CONNECTIVITY-AWARE AND ADAPTIVE MULTIPATH ROUTING ALGORITHM FOR MOBILE ADHOC AND SENSOR NETWORKS

Bayrem TRIKI[1], Slim REKHIS[2] and Noureddine BOUDRIGUA[2]

[1] University of Sousse
[2] University of Carthage

ABSTRACT

We propose in this paper a connectivity-aware routing algorithm and a set of related theorems. This algorithm allows nodes in Mobile Adhoc and Sensor Networks (MASNets) to provide the highest connectivity life time to a specific destination since the issuance of data becomes a necessity for MASNets. In the proposed Solution, nodes in MASNets are able to specify the disjointness degree of the available paths allowing the discovery of the optimal set of backup routes and consequently enhance the survivability of the connectivity. These nodes perform an on-demand discovery and a generation of a set of routes, by specifying a disjointness threshold, representing the maximal number of nodes shared between any two paths in the set of k established paths. The proposed multipath routing algorithm, is adaptive, secure, and uses labels to carry the disjointness-threshold between nodes during the route discovery. A set of security mechanisms, based on the Watchdog and the digital signature concepts, is used to protect the route discovery process.

KEYWORDS

k-x-connectivity, adaptive routing, multipath, watchdog, Mobile Adhoc And Sensor Networks

1.INTRODUCTION

Mobile Adhoc and Sensors Networks (MASNets) are highly demanded in the current Internet of Things projects. In such projects the relationship between the physical and the virtual environments is too close. For example, wireless sensors networks are in deep relationship with cloud computing networks to store collected data [20]. Consequently, MASNets require more and more bandwidth and an important adaptivity skill to alleviate the resource constraints nature of MASNets and to provide a guaranteed data delivery [26].

Used sensors nodes in MASNets have several limiting factors related to infrastructure-less architecture and dynamic network topology [18]. Maintaining the availability of routes in Mobile and Adhoc Sensor Networks (MASNets) during network runtime represents a challenging problem [23, 3, 8]. In fact, they may be easily broken due to nodes mobility, links and nodes failure, and radio interference. In addition, links could have limited bandwidth, and potential network congestion could lead to resource starvation in nodes. Multipath routing algorithms were proposed as a solution for these issues[22, 28], making a node able to establish and use simultaneously k paths to the destination [24, 30]. The main objectives of algorithms is to enforce fault tolerance [29], load balancing [19, 27] and minimize end-to-end delivery delay [13]. Once the k paths are established, a node sends datagrams redundantly over the alternative paths to mitigate failure problems [1, 17].

DOI : 10.5121/ijwmn.2015.7104

For example, The Multipath On-Demand Routing Algorithm (MDR) proposed in [4] ensures the establishment of disjoint paths between the source and the destination. It splits the original data packet into k parts and sends these new sub-packets instead of the whole packet across available paths. The destination, which eventually receives one of the route request messages, will only be aware of the existence of one path. It returns a route reply containing a supplementary field that indicates the number of hops it traveled so far. Each node that receives a route reply, increments the hop count of the message and then forwards the message to the neighbor from which it got the original route request. This solution may cause a lot of overhead in the network. In dynamic multi-path source routing (DMSR) [21], each node must write its bandwidth into forwarded packets in order to find better paths based on the available bandwidth. Best Effort Geographical Routing Protocol (BEGHR) exploits nodes position to forward data, and requires the use of a positioning system such as Global Position Systems (GPS). However, the demands for resources at the different nodes are quite high, which affect battery lifetime. Label-based Multipath Routing (LMR) [6] broadcasts a control message throughout the network for a possible alternative path. It defines a lexicographic total order on the label for each node. The destination should have the minimum label. If a node is a local minimum with respect to its neighbors, it will not have a path to the destination. In this case, it increases its label and reverses some or its entire links. The number of routing paths can be higher than two, while showing relatively long end to end transmission delay.

The maximal number of possible paths that could be established between any source and destination nodes is very sensitive to the mobility scheme of sensors nodes, which may affect the extent to which these paths are disjoint [12, 9]. The sensitivity of the used applications and the variation of the nodes density from a network location to another, makes it important to extend the multipath routing algorithms so that a new parameter, called disjointness factor (representing the maximal number of nodes shared between any two paths in the set of established paths), will be specified and exploited during the routes discovery to create a trade-off between fault-tolerance and performance [9].

We develop in this paper a set of validation theorems of the Secure Multipath Routing Algorithm for Mobile Adhoc and Sensor Networks (SeMuRAMAS) proposed in [2], which allows nodes in MASNet to perform an on-demand discovery and generation of a set of paths, while specifying both of the number of separate paths k to establish and the disjointness factor *x*. A new concept called *k*-*x*-connectivity is therefore introduced. We propose in this paper a set of theorems used for the adjustment of parameters *k* and *x* allowing an optimized connectivity life time. The algorithm is adaptive, secure, and uses labels to carry the disjointness-threshold between nodes during the route discovery. SeMuRAMAS exploits the watchdog concept to tolerate several types of routing attacks, and uses threshold signature, including the elliptic threshold signature algorithms, to protect the integrity of the exchanged datagrams and prevent attackers from forging routes. The conducted simulation estimates the additional overhead, showing the efficiency of the algorithm.

The paper contribution is seventh-fold. First, the proposed multipath routing algorithm is adaptive. It allows a source node to tune the disjointness factor to a suitable value before establishing a path and sending data. The value of the threshold may depend on the sensitivity of the message to be sent and the rate of routes failures during previous communications.
Second, we provides a set rules allowing the adjustment of the parameters k and x needed for the tuning of the disjointness level between discovered paths. Third, thanks to the use of the watchdog mechanism and digital signature, the algorithm is secure. In the WSNs, threshold signature does not require an extensive number of stored keys per sensor node. The technique of

sur-signature could be used for the generation of evidences, which could be used by a digital investigation scheme to prove the identity of malicious nodes and trace and analyze the attack. Fourth, the algorithm is tolerant to a large set of routing attacks such as wormhole. Fifth, SeMuRAMAS takes into consideration the characteristics of Adhoc and wireless sensor networks, in terms of architecture, node resources limitation, and attacks categories. Sixth, the variation of the network topology could make nodes unable to establish multiple disjointness paths to the destination. If the MASNet is used for some application requiring a high level of tolerance to nodes and link failures, by tuning the value of the disjointness factor, nodes could cope with topology variation and continue benefiting from a degraded level of fault tolerance. Seventh, when it is impossible to generate a route for the specified threshold, the proposed algorithm advices the source node about the optimal disjointness value to be used by the destination mobile nodes.

This paper is organized as follows. The next section describes the requirements to be fulfilled by a secure and multipath routing algorithm. Section 3 presents the problem formulation related to the multipath routing process. Section 4 describes the proposed routing algorithm. Section 5, presents security mechanisms used by the proposed algorithm. In Section 6, a validation of the proposed algorithm is addressed. Section 7 presents and discusses simulation results. The last section concludes the work.

2. REQUIREMENTS FOR ADAPTIVE MULTIPATH ROUTING IN AD HOC AND SENSOR NETWORKS

This section presents seven important requirements to be fulfilled by an adaptive multipath routing algorithm for mobile ad hoc and sensor networks

First, although ad hoc and wireless sensor networks could be infrastructure-less based, and use multihop communication [10], they may show many differences including, but not limited to, the availability of computational resources, battery energy availability [15], and the security of the area in which the network is deployed. Since the proposed algorithm should be operational on both Ad Hoc and Wireless Sensor Networks, it should take into consideration those constraints specific to WSN.

Second, it would be better that the algorithm be reactive rather than proactive. In fact, the source node is usually the node that specifies the disjointness factor value. This value may depend on the sensitivity of the data to be transmitted from a session to another.

