a link-local interface identifier right at the beginning of node operation, and DAD would follow with an address created in this way. ND++ design is flexible and can be combined with different prefix management and assignment techniques, of which the ones presented above are most common and can serve a variety of MANET set-ups [3].

Since ND++ aims mainly at DAD, a number of different strategies can be deployed to support network merging. We propose to include additional logical layer on top of ND++ to handle the detection of possible merging events. This could be done within a dedicated management framework, similar to the one proposed by us for the auto-configuration needs in [46]. In this approach, after network merging, DAD should be triggered by each node in at least one of the previous sub-domains in order to verify if currently used IPv6 addresses are indeed unique. One possible solution for the detection of merging event is to monitor neighbor information collected by MPRs and trigger DAD once significant changes have been detected. For this purpose, the topology could be monitored in the scope wider than 2-hop neighborhood, e.g., by exploiting the Topology Control messages as proposed by us in [24]. We have presented an example of network merging properly handled with ND++ and the higher-level management logic in [25]. The ND++ evaluation presented in this paper suggests that an overhead arising from such a joint DAD initiation started by several nodes at a time will not lead to a broadcast storm. A detailed evaluation of network merging scenarios is, however, out of the scope of this article.

On the other hand, network partitioning scenarios do not require any additional action from the ND++ protocol – once a group of nodes separates from an initial network, network topology and neighbor relations change, but the nodes remain properly configured and both networks (“old” and “new”) can function separately. As opposed to stateful AAC approaches, there is no need to release address resources during network partitioning. Substantially, there is no cost involved in terms of overhead related to this operation [3].

In the remaining part of this article we will focus on the new DAD approach, which we introduce with ND++. We present below an overview of the key ND++ functionality, while protocol parameters, set-up and detailed algorithmic procedures will be elaborated in more detail in the following sections.

A. ENHANCED DUPLICATE ADDRESS DETECTION

In order to overcome the limitations of the basic ND protocol, discussed in Section II-B, and address challenges for the efficient AAC in MANETs, our proposed solution – ND++ – provides enhanced duplicate address detection (DAD++) functionality. This enhancement is a novelty introduced with ND++ and is realized twofold [3]:

- Firstly, the range of NS and NA messages is extended to cover the whole MANET domain. For this purpose, the new, modified multihop Neighbor Solicitation (mNS) and multihop Neighbor Advertisement (mNA) messages are exchanged throughout the DAD process. The extended capabilities of those messages are achieved by sending them to the all-nodes-link-local address [47] and specifying relatively large *HopLimit* field of an IPv6 header [48], indicating the number of hops the message can reach. Thus, mNS and mNA messages can be forwarded during DAD and can reach all network nodes.
- Secondly, in ND++ the DAD++ is performed in two steps: first in the range of 1-hop, similarly as in the standard ND protocol (1-hop DAD), then in the extended range covering a whole MANET domain (n-DAD). This two-stage approach enables to efficiently incorporate the MPR-based flooding, even in the presence of potentially duplicated node’s addresses within the network.

DAD++ would not be possible in a MANET network without one additional element – a random identifier, which is used to distinguish two nodes having duplicated IPv6 addresses and, in particular, to detect by the node the copies of its own messages forwarded back. Substantially, almost all of the AAC solutions proposed for MANET networks incorporate some notion of a random node identifier, in addition to the node’s IP address. This is demanded by the specific MANET communication model where nodes are able to receive “echos” of their own messages. However, as opposed to other solutions, we have proposed Random ID to be carried in the IPv6 Extension Headers [48] option of each ND++ packet. This enables its fast and simple processing in IPv6 networks [3].

Thanks to the above enhancements proposed by us to the ND-based DAD, ND++ offers efficient DAD++ which can serve the needs of ad hoc networks. An overview of DAD++ functionality is presented in Figure 1.

B. FLOODING OPTIMIZATION

Flooding optimization during AAC is not straightforward, since most flooding control mechanisms require basic information about network topology and neighbors to be present at the very initial stage (e.g., [49]–[51]). However, this type of information cannot be known at the time when AAC takes place due to the lack of confirmed unique node identifiers. Therefore, many flooding optimization techniques are not

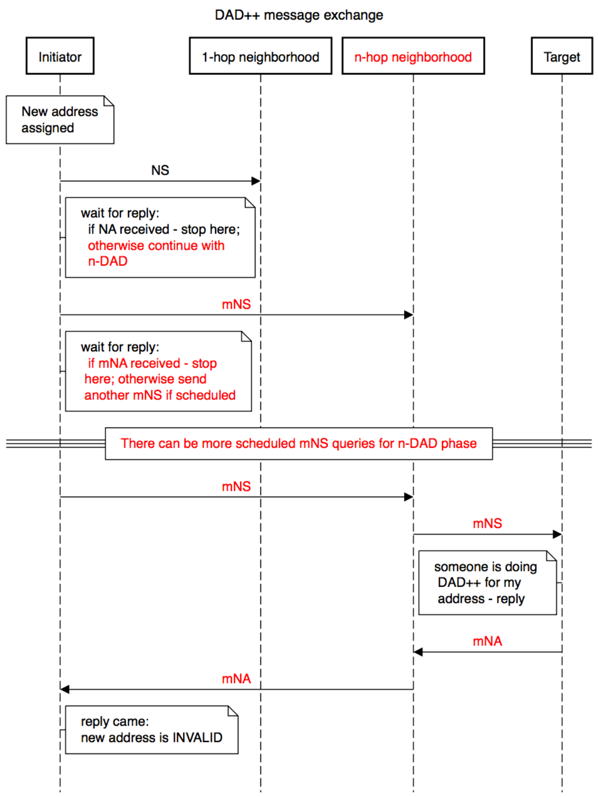

DAD++ message exchange
Initiator
1-hop neighborhood
n-hop neighborhood
Target
New address assigned
NS
wait for reply:
if NA received - stop here;
otherwise continue with n-DAD
mNS
wait for reply:
if mNA received - stop here; otherwise send another mNS if scheduled
There can be more scheduled mNS queries for n-DAD phase
mNS
mNS
someone is doing DAD++ for my address - reply
mNA
mNA
reply came:
new address is INVALID

FIGURE 1. Example of message exchange during DAD++ (Initiator - node starting DAD++; target - node which already uses the address being verified by the initiator). The messages and actions marked red depict the novelties introduced with ND++.

feasible in the considered case. A good candidate to solve this problem is the MPR algorithm [28], which requires the knowledge of only 2-hop neighborhood of each node and limits the exchange of necessary control messages to the link-local scope (i.e., 1-hop neighborhood). Additionally, it offers good results as compared to other flooding algorithms and exhaustive flooding [52]. Therefore, it was selected for flooding optimization in ND++ [3]. However, it has to be underlined that new algorithmic procedures were necessary in ND++ in order to exploit this flooding technique in AAC and DAD. Their design is a novelty introduced in this solution.

The key enabler for the MPR-based flooding is the division of AAC in ND++ into two steps: DAD and *n*-DAD, with the MPR selection being executed between those phases. In ND++, the first DAD step enables the uniqueness of the address to be confirmed among the node's direct neighbors. After this process is successfully finished, the node owns a valid IPv6 address. It can be used for the link-local communication with other nodes (previously, during DAD operations, NS messages were sent with the so-called "unspecified" source address [47]). Hence, at this point the node can advertise itself, and its known neighbors, in the direct neighborhood (1-hop scope). This process is essential in order to introduce the MPR-based flooding. It is performed by exploiting the new NA option proposed within ND++ – the MPR Parameters option, which is sent along with 1-hop NA messages. MPR Parameters options are exchanged periodically by each node and contain information similar to the ones collected by OLSR routing protocol for the need of MPR-based message flooding [28]. One exception is, however, that each advertised IPv6 address is complemented by Random ID of the node possessing it. The tuple of Random ID and node's main IPv6 address distinguish nodes, even in the presence of address duplications [3].

By collecting MPR Parameters options of NA messages received from its neighbors, each node gathers information about its 2-hop neighborhood. Based on these data each network node can choose MPRs out of its direct neighbors – nodes that will forward ND++ information on its behalf. The MPR selection algorithm in ND++ is based on the modified heuristic defined in [28]. Our modification enables to take into account not only IPv6 addresses, but also Random IDs and, thus, enables proper MPR selection even in the presence of address duplications. MPRs are chosen from 1-hop neighborhood of each node in such a way that they cover 2-hop neighborhood in full with a possibly small set of MPRs [3], [28].

Once the node has chosen its MPRs, it will advertise this selection during the next periodic exchange of control NA messages – by using the new MPR Announcement option proposed in ND++. Hence, from now on, periodic NA messages will contain both MPR Parameters and MPR Announcement options. Substantially, MPR announcement process does not introduce any additional message overhead in terms of the number of sent messages [3].

All nodes in the network receiving MPR options register the neighbors which have chosen them as their MPRs. Hence, each node maintains the list of its MPR Selectors (the MPR Selectors Set) and will forward the packets received from the nodes on this list. Thus, only selected network nodes participate in the flooding of ND++ messages during *n*-DAD phase. An example of a network in which a new node is performing AAC by using ND++ with MPR-based flooding optimization is presented in Figure 2 [3].

C. ND++ IN THE CONTEXT OF RELATED WORKS

In Table 1 we present a comparison of our ND++ solution with the most relevant related works. Due to many functional and structural differences in the key assumptions of the investigated methods, we have chosen a set of qualitative criteria, which best differentiate and characterize each of the selected AAC solutions.

Among the selected metrics the primary focus is on the network-wide flooding events, since this factor is directly translated into the amount of overhead costs incurred by each method. Owing to the fact that too high message overhead may corrupt the whole network and even disable any communication among nodes, we perceive this factor as the most influential. Another important metric is routing independence, since we focus on the AAC solutions which (1) do not require any particular routing protocol and (2) do not restrict

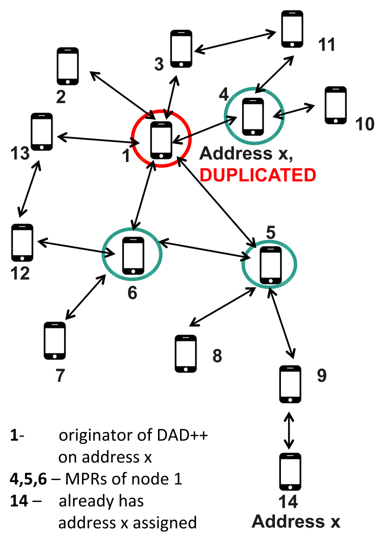


FIGURE 2. Example of scenario demonstrating ND++ functionality.

routing to be used in the network to any particular protocol(s). Routing independence also means that no restrictions are made on the new node arrival patterns and abrupt network initialization should be possible. Other features, although also important, are less decisive. In particular, the effect of delay resulting from high latency (expressed as the amount of time needed to resolve an address query) is not critical to network operation and its potentially negative impact is limited to the relatively short period when a node queries for a first address. Memory consumption may be important for some MANETs where the nodes are small devices with limited capabilities. Thus, storing network-wide configuration information should be avoided [3].

As compared to the most relevant related works, ND++ is characterized by the features which enable it to outperform the rest – not only in terms of flooding optimization capabilities and minimized overhead of additional message processing, but also from the perspective of practical applicability. This is a result of its independence of "super nodes" and external higher-level configuration as well as of particular routing protocol to be used in a network. Additionally, ND++ requires that each node stores only the information about its 2-hop neighborhood, which is not significant. This limits the necessary memory consumption, since the information about further nodes does not need to be collected, stored and processed. The features of ND++ reveal that it has a potential to offer the lowest overhead results, while still performing very well on other identified metrics (or at least acceptably in the case of latency) [3]. In Section VI-D we will present a more detailed discussion on quantitative overhead and latency evaluation.

IV. DETAILED ND++ DESCRIPTION

In this section we will present the details referring to the ND++ messages, changes incorporated to standard ND in our proposed solution as well as the algorithmic procedures related to DAD++ and MPR-based flooding. We will also elaborate on the ND++ parameters set-up, including the critical values of protocol's range and the exploited timers.

A. ND++ MESSAGES

The specification of ND++ messages is based on the extended ND message format. The newly proposed modifications introduce new sub-types of messages with multi-hop capabilities, new options supporting MPR processes and a new Hop-by-hop Extension Header option that is carried in

TABLE 1. Comparison of ND++ with other state of the art solutions according to the identified key metrics: (1) flooding needed, (2) flooding optimization, (3) node hierarchy, (4) overhead, (5) latency, (6) memory use, (7) routing dependency; (n/a - not applicable, n/d - not enough data).

