

Finding minimum transmission radii for preserving connectivity and constructing minimal spanning trees in ad hoc and sensor networks

Francisco Javier Ovalle-Martínez^a, Ivan Stojmenović^{a,*}, Fabián García-Nocetti^b,
Julio Solano-González^b

^a*SITE, University of Ottawa, Ottawa, Ont., Canada K1N 6N5*

^b*DISCA, IIMAS, UNAM, Ciudad Universitaria, México D.F. 04510, México*

Received 10 September 2004

Abstract

Ad hoc networks are normally modeled by unit graphs, where two nodes are connected if and only if their distance is at most the transmission radius R , equal for all nodes. Larger than necessary values of R cause communication interference and consumption of increased energy, while smaller values may disable data communication tasks such as routing and broadcasting. It was recognized that the minimum value of R that preserves the network connectivity is equal to the longest edge in the minimum spanning tree. However, all existing solutions for finding R rely on algorithms that require global network knowledge or inefficient straightforward distributed adaptations of centralized algorithms. This article proposes to use the longest *LMST* (local minimum spanning tree, recently proposed message free approximation of *MST*) edge to approximate R using a wave propagation quasi-localized algorithm. The differences between exact and so approximated values of R are estimated for two and three-dimensional random unit graphs. Despite small number of additional edges in *LMST* with respect to *MST* (under 5%), they can extend R by about 33% its range on networks with up to 500 nodes. We then prove that *MST* is a subset of *LMST* and describe a quasi-localized scheme for constructing *MST* from *LMST*. It needs less than 7 messages per node on average (for networks up to 500 nodes). The algorithm eliminates *LMST* edges which are not in *MST* by a loop breakage procedure, which iteratively follows dangling edges from leaves to *LMST* loops, and breaks loops by eliminating their longest edges, until the procedure finishes at a single node (as a byproduct, this single node can also be considered as an elected leader of the network). This so elected leader also learns longest *MST* edge in the process, and may broadcast it to other nodes. We also describe an algorithm for updating *MST* when a single node is added to the network.

© 2004 Elsevier Inc. All rights reserved.

Keywords: Ad hoc networks; Sensor networks; Minimum spanning tree

1. Introduction

Due to their potential applications in various situations such as battlefield, emergency relief, environment monitoring, etc., wireless ad hoc networks have recently emerged as a prime research topic. Wireless networks consist of a set of wireless nodes which are spread over a geographical area.

These nodes are able to perform processing and are capable communicating with each other by means of a wireless ad hoc network. Wireless nodes cooperate to perform data communication tasks, and the network may function in both urban and remote environments.

Energy conservation is a critical issue in wireless networks for the node and the network life, as the nodes are powered by batteries only. Each wireless node typically has transmission and reception processing capabilities. To transmit a signal from a node A to other node B , the consumed power (in the most common power-attenuation model) is proportional to $\|AB\|^z + c$, where $\|AB\|$ is the Euclidean

* Corresponding author. Fax: +1 613 739 4396.

E-mail addresses: fovalle@site.uottawa.ca (F.J. Ovalle-Martínez), ivan@site.uottawa.ca (I. Stojmenović), fabian@uxdea4.iimas.unam.mx (F. García-Nocetti), julio@uxdea4.iimas.unam.mx (J. Solano-González).

distance between A and B ; α is a real constant between 2 and 5 which depends on the transmission environment, and constant c represents the minimal power to receive, store and process a signal. For simplicity, this overhead cost can be integrated into one cost, which is almost the same for all nodes. The expression represents merely the minimal power, assuming that the transmission radius is adjusted to the distance between nodes. While adjusting transmission radius is technologically feasible, medium access layer (e.g. the standard IEEE 802.11) works properly only when all nodes use the same transmission radius. Otherwise hidden terminal problem is more difficult to control and magnifies its already negative impact on the network performance.

Ad hoc networks are normally modeled by unit graphs, where two nodes are connected if and only if their distance is at most R , where R is the transmission radius, equal for all nodes. Finding the minimum R that preserves connectivity is an important problem for network functionality. Larger than necessary values of R cause communication interference and consumption of increased energy, while smaller values of R may disable data communication tasks such as routing and broadcasting.