Third, the algorithm must be distributed where intermediate nodes should start learning and gathering information regarding potential available path as long as the route request datagrams is forwarded. In fact, distributed computation has a better chance to withstand failure in the case of attacks.

Fourth, the routing algorithm must preserve the network performance. Especially, the overhead caused by the storage of information regarding potential usable routes, and the distributed computation of the routing paths by intermediate nodes in the networks, should be reduced.
Fifth, if the source node fails to establish a set of paths with the specified threshold value, the algorithm should allow it to receive an indication regarding the minimal disjointness value supported by the current network topology. Such a mechanism will prevent the source node from generating additional failures and overloading the network bandwidth.

Sixth, the algorithm should include the generation of evidences regarding the identities and behavior of nodes involved during the establishment of the multiple routes. This is of utmost importance if a digital investigation scheme is used to trace back an occurred attack, locate the malicious nodes, and prove the existence of fake routes.

Seventh, the proposed algorithm should be tolerant to attacks targeting routing protocols. In wormhole attacks, for instance, a node can perform a high-powered transmission of a route request message to a non neighbor node, forcing the routing algorithm to include it in the established routing paths. This attack makes the malicious node appear as a highly connected node, while in reality, it is connected to a few number of nodes. In this context, the proposed multipath algorithm should verify that packets are properly forwarded in the network and identities of intermediate nodes are appended securely to the routing requests.

3. MULTIPATH ROUTING: PROBLEM FORMULATION

We consider a set of deployed mobile nodes in a mesh network. Each two nodes, say *S* and *D*, can communicate together in a multi-hop fashion. Depending of the density of the network, a set of intermediate nodes could be used to forward the exchanged datagrams between *S* and *D*. Several paths could be available between these nodes. The source node *S* can discover *k* paths in the set of generated routes, which can be disjoint or share some nodes. We assume that the number of shared nodes between any two paths in the set of discovered *k* paths, could not share a number of nodes more than a predefined disjointness factor, denoted in the sequel by *x*.

Its important to formally express the relationship between *x* and *k* values. For this purposes we represent all possible paths between the source and the destination as a graph of nodes. As shown in Figure 1, this graph is used in the form of a tree where the head is the destination node *D* and all leafs represent the same source node *S* accessed over different paths. A path in the tree from the source to the destination node represents a routing path. The parent of a node in this path represents the next hop.

In the tree, all paths can share some intermediate nodes excepting the source node at the leaves. We consider that each hop from a node to another represents an increment of the depth level. We denote by L_i the i^{th} level in the tree. This means that each path is represented by a number of levels equal to the number of used intermediate nodes. The first level, denoted by L_0, is assigned to the destination node and the last level is assigned to the source node. Each time that there is one step backward in a specific path to the source node, we increment this level by 1. We denote by a bifurcation a set of edges connection between a parent node and at least two children nodes.

In a first analysis, we express the possible disjointness *k* paths in terms of the disjointness factor value *x* using a specific parsing algorithm. The latter can apply a depth-first search on the tree to look for the maximal value of *k* (i.e., number of paths) sharing at most *x* nodes. The input of this algorithm is the constructed tree and the value of the disjointness factor *x*. The output of this algorithm will be the value of the parameter *k* allowing estimating the possible value of *k* given the value of *x*. Breadth First Search is performed on the tree by counting for every bifurcation *B* encountered until level L_x, the value C_B representing the number of child nodes obtained from that bifurcation minus one. The value *k* will be equal to one higher than the value of the obtained C_B.

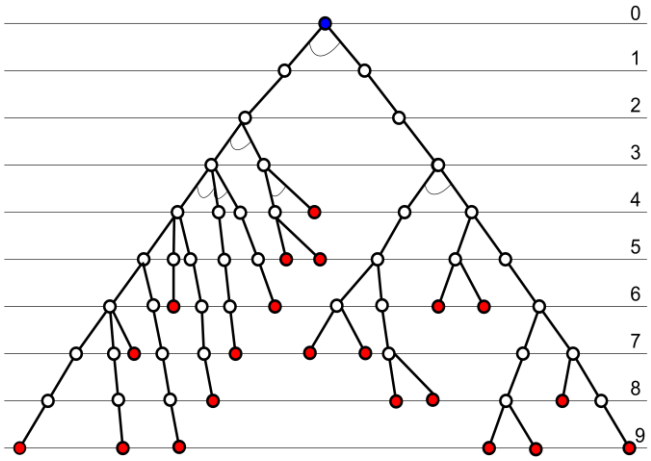

Figure 1. Tree of paths between the source and the destination node

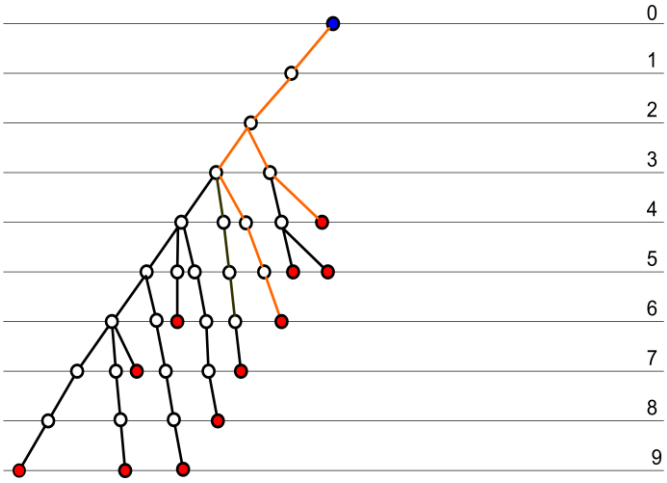

Figure 2. Example of paths between the source and the destination node

In a second analysis, we express the maximal value of the disjointness factor *x* given the value of *k* representing the number of required paths by applying a depth-first search on a branch of the tree to determine *k* paths sharing at most *x* nodes. The input of this algorithm is the constructed tree and the value of *k*. The output will be the value of the disjointness factor x. This algorithm consists in determining the minimal level, say l_{min}, in the tree showing a total number of leaf nodes higher or equal to *k* (since each leaf represents a path). The value of *x* will be equal to $l_{min} - 1$.

The routing algorithm that we are proposing in this paper allows to determine the value of *k* starting from the specified value of *x*. Therefore, it performs a breadth-first search on the network topology modeled by the tree described above. In order to determine the number of bifurcations, each intermediate node should store the different paths allowing packets to reach the source node. The destination node should inform nodes in the network about the value of the disjointness factor it determined. In addition, all leaves (representing instances of the source node) should receive all paths computed by those intermediate nodes, in order to be able to reconstruct the tree of available paths. Several datagrams will be used to this purpose and will be described in the next section.

4. DESCRIPTION OF THE PROPOSED ADAPTIVE ROUTING ALGORITHM

The proposed SeMuRAMAS [2], as a multipath adaptive routing algorithm, extends the Dynamic Source Routing (DSR) algorithm [11], which is a reactive routing approach widely used as a basis for a large set of extended routing protocols. With regard to the existing works in the literature, SeMuRAMAS has several contributions.

First, it is adaptive to the network topology variation. It allows a source node to tune the disjointness threshold to a suitable value before establishing a path and sending data. The value of the threshold may depend on the sensitivity of the message to be sent or the rate of broken routes during previous communications. Second, thanks to the use of the watchdog mechanism and digital signature, the algorithm is secure. In the WSNs, threshold signature does not require an extensive number of stored keys per sensor node. The technique of sur-signature could be used for the generation of evidences, which could be used by a digital investigation scheme to prove the identity of malicious nodes and trace and analyze the attack.

Third, the algorithm is tolerant to a large set of routing attacks such as wormhole. Fourth, SeMuRAMAS takes into consideration the characteristics of Ad Hoc and wireless sensor networks, in terms of architecture, nodes resources limitation, and categories of attacks.