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	Other remarks
MANETConf [15]	Yes	No	No	Very high	Moderate	High	No	several flooding events covering whole network necessary for a single new address query
Fernandes *et al.* [34]	Yes	No	No	High	Moderate	Low	No	flooding is also necessary while releasing address resources
Gammar *et al.* [35]	Yes	No	Yes	High	Low	Low-Moderate	No	possible issues with address space management and lack of free resources
Perkins *et al.* [20] ('strong DAD")	Yes	No	No	Very high	High	Low	No	no restrictions on forwarding copies of previous messages – broadcast storm likely
Park *et al.* [23]	Yes	No	No	High	High	Low	No	similar to Perkins *et al.*, but message forwarding limited to one copy of each message
Weniger *et al.* [21], [22]	Yes	No	Yes	Moderate	n/d	Low	No	establishing and maintaining leader's hierarchy requires network-wide flooding and periodic exchange of multihop control messages (with scope limited to approx. 3 hops)
Wang *et al.* [16]	Yes	No	Yes	Moderate	Low	Low	No (*)	link failures and node mobility may corrupt node hierarchy creation; possibility of frequent reconfiguration of node's addresses; depends on pre-configured routing for unicast multihop transmissions – problems with abrupt node initialization likely
Boudjit [41]	Yes	Yes	No	Low-Moderate	Moderate	Low	Yes	usage restricted to (modified) OLSR routing protocol; the proposed modifications may fail to support some network set-ups
ND++	Yes	Yes	No	Low	Moderate-High	Low	No	Minimizes number of necessary flooding events and optimizes flooding itself, incorporated into IPv6 ND

the IPv6 packet header. All those new features of ND++ are following the design paradigms specified for the IPv6 protocol stack [48], [53]. Since the ND++ is an extension to the basic ND, the core functionality of ND remains unchanged and must be supported to ensure proper interactions with nodes that do not support ND++ implementation. The flexibility of IPv6 design enables such coexistence and ensures backward compatibility.

For the needs of DAD++, ND++ exploits both 1-hop Neighbor Solicitation (NS) and Neighbor Advertisement (NA) messages from the standard ND specification [18] as well as their multihop counterparts – mNS and mNA messages being the new message sub-types proposed for ND++. The latter ones are differentiated by the Code field of ND (ICMPv6) header that is set to 1 instead of 0 in the case of multihop messages. In general, ND++ uses the following message types [3]:

- 1-hop NS messages: used to send address query in the link-local scope (1-hop DAD); the new address of interest is specified as a target address and the source address of this message is the unspecified address (the so-called "::" [47]); destination address is the solicited-node-link-local address
- 1-hop NA messages: used to send a reply to the 1-hop address query by the node which already has the target address assigned; the source address is the address of message originator
- Multihop NS (mNS) messages: used to send address query in the extended range of several hops (n-DAD); sent from the node's IPv6 address as a source address to the destination of all-nodes-link-local address (the source address is either the valid link-local address confirmed during 1-hop DAD or another node's IPv6 address validated earlier); the source address is not the unspecified address – to allow proper MPR-based message flooding
- Multihop NA (mNA): used to reply to the multihop address query in the case the mNS target address matches one of node's IPv6 addresses (n-DAD); the source address is the address of message originator, the destination address is the all-nodes-link-local-address

There are also new option types proposed in ND++ to be sent periodically with 1-hop NA messages (i.e., they cannot be forwarded). They are introduced to enable (1) the MPR selection, (2) the usage of MPRs throughout the process and (3) information sharing among direct neighbors [3]:

- MPR Parameters option – used by each node in a network to advertise its neighbors and its forwarding capabilities. It specifies willingness of a node to act as an MPR for other nodes and lists node's 1-hop neighbors along with the Link Code tuple containing the information about their type (Neighbor Type) and the type of link (Link Type) to these neighbors. The option is based on the functionality from [28], however modifications were proposed to the message structure. In ND++ multiple options are used to share information about neighbors with different Link Codes. Link Code for each neighbor is set according to the rules specified in [28].
- MPR Announcement option – proposed for announcing which nodes have been chosen to act as an MPR for a certain MPR Selector (message originator).

We have proposed a detailed message format in [24], and we refer the reader to this work for further details. However, in order to incorporate our latest improvements in the MPR selection algorithm, we propose here to extend the format of MPR Parameters and MPR Announcement options in order to encapsulate also Random IDs along with IPv6 addresses. The padding may be necessary at the end of the option payload to conform to the rules specified in [18].

B. RANDOM ID OPTION

The Random ID is an essential identifier to be carried by each packet sent by a node. The tuple of Random ID and node's main IPv6 address is necessary in MANETs to allow each node to distinguish copies of own messages which were forwarded back to the message originator. We also exploit it to support the MPR selection procedure during the network bootstrapping phase. Therefore, Random ID Option is proposed that is attached to the Hop-by-hop Extension Header (included in the IPv6 packet header) [24], [25]. It contains the 32-bit Random ID, which is supposed to be a randomly generated number. Care should be taken when selecting a seed for the Random Numbers Generator (RNG), which should allow to decorrelate random selections made by each network node (e.g., timestamp, host ID, device serial number and similar data could be taken as a starting point for generating a Random ID). We envision the probability of two nodes having the same 32-bit Random ID and 64-bit interface identifier to be negligible [3] (typically IPv6 address is composed of a 64-bit prefix similar for all MANET nodes and a 64-bit interface identifier). The theoretical evaluation of this situation can be derived from the known birthday paradox [54]. It depicts the probability that there will be at least one address duplication among k network nodes with addresses randomly generated (with a uniform distribution) from the address space of size n. By applying this concept, we evaluate the probability of duplications among the tuples of a 64-bit identifier and a 32-bit Random ID to be in the order of 10^{-28} for $k = 5$ nodes and 10^{-22} for $k = 10.000$ nodes. With the decrease of IPv6 address space this probability increases to at most 1.78E-07 for a 16-bit identifier and 10.000 nodes, which is still a negligibly small value [3].

Random ID Option also contains the sequence number field with the message Sequence Number (SN) – an integer incremented with each message sent by a node, which is useful for the detection of copies of previously sent messages.

C. FORWARDING RULES

As mNS and mNA messages traverse across the network, the following forwarding rules were specified in ND++ for these messages to be executed by each node:

1) if the sender address is in the MPR Selectors set – consider for forwarding, otherwise – do not forward the packet but process it in this node;
2) if considered for forwarding in step 1): verify Forwarding Tuple for this message: {*target address, Random ID, packet sequence number*} – if the tuple is not found – forward message with a source address set to this node's main address and process the packet, otherwise – drop the packet (do not process, since one copy of this message was already read by this node);

During forwarding the source address field of IPv6 header is modified to the address of a forwarder. This is necessary to allow the other nodes to distinguish which messages were forwarded by their MPR Selectors. For the same reason mNS message is sent from its originating node not with the unspecified address, as in the case of 1-hop DAD, but from one of the already confirmed IPv6 addresses of this node. Moreover, the MPR Selectors set always contains the unspecified address to accompany for some possible situations when the intermediate forwarding node does not have a valid IPv6 address yet – it will then forward the packet with the unspecified address. This may happen, e.g., when a node has successfully finished 1-hop DAD for its new (and only) IPv6 address, starts the exchange of MPR-related information, but then receives a negative reply during *n*-DAD and has to mark the previously announced address as invalid. As a result, it does not have any valid address which could be used to forward the received message, since it has not managed to finish DAD for the next address it has chosen. The solution presented above with regard to the sender address selection enables such a situation to be properly handled and to keep up an MPR-based flooding even in the case of abrupt (simultaneous) network initialization when a large number of duplicated nodes join a network at the same time and frequent address changes may occur [3]. This behavior is the most recent functionality of ND++, which was not presented before.

In ND++ we use Forwarding Tuple based on a target address and originator's Random ID, since the originator's source address is mostly not known and is considered not ready for network-wide usage during address auto-configuration. In this approach, two messages that have originated from the same source and reached the current MPR via two different paths are treated as duplicated and only one of them will be forwarded further [3]. Moreover, each forwarder is also an envisioned recipient of the message, so the copy of the message is forwarded, but the message itself is processed by the receiving node as well.

D. DAD++ IN DEMANDING MANET ENVIRONMENTS

In the forwarding policy presented above a network node forwards only one copy of each message and the forwarding rules are set so that the number of forwarded messages is minimized. However, in the course of our previous research [30], [31] it turned out that the probability of successful duplication detection was decreased under more demanding channel environments – the levels of 100% achievable for ideal channel could not be met. Hence, we have investigated several remedy approaches to increase ND++ reliability. Our detailed investigations [30], [31] reveal that the most beneficial solution is to follow the initial 1-hop DAD stage by cyclic *n*-DAD queries repeated for a specified amount of time, instead of just a single *n*-DAD query (please refer to Figure 1). The additional advantage of such an approach is that some of the scheduled trials of *n*-DAD are neglected in case a positive reply to DAD++ query is received (i.e., a duplication is found).

Following the above approach, it is also necessary to specify a policy regarding the mNAs sent with a reply to DAD++ address query. Therefore, we propose to record the number of replies and limit them to a single reply for each consecutive *n*-DAD address query.

E. PROTOCOL CONFIGURABLE PARAMETERS

The particular ND++ set-up has strong impact on the protocol overhead and performance. Our previous investigations have identified the most accurate parameter values [27], [29]–[31], so that ND++ is capable of addressing the needs of diversified network types and sizes. Figure 3 presents the execution (in time) of the DAD++ procedure and the parallel MPR processes performed by a given node for the exemplary network situation. Table 2 presents the values chosen for the key ND++ parameters along with their short description.

V. EVALUATION

During our previous research, we performed a detailed evaluation which allowed us to 1) identify the most suitable ND++ parameters set-up and 2) estimate ND++ performance in diversified network types, sizes and under different channel conditions. In [25] we presented the proof-of-concept evaluation of ND++ in the testbed platform comprising of real hardware machines on which MANET environment was emulated. This step constituted a formal verification of key concepts and assumptions. The results showed that ND++ achieved good performance in small network set-ups. Hence, we followed with a simulation-based evaluation to observe ND++ behavior in medium- and large-scale networks under diversified networking conditions [26], [27], [29]–[31]. This study enabled us to improve ND++ algorithmic schemes and behavior as well as to identify the most flexible and network-independent protocol set-up. However, it concerned static scenarios with one node starting a DAD++ for a newly assigned address.

Therefore, in this article we present our most recent results reflecting protocol performance in case of abrupt network initialization under diversified network conditions with node mobility. Hence, the obtained results provide a more detailed performance estimation, including scalability evaluation.

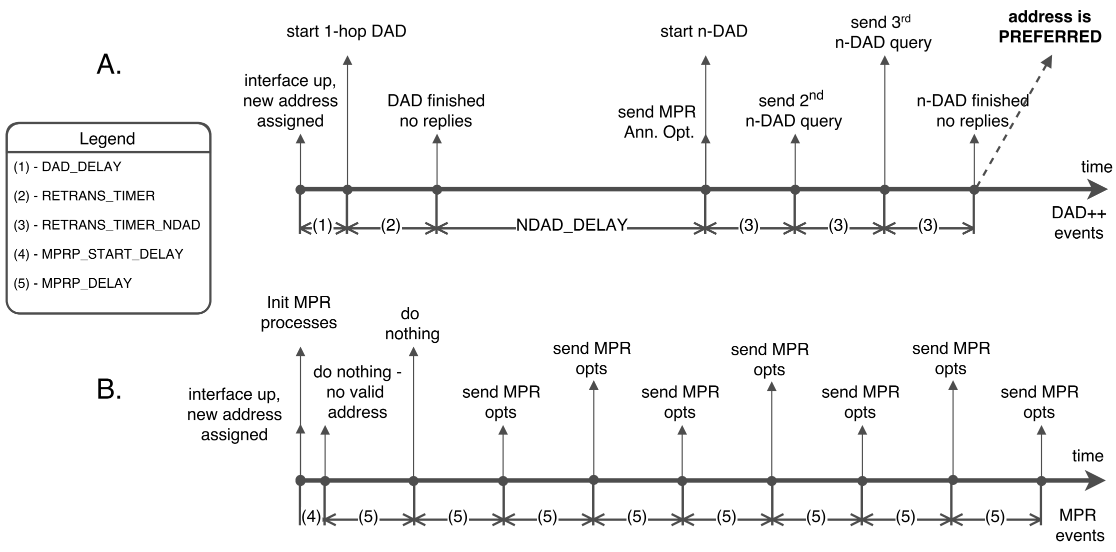


FIGURE 3. The execution of DAD++ (A) in relation to the MPR processes (B) for a selected network node, during a DAD++ query for a single, new IPv6 address, which is the only address assigned to this node [3].

TABLE 2. ND++ parameter values.