The two objectives that have been mainly considered in literature are: minimizing the maximum power at each node (as in this paper) and minimizing the total power assigned if transmission ranges can be adjusted (for example, [1] shows that MST used as non-uniform power assignment yields a factor of 2 approximation). Instead of transmitting using the maximum possible power, nodes in an ad hoc network may collaboratively determine the optimal common transmission power. It corresponds to the minimal necessary transmission radius to preserve network connectivity. It was recognized [8–11] that the minimum value of R that preserves the network connectivity is equal to the longest edge in the minimum spanning tree (MST). However, all existing solutions for finding R rely on algorithms that require global network knowledge or inefficient straightforward distributed adaptations of centralized algorithms. Therefore almost all existing solutions for energy-efficient broadcast communications are globalized, meaning that each node needs global network information.

Global network information requires huge communication overhead for its maintenance when nodes are mobile or have frequent changes between active and sleeping periods. Localized solutions are therefore preferred. In a localized solution to any problem, the node makes (forwarding or structure) decisions solely based on the position of itself and its neighboring nodes. In addition, it may require constant amount of additional information, such as position of destination in case of routing. In case of $LMST$, both construction and maintenance are fully localized. In some cases (such as MST), however, local changes to a structure may trigger changes in a different part of the network, and therefore could have global impact to a structure. We refer to such protocols as being *quasi-localized*, if any node in the update

process still makes decision based on local information, but a ‘wave’ of propagation messages may occur.

Nodes in ad hoc network are assumed to either know their geographic position (using, for example, GPS), or be able to determine mutual distances based on signal strength or time delays. This assumption is in accordance with literature.

Li et al. [5] recently proposed a localized algorithm to approximate MST . The algorithm constructs local minimal spanning tree, where each node finds MST of the subgraph of its neighbors, and an edge is kept in $LMST$ (localized minimal spanning tree) if and only if both endpoints have it in their respective local trees. In this article, we propose to use the longest edge in $LMST$ edge to approximate R using a wave propagation quasi-localized algorithm. The wave propagation algorithm is adapted from [4], where it was used for leader election. We simply propagate the longest $LMST$ edge instead of propagating the winning leader information.

In order to determine whether the longest $LMST$ edge is a reasonably good approximation of desired R , we study some characteristics of $LMST$ in two dimensions (2D) and three dimensions (3D). We observed that, although the number of additional edges in $LMST$ respect to MST is very small (under 5% of additional edges), they tend to be relatively large edges, and can extend R by about 33% of its range on networks with up to 500 nodes.

In some applications, such as mesh networks for wireless Internet access, or sensor networks for monitoring environment, the nodes are mostly static and network does not change too frequently (which is the case if mobility is involved). The increased transmission range by 33% may easily double the energy expenditure, depending on constants α and c in the energy consumption model, and increases interference with other traffic. On the other hand, the increased value for R provides redundancy in routing, which is useful especially in a dynamic setting. These observations about the structure of $LMST$ and increased power consumption motivated us to design an algorithm for constructing MST topology from $LMST$ topology, without the aid of any central entity. The proposed algorithm needs less than 7 messages per node on average (on networks up to 500 nodes). It eliminates $LMST$ edges which are not in MST by a loop breakage procedure, which iteratively follows dangling edges from leaves to $LMST$ loops, and breaks loops by eliminating their longest edges, until the procedure finishes at a single node (as a byproduct, this single node can also be considered as an elected leader of the network). This so elected leader also learns longest MST edge in the process, and may broadcast it to other nodes.

We made two sets of experiments (using Matlab environment) for the MST construction from the $LMST$. In one scenario, nodes are static and begin constructing MST from $LMST$ more or less simultaneously. In the second set of experiments, we study the maintenance of already constructed MST when a new node is added to the network.

This paper is organized as follows: Section 2 presents the related work for the stated problem. In Section 3 we present

some characteristics of *MST* and *LMST* obtained by experiments, for 2D and 3D. The main characteristics of interest to this study are the lengths of the longest *MST* and *LMST* edges. In Section 4, we describe the adaptation of wave propagation protocol [4] for disseminating R throughout the network. The algorithm for constructing *MST* from the *LMST* is explained in detail in Section 5. Section 6 gives performance evaluation of the algorithm for constructing *MST* from *LMST*. Section 7 describes an algorithm for updating *MST* when a single node is added to the network, and gives results of its performance evaluation. Section 8 concludes this paper and discusses relevant future work.