Fifth, the variation of the network topology can make nodes unable to establish multiple disjointness paths to the destination. If the MASNet is used for some application requiring a high level of tolerance to nodes and link failures, by tuning the value of the disjointness threshold, nodes could cope with topology variation and continue benefiting from a degraded level of tolerance. Sixth, in the case of the absence of paths with specified threshold, the proposed algorithm advises the source node about the optimal disjointness value supported by the current MASNet topology to ensure the connectivity between the source and the destination mobile nodes.

4.1 SeMuRAMAS phases

This protocol includes two steps: route discovery and route maintenance, which allow to discover and maintain possible multiple paths between any two mobile source node, say *S*, and destination mobile node, say *DM*. Especially, in the context of wireless sensor networks, *DM* should represent the base station.

Route Discovery: This is the mechanism used when *S* wants to establish a set of paths with *DM*. Route Request datagrams, say *RReq*, is sent by *S* when it does not already have a route to *DM*. The entirely on-demand properties allow SeMuRAMAS to minimize the overhead and specify the path-disjointness factor value. After receiving a list of potential paths, *S* computes all paths to the destination satisfying the specified threshold, chooses the list of paths to be used, caches the remaining ones, and starts sending the data. Keeping information regarding unused paths, allows a rapid reaction to route modification and decreases the overhead related to the generation of a new *RReq*.

Route Maintenance: This is the mechanism used by the intermediate nodes to let *S* update the list of paths in use when the network topology changes or when some routes are broken due to a routing attack or because nodes can change their state from active to sleeping. This mechanism is based on allowing intermediate nodes using the watchdog concept for every packet they forward [14] to detect the identities of misbehaving nodes or detect route errors. If the next hop appears to

be broken, a route error packet, say *RErr*, is generated and sent to *S* in order to decrease the number of possible path to the destination. *S* will consider all the paths as broken and will attempt to use another route stored in its cache, which allows maintaining the *k*-*x*-connectivity. If none backup route to *DM* is in the cache, the source node generates again the Route Discovery mechanism.

4.2 SeMuRAMAS route Discovery

SeMuRAMAS uses six kinds of messages during the route discovery:

- Route Request Datagram (*RReq*): It is the first packet to be broadcasted by a mobile node, which wants to establish multiple routes to a destination node. Every intermediate node exploits this datagram to discover incomplete routes in the network. It also appends its identity to the *RReq* and broadcasts it to its neighbors.
- Route Response Datagram: It is sent back by the destination upon reception of the *RReq*. This datagram contains the optimal path and is source routed to the node which generated the *RReq*.
- Notification Datagram: It is used by the destination node to ask intermediate nodes to forward the information they learned regarding the routes to the source node. The information would have been invisible by the destination mobile node when it received the *RReq* datagram.
- List forwarding Datagram: It is used by intermediate nodes to forward the information they stored regarding the existing paths in the network.
- Route Error Datagram: It is sent by an intermediate node to the source node when it detects a route failure. It also lets the source node update the set of paths it uses to reach the mobile destination node.
- Threshold Tuning Datagram: is used by an intermediate node to indicate the value by which the threshold should be increased to let the establishment of the multiple requested paths be possible.

Network discovery: When a mobile node, say *S*1, joins the network, it broadcasts a two-hop HELLO message, which includes its identity and has a Time To Live (TTL) value equal to 2. Any node, say *S*2, which hears the message, includes the identity of *S*1 in its list of one-hop neighbors, sets the TTL value of the HELLO message equal to 1 lower than its received value, and forwards the datagram. Any node, say *S*3, that hears the message includes the identity of *S*2 in its list of one-hop neighbors and *S*1 in its list of two-hop neighbors, sets the TTL value of the HELLO message equal to 1 lower than its received value, and discards the datagram. To be considered as active, every node should periodically send a two-hop HELLO message and follow the above described process. This allows each node to maintain two up-to-date lists. The first is the list of neighbors and the second shows for each neighbor the list of its neighbors. The two lists will support the detection of routing attacks (described later in Section 5).

Route request generation and forwarding: A node which wants to establish a path to the mobile destination node, say *MD* initiates the route discovery by generating a *RReq* datagram to *MD*, and broadcasting it in the network. Every generated *RReq* includes a three-tuple information: ⟨*Seq*, *RRec*, *Dt*⟩. *Seq* stands for a sequence number which should be different for every new generated *RReq*. The sequence number together with the IP address of the sender allow to uniquely identify the *RReq* and associate it to the subsequent generated responses. *Dt* is the disjointness factor which is set by the sender to specify the maximal number of nodes that could be shared by any two paths among the set of paths to establish with the destination node. The value of *Dt* remains unchanged during the forwarding of the *RReq* to the destination. *RRec* is a

route record which is used to include the path followed by the *RReq* to reach *MD*. In fact, when a node in the network receives a copy of this datagram for the first time, it appends its identity to the *RRec* field, broadcasts it to its neighbors. Every node, say *N*, including *MD*, which receives a second copy of the datagram, extracts the content of the *RRec* field. The latter provides a path from the sender to the node *N*. The node *N* will append the content of *RRec* together with the value of *Seq* to a list stored locally, entitled *RP*, which stands for list of Received Paths. Then it discards the datagram.

Route response generation and forwarding: The Route Response includes five-tuple information < *Seq*, *RRec*, *nh*, *RNC*, *RPD* > , where Seq represents the value of the sequence number that appeared in the received Route Request datagram, and RRec is the shortest Route Record, in terms of number of records, which is collected from the different copies of the received Route Requests. The value of *nh* represents the number of hops traversed by the route response datagram within a link that will be shared by at least two routes. Initially, nh is set to 0 by the destination node. *RNC* indicates the remaining number of nodes that could still be shared by the fragments of routes to generate. It is set by *MD* to the value of the disjointness factor received in the route request. Finally, *RPD* contains the *RP* list stored in d after replacing any two entries *E* and *E*' in *RP* list, such that *R*(*E*) shares more than *RNC* nodes with *R*(*E*'), by the shortest one. The Route Response datagram will be source routed using the reverse series or nodes in *RRec*. Before forwarding the received Route Response datagram, every intermediate node checks if it has a non-empty *RP* list associated to the triplet (*Seq*, *s*, *MD*). If it is the case, it decreases the value of *RNC* by *nh*.

Notification Datagrams generation and forwarding: Any intermediate node, including *MD*, should generates a Notification Datagram when to every entry in its *RP* list when has already forwarded the Route Response datagram and he has a non-empty RP list associated to the triplet (Seq,s,MD) specified in this datagram. A Notification Datagram contains seven-tuple information < *s*, *d*, *Seq*, *nh*, *RNC*, *RPD*, *P* > , where *s*, *d*, *Seq*, *nh*, and *RPD* have the same value already specified in the routed Route Response datagram. *RNC* is set equal to the disjointness factor value received in the Route Response datagram. *P* stands for the reverse route extracted from the shortest entry (in terms of number of nodes) in the *RP* list associated to the triplet (*Seq*, *s*, *d*). *P* will be used to source route the datagram to s. Every intermediate node, which receives this datagram, checks whether its RP list, associated to the triplet (*Seq*, *s*, *d*), is empty. If it is the case, it increments nh, otherwise, it decreases RNC by nh and sets nh to 0. If RNC is still higher than 0, it forwards it to the next node in the route and generates another Notification Datagram for every entry in its *R*P list, and source route it to node s. SeMuRAMAS keeps the parameters s, d, Seq, nh, and RPD will have the same values in the already received Notification Datagram except the updated value of RNC. It allows discovering any existent solution in the form of concatenation of two kinds of sub-paths: the first sub-path is shared by another route to the destination with a length shorter than the disjointness factor *x*, the second sub-path never occurs in other route to the destination, discovered by SeMuRAMAS.