Parameter name	Value	Description
HopLimit	1.5 * max. expected node count	specifies the range of multihop messages; the value should be large – overestimating it does not have impact on ND++ overhead
NDAD_COUNT	3	number of consecutive *n*-DAD trials
DAD_DELAY	random $\in \langle 0, 1)$s	random delay before starting 1-hop DAD scheduled to avoid synchronization between nodes during abrupt network initialization
NDAD_DELAY	3s	the time between the end of 1-hop DAD and the start of the first *n*-DAD query – it is necessary to allow for MPR processes to be able to select MPRs in case of abrupt network initialization or in the node bootstrapping phase
RETRANS_TIMER	1s	time to wait for a NA with a reply to the previously sent 1-hop DAD query – it is assumed that in case no reply is received within this time, the address is valid in a link-local scope
RETRANS_TIMER_NDAD	1s for small and medium networks; 3.5s for large networks	time to wait for a mNA with a reply to the previously sent *n*-DAD query – it is assumed that in case no reply is received within this time, the address is valid; the value should be adjusted to network size to accommodate for the time needed to traverse the network to and from the target node; overestimating this value would not corrupt ND++ functionality
FRW_DELAY	random $\in \langle 0, 100)$ms for small and medium networks; $\in \langle 0, 200)$ms for large networks	random delay before forwarding a message using MPR-based flooding – it is necessary to avoid synchronization between neighbors during forwarding which leads to collisions in MAC layer
MPRP_DELAY	1s or 0.5s in mobile set-ups	time between sending consecutive periodic NA messages with MPR Parameters option
MPRP_START_DELAY	random $\in \langle 0, 1)$s	random delay before sending first MPR Parameters option by a node. It is necessary in order to avoid synchronization between nodes in case of abrupt network initialization

A. EVALUATION CRITERIA

For a thorough evaluation of ND++ functionality and performance, we identified the key metrics which were then estimated during our simulation experiments [3]:

- *Reliability* – probability of a successful duplication detection. This metric is very important in the case of stateless AAC mechanisms, since it presents how reliable the protocol is under diversified networking conditions. It is particularly important to assess the reliability not only under ideal channel conditions, but also in realistic environments, when collisions and related message drops occur in lower layers of the protocol stack. Unfortunately, a detailed evaluation of this measure, especially in real channel conditions, is usually neglected by other authors describing stateless AAC solutions.
- *Overhead* – the number of ND++ messages observed in the network during DAD++ started by all nodes in the network which assign themselves a new address (i.e., the sum of all mNS, mNA, NS and NA messages generated by these nodes). In the case where there is no duplication, the overhead measure counts only mNS and NS messages. Substantially, this measure includes the initial messages sent by a DAD++ originator and all of its copies forwarded in the network during a message lifetime. Such an approach gives a good overview of the flooding efficiency being the key factor which

influences AAC solution performance. Substantially, this aspect tends to be omitted by other authors, who often present the results reflecting only the messages that are sent by their initial originators. Additionally, in our approach the messages necessary for MPR-related message exchange are counted in a separate measure depicted as *control overhead*.

- *Latency* – time needed to resolve an address query, starting from a point in time when a new address is assigned (duplicated or not), and finishing when address auto-configuration is done. The latency value will vary significantly in the case of duplicated and not duplicated addresses. Therefore, we present the latency results separately for the nodes which were assigned unique (*Latency_NoDupl*) and duplicated addresses (*Latency_Dupl*).

We anticipate latency as an important factor to identify ND++ features, though for the general performance, overhead and reliability are the most significant metrics, since the results of their improper levels are much more severe to the network as a whole. Those 3 metrics are the key factors which should be a subject of quantitative ND++ protocol evaluation.

B. BASELINE ND++ SET-UP AND NETWORKING ENVIRONMENT

During our experiments, we considered a single-domain, stand-alone, IPv6-only MANET network with single-interface nodes connected through 802.11g Wi-Fi connections. The assumption was that there were no malicious nodes in the network. We investigated abrupt node initialization scenarios, since these are more difficult to handle and our goal was to evaluate ND++ performance in demanding networking environments. Graceful node initialization, where new nodes are set-up within the specified time distance, is less severe and would result in figures lying between abrupt initialization scenarios and a single-node scenario, when there is only one node assigning a new address at a time.

From the ND++ perspective, network partitioning scenarios do not require any additional action from the ND++ protocol, hence there is no need to consider them during this evaluation. Regarding network merging, it is very similar to the considered abrupt initialization from the perspective of the above evaluation criteria, especially in the case of a generated overhead. Since in ND++ itself we focus mainly on the DAD++ and envision that different strategies may be deployed to address network merging issues, the evaluation in the above scenarios should be sufficient to estimate ND++ performance in the event of network merging.

In particular, we consider two types of set-ups, which are applied to the target network of different sizes [3]:

- *New Node* (*nN*) – there is a specified percentage of x new nodes out of the targeted number of network nodes, which are joining a network and assign themselves a new IPv6 address. This situation can occur when a new group of nodes comes up and requires initial configuration in a network. Out of the new nodes, the percentage of y nodes has duplicated addresses. In the case where there were duplicated addresses, z percent of them were assigned the same value – this is important, since DAD++ can be finished faster if the number of the same duplications is higher (when a node receives a query from another node for the same address that is currently being verified by this node, this address is marked invalid regardless of the reply to node's own NS or mNS sent earlier). This case is likely to happen in practical network set-ups, since a common practice is still the IPv6 address generation based on the MAC addresses, for which often the same value is assigned to the whole series of machines.
  When new nodes are joining a network, all aspects of ND++ can be verified, including the effectiveness of the MPR selection – the new nodes were not present in the network before, hence they need to advertise themselves and properly establish MPRs in line with the DAD++ activities. In case the duplication is detected, the node randomly selects another new address (which is always unique in our simulation set-up). Otherwise, it would not have any operational address and could not be a part of the MPR selection procedure. This could cause false negative results, e.g., with regard to the DAD++ reliability, resulting from improper MPR flooding. This is particularly visible with a large number of new nodes where the network would be partitioned without part of the nodes becoming operational. We intentionally wanted to avoid such scenarios.
- *New Address* (*nA*) – there is a specified percentage of x nodes out of the targeted number of network nodes which assign themselves another, new, IPv6 address. Such a situation can occur, e.g., when network renumbering takes place. Out of the new addresses, the percentage of y is duplicated. In the case of duplications, z percent of them were assigned the same value.
  This set-up verifies mainly the performance of DAD++ procedure, in isolation from MPR processes – since the nodes already have one operational address, the MPRs are properly established before DAD++ starts. In the case a new address is duplicated, there is no other address selected, since the node already has at least one valid IPv6 address.

In general, we describe the particular scenarios of the above set-ups as x-y-z-nN or x-y-z-nA, respectively.

The mobility pattern exploited in the experiments was based on a Random Waypoint mobility model, where nodes move in a rectangular area. This mobility model is widely used in the evaluation of MANET AAC solutions (e.g., in the evaluation of the well-known OSPF and OLSR protocols for MANETs [55], [56]). It offers good coverage of the area of operation and average results on most of the metrics identified in [57], such as packet delivery ratio and end-to-end delay (a mobility model which would create too easy or too demanding networking environment could lead to overestimated or underestimated results during the simulations depicted in this article). In addition, the authors of [57] characterize it as flexible and capable of creating realistic mobility patterns which reflect the way people may move. In the selected model the nodes where initially laid out uniformly in the cross-grid topology in the area of a square. The node density was 926 nodes/km^2 – the node's range was fixed at the value of 50m and, hence, the square area was adjusted according to the expected node count. Then, Random Waypoint mobility model was applied, so that each node was randomly choosing a pause time and a speed within the specified interval. At the simulation starting point, the network topology was a random layout resulting from the application of a mobility pattern for a specified amount of time to the initial layout. Our detailed observations revealed that for the prevailing amount of time the network remained fully connected during mobility (isolated nodes were very rare).

ND++ was evaluated in two types of channels [3]:

- *IDEAL* – the ideal channel limits the influence of channel characteristics in order to allow for the evaluation of protocol behavior from the network layer perspective – independently of many MAC and PHY layer artifacts. Hence, the channel propagation loss is specified as a simple model where the signal is either received with the maximum power when it is within a particular range, or with the power close to zero outside of it.
- *REAL* – the realistic channel model introduces additional packet losses, which result from unideal channel characteristics. This channel type enables ND++ observation in more realistic environment where lower layers of the Internet architecture degrade protocol's performance, especially in terms of its reliability. We modeled the channel as a log-distance path loss model.

Due to our evaluation objectives and the results of our previous investigations, fading effects were not added in a *REAL* channel model – introducing fading channel without the long-distance path loss have not influenced the ND++ reliability. Moreover, apart from signal degradation, fading channels introduce gain, which significantly influences topology by adding temporary, new, wide-range connections between nodes. However, for the ND++ evaluation we have taken the objective to keep the investigated topologies as stable as possible [3].

We were simulating networks of different sizes: small with 16 nodes, moderate with 36 nodes and large with 100 nodes.

C. SIMULATION ENVIRONMENT

From a large number of investigated MANET simulators [58], we chose the NS-3 [59] simulator, since it best fulfilled the requirements of the envisioned networking environment – with regard to topology, node mobility and channel propagation models. Moreover, it is an open-source simulator allowing for direct modifications to the IPv6 stack, which was crucial for the ND++ implementation.

The 802.11g network was configured in NS-3 with OFDM mode and the rate of 54Mbps to achieve the maximum throughput. This was driven by the objective to limit the impact of the PHY and MAC layer parameters on the overhead and other metrics. The channel was specified as either *IDEAL* or *REAL*, as described above. The network layer was an IPv6-only network with basic IPv6 protocol stack enhanced with the proposed ND++ solution. There was no routing established in the network, since at the stage of address auto-configuration it is not necessary and likely could not be operational yet.

For each simulation a single node or a group of nodes was randomly selected to start DAD++. The final results are an average over 5 different simulations, each with different set of randomly selected nodes. This way, the results are independent of node positions. Five different sets of random nodes in the mobile environment are in our opinion enough to achieve the required diversity of set-ups.

For the final averaged results the 95% confidence intervals were specified and assessed with the goal that the half of the two-sided confidence interval should be an order of magnitude lower than the estimated value, preferably not more than 10 times lower. This condition was met in the vast majority of cases, excluding some special set-ups where due to network configuration the observed results were significantly different (for these situations, an explanation is given further on in this article). The confidence intervals are drawn on the presented graphs, unless they are too small to be visible [3].

D. SIMULATION PARAMETERS

The values chosen for the key simulation parameters are presented in Table 2. *RETRANS_TIMER_NDAD* and *FRW_DELAY* are the only parameters which have different values for the 100-node network than for the other investigated cases. We recommend to set them according to the estimated network size, however bigger values could also be chosen for all topologies. This could, however, result in lower ND++ performance, especially with regard to latency. MPR-related parameters and *NDAD_DELAY* are chosen in relation to *DAD_DELAY* and *RETRANS_TIMER*, so that proper MPR selection is performed in parallel to DAD++ procedures.

In order to best assess ND++ functionality and performance under diversified load we have identified several configuration set-ups. They reflect the plethora of situations and networking environments that have the biggest influence on ND++ behavior. The scenarios can be divided into 4 categories and described according to the *x-y-z-nN* or *x-y-z-nA* format, as specified in Section V-B:

- category A: a single node is randomly selected to start DAD++ for a new address. Four variants are considered: with duplication present or not, in a situation of new node joining a network or a new address assignment to an existing node: singleNode-nN-noDupl, singleNode-nA-noDupl, singleNode-nN-dupl, singleNode-nA-dupl.

- category B: low load scenario 5 – 100 – 0–nN – 5% of new nodes out of which all are duplicated
- category C: moderate load, 20% of new nodes out of which maximum of 50% are duplicated; variable other parameters: 20-50-0-nN, 20-50-0-nA, 20-50-100-nN, 20-20-0-nN.
- category D: very high load, 50% of new nodes and a variable number of duplications among them. In this group we included scenarios with both new nodes joining a network and nodes assigning themselves new addresses: 50-100-0-nN, 50-100-0-nA, 50-20-0-nN and 50-20-100-nN.

For the Random Waypoint mobility model set-up we chose node pause time to be uniformly distributed in the interval $\langle 2, 5)$s. Such values enabled smooth move pattern. Each of the above scenarios was evaluated for the node speed corresponding to either *walk* ($\in \langle 3, 7)$km/h) or *run* ($\in \langle 10, 35)$km/h), since we envision that solutions like ND++ would be mostly used under such conditions. Moreover, we were considering 2 different set-ups of MPR parameters, namely *MPRP_DELAY* and the related *MPRP_START_DELAY* – the first one as specified for static scenarios (1s) and the other one with more frequent neighbor information collection achieved by setting these parameters to 0.5s. Hence, each scenario was repeated in 4 different configurations with regard to node speed and *MPRP_DELAY* for each of the 3 network sizes.