2. Related work

In [3], Dai and Wu proposed three different algorithms to compute the minimal uniform transmission power of the nodes, using area-based binary search, Prim's minimum spanning tree, and its extension with Fibonacci heap implementation. However, all solutions are globalized, where each node is assumed to have full network information (or centralized, assuming a specific station has this information and informs network nodes about *MST*). We are interested in quasi-localized algorithm, where each node uses only local knowledge of its 1-hop neighbors, and the communication propagates throughout the network until *MST* is constructed.

In [7], Narayanaswamy et al. presented a distributed protocol that attempts to determine the minimum common transmitting range needed to ensure network connectivity. Their algorithm runs multiple routing daemons (RDs), one at each power level available. Each RD maintains a separate routing table where the number of entries in the routing table gives the number of reachable nodes at that power level. The node power level is set to be the smallest power level at which the number of reachable nodes is the same as that of the max power level. The kernel routing table is set to be the routing table corresponding to this power level. The protocol apparently requires more messages per each node, and higher power levels, than the protocol presented here.

Penrose [8,9] investigated the longest edge of the minimal spanning tree. The critical transmission range for preserving network connectivity is the length of the longest edge of the Euclidean *MST* [8–11]. The only algorithm these articles offer is to find *MST* and then its longest edge, without even discussing the distributed implementation of the algorithm.

Santi and Blough [11] show that, in two and three dimensions, the transmitting range can be reduced significantly if weaker requirements on connectivity are acceptable. Halving the critical transmission range, the longest connected component contains 90% of nodes, approximately. This means that a considerable amount of energy is spent to connect relatively few nodes.

A localized *MST* based topology control algorithm for ad hoc networks was proposed in [5] by Li et al. Each

node u first collects the positions of its one-hop neighbours $N1(u)$. Node u then computes the minimum spanning tree $MST(N1(u))$ of $N1(u)$. Node u keeps a directed edge uv in *LMST* if and only if uv is also an edge in $MST(N1(v))$. If each node already has 2-hop neighbouring information, the construction does not involve any message exchange between neighboring nodes. Otherwise each node contacts neighbors along its *LMST* link candidates, to verify the status at other node. The variant with the union of edge candidates rather than their common intersection is also considered in [5], possibly leading to a directed graph (no message exchange is then needed even with 1-hop neighbour information). In [6], Li et al. showed that *LMST* is a planar graph (no two edges intersect). Then they extended the *LMST* definition to k -hop neighbours, that is, the same construction but with each node having more local knowledge. They also prove that *MST* is subset of 2-hop based *LMST*, but not that *MST* is a subset of 1-hop based *LMST* considered in this article. We observed, however, on their diagrams that *LMST* with 2-hop and higher local knowledge was mostly identical to the one constructed with merely 1-hop knowledge, and decided to use only that limited knowledge, therefore conserving the communication overhead needed to maintain k -hop knowledge.

In [4], Dulman et al. proposed a wave leader election protocol. Each node is assigned a unique ID from an ordered set. Their algorithm selects as leader the node with minimum ID. In the wave propagation algorithm [4], each node maintains a record of the minimum ID it has seen so far (initially its own). Each node receiving a smaller ID than the one it kept as currently smallest updates it before the next round. In each round, each node that received smaller ID in the previous round will propagate this minimum on all of its outgoing edges. After a number of rounds, a node elects itself as leader if the minimum value seen in this node is the node's own ID; otherwise it is a non-leader.

We will apply *FACE* routing algorithm [2] in our protocol for converting *LMST* into an *MST*. *FACE* routing guarantees delivery and needs a planar graph to be applied. Starting from source node, faces that intersect imaginary line from source to destination are traversed. The traversal of each face is made from the first intersecting edge (with mentioned imaginary line) to the second one. Reader is referred to [2] for more details.