List Forwarding Datagrams generation and forwarding: Upon generation of the Notification Datagram, every node, which has a non-empty *RP* list associated to (*s*, *d*, *Seq*), consecutively applies two filters, say *F*1 and *F*2, on that list. The filter F1 eliminates from *RP* any entry, say E, such that there is E in RPD where *R*(*E*) shares more than *RCN* − 1 nodes with R(E'). Note that the value of RPD is the same included in the already generated Notification Datagram. The filter *F*2 replaces any two entries *E* and *E*' in the *RP* list, such that R(*E*) shares more than *RCN* − 1 intermediate nodes with *R*(*E*'), by the shortest one of them. Paths filtered by F2 from the RP list will be sent later to the source node with the RP list in order to be used as backup paths if the source node is not able to find any path to the destination node. After applying the two filters, if the RP list is not empty, the node generates a List of Paths datagram containing four-tuple

information < *s, d, Seq, RP, Rte* > , where Rte is the shortest path obtained from the remaining entries in RP. It will be used to source route the datagram from the current node to s. After generation, the RP list is deleted. Two categories of nodes can be used: nodes with high storage capacity and nodes with limited storage capacity. Each node knows the category of its neighbors. A node with a low storage capacity has the possibility to send parts of the data it stores only to neighbor nodes with high storage capacity. In fact, the receiving node should send back the data to the sender before it sleeps or goes out of its coverage. If the sender memory is still full, the receiving node should find a neighbor node, which is also a neighbor of the sender and has a high storage capacity, transfer the data to that node, and inform the sender about its identity.

Threshold Tuning Datagram generation and forwarding: Each time that a mobile node sets the *RCN* to 0, it executes the two filters *F*1 and *F*2 on the content of the *RP* to discard paths sharing more than the authorized disjointness factor. The mobile node computes the minimal number of shared nodes, say *n*, between the remaining paths in the *RP* and sends this value to the source node within the *TTD* datagram. This value could be exploited by the source node, in the case where it is unable to establish the set of paths satisfying the requested threshold. It let to determine the best suitable threshold value that could be guaranteed by the network topology. This value will be *n* greater than the last used threshold value.

4.3 Reconstruction of routes by the source

The aim of the reconstruction process is to generate *k* paths satisfying the disjointness factor *x*. If several combinations are possible, the sender can select the one which uses the minimal number of shared nodes, or select the one which uses the shortest *k* possible paths. The first alternative could be chosen if availability is more sensitive than delivery delays. We remind, that the value of *x* specified in the route request *RReq*, has prevented intermediate nodes to forward useless list of paths, which, if assembled together by the source node, would generate routes that include a number of shared nodes higher than what is expected by *x*.

To reconstruct the set of paths, the following algorithm is executed. Let L_{cp} be the list of complete paths satisfying the threshold *x* to generate by the algorithm, and *LP* be the series of path received in the different List Forwarding Datagrams sent by intermediate nodes in the network. L_p is a n-tuple of the form $< p_1, ..., p_n >$, where every p_i ($\in LP$) represents any path of the form of $< S_1, ..., S_{bl} >$, where S_1 represents the identity of the node, which initiated the route discovery, S_2 to S_{bl-1} represent intermediate nodes and S_{bl} represents the identify of any intermediate node in the network including the destination mobile node *MD*, which discarded a second copy of the *RReq* datagram.

Starting with a FIFO queue, say *Q*, containing the set of paths in *RPBS* received by *MD*. Until $Q = \langle\rangle$ or $L_p = \emptyset$, the algorithm executes the following tasks: Let $R = < S_1', ..., S_n' >$, be the first routing path in *Q*, where S_1' stands for the source node and S_n' stands for *MD*. Starting from a value of *i* equal to *n*, and until *i* becomes equal to 1 (i.e., *i* is decreased by 1 in every loop), the algorithms checks for every $p \in L_p$ if $S_i \in R$ corresponds to the last node $S_{bl} \in p$. If it is the case, the source node will generate a path equal to $\langle S_1, ..., S_{bl}, S_{i+1}', ..., S_n' \rangle$, appends it to L_{cp} and *Q*, and deletes *p* from L_p. If *i* becomes equal to 1, the algorithm deletes *R* from *Q*. For the particular case where *MD* has generated an *RPBS* containing paths which share more than *x* nodes, the L_{cp} needs to be filtered by keeping the shortest path from those having more than *x* nodes shared with paths in the *RPBS*. Based on the content of L_{cp}, the source node will be able to select the best combination of paths, in terms of number of hops, satisfying the values of *k* and *x*.

If $L_{cp} = \emptyset$, meaning that the request k paths cannot be established while satisfying the threshold x, the source node should have received at least a *TTD* datagram. It selects the minimal value, say v, from those indicated by the received *TTD* datagrams. Later, it can generate again a new route discovery where x will be v greater than the last used x value.

4.4 Route maintenance

A path can fail due to collisions and/or nodes mobility. It is essential to recover broken paths immediately to ensure the reliability of data. After route establishment and during data forwarding, when a node in an established route fails to send the packet to the next hop, or detects that a neighbor is not forwarding the datagram, SeMuRAMAS considers the route as broken and sends a Route Error *RErr* to the source node to inform it about the identity of the unavailable node. This mechanism is strengthened by applying a watchdog mechanism described in the Section 5. Upon reception of this RErr packet, the source node deletes any alternative route in the k established path that contains the broken node. It tries to replace it by one of the available routes in its cache, and verifies if the set of k paths to use still have, at the maximum possible, x common nodes. If it is not the case, the route discovery step is initiated again.

4.5 Illustrative example

In the example presented in Figure 3, the source node *A* needs to establish three ($k = 3$) routes to the destination node *MD* sharing at maximum two nodes between them (disjointness factor $x = 2$). *A* generates and broadcasts a *RReq* with a disjointness factor x equal to 2. In the Figure, nodes are represented by circles, and an edge connects two nodes if they are able to directly communicate together. The *RP* lists stored by each node are represented by rectangles. Anode *B* receives the first packet directly from *A* and receives a second copy though two other paths which are $< A, C, B >$ and $< A, C, E, B >$. As each time node *B* drops a duplicated packet, it stores the routing path used by this copy in its *RP* list, the routing path *RP* will be equal to $< < A, C, B > < A, C, E, B > >$. The *RReq* is forwarded to neighbor nodes until it reaches *MD* on the shortest path $R_{sp} = < A, B, Din - depth, F, MD >$.

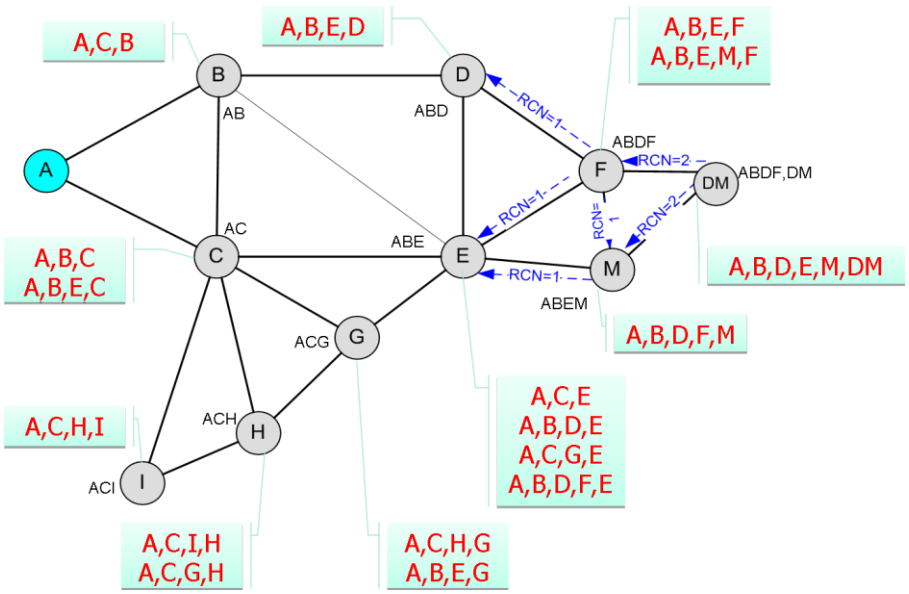

Figure 3. Example of received path lists stored nodes in a network using SeMuRAMAS

As shown in Figure 3, based on the number of minimum common nodes *RCN* which is set to 2 in the Notification Packet *ND*, the *RP* lists are only sent by nodes *F*, *M*, *D* and *E*. In fact, at the response step, when Node *F* receives the *ND* from *MD*, it decreases the value of *RCN* to 1 and forwards this packet to its neighbors. This node applies the filter *F*1 on its *RP* list based on the content of the *RPBS* list received for the destination node. Note that this *RPBS* contains the two following paths $< A, B, D, F, MD >$ and $< A, B, D, E, M, MD >$.