VI. RESULTS

In this section we present the simulation results for the scenarios A-D with regard to the key identified metrics - protocol reliability, overhead and latency. Our goal was, however, not only to investigate ND++ performance with regard to those metrics, but also to assess if more frequent neighbor information collection is necessary to properly handle node mobility during MPR processes. The measurements are made in two types of channels – under ideal and realistic conditions.

A. RELIABILITY

During the reliability evaluation our main goal was to verify if the probability of successful duplication detection in ND++ permits practical application in diversified network scenarios. Our assumption was to reach reliability as close as possible to 100%, not lower than 80% for each investigated case.

For the scenarios with node mobility, reliability is the critical metric. In Figure 4 and Figure 5 we present the results for both *IDEAL* and *REAL* channel. The figures enable us to compare the ND++ ability to properly detect duplications under two different set-ups –with *MPRP_DELAY* set to 1s versus more frequent information collection with *MPRP_DELAY* set to 0.5s (the related maximum *MPRP_START_DELAY* is always set accordingly – to the value equal to *MPRP_DELAY*).

Significant increase in the reliability levels can be observed when MPR-related information is exchanged more frequently, i.e., once per 0.5s. This effect is present for both *IDEAL* and *REAL* channel conditions as well as for both levels of node speed. It is particularly visible for smaller network sizes. Moreover, the results for *New Address* scenarios also tend to be worse than in the case of *New Node* scenarios. Those two effects are most likely related to the MPR-based forwarding performance in the changing environment imposed by node mobility – in small networks the number of MPRs is relatively small. Hence, the impact of potentially unreliable links (due to node mobility) is more severe. Additionally, it seems that newer MPR information taken as a basis for MPR selection algorithm, results in a better performance – this is visible in case of *nN* scenarios [3].

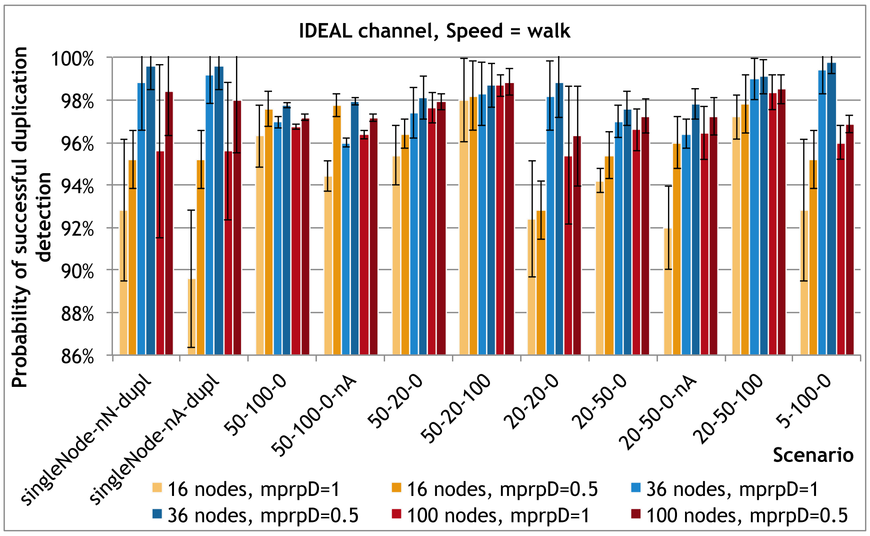
For *IDEAL* channel conditions in the set-up with *MPRP_DELAY* equal to 0.5s, the ND++ reliability was above 90% for each scenario, often being more than 96%. Hence, this set-up should be recommended. For the *REAL* channel, the 90% threshold was exceeded for the vast majority of scenarios, though for the category D scenarios (except 50-20-100), reliability was decreased in the case of the 100-node network. Taking into consideration the harsh mobile network environment, where the nodes are in practice in nearly constant move, the achieved levels are acceptable. Nevertheless, they seem to represent the boundaries of the ND++ performance possible to be reached with 802.11g PHY/MAC protocol. Looking at the results for *IDEAL* channel conditions, the ND++ itself has the capacity for a possibly better performance, however it would need to be accompanied with improved Wi-Fi network access protocols. In general, we envision such reliability levels sufficient to address the needs of diversified MANET set-ups [3].

B. OVERHEAD

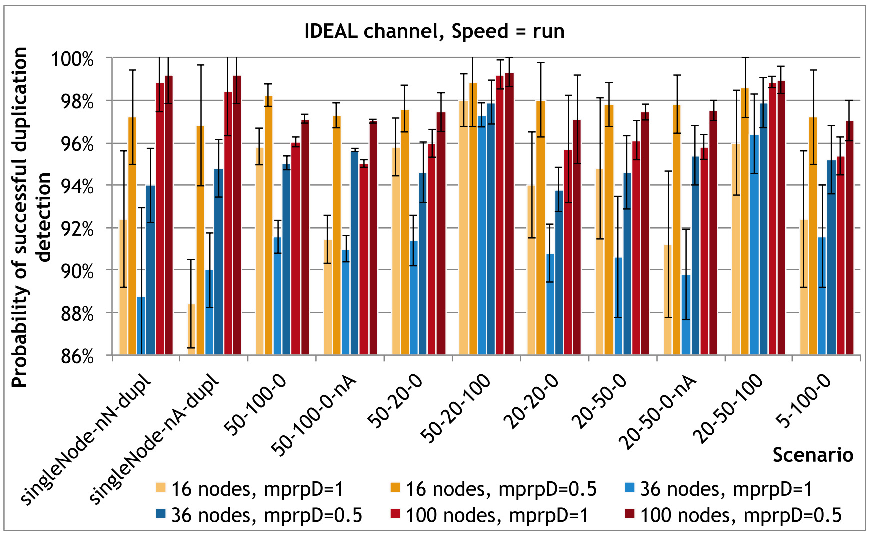
Our primary focus is on the comprehensive and generic ND++ performance evaluation. Therefore, in this article we focus on the overhead metric referring to the total number of all ND++ messages generated after a new address was assigned. However, with this assumption it is necessary to bear in mind that the obtained values may vary significantly depending on the scenario, in particular on the fraction of new nodes having a duplicated address (mNA messages are sent only when a duplication is detected).

Moreover, in the scenarios with *New Node* set-up when a duplication is detected, another address is chosen (which in our experiments is always unique) and verified by the DAD++. Hence, in such a situation the presented results depict, in fact, the overhead generated by two DAD++ procedures for two different addresses.

Scenarios, even within the same category, vary in the number of nodes which start DAD++. Moreover, this value is specified as a percentage of total expected network size. Hence, its absolute value will be also different for each network size. In order to be able to compare results between particular scenarios, we present the obtained overhead values per address allocation. For this purpose, an address allocation is understood as a full trial to obtain a valid address – i.e., in the *New Node* set-up a single address allocation is



(a)



(b)

FIGURE 4. ND++ reliability in ideal channel conditions for node mobility set-ups in scenarios of category A-D in the networks of size 16, 36 or 100 nodes and two different set-ups – with *MPRP_DELAY* equal to 1s or 0.5s. (a) Speed = *walk*. (b) Speed = *run*.

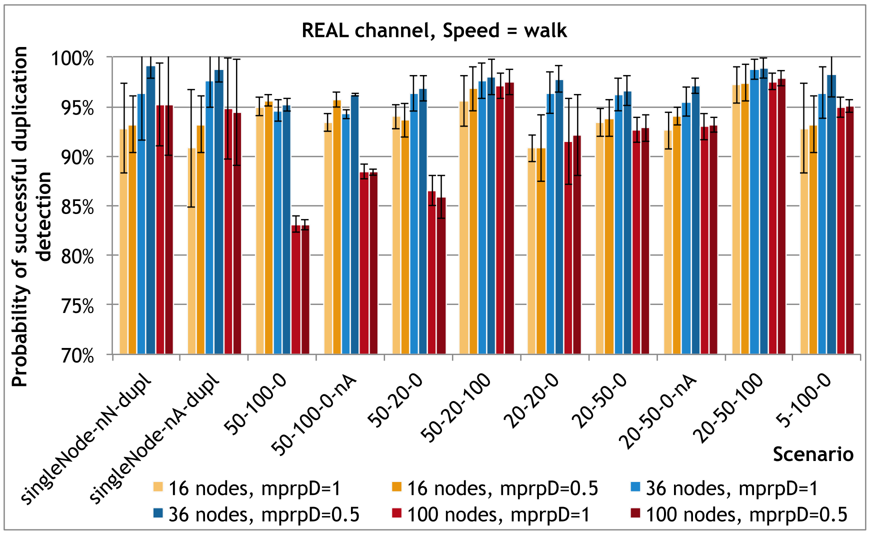
perceived as the entire procedure performed after first address assignment (comprising in fact of two DAD++ queries), while in the *New Address* set-up a single DAD++ procedure is covered [3].

1) PERFORMANCE EVALUATION

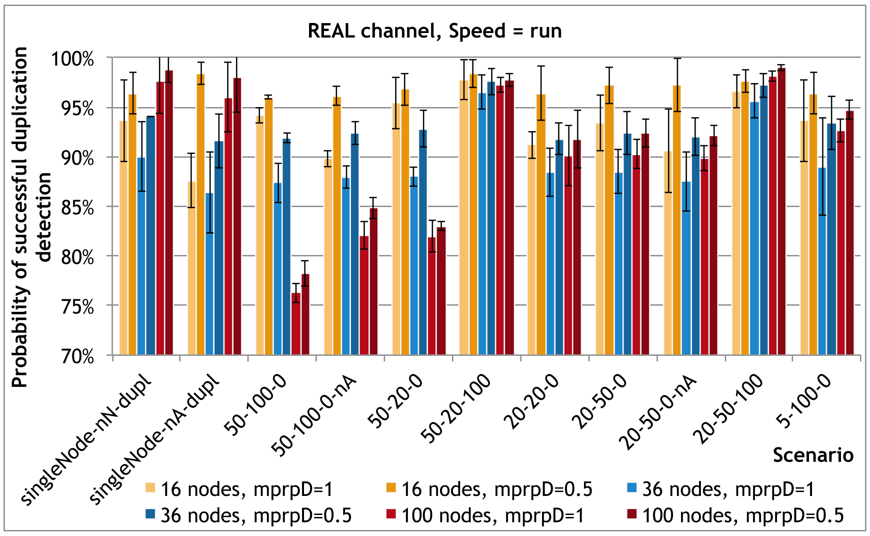
We present the ND++ message overhead versus the network size. The overhead values are the sum of all ND++ messages sent by all the nodes in the network during the scheduled address allocation attempts. They are then divided by the number of those attempts which is similar to the number of nodes that are either joining a network or assigning themselves a new address. Such "per address allocation" figures are then divided by the expected network size to obtain "per node per address allocation" results. In particular, in category A scenarios there is only one address allocation in each scenario.

Figures 6, 7 and 8 present the overhead of all ND++ messages for different situations. The total overhead values are comparable in all the considered cases for both investigated *MPRP_DELAY* values, with a slight decrease (usually within the confidence interval boundaries) in the case of *MPRP_DELAY* equal to 0.5s. Hence, taking into consideration the reliability results (as well as latency results, which also confirm this conclusion), we would recommend to select lower *MPRP_DELAY* values for the mobile scenarios. Therefore, we further present the results for this set-up.

For category A scenarios, the difference between *IDEAL* and *REAL* channel conditions was negligible, hence we present only the results for the latter case in Figure 6.



(a)



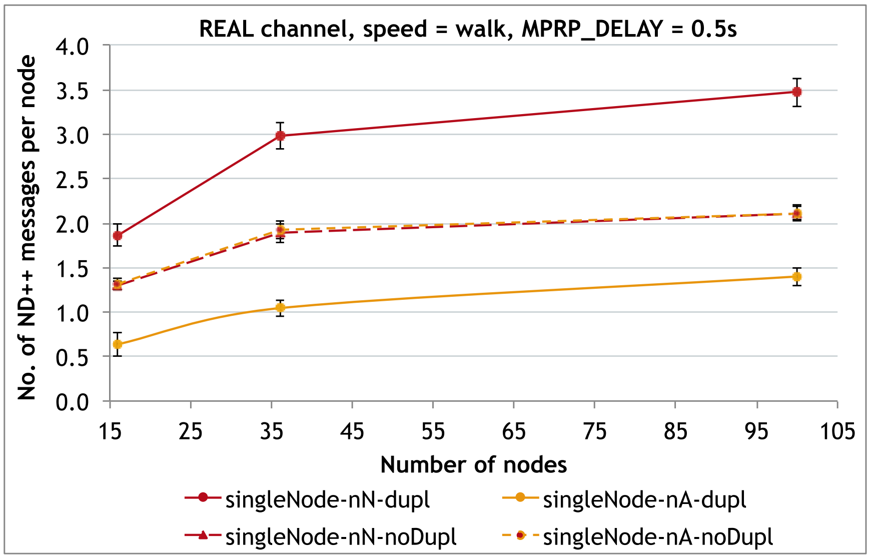
(b)

FIGURE 5. ND++ reliability in realistic channel conditions for node mobility set-ups in scenarios of category A-D in the networks of size 16, 36 or 100 nodes and two different set-ups – with *MPRP_DELAY* equal to 1s or 0.5s. (a) Speed = *walk*. (b) Speed = *run*.