Using the filter *F*1, paths $< A, B, E, F >$ and $< A, B, E, M, F >$, in the *RP* list of the node *F*, will be eliminated because they have more than $RCN - 1 = 1$ shared node with the *RPBS*. As the *RP* list of node *F* is empty, its *RP* list will be sent back to the source node. When node *M* receives the *NP* from *MD*, it decreases *RCN* to 1 and applies the two filters *F*1and *F*2 on its *RP* list. By applying *F*1, node *M*, will discard the path $< A, B, D, F, M >$ from its *RP* list, because it has more than $RCN - 1 = 1$ shared nodes with the first path $< A, B, D, F, MD >$ of the *RPBS* list. When the node *M* receives a second copy of *ND* from the node *F* it will discard it. When nodes *D* and *E* receive *ND* from node *F*, they decrease the value of *RCN* to 0. By applying the two filters *F*1 and *F*2, node *D* will not send the path $< A, B, E, D >$ stored in its *RP* list. In fact, the use of filter *F*1 lets node *E* keep only paths $< A, C, E >$ and $<A, C, G, E>$ in its *RP* list. However, $< A, C, E >$ and $< A, C, G, E >$ share node *C* and node *E* accepts only $RCN - 1 = 0$ common nodes. Therefore, when it applies the filter *F*2, node *E* is forced to keep only the shortest path it knows, which is $< A, C, E >$.

When the node *E* receives a second copy of the *ND* from the node *M*, it discards it. When the neighbors of nodes *D* and *E* receive the *NP* with an *RCN* set to 0, they discard this packet without sending any information. At the end, the source node *A* has a list of paths available to *MD* which is equal to $L_p = \{< A, C, E >\}$. The *RPBS*, extracted from the *RRep*, together with the list of paths L_p, will be used to determine the set of routes to *MD*, characterized by $k = 3$ and $x = 2$. By applying the reconstruction algorithm described in Subsection 4.3, the final list of reconstructed paths L_{cp} will be:$\{<A, C, E, F, BS > <A, B, D, F, BS> <A, B, E, M, BS >\}$.

5. RELIABILITY OF SEMURAMAS

When a failure occurs, the proposed algorithm assumes that the source node switches to another path that does not include the failed links or nodes. The detection of node failure is performed by every intermediate node periodically through neighbor discovery procedure. When a node tries to forward a packet to the next node according to the specified route, the node informs the source node about the failure. A recovery of failure is made by the source upon receiving failure node information as depicted in Subsection 3.4. Another alternative for the recovery can be done distributively. In this case, Instead of informing the source node about the node failure, the node detecting the failure can try to bypass the failing node by checking whether it is the source of a request for k-x paths to a destination for which one of the constructed paths has an intersection with the active route. In fact, the intermediate node which detects the node failure when it tries to send datagrams related to the route request, say R_1, could be the source node of another route request, say R_2, sent for another destination node. In this case it should have a local view of the available routes to nodes in the network. Consequently, some of the nodes used for the source routed datagrams related to *R*1 could be used for source routed datagrams related to R_2. Hence, the intermediate node, which detected the failure, checks in the set of paths reconstructed by R_2, if there is a bypass links to the remaining nodes in the route specified in datagrams related to R_1. In this case, the intermediate node sends to the source node of R_1, the route error, and the possible bypass link to be used if they satisfy the parameters k and *x*. If a path is still available to the destination after the intermediate node failure, but the ceil of the number of paths available before

the failure divided by 2, is equal to 1, the source node will generate a new route request while continuing to use the current path until new routes are obtained by that request.

If all paths are disjoints (i.e., the disjointness factor used to discover them is equal to 0), the proposed algorithm guarantees a maximal availability period. However, if the discovered routes share some number of nodes (i.e., the used disjointness factor is higher or equal to 1), a failure has occurred, and there is at least a route that does not share the failed node, the connection between the source and destination nodes will survive. In the worst case, the failed node is part of the set of reconstructed path. Therefore, the algorithm will not be able to enhance the routing survivability and the connection will no longer be available. This situation can occur specially when there is a star topology and all routes share the central node. This situation should not be considered as a drawback of SeMuRAMAS because it is a consequence of the physical topology of the network.

In addition, nodes could be captured by attackers in order to compromise the route discovery process. To provide a secure routing algorithm against attacks, which prepare for the investigation, three main properties should be satisfied. First, nodes should be able to authenticate each others during the process of routes establishment. Datagrams generated with forged information should be discarded before reaching the destination mobile node *MD*. Second, every node should not only be in charge of generating and forwarding datagrams to *MD*, but also should be able to watch what its neighbors receive and forward. Second, when a node detects a malicious neighbor, both the source and the destination nodes should be informed.

To provide the third propriety, the watchdog technique is used to detect nodes that do not forward the datagrams as expected [7]. A node which uses the watchdog technique is able to determine whether its neighbor nodes are forwarding the datagram they receive or not. If the packet is not forwarded within some predefined period, this neighbor is considered as malicious [14]. Every node should maintain two lists: a list of one-hop neighbors and a list of two hop neighbors. The two lists are created by letting every node periodically perform a two-hop broadcast of a Hello Message (i.e., by setting the TTL equal to 2). A node, say n_1, which receives a generated Hello message by a node, say n_0, with a TTL equal to 2, appends the identity of n_0 to its list of neighbors, appends its own identity (i.e., n_1), decreases by one the TTL, and forwards the packet. A node, say n_2, which receives a datagram with a TTL equal to 1 from the neighbor node n_1, appends the identity of the sender (i.e., n_0) to its list of two hop neighbors, and marks this node as being reachable through the immediate sendern_1.

To protect the routing algorithm against forgery of false routing information, we use a signature scheme to authenticate nodes and guarantee the integrity of the information they exchange. We suppose that, in the case of WSN, every node joining the network is authenticated by the *BS*. Intermediate verification of packets signature (including the hello messages) allows discarding compromised packets before they reach the destination nodes, which optimizes the used energy and communication resources, and reduces the overhead of the signature verification process performed by the destination node. During the routes establishment, every node which generates or forwards the *RReq*, appends its identity, the identity of the next receiving nodes, and sur-signs the route record. A node receiving the forwarded message verifies whether the last appended signature is correct, checks if it is the presumed destination, determines the immediate sender (the neighbor node from which the packet is being forwarded) of that datagram and makes sure that it is a neighbor. If it is the case, it appends its identity, the identity of the possible next hops and appends its signature the datagram (the new signature is applied on the whole received datagram including the old signature).

In the context of WSN, signature is performed using the elliptic threshold signature algorithm provided by [16], which allows to generate for one public key k_{pub}, n associated secret keys k_{pr1}, ..., k_{prn}. Every signature created using one of the private keys, say k_{pri}, can be checked using k_{pub}. Every node uses its own private key for signature, while the same public key is used by all the other nodes for the purpose of signature verification. In the context of general types of Ad Hoc networks, where nodes can enter and leave the network at any time, and no resource limitations exist, regular signature can be used. Every node should have its own private and public keys. For this, a certificate containing the public key of each node should be delivered to the node by a certification authority. Mobiles nodes will use the certificates of the senders to check the integrity of the signed datagrams. We assume that each mobile node has the ability to contact a certification authority repository to download the certificate.

These techniques increase the resilience to node compromising, especially in the context of WSN. The protection is done by: a) using digital signature to authenticate packet content and discard invalid datagrams; b) identifying captured nodes using an applied watchdog mechanism and intermediate signature; and c) Discarding compromised nodes using SeMuRAMAS. Consequently, even when parts of the nodes have been captured, the rest of the network remains secure.

6. VALIDATION OF SEMURAMAS PROPRIETIES

In this section, we will show that SeMuRAMAS has some properties including coherence, ability of the algorithm to find a path in the network, and security in SeMuRAMAS. We proposed a set of theorem related to these proprieties.

Theorem 1: (Correctness) Let δ be the set of reconstructed paths satisfying the disjointness factor x generated using the SeMuRAMAS algorithm. Any two paths in δ do not share more than x nodes.

Proof:

- Assuming that there are two reconstructed paths $p1$ and $p2$ at the source node, the aim is to show that the number of shared nodes between $p1$ and $p2$ does not exceed the pre-defined disjointness factor, denoted by x. The theorem will be proved by contradiction. Let us suppose that p_1 and p_2 share more than x nodes and represent a solution, and we will disprove the latter proposition.

- Let s_1 be the first shared node between $p1$ and $p2$ which received a second copy of the same route request generated by the source node. We remind that:
    - Each shared node receiving the notification packet or the route response and its RP list (i.e., it previously received at least two copies of the same route request) should decrease the value of x and forward this packet with the new value. In addition, the use of filter $F2$ allows any intermediate node, which receives the value of x, to check if $p1$ and $p2$ do not share more than $x - 1$ nodes from the source node to it. If it is the case, it announces to the source node that $p1$ and $p2$ represent two possible routes which do not share more than x nodes;
    - Each shared node receiving a notification packet with a value of x equal to 1 will discard it, as the value of x will be decreased to 0.
- The first shared node s_1, should receive the notification packet. This means that the received value of x is greater than or equal to 0. If the number of shared nodes exceeds the value of$x$, this means that s_n has received a negative value of x, which is impossible.

Theorem 2: Let P be the shortest path between two nodes s and d, and δ be the set of paths generated by s using SeMuRAMAS with a disjointness factor set to x. Any path between s and d, which does not share more than x nodes with P, belongs to δ.

Proof: Assuming that there is a path $p1$ sharing with the shortest path P a number of nodes which does not exceed the pre-defined disjointness factor, denoted by x. Let δ be the set of paths generated by s using SeMuRAMAS. We suppose that $p1$ will not be discovered by SeMuRAMAS and we will prove that this is impossible. We remind that in the shortest path each node receiving the route response will decrement the value of the threshold x. If the value of x is higher than 0 and the node has already received a second copy of the route request, it will send the stored paths to the source and generate a notification packet containing segment of paths from the source to the related shared node. If the path $p1$is not added to δ, this means that the node involved in P has not received the route response, which is impossible.

Theorem 3. Given a route request generated by a node s to reach a destination d. Let RP be the Received Path list, obtained at an intermediate node i, consequently to the reception of several copies of route requests sent from s to d. Then RP has the following properties

- The number of paths in RP is equal to the number of copies of the route request received by node i
- Any path in RP is a concatenation of two subpaths. The first subpath is shared by at least another path in RP, while the second subpath never intersects with another path in RP.

Proof:

- We remind that a $RReq$ datagram contains the identities of nodes through which the datagram has been routed. During the route discovery, a node broadcasts the $RReq$, and all its neighbors receive a copy. The $RReq$ can follow different paths starting from one node and only one copy of the $RReq$ could be routed through a segment of consecutive nodes. We define for every path in the list of received paths RP two subpaths. This first subpath is composed of nodes shared with at least a path in the same RP and the second subpath is composed of nodes which are not involved in any other path within the same RP. We demonstrate the Lemma based on recurrence induction. We consider a path p in RP, which can be written in the form of $p = < S_1, .., S_i, .., S_n, >$ as the concatenation of several path segments, where every segment S represents the longest series of nodes that belong to the maximal possible number of paths in the RP.
- Every S_i is in the form of $< n_x, .., n_y >$ where n_y is the node from which the $RReq$ was forwarded to a number of neighbors, say Nn_i ($Nn_i \geq 2$), such that the neighbors in Nn are part of different paths to the node storing the RP. We suppose that the list S_i is shared by Np_i paths in the RP. Therefore, the segment path S_{i+1} will be shared by $Np_i - Nn_i + 1$ paths and we have $Np_{i+1} = Np_i - Nn_i + 1$. This means that the $RReq$ has followed different paths, where every path, which starts with a node in N_n, does not share any node with the next segment $S_{i+1}, ..., S_n$ excepting the node storing the RP list. Roughly speaking, for every i (where $1 \leq i \leq n$) if $Np_i > 1$ the segment S_i will be part of the first subpath. The same reasoning is applied by recurrence until S_n. Since a node does not forward two copies of the same Route Request Datagram, the segment S_n will inevitably belong to the second subpath and cannot be empty.

The construction of paths in SeMuRAMAS is based on a distributed mechanism where information regarding available links is collected by the BS, or stored by intermediate nodes. The

coherence of the routes generation is vindicated by the fact that exchanged information during the algorithm execution are signed by participating neighbors and verified by neighbors. In addition, the Notification datagrams generated by the destination mobile node *MD* or by nodes incorporate the disjointness factor related to the set of the k established paths. Since this value is decreased whenever the notification datagrams is forwarded by a node, which maintains a path to the source node, all pairs of paths that have more than *x* shared nodes will not be generated. This allows that nodes to store and forward only the useful information and reduce the network overhead. This propriety was proved by theorem 2.
In addition, the source node is able to reconstruct all existing shortest paths to the destination. In fact, SeMuRAMAS broadcasts the *RReq* over all nodes in the network and makes all the nodes, including *MD*, able to save all information regarding possible routes from the source node to them. In addition, notification datagrams are broadcast to all neighbor nodes, which have discarded a copy of the *RReq*, to let them send to the source node the list of paths they discovered. Using SeMuRAMAS the network overload may increase depending on the number of nodes and the value of the disjointness factor *x*. Especially, a high number of lists of paths may be generated and the number of List of Paths datagrams may increase. The traffic overload will be estimated in the next section. In WSN, if nodes are in sleeping state they will not be involved in the route discovery. If the node changes its state at the end of this process, it will be considered as a novel node joining the network, and it will be involved in the next route discovery.

The security of SeMuRAMAS is based on the use of watchdog mechanism and digital signature. The watchdog mechanism allows capturing several types of routing attacks such as the wormhole attack [7]. To efficiently use this mechanism, bi-directional links should be used to communicate between mobile nodes in order to let a node detect whether its neighbor is forwarding the datagram received from another node. False positives may occur if a node detects that its neighbor is malicious because it has not forwarded the datagram, while, in reality, it happens due a collision. False negatives may occur if the state, of some nodes involved in the route to *MD*, becomes sleeping or runs out of energy. Such node, which is not detected as inactive yet, may be considered as malicious and leads to the generation of false negative alerts.

The values of the two parameters *x* and *k* are highly correlated and both of them depend on the node density. Typically, if the source node chooses a high value of *k*, it tends to decrease the value of *x* to guarantee the establishment of all the paths. It is worth to notice that, for a fixed value of *k*, the lower is the node density, the lower will be the value of *x*. Conversely, if the nodes density is high, the value of *x* could increase with regard to the latter situation.