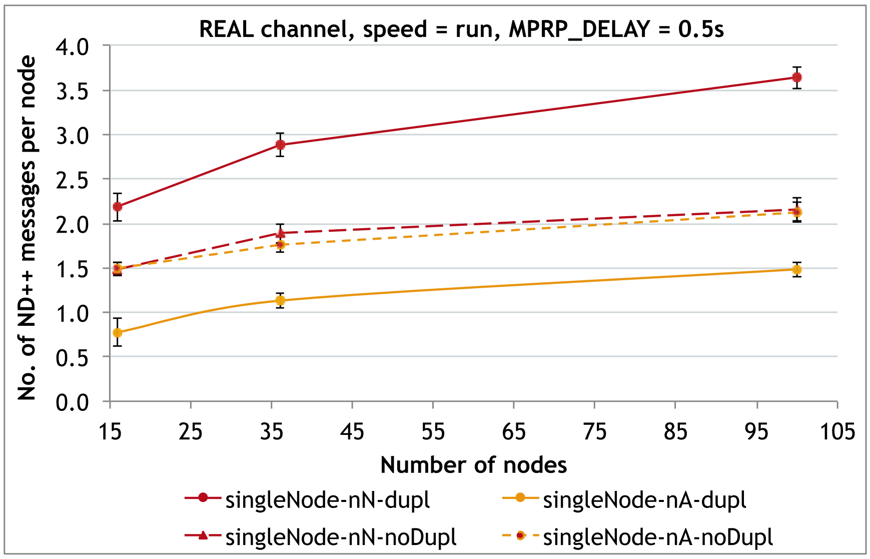
It depicts category A scenarios with a single node starting DAD++ procedure. The lowest overhead values are observed regardless of the network size for the scenario singleNode-nA-dupl, since in this case, after duplication is found, the procedure finishes with the address marked invalid. In this scenario, in most cases, a single n-DAD query is enough to confirm duplication. However, in larger networks the necessary number of n-DAD queries is usually higher, which leads to the slight increase in overhead, above the values suggested by the linear increase proportional to the difference in network size (as compared to smaller networks). This observation is then also confirmed by the latency evaluation presented in Section VI-C.

SingleNode-nA-dupl scenario presents the baseline ND++ performance with a minimum number of necessary n-DAD queries, with both mNS and mNA messages present. On the contrary, singleNode-nN-dupl scenario presents a case when address assignment finishes after finding an address which is valid (in our case with necessary two consecutive DAD++ procedures). This implies that for this scenario the overhead levels will be higher. Moreover, they will usually be more than 2 times higher, since the second address is valid. This implies that all 3 n-DAD trials will be executed to confirm it, while in the case of duplication detection, the remaining scheduled n-DAD trials are omitted, which reduces overhead [3].

The last two scenarios presented in Figure 6, namely singleNode-nN-noDupl and singleNode-nA-noDupl, depict a situation when a new address is not duplicated. Hence, the overhead performance in both of these cases is similar,



(a)



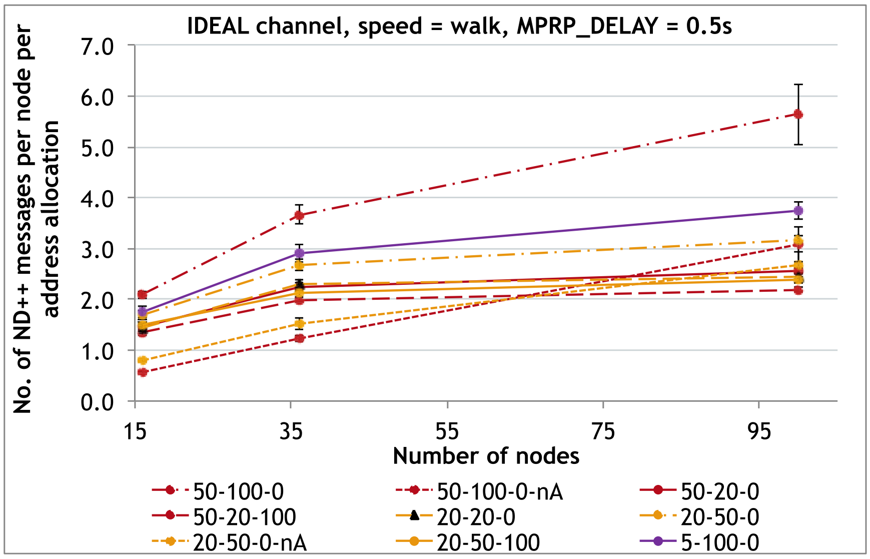
(b)

FIGURE 6. Number of ND++ messages per node in realistic channel conditions for scenarios of category A in the networks of size 16, 36 or 100 nodes with *MPRP_DELAY* equal to 0.5s. (a) Speed = *walk*. (b) Speed = *run*.

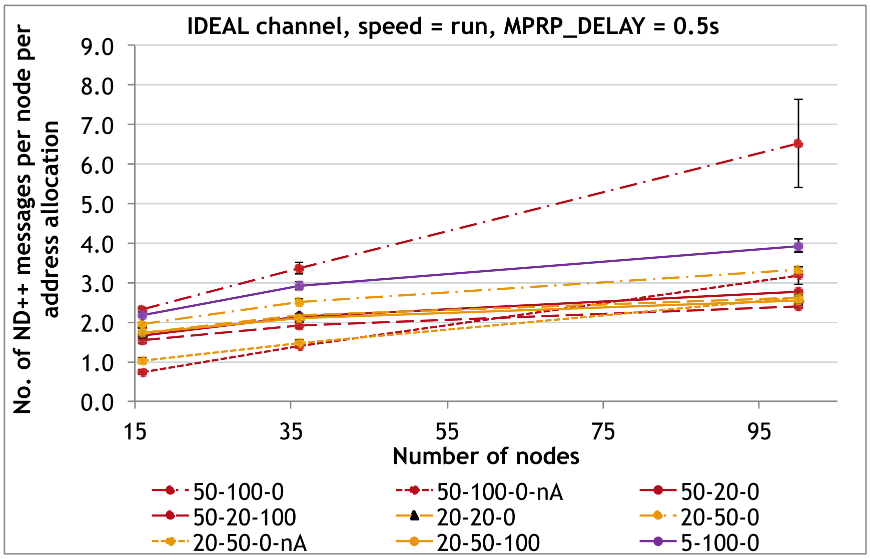
since DAD++ is executed in the same manner, regardless of the new IPv6 address being the first or consecutive node's address. When the new address being verified is unique, all 3 *n*-DAD trials are necessary, which implies that overhead observed for this case is higher than for the singleNode-nA-dupl scenario. However, it should be also underlined that in ‘‘-noDupl’’ scenarios mNA messages are not present [3].

For scenarios of category A, the single query for a duplicated address, in singleNode-nA-dupl scenario, was resolved with less than 1.5 messages per node. Approximately 2 messages per node ware needed when querying for a unique address – for both investigated speeds. For a singleNode-nN-dupl scenario, the total overhead did not exceed approx. 3.5 messages per node.

Figure 7 and Figure 8 present the overhead results for the scenarios of categories B, C and D, where multiple nodes start DAD++ at the same time. Similarly to the category A scenarios, the lowest values are observed for the scenarios with *New Address* set-up (20-50-0-nA and 50-100-0-nA), where the DAD++ is not repeated after initially assigned address has been identified as duplicated. In general, the bigger fraction of nodes has a duplicated addresses among all the nodes which assigned themselves a new address (i.e., the *y* value), the higher the overhead. Hence, the scenarios 5-100-0-nN and 50-100-0-nN have reached the highest overhead values among all investigated cases. Remarkably, the overhead for these two cases is relatively similar, regardless of the fact that 10-times more nodes were starting address queries in



(a)



(b)

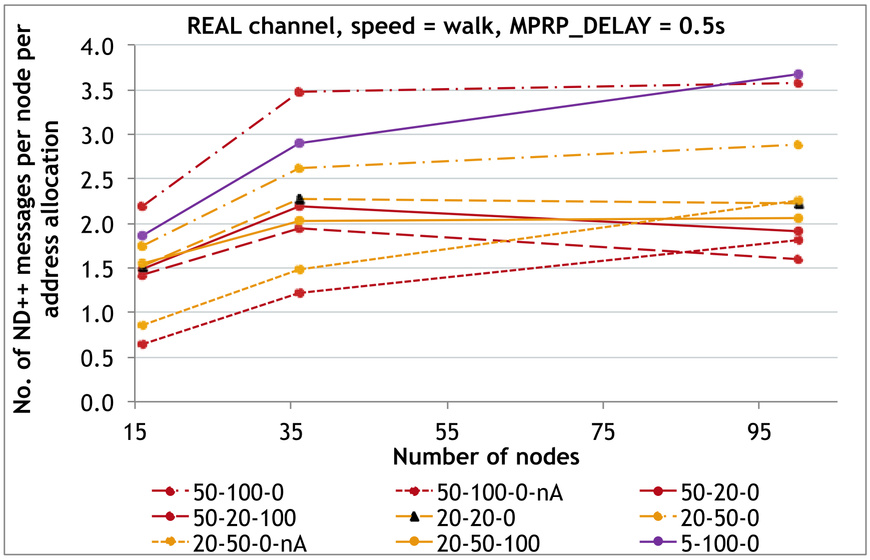
FIGURE 7. Number of ND++ messages per node per address allocation in ideal channel conditions for scenarios of categories B (marked violet), C (marked yellow) and D (marked red) in the networks of size 16, 36 or 100 nodes with *MPRP_DELAY* equal to 0.5s. (a) Speed = *walk*. (b) Speed = *run*.

scenario 50-100-0-nN (50% of all nodes) as compared to scenario 5-100-0-nN (5% of all nodes). This feature proves ND++ scalability, which will be discussed in more detail in the remaining part of this section. Moreover, while comparing scenarios 20-50-100-nN with 20-50-0-nN and 50-20-100-nN with 50-20-0-nN, it can be observed that lower overhead is generated when the new nodes are assigned the same address each (i.e., the value *z* in *x-y-z-nN* scenario is 100%) – this is driven by the ND++ specification where a node marks invalid an address for which DAD++ is ongoing after detecting queries for this address which originate from another node.

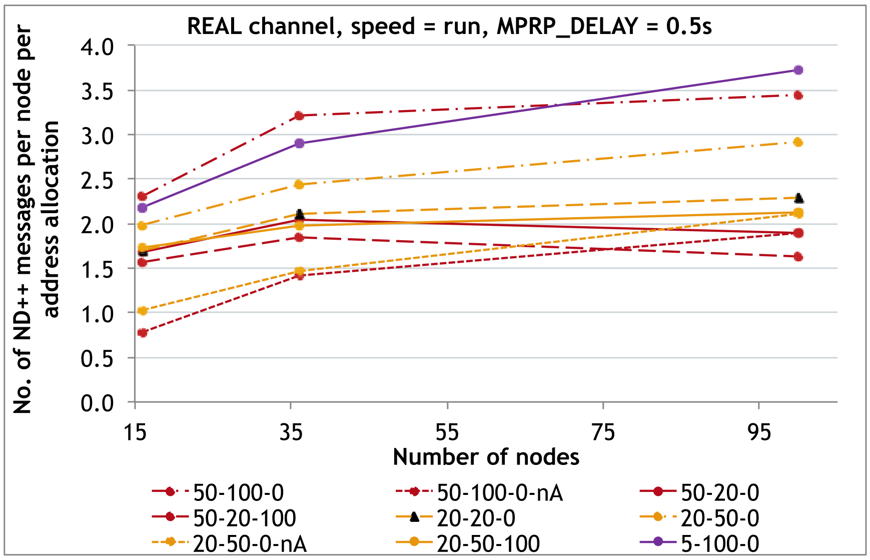
The maximum and minimum overhead values observed for the scenarios B, C, and D in *REAL* channel conditions, as presented in Figure 8, are similar to those obtained for category A scenarios. However, for *IDEAL* channel conditions (Figure 7), an increase in the number of messages is observed for the category D scenarios, in particular for the most demanding scenario – 50-100-0-nN. This effect is most likely related to the capabilities of MAC and PHY layer protocols in the very demanding environment. There is no significant increase observed in the overhead values in the scenarios of different categories, i.e., under variable load. The results for different node speeds are also comparable.

2) SCALABILITY EVALUATION

A scalable system is supposed to remain operational under increased load. In the case of ND++ and AAC protocols,



(a)

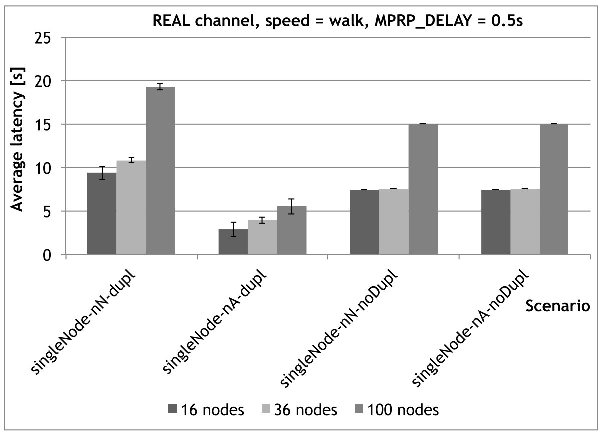


(b)

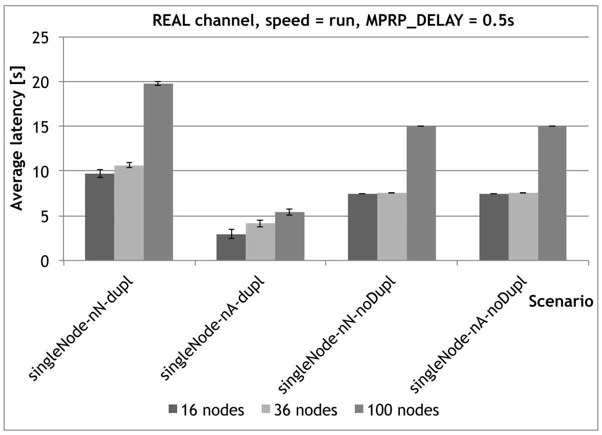
FIGURE 8. Number of ND++ messages per node per address allocation in realistic channel conditions for scenarios of categories B (marked violet), C (marked yellow) and D (marked red) in the networks of size 16, 36 or 100 nodes with *MPRP_DELAY* equal to 0.5s. (a) Speed = *walk*. (b) Speed = *run*.

we envision that with the increased network size or in the presence of higher number of simultaneous address allocations the protocol overhead per node should preferably remain approximately constant, or at most grow slowly linearly. Considering the conclusions from the analysis presented above, in particular with regard to the comparison of overhead values for the corresponding scenarios of different load, ND++ has the scalability properties. The results for each particular scenario remain approximately constant with the increased network node count. Some visible exceptions to this observation may result from the necessity to execute more n-DAD address queries in more demanding environments of larger networks. Moreover, in smaller networks the fraction of nodes being on the edge of the network (which are less likely to become MPRs) is relatively big as compared to the total number of network nodes. This may enhance more efficient MPR structure (i.e., with a relatively small number of MPRs it is possible to cover all network nodes), which results in reduced overhead. Hence, in general, it can be assessed that ND++ is scalable in all considered scenarios, while looking at its behavior in the networks of increased size.

However, ND++ scalability is also visible when comparing particular scenarios from different categories. Overhead for the corresponding scenarios (e.g., 5-100-0-nN, 50-100-0-nN and singleNode-nN-dupl) is very similar, even though the scenarios represent different categories and,



(a)



(b)

FIGURE 9. **Average latency in realistic channel conditions for category A scenarios in the networks of size 16, 36 or 100 nodes with *MPRP_DELAY* equal to 0.5s. (a) Speed = *walk*. (b) Speed = *run*.**

hence, different load of address queries intensity. An exception is the 100-node case for *IDEAL* channel conditions where more *n*-DAD trials are likely to be executed and related flooding is not suppressed. Nevertheless, we can generalize that with an increased load, in terms of the number of simultaneous address queries, ND++ remains scalable. The overhead may vary depending on a particular scenario set-up and its conditions (e.g., depending on the fraction of nodes which are duplicated), but will behave almost the same in the case the number of simultaneous address queries is increased within the similar scenario conditions. Hence, the results presented in Figures 6, 7 and 8 show the overhead boundaries among which the vast majority of possible network situations will be laid. Therefore, we can specify with high confidence that for most cases ND++ overhead performance in realistic environments will remain within 0.6 (16 nodes, speed=*walk*, singleNode-nA-dupl) and 3.7 (100 nodes, speed=*run*, 5-100-0-nN) messages per node per address allocation. The upper boundary in *IDEAL* channel conditions was evaluated as 6.5 messages (100 nodes, speed=*run*, 50-100-0-nN).

3) CONTROL OVERHEAD

Apart from the overhead figures presented above, ND++ has also control overhead which depends on a particular set-up of MPR-related processes. Control overhead is limited to a 1-hop scope of each node. It is related to NA messages sent to communicate neighbor information and MPR parameters. Hence, the control overhead is limited to 1 message sent each *MPRP_DELAY* seconds. In ND++ it is one message per second for static scenarios (as depicted by our previous research [27], [31]) or 2 messages per second for mobile scenarios. Additionally, 1 asynchronous NA is sent with MPR parameters before first *n*-DAD is started, which results in additional 1 message per each DAD++ procedure. Due to its limited scope and not frequent, scheduled message generation which is not synchronized among nodes, this factor is not limiting the network performance when ND++ is operational [3].

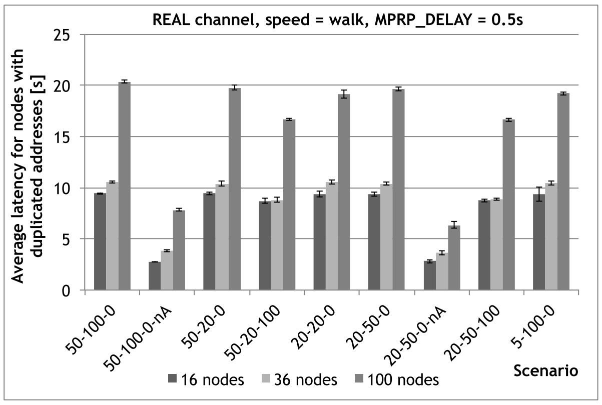
Interestingly, the IETF proposed recently the Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP) [60], where it promotes the OLSR-like 2-hop neighborhood information collection. This RFC proposes to gather such information in each MANET network as a standard behavior. If in use in the network, this protocol could, to some extent, replace the NA-based control information collection in ND++, which would be done in the network anyway by the NHDP protocol. In this way ND++ would not bring additional control overhead, except for an asynchronous NA sent before the first scheduled *n*-DAD. Hence, it could be further integrated with other IPv6 standardized solutions [3].

C. LATENCY

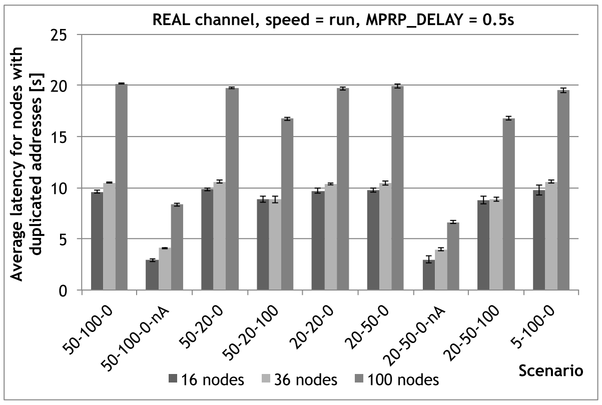
Latency evaluation aims at presenting the duration of AAC procedures in different scenarios. The initial time point for latency assessment is always this moment when the first DAD++ procedure is started, equal to the time when node's first or consecutive address is assigned (i.e., when the experiment is started). The endpoint for latency evaluation is considered to be one of the two following cases: 1) the newly assigned address is marked as either PREFERRED or INVALID in the case of the *New Address* scenario set-up and 2) the node is assigned a new address which is marked PREFERRED in the case of the *New Node* scenario set-up. Hence, latency reflects the total length of address assignment and duplicate address detection procedures performed with ND++ in each scenario [3].

In Figure 9 and Figure 10 we present the latency results obtained for the selected set-up with *MPRP_DELAY* equal to 0.5s for the *REAL* channel (the differences between *REAL* and *IDEAL* conditions are negligible).

For the *Latency_NoDupl* metric, the obtained values are very similar for each case. This is driven by the fact that the expected length of DAD++ procedure in the case of unique



(a)



(b)

FIGURE 10. Average latency for nodes with duplicated addresses in realistic channel conditions for scenarios of categories B,C,D in the networks of size 16, 36 or 100 nodes with *MPRP_DELAY* equal to 0.5s. (a) Speed = *walk*. (b) Speed = *run*.

address is determined only by the protocol set-up – the node has to wait till the specified timeout before it can consider a new address valid. This also implies that latency values observed for 16- and 36-node network will be lower than for 100-node network, where the set-up has to be different to accommodate for bigger network size. The only variations in the observed latency values result from the small random delays introduced to avoid synchronization. Hence, the confidence intervals lengths are negligible. The results for *Latency_NoDupl* are approx. 7.5s for 16- and 36-node network and 15s for 100-node network. They are visible in Figure 9 for the singleNode-nN-noDupl and singleNode-nA-noDupl scenarios. The results do not depend on node speed.

Figure 9 and Figure 10 present the results for the *Latency_Dupl* metric – the values refer to the fraction of nodes which were originally assigned a duplicated address. For the *New Address* scenarios, namely 50-100-0-nA and 20-50-0-nA, we can observe latency values obtained in a single DAD++ query. These show that the address status was resolved in less than 4s in the case of 16- and 36-node network and less than 8.3s for the 100-node network. The values obtained for small and medium networks are similar for all investigated *New Address* scenarios, including singleNode-nA-dupl. An increase is, however, noted for the 100-node network while comparing a single node case from category A to scenarios of category C and D. This shows that an increase in the number of simultaneous address queries does not influence the ND++ performance in smaller networks, but becomes visible in large networks. This effect results from PHY and MAC layer artifacts, since it was not visible under *IDEAL* channel conditions. Nevertheless, the observed difference is not significant. Hence, we perceive that this effect does not affect scalability (this is additionally confirmed by a comparison of the 50-100-0-nN, 5-100-0-nN and singleNode-nN-dupl results) [3].

For the *New Node* scenarios, the *Latency_Dupl* metric refers to the DAD++ performed twice - until to

TABLE 3. The summary of key ND++ performance measures.

Reliability [%]	*IDEAL* channel: >90%, mostly >94%; *REAL* channel: >77%, mostly >90%
Overhead [messages/ node/ address allocation]	*IDEAL* channel: 0.6 – 6.5; *REAL* channel: 0.6 – 3.7
Latency_NoDupl [s]	7.5 for 16- and 36-node network; 15 for 100-node network
Latency_Dupl, *New Address* scenarios [s]	<4 for 16- and 36-node network; <8.3 for 100-node network
Latency_Dupl, *New Node* scenarios [s]	8.7–10.5 for 16- and 36-node network; 16.6–20.3 for 100-node network

a valid address is obtained. Hence, it is approximately the sum of the corresponding *Latency_NoDupl* values and the *Latency_Dupl* figures obtained for the *New Address* scenario. In general, for 16- and 36-node set-up, the latency was oscillating between approx. 8.7s and 10.5s. The differences between particular scenarios, however, slightly increased for 100-node network reaching the values between 16.6s and 20.3s (for the same reason as depicted above for the *New Address* scenario set-ups). Nevertheless, we can still perceive the scalability requirements to be fulfilled [3].

No significant differences were observed in the evaluation of latency between the "walk" and "run" cases. Hence, it can be concluded that the influence of node speed on the latency figures is negligible.