Two particular topologies can reduce considerably the performance of SeMuRAMAS. The first is obtained when nodes are so close to each other and all of them are located around *MD*. In this topology, a *RReq* generated from any node will reach all nodes in the network. As a result, a node can receive the same copy of datagrams from all nodes in the network. The nodes memory will be overloaded due to the highest number of paths to store in the *RP* list. The second is obtained when nodes are not deployed with a sufficient number, and most of nodes do not have more than two neighbors. In that case, the routing paths to generate will be relatively long. While the memory occupation rate in nodes is highly reduced with regard to the previous topology, a multipath will require a high delay to be established.

7. SIMULATION RESULTS

In this section, we describe the simulations performed to estimate memory overhead and communication overhead generated by proposed multipath routing algorithm.

7.1 Memory overhead estimation

Every node receives a copy of a *RReq* stores the route recorded in its *RP* list of received paths. Since this list is temporary stored within the mobile node memory, we need to estimate the average number of stored paths in each node as a function of the number of nodes. For this, we consider an area of size 72 × 72 length units square. The nodes communication radius is set to 10 length units and the *x*'s value is set to 2. Based on these values, the optimal number of deployed nodes *No*, as defined by the following formula, is approximately equal to 50 ($\backsimeq 3*72^2/(10^2\pi)$).

No = 3**Total_Network_AreaNode's/ Coverage_Area (1)*

To estimate the average number of stored paths, we develop a simulation considering a network where we deploy a number of nodes using a uniform distribution of random values, with respect to variation ranging from 10 to 150 nodes. We note that, the use of uniform distribution is very common in such situation to distribute nodes over the network area and ensure a homogeneous coverage [5]. If the nodes would have been deployed in non-flat surface containing valleys, rivers and lakes, the use of another distribution such as normal distribution [25] would be a better solution. In each simulation phase, we randomly choose a source node and a destination node to send the route request. The simulation is repeated 500 times to estimate the average value.

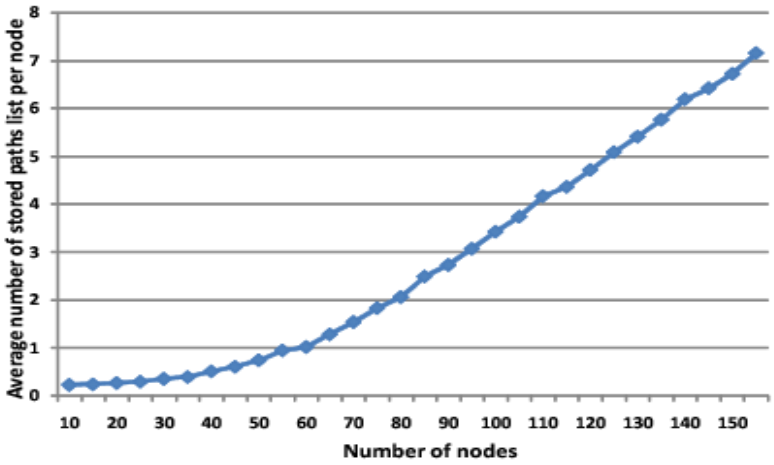

Figure 4. Influence of nodes density on length of stored paths list

Figure 4 shows the variation of the average number of stored paths per node, says *An*, with respect to a number of nodes ranging from 10 to 150. This average is computed at the end of the route discovery phase. It is equal to:

An = Number_of_stored_paths_in_all_nodes / Number_of_deployed_nodes (2)

The simulation results indicate that, the average number increases significantly when the node communication radius increases. Starting from a value of deployed nodes equal to10, and until this number reaches *No* (i.e., 50), the number of received copies of the same *RReq* increases slowly. In fact, since the node number is lower than the optimal value, the number of unreachable nodes is important. The node, which does not receive any copy of *RReq*, will have a number of stored paths equal to 0. On the other hand when the number of nodes is higher than *No*, all the

nodes in the network become reachable and the average of stored paths significantly increases as long as the number of nodes in the network increases.

7.2 Communication overhead estimation

In this simulation we estimate the communication overhead generated by SeMuRAMAS. To do so, we consider an area equal to 39 × 39 length units square, where a variable number of nodes was uniformly distributed. The node communication radius is set to 10 length units and the value of the disjointness factor *x* is set to 2. Based on these values, the optimal number of deployed nodes is equal to 15. The curve depicted by Figure 5 shows the estimated average time per node required to generate a set of routes satisfying the chosen disjointness factor, in terms of number of nodes in the network and number of hops separating the source node to the *MD*. The number of nodes was varied from 5 to 195, and the estimated time was computed for different possible hop values.

The simulation shows that the estimated average time, per node, to establish a path, initially increases as the number of nodes increases from 5 to 35, decreases from 35 to 355and becomes stable starting from 30. In fact, the network becomes more and more covered, and nodes far from the destination mobile node *MD* become able to establish a path, which increases the average of the estimated time. As the number of nodes becomes far from the optimal value, the estimated average time becomes approximately constant. This is due to the fact that datagrams generated by the algorithm are always routed through the shortest path. Even during the route establishment phase, the copies of the *RReq* datagrams arriving to *MD*, are the ones which were forwarded through the minimal number of nodes. The simulation shows also that the highest values of hops are obtained for a number of nodes ranging from 20 to 35. In fact, since the network is not sufficiently dense, nodes that are far from the *MD* will require a high number of hops to reach the *MD*. As the network becomes dense, these nodes become able to reach the *MD* using shorter paths. Finally, it is also noticed that as the number of hops increases by 1, the estimated average time regularly increases linearly.

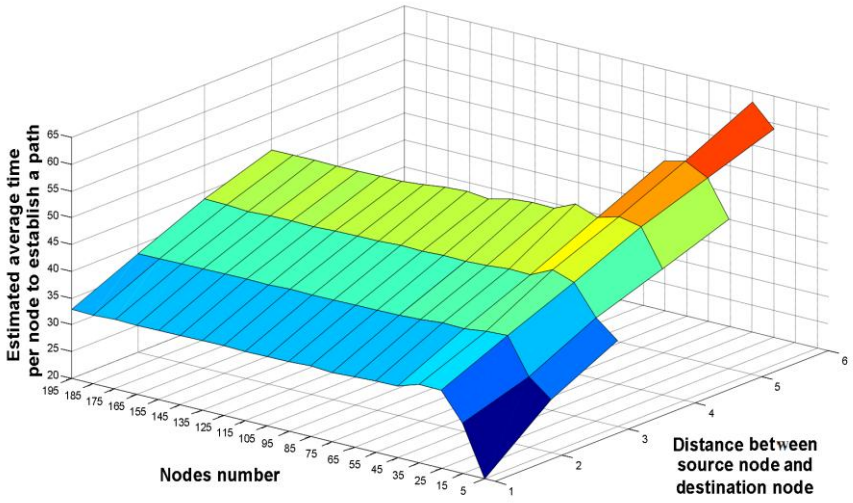

Figure 5. Influence of nodes density on the overhead and the length of paths

8. CONCLUSION

We developed in this paper a novel algorithm called SeMuRAMAS to let nodes in Wireless Adhoc and Sensor Networks be able to specify both of the number of paths that should be available between a source and a destination, and the maximal number of nodes to be shared by these paths.. This allows source nodes to have the ability to route their data using optimized multiple paths. SeMuRAMAS uses watchdogs and digital signature to detect malicious neighbor nodes and authenticate exchanged packets. A simulation is conducted to estimate the memory and communication overhead. Directions for future works will address the estimation of the threshold value depending on the network density and the expected number of paths.

REFERENCES

[1] Almasaeid Hisham, Ahmed Kamal: "On the Minimum k-Connectivity Repair in Wireless Sensor Networks", Proceedings of the IEEE International Conference on Communications (ICC), 2009.