D. COMPARISON WITH RELATED WORKS

The functional differences between ND++ and the most significant related works are discussed in Section III-C. Table 3 summarizes the simulation results of ND++ performance which were presented in Section VI. In this section we will provide an overview of a quantitative comparison, though the

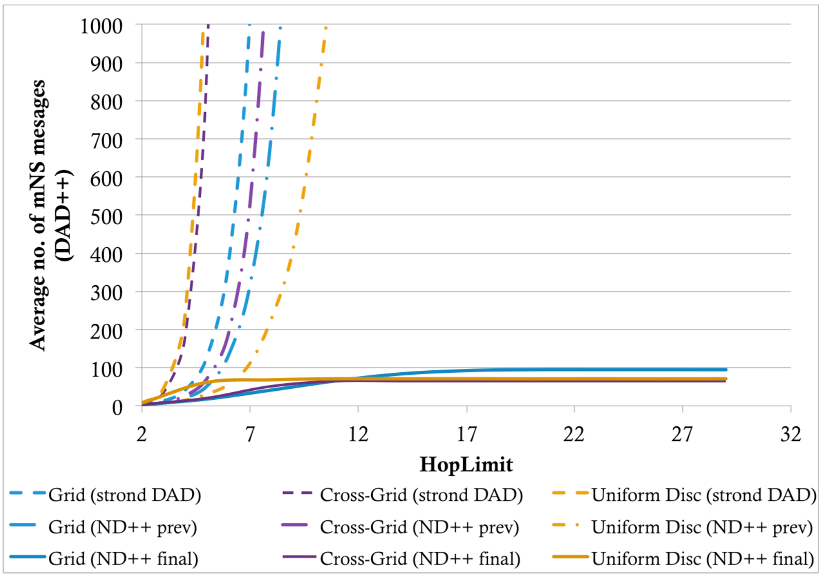


FIGURE 11. Comparison between "strong DAD", latest ND++ version (ND++ final) and previous ND++ version (ND++_prev) – scenario of a single DAD++ query with one *n*-DAD phase repetition for an exemplary Grid topology in *IDEAL* channel conditions for a 16-node network.

presented figures will be an estimation, since each state of the art solution was evaluated in a different manner and under different scenarios. In particular, the evaluation of stateless protocols is focused mainly on the overhead estimation, while in the case of stateful protocols, the focus is shifted toward address pool accessibility. Moreover, in most cases the pieces of information given in the papers presenting other solutions are not detailed enough to replicate them, while the conditions under which the presented values were obtained are often not clear. For the ND++ evaluation, the most important is the comparison with other stateless AAC solutions, however we also present a discussion on stateful mechanisms and the most relevant hybrid approaches, focusing mainly on the key metric – overhead.

The ND++ proposes a flooding control method, hence, it will outperform all the stateless solutions which rely on exhaustive flooding, in particular the approach by Perkins *et al.* [20] and Park *et al.* [23]. The solution of Perkins *et al.* [20] is based on purely exhaustive flooding, while Park *et al.* [23] proposes to drop the copies of previously forwarded messages in order to limit a broadcast storm. Figure 11 presents a comparison between [20] (*strong DAD*), latest ND++ version (*ND++ final*') and one of previous ND++ versions (*ND++ prev*), where MPR-based flooding was introduced without suppression of copies of previously sent messages. It reveals that ND++ highly outperforms the *strong DAD* solution and shows the clear benefit of MPR-based flooding in comparison to exhaustive flooding. This proves that ND++ would also perform better than the approach of Park *et al.* [3], [23]. The comparison presented in Figure 11 depicts the results for an exemplary 16-node networks of different topologies, however, the same relation and character of differences was observed by us also in larger network sizes of 32 and 100 nodes.

Boudjit *et al.* [40], who also proposed a flooding control approach, presented a comparison between their proposed flooding method and MPR-based flooding as proposed in OLSR and also used in ND++. The results present that in the flooding proposed for the AAC purposes in [40], the number of selected forwarding nodes is higher and that this increase is growing with a node count. Forwarding in ND++ is, however, as efficient as in the OLSR protocol in the majority of cases. Hence, ND++ outperforms the approach of Boudjit *et al.* in terms of overhead, notwithstanding to the other functional advantages of ND++ over this method [3] (presented in Section III-C).

As for the stateful address auto-configuration methods, a comparison between a number of them is presented in [34]. The results reveal that with regard to the protocol overhead those protocols performed worse than Perkins' DAD approach [20]. In particular, the worst-case ManetConf solution [15] required approx. 5 network-wide exhaustively flooded control messages per each address allocation. This is significantly more than in the case of ND++ or even Perkins' method. However, the stateful solutions outperformed *strong DAD* in the context of latency, which is a typical feature of this type of AAC methods [3]. Other benefits of stateful mechanisms were rather functional and were already discussed by us in the beginning part of this article.

Referring to hybrid and other AAC solutions, the most relevant are the approaches by Wang and Qian [16] and by Weniger and Zitterbart [21]. In the case of those methods the comparison to ND++ and other stateless AAC mechanisms is not straightforward, since they rely on the hierarchical structure of selected "super nodes" with enhanced capabilities. This structure, however, needs to be created and maintained during network lifetime, which requires exhaustive network-wide flooding and, hence, constitutes the main cost of the solution in terms of overhead. Therefore, ND++ will outperform these methods in the long run, since 1) it uses optimized flooding and 2) flooded mNS and mNA messages are sent only during DAD++ when a new address is being verified. Weniger and Zitterbart [21] does not present any quantitative results referring to their solution. However, Wang and Qian [16], whose proposal is perceived as more efficient, presents control traffic to be about 120 messages per node within the time period of 2000s. Additionally, in [16] each address query is resolved with approximately 38 messages per node among which some are flooded network-wide. Substantially, the authors have counted only the initially generated messages and not their forwarded copies as we have presented for ND++. Among them some were 1-hop-scoped, while some others were multihop and were forwarded by the majority (or even all) nodes in the network increasing the figures for the total number of messages sent by all the nodes in this network. Unfortunately, the authors do not present any detailed information about how many of them where flooded through the network. Nevertheless, the presented results suggest that ND++ with maximum 6.5 messages per address query per node and control traffic limited to 1 link-local message per node per second (or 2 messages per second in the case of mobility) should provide significantly better results in terms of overhead. As regards the latency figures, the hybrid approaches provide better results than ND++ – a query for a new address is not sent to the entire network at the cost of a control overhead necessary for establishing and keeping hierarchical configuration structure of "super nodes" [3].

VII. CONCLUSION

We have presented the evaluation of ND++ stateless AAC protocol under diversified channel conditions, network set-ups, sizes and topologies in node mobility scenarios. The detailed analysis of ND++ behavior targets the aspects of the key metrics – reliability, overhead and latency. The results of simulation experiments reveal that ND++ not only provides close to 100% reliable AAC services under ideal channel conditions, but also has the required capabilities and mechanism to keep very high probability of successful duplication detection in demanding channel environments and mobility patterns. Moreover, the overhead analysis exposed that the protocol performance was not more than approx. 6.5 messages per address allocation per node in diversified scenarios, including those of abrupt network initialization with many new nodes or addresses. In many cases, as few as 2-3 messages per address allocation per node were enough to perform DAD++ successfully, with the minimal values less than 1 message. Those results were achieved for two different levels of node's speed reflecting the variabilities in mobility patterns and scenarios. The latency values are higher than for stateful and hybrid AAC related works, however this is typical for stateless solutions where all AAC efforts are performed right after new address assignment. The values of approximately 3-11s for smaller networks and approx. 6-20s for large networks are acceptable and should not limit the practical applicability of ND++, especially that the uniqueness of an address in the link-local scope is confirmed within 1s and from this moment this address can be used in 1-hop message exchange. The observed large deviations in latency values are related to particular scenarios, which either contain the duplicated nodes or not, perform consecutive query for another address after duplication is detected or finish with marking new address as invalid.

Substantially, the latency would delay other node activities usually only in the beginning of node's operation. We perceive reliability and overhead as more significant metrics, since they reflect the ND++ ability to fulfill AAC goals and the ability for a network to carry related amount of traffic. In particular, ND++ performance was investigated in diversified types of scenarios, with different network sizes which allowed the solution to be assessed under both the typical operation conditions and in the very demanding environment. This proves the applicability of ND++ also for large-scale MANETs. The presented analysis also confirms that ND++ design offers sufficient mobility support, which was assessed in different mobile environments – with network nodes moving both rather slowly and relatively fast while the network environment is a subject to frequent changes. Hence, the results also give an overview of the boundaries of ND++ performance, which will be observed in harsh operational set-ups. Such evaluation has also proved that ND++ mechanisms offer the sufficient robustness to provide an efficient and reliable operation in all of the investigated cases. Essentially, ND++ has proved to be scalable with regard to both an increased network size and the growing number of nodes starting simultaneous address queries. This feature was verified under diversified set-ups, scenarios and networking conditions, including node mobility. Finally, the evaluation of qualitative metrics led to a conclusion that the memory consumption of the proposed protocol is small comparing to other approaches and the solution is independent from routing used in the network.

The detailed evaluation presented in this paper allows us to ascertain that ND++ achieves very low protocol overhead levels along with the required reliability and acceptable latency figures, while properly addressing scalability and reliability concerns. In particular, very good overhead results allowed ND++ to overcome the key drawback of the stateless AAC methods proposed to date. Unlike those methods, however, ND++ has the required efficiency and robustness to allow practical implementations to be developed. Furthermore, its performance shows that it is one of the most efficient AAC protocols – including not only stateless, but also stateful methods. Moreover, ND++ can be easily integrated with existing intrinsic IPv6 solutions for AAC in fixed networks and its control overhead could be potentially limited by further integration with the IETF protocols proposed for MANET networks. Hence, we believe that ND++ could be used as one of the key enablers of IPv6-based future IoT networks serving the needs of future IoT networks.

REFERENCES

[1] *CISCO Website—Internet of Things (IoT)*. Accessed: Jan. 5, 2016. [Online]. Available: http://www.cisco.com/web/solutions/trends/iot/overview.html

[2] R. Rubin, "Business insider IoT report. Here comes the Internet of Things," BI Intell., DE, USA, Tech. Rep., Oct. 2013. Accessed: Jan. 5, 2016. [Online]. Available: http://www.slideshare.net/marklittlewood/business-insider-iot-report

[3] M. Grajzer, "IPv6 address auto-configuration for wireless mobile ad hoc networks," Ph.D. dissertation, Fac. Electron. Telecommun., Chair Commun. Comput. Netw., Poznan Univ. Technol., Poznań, Poland, 2016.

[4] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of Things: Vision, applications and research challenges," *Ad Hoc Netw.*, vol. 10, no. 7, pp. 1497–1516, Sep. 2012.

[5] R. van der Berg, "Machine-to-machine communications. Connecting billions of devices," OECD, Paris, France, OECD Digit. Economy Papers 192, Jan. 2012. doi: 10.1787/5k9gsh2gp043-en.

[6] R. Pozza, M. Nati, S. Georgoulas, K. Moessner, and A. Gluhak, "Neighbor discovery for opportunistic networking in Internet of Things scenarios: A survey," *IEEE Access*, vol. 3, pp. 1101–1131, 2015.

[7] J. Chase, "The evolution of the Internet of Things," Texas Instrum., Dallas, TX, USA, White Paper, 2013. Accessed: Jan. 5, 2016. [Online]. Available: http://www.ti.com/lit/ml/swrb028/swrb028.pdf

[8] D. Lake, A. Rayes, and M. Morrow, "The Internet of Things," *Internet Protocol Journal*, vol. 15, no. 3, Cisco Syst., Sep. 2012. Accessed: Jan. 5, 2016. [Online]. Available: http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_15-3/153_internet.html

[9] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013.

[10] A. Misra, S. Das, A. McAuley, and S. K. Das, "Autoconfiguration, registration, and mobility management for pervasive computing," *IEEE Pers. Commun.*, vol. 8, no. 4, pp. 24–31, Aug. 2001.

[11] R. Wakikawa, A. Tuimonen, and T. Clausen. (Mar. 2006). *IPv6 Support on Mobile Ad-hoc Network*. Accessed: Jan. 5, 2016. [Online]. Available: https://www.ietf.org/archive/id/draft-wakikawa-manet-ipv6-support-02.txt

[12] S. Thomson, T. Narten, and T. Jinmei, *IPv6 Stateless Address Autoconfiguration*, document RFC 4862, IETF, Fremont, CA, USA, Sep. 2007. Accessed: Jan. 5, 2016. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4862.txt

[13] R. Droms, J. Bound, B. Volz, T. Lemon, C. Perkins, and M. Carney, *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*, document RFC 3315, IETF, Fremont, CA, USA, Jul. 2003. Accessed: Jan. 5, 2016. [Online]. Available: https://tools.ietf.org/html/rfc3315

[14] *The Internet Engineering Task Force (IETF)*. Accessed: Jan. 5, 2016. [Online]. Available: http://www.ietf.org

[15] S. Nesargi and R. Prakash, "MANETconf: Configuration of hosts in a mobile ad hoc network," in *Proc. 21st Annu. Joint Conf. IEEE Comput. Commun. Soc.*, vol. 2, Jun. 2002, pp. 1059–1068.