[2] Bayrem Triki, Slim Rekhis, Noureddine Boudriga: "Threshold Based Multipath Routing Algorithm in Mobile Adhoc and Sensor Networks", Communications in Computer and Information Science (CCIS), pp. 54-70, 2012.

[3] Bokareva, Tatiana: "Wireless Sensor Networks for Battlefield Surveillance", Proceedings of the Land Warfare Conference, 2006.

[4] Dulman Stefan Jian Wu, Paul Havinga: "An Energy Efficient Multipath Routing Algorithm for Wireless Sensor Networks", Proceedings of the IEEE International Symposium on Autonomous Decentralized Systems (ISADS 2003), 2003.

[5] H. Karl, A. Willig: Protocols and Architectures for Wireless Sensor Networks. Wiley, 2007.

[6] Hou Xiaobing, David Tipper, Joseph Kabara: "Label-based Multipath Routing in Wireless Sensor Routing", Proceedings of the 6th International Symposium on Advanced Radio Technologies (ISART 04), 2004.

[7] Huang Lei, Lixiang Liu: "Extended Watchdog Mechanism for Wireless Sensor Networks", Journal of Information and Computing Science, pp. 39-48, 2008.

[8] Huang Lu, Jie Li, Mohsen Guizani: "Secure and Efficient Data Transmission for Cluster-Based Wireless Sensor Networks", IEEE Transactions on Parallel and Distributed Systems, pp. 750-761, 2014.

[9] Hyun Woo Oh, Jong Hyun Jang, Kyeong Deok Moon, Soochang Park, Euisin Lee, Sang-Ha Kim: "An explicit disjoint multipath algorithm for Cost efficiency in wireless sensor networks", Proceedings of the 1st International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC), pp. 1899 - 1904, 2010.

[10] Jean Stankovic: "Research Challenges for Wireless Sensor Networks", ACM SIGBED Review, pp. 9-12, 2004.

[11] Johnson David, David Maltz: Mobile Computings. Kluwer Academic Publishers, 1996.

[12] Khelifa Benahmed, Hafidh Haffaf, Merabti Madjid, and David Llewellyn-Jones: "Monitoring connectivity in Wireless Sensor Networks", Proceedings of the IEEE Symposium on Computers and Communications (ISCC'09), 2009.

[13] Law Yee Wei, Li-Hsing Yen, Roberto Di Pietro, Marimuthu Palaniswami: "Secure k-Connectivity Properties of Wireless Sensor Networks", Proceedings of the IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems, 2007.

[14] Lee Jin Wook, Yann-Hang Lee, Violet R. Syrotiuk: "The performance of a watchdog protocol for wireless network security", International Journal of Wireless and Mobile Computing, pp. 28-36, 2007.

[15] Lei Wang, Yuwang Yang, Wei Zhao: "Network coding-based multipath routing for energy efficiency in wireless sensor networks", EURASIP Journal on Wireless Communications and Networking, pp. 1-15, 2012.

[16] Maha Sliti, Mohammed Hamdi, Noureddine Boudriga: "An elliptic threshold signature framework for k-security in wireless sensor networks", Proceedings of the 15th IEEE International Conference on Electronics, Circuits and Systems, 2008.

[17] Marjan Radi, Behnam Dezfouli, Kamalrulnizam Abu Bakar, Malrey Lee: "Multipath Routing in Wireless Sensor Networks: Survey and Research Challenges", Journal on the science and technology of sensors and biosensors, pp. 650-685, 2012.
[18] Melike Yigit, Ozlem Durmaz Incel, Vehbi Cagri Gungor: "On the interdependency between multi-channel scheduling and tree-based routing for WSNs in smart grid environments",Computer Networks, 2014.
[19] Ming Tao, Dingzhu Lu, Junlong Yang: "An Adaptive Energy-aware Multi-path Routing Protocol with Load Balance for Wireless Sensor Networks", Journal of Wireless Personal Communications, pp. 823-846, 2012.
[20] Mohammed Abazeed, Norshiela Faisal, Suleiman Zubair, and Adel Ali: "Routing Protocols for Wireless Multimedia Sensor Network: A Survey", Journal of Sensors, 2013.
[21] Peng Yang, Biao Huang: "Multi-path Routing Protocol for Mobile Ad Hoc Network", Proceedings of the International Conference on Computer Science and Software Engineering,2008.
[22] Shanping Li, Zhendong Wu: "Node-Disjoint Parallel Multi-Path Routing in Wireless Sensor Networks", Proceedings of the Second International Conference on Embedded Software and Systems, pp. 432-443, 2005.
[23] Shi Jinglun: "A Multipath Routing Algorithm for Wireless Sensor Networks", Lecture Notes in Computer Science, pp. 438-446, 2006.
[24] Shpungin Hanan, Segal Michael: "k-Fault Resistance in Wireless Ad-Hoc Networks", Proceedings of the 2005 joint workshop on Foundations of mobile computing, pp. 89—96, 2005.
[25] Sireesha Krupadanam, Huirong Fu: "Beacon-less Location Detection in Wireless Sensor Networks for Non-flat Terrains", International Journal of Software Engineering and Its Applications, pp. 55-76, 2008.
[26] Stefano Chessa, Soledad Escolar, Susanna Pelagatti, Jesus Carretero: "Multi-dimensional recursive routing with guaranteed delivery in Wireless Sensor Networks", Computer Communications, 2014.
[27] Tao Shu, Marwan Krunz, and Sisi Liu: "Secure Data Collection in Wireless Sensor Networks Using Randomized Dispersive Routes", The IEEE Transactions on Mobile Computing, pp. 941-954, 2009.
[28] Wenjing Lou, Younggoo Kwon: "H-SPREAD: A Hybrid Multipath Scheme for Secure and Reliable Data Collection in Wireless Sensor Networks", IEEE Transactions on Vehicular Technology, pp. 1320-1330, 2006.
[29] Zhang Honghai, Jennifer C. Hou: "Asymptotic critical total power for k-connectivity of wireless networks", Proceedings of the IEEE/ACM Transactions on Networking (TON), pp. 347-358, 2008.
[30] Mayur C. Akewar, Nileshsingh V. Thaku , "Grid based wireless mobile sensor network deployment with obstacle adaptability", International Journal of Wireless & Mobile Networks (IJWMN) Vol. 4, No. 5, October 2012.

Authors

Dr. Bayrem Triki received the Ph.D. in Telecommunications from the Engineering School of Communications (Sup'Com), University of Carthage (Tunisia) in 2013. He is currently an Assistant in ISITCom at the university of Sousse. Dr Triki conducting research activities in the areas of is conducting research activities in the area of digital investigation of security incidents , intrusion detection systems, security traffic shaping, computer and network attacks.

Dr. Slim Rekhis received the. University Habilitation (DSc.) and the Ph.D. in Telecommunications from the Engineering School of Communications (Sup'Com), University of Carthage (Tunisia) in 2007. He is currently an Assistant Professor in Sup'Com and member of the Communication Networks and Security (CN&S) research laboratory. Dr. Rekhis is conducting research activities in the area of digital investigation of security incidents, formal verification of security protocols, intrusion detection and tolerance, traceback of host and network attacks, wireless sensor networks, and security of vehicular ad-hoc networks.

Pr. Nouredddine Boudrigua received his Ph.D. in algebraic topology from the University of Paris (France) and his Ph.D. in computer science from the University of Tunis (Tunisia). He is currently a professor of telecommunications at the University of Carthage, Tunisia, and the director of the Communication Networks and Security Research Laboratory (CNAS). Pr. Boudriga is the recipient of the Tunisian Presidential Award in Science and Research (2004). He has served as the general director and founder of the Tunisian National Digital Certification Agency. He has been very actively involved in research and has authored or co-authored many chapters and books, including more than 250 published journal and conference papers. Pr. Boudriga is the President of the Tunisia Scientific Telecommunications Society.