[16] X. Wang and H. Qian, "Dynamic and hierarchical IPv6 address configuration for a mobile ad hoc network," *Int. J. Commun. Syst.*, vol. 28, no. 1, pp. 127–146, Jan. 2015.

[17] Cisco. (May 1997). *Duplicate MAC Addresses on Cisco 3600 Series*. Accessed: Jan. 5, 2016. [Online]. Available: http://www.cisco.com/c/en/us/support/docs/field-notices/misc/7.html

[18] T. Narten, E. Nordmark, W. A. Simpson, and H. Soliman, *Neighbor Discovery for IP Version 6 (IPv6)*, document RFC 4861, IETF, Fremont, CA, USA, Sep. 2007. Accessed: Jan. 5, 2016. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4861.txt

[19] E. Baccelli. (Feb. 2008). *Address Autoconfiguration for MANET: Terminology and Problem Statement*. Accessed: Jan. 5, 2016. [Online]. Available: https://tools.ietf.org/html/draft-ietf-autoconf-statement-04

[20] C. E. Perkins, J. T. Malinen, R. Wakikawa, E. M. Belding-Royer, and Y. Sun. (Nov. 2001). *IP Address Autoconfiguration for Ad Hoc Networks*. Accessed: Jan. 5, 2016. [Online]. Available: https://www.ietf.org/archive/id/draft-perkins-manet-autoconf-01.txt

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

[22] K. Weniger and M. Zitterbart. (Feb. 2002). *IPv6 stateless Address Autoconfiguration for Hierarchical Mobile Ad Hoc Networks*. Accessed: Jan. 5, 2016. [Online]. Available: http://tools.ietf.org/id/draft-weniger-manet-addressautoconf-ipv6-00.txt

[23] J.-S. Park, Y.-J. Kim, and S.-W. Park. (Jul. 2001). *Stateless Address Autoconfiguration in Mobile Ad Hoc Networks Using Site-Local Address*. Accessed: Jan. 5, 2016. [Online]. Available: http://tools.ietf.org/id/draft-park-zeroconf-manet-ipv6

[24] M. Grajzer. (Mar. 2011). *ND++—An Extended IPv6 Neighbor Discovery Protocol for Enhanced Duplicate Address Detection to Support Stateless Address Auto-Configuration in IPv6 Mobile Ad Hoc Networks*. Accessed: Jan. 5, 2016. [Online]. Available: https://tools.ietf.org/id/draft-grajzer-autoconf-ndpp-00.txt

[25] M. Grajzer, T. Żernicki, and M. Głąbowski, "ND++—An extended IPv6 neighbor discovery protocol for enhanced stateless address autoconfiguration in MANETs," *Int. J. Commun. Syst.*, vol. 27, no. 10, pp. 2269–2288, Oct. 2014.

[26] M. Grajzer and M. Głąbowski, "Performance evaluation of neighbor discovery++ protocol for the provisioning of self-configuration services in IPv6 mobile ad hoc networks," in *Proc. 16th Int. Telecommun. Netw. Strategy Planning Symp.*, Sep. 2014, pp. 1–6.

[27] M. Grajzer and M. Głąbowski, "Neighbor discovery++: A low-overhead address auto-configuration to enable robust Internet of Things architectures," in *Proc. Int. Conf. Inf. Syst. Archit. Technol.*, L. Borzemski, A. Grzech, J. Swiatek, and Z. Wilimowska, Eds. Cham, Switzerland: Springer, 2015.

[28] T. Clausen and P. Jacquet, *Optimized Link State Routing Protocol (OLSR)*, document RFC 3626, IETF, Fremont, CA, USA, Oct. 2003. Accessed: Jan. 5, 2016. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3626.txt

[29] M. Grajzer and M. Głąbowski, "Enhanced stateless address auto-configuration solution for low-overhead network self-configuration in IPv6 mobile ad hoc networks," in *Proc. IEICE Gen. Conf.*, Japan, Mar. 2015, pp. 10–13.

[30] M. Grajzer and M. Głąbowski, "On the probability of duplicate address detection with neighbor discovery++ protocol in IPv6 mobile ad hoc networks," in *Proc. IEICE Gen. Conf.*, Japan, Mar. 2015, p. 43.

[31] M. Grajzer and M. Głąbowski, "On the reliability of duplicate address detection in mobile ad hoc networks with neighbor discovery++ address auto-configuration protocol," in *Proc. IEICE Inf. Commun. Technol. Forum*, Manchester, U.K., Jun. 2015, pp. 3–5.

[32] C. Bernardos, M. Calderón, and H. Moustafa. (Jun. 2010). *Survey of IP Address Autoconfiguration Mechanisms for MANETs*. Accessed: Jan. 5, 2016. [Online]. Available: https://tools.ietf.org/html/draft-bernardos-manet-autoconf-survey-05

[33] L. J. G. Villalba, J. G. Matesanz, A. L. S. Orozco, and J. D. M. Díaz, "Auto-configuration protocols in mobile ad hoc networks," *Sensors*, vol. 11, no. 4, pp. 3652–3666, Mar. 2011.

[34] N. C. Fernandes, M. D. D. Moreira, and O. C. M. B. Duarte, "An efficient and robust addressing protocol for node autoconfiguration in ad hoc networks," *IEEE/ACM Trans. Netw.*, vol. 21, no. 3, pp. 845–856, Jun. 2013.

[35] S. M. Gammar, A. Abidi, and F. Kamoun, "Distributed address auto configuration protocol for MANET networks," *Telecommun. Syst.*, vol. 44, nos. 1–2, pp. 39–48, Jun. 2010.

[36] C. Huitema and B. Carpenter, *Deprecating Site Local Addresses*, document RFC 3879, IETF, Fremont, CA, USA, Sep. 2004. Accessed: Jan. 5, 2016. [Online]. Available: http://www.ietf.org/rfc/rfc3879.txt

[37] N. H. Vaidya, "Weak duplicate address detection in mobile ad hoc networks," in *Proc. 3rd ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, Jun. 2002, pp. 206–216.

[38] K. Weniger, "Passive duplicate address detection in mobile ad hoc networks," in *Proc. IEEE Wireless Commun. Netw.*, vol. 3, Mar. 2003, pp. 1504–1509.

[39] S. R. Hussain, S. Saha, and A. Rahman, "SAAMAN: Scalable address autoconfiguration in mobile ad hoc networks," *J. Netw. Syst. Manage.*, vol. 19, no. 3, pp. 394–426, Sep. 2011.

[40] S. Boudjit, A. Laouiti, P. Mühlethaler, and C. Adjih, "Duplicate address detection and autoconfiguration in OLSR," *J. Universal Comput. Sci.*, vol. 13, no. 1, pp. 4–31, 2007.
[41] S. Boudjit, "Autoconfiguration algorithm for a multiple interfaces adhoc network running OLSR routing protocol," *Int. J. Comput. Netw. Commun. (IJCNC)*, vol. 5, no. 1, pp. 153–170, Jan. 2013.
[42] P. Ruiz and P. Bouvry, "Survey on broadcast algorithms for mobile ad hoc networks," *ACM Comput. Surv.*, vol. 48, no. 1, Sep. 2015, Art. no. 8.
[43] S. Boudjit, C. Adjih, A. Laouiti, and P. Muhlethaler, "A duplicate address detection and autoconfiguration mechanism for a single-interface OLSR network," in *Technologies for Advanced Heterogeneous Networks*. Berlin, Germany: Springer-Verlag, 2005, pp. 128–142.
[44] S. Miyakawa and R. Droms, *Requirements for IPv6 Prefix Delegation*, document RFC 3769, IETF, Fremont, CA, USA, Jun. 2004. Accessed: Jan. 5, 2016. [Online]. Available: https://tools.ietf.org/html/rfc3769
[45] R. Droms and O. Troan, *IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) Version 6*, document RFC 3633, IETF, Fremont, CA, USA, Dec. 2003. Accessed: Jan. 5, 2016. [Online]. Available: https://tools.ietf.org/html/rfc3633
[46] M. Grajzer, T. Żernicki, and A. B. Cavalcante, "Autokonfiguracja oraz zarzadzanie uszkodzeniami w autonomicznych mobilnych sieciach ad hoc (address autoconfigurationa and fault management in autonomic mobile ad hoc networks)," *Przeglad Telekomunikacyjny, Telecommun. Rev.*, vol. 2011, pp. 424–427, Jun. 2011.
[47] R. Hinden and S. Deering, *IP Version 6 Addressing Architecture*, document RFC 4291, IETF, Fremont, CA, USA, Feb. 2006. Accessed: Jan. 5, 2016. [Online]. Available: https://tools.ietf.org/html/rfc4291
[48] S. Deering and R. Hinden, *Internet Protocol, Version 6 (IPv6) Specification*, document RFC 2460, IETF, Fremont, CA, USA, Dec. 1998. Accessed: Jan. 5, 2016. [Online]. Available: http://www.rfc-editor.org/rfc/rfc2460.txt
[49] L. Hogie, "Mobile ad hoc networks: Modelling, simulation and broadcast-based applications," Ph.D. dissertation, Univ. Luxembourg, Luxembourg, 2007, pp. 102–105.
[50] W. Peng and X. Lu, "AHBP: An efficient broadcast protocol for mobile ad hoc networks," *J. Comput. Sci. Technol.*, vol. 16, no. 2, pp. 114–125, Mar. 2001.
[51] R. Ogier and P. Spagnolo, *Mobile Ad Hoc Network (MANET) Extension of OSPF Using Connected Dominating Set (CDS) Flooding*, document RFC 5614, IETF, Fremont, CA, USA, Aug. 2009. Accessed: Jan. 5, 2016. [Online]. Available: http://wiki.tools.ietf.org/html/rfc5614
[52] J. Lipman, P. Boustead, J. Chicharo, and J. Judge, "Optimized flooding algorithms for ad hoc networks," in *Proc. 2nd Workshop Internet, Telecommun. Signal Process.*, Gold Coast, QLD, Australia, 2003, pp. 1–7.
[53] A. Conta, S. Deering, and M. Gupta, *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*, document RFC 4443, IETF, Fremont, CA, USA, Mar. 2003. Accessed: Jan. 5, 2016. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4443.txt
[54] M. Sayrafiezadeh, "The birthday problem revisited," *Math. Mag.*, vol. 67, no. 3, pp. 220–223, 1994.
[55] T. R. Henderson, P. A. Spagnolo, and G. Pei, "Evaluation of OSPF MANET extensions," Boeing, Chicago, IL, USA, Tech. Rep. D950-10897-1, 2005.
[56] P. Jacquet, A. Laouiti, P. Minet, and L. Viennot, "Performance analysis of OLSR multipoint relay flooding in two ad hoc wireless network models," INRIA, France, INRIA Res. Rep. 4260, 2002. [Online]. Available: http://hal.inria.fr/docs/00/07/23/27/PDF/RR-4260.pdf
[57] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Commun. Mobile Comput.*, vol. 2, no. 5, pp. 483–502, Sep. 2002.
[58] M. Grajzer and M. Głąbowski, "On IPv6 experimentation in wireless mobile ad hoc networks," *J. Telecommun. Inf. Technol.*, vol. 3, pp. 71–81, Sep. 2014.
[59] *NS-3 Simulator Website*. Accessed: Jan. 5, 2016. [Online]. Available: http://www.nsnam.org
[60] T. Clausen, C. Dearlove, and J. Dean, *Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP)*, document RFC 6130, IETF, Fremont, CA, USA, Apr. 2011. Accessed: Jan. 5, 2016. [Online]. Available: https://tools.ietf.org/html/rfc6130

MONIKA GRAJZER received the M.Sc. (Hons.) and Ph.D. degrees in telecommunications from the Poznań University of Technology, Poland, in 2008 and 2016, respectively. She is a Co-Owner and Chief Scientist with Gido Labs sp. z o.o. R&D Company. Her research interests include wireless technologies, in particular in mobile ad hoc networking, network configuration protocols and autonomic, and self-managing networking for the future Internet and the Internet of Things. She was involved in several European and national collaborative research projects. Her research resulted in the provisional patent application, the IETF Internet Draft, as well as several research articles and papers.

MARIUSZ GŁĄBOWSKI received the M.Sc., Ph.D., and D.Sc. (Habilitation) degrees in telecommunication from the Poznań University of Technology, Poland, in 1997, 2001, and 2010, respectively. Since 1997, he has been with the Department of Electronics and Telecommunications, Poznań University of Technology. He is engaged in research and teaching in the area of performance analysis and modeling of multiservice networks and switching systems. He has authored/coauthored four books, seven book chapters, and over 150 papers. He has refereed articles for many international conferences and magazines.