3. Comparing longest edges of *MST* and *LMST*

Theorem 1. *MST is a subset of LMST.*

Proof. The well-known Kruskal's algorithm for constructing *MST* sorts all edges in the increasing order, and considers these edges one by one for inclusion in *MST*. *MST* initially has all vertices but no edges. An edge is included into already constructed *MST* if and only if its addition does not create a cycle in the already constructed *MST*. Let $LMST(A)$ be the

Table 1
MST vs. LMST for 500 nodes

	MST 2D	LMST 2D	MST 3D	LMST 3D
Average degree	1.996	2.114	1.996	2.192
Average max number of neighbors	3.900	3.900	4.400	4.500
Highest degree	4.000	4.000	5.000	5.000
Average maximal radius	0.081	0.103	0.177	0.214
Std deviation maximal radius	0.010	0.005	0.014	0.017

Table 2
Ratio of average maximal radius of MST and LMST

N	MST/LMST average maximal radius, 2D	MST/LMST average maximal radius, 3D
10	0.9392	0.9421
20	0.8635	0.8650
50	0.7923	0.8043
100	0.7944	0.7978
200	0.7515	0.7588
500	0.7864	0.8271

minimal spanning trees constructed from $n(A)$, which is set containing A and its 1-hop neighbors. We will show that if an edge from MST has endpoints in $n(A)$ then it belongs to $LMST(A)$. Suppose that this is not correct, and let e be the shortest such edge. $LMST(A)$ may also be constructed by following the same Kruskal's algorithm. Thus edges from A to its neighbors and between neighbors of A are sorted in the increasing order. They are then considered for inclusion in $LMST(A)$. Thus, when e is considered, since it is not included in $LMST(A)$, it creates a cycle C in $LMST(A)$, with e being the longest edge in that cycle. Some of the edges from C are not in MST . Consider now expanded cycle C' constructed from C as follows. Let f be an edge from C which is not in MST . Addition of f into MST creates a cycle B , with f being the longest edge in the cycle. The cycle consists of f and a path consisting of edges from MST . Replace f in C with all the edges from that path. Each such replacement enlarges the cycle C , but does not add any edge longer than f , and consequently longer than e . At the end of this process, after replacing all non- MST edges with the corresponding paths of MST edges, e remains the longest edge of C' , but all the other edges of C' are now also in MST . This is a contradiction, since MST has no cycles. Therefore each edge AB from MST belongs to both $LMST(A)$ and $LMST(B)$, and therefore to $LMST$. \square

We are interested in the viability of using the $LMST$ topology for approximating the minimal transmission radius R . Matlab was used to derive some characteristics of the $LMST$ topology in 2D and in 3D.

We generate unit graph of n nodes ($n = 10, 20, 50, 100, 200$ and 500), each randomly distributed over an area of 1×1 for the 2D case and over a volume of $1 \times 1 \times 1$ for the 3D case. The following characteristics (some of them

are presented for possible other applications) of $LMST$ and MST were compared:

- Average degree (average number of neighbors for each node),
- Average maximal number of neighbors,
- Percentage of nodes which have degrees 1, 2, 3, 4, 5, degree > 5 ,
- Highest degree of a node ever found in any of tests,
- Average Maximal radius,
- Standard deviation of average maximal radius.

For each n we ran 200 tests in order to have more confident results. Tables 1 to 6 show the obtained results. It is well known that the average degree (average number of neighbors per node) of an MST with n nodes is always $2 - 2/n$, since it has $n - 1$ edges and n nodes. This value is entered in tables below. Table 1 shows the results for 500 nodes (similar data are obtained for other values of n). Note that $LMST$ has $< 5\%$ more edges than MST .

Table 2 presents the ratios of the longest MST and $LMST$ edges, for various numbers of nodes. The ratio is always > 0.75 . This means that, on average, longest $LMST$ edge may have about $1/3$ longer length than the longest MST edge. It may lead to about twice as much additional energy for using the longest transmission radius from $LMST$ instead of MST . Such discovery motivated us to design a procedure for converting $LMST$ into an MST .

As we can see from Table 1, the maximum degree of any node obtained for $LMST$ in all the tests was 5. This means that $LMST$ maintains a relatively low degree independently on the size of the network and its density (our study is based on maximal density, or complete graphs). Since the area where nodes were placed remained fixed, and

Table 3
Percentage of nodes which have degree 1, 2, 3, 4, 5, and > 5 for the 2d MST

Degree	1	2	3	4	5	> 5
$N = 10$	35.5%	49.5%	14.5%	0.5%	0	0
$N = 20$	28.75%	53%	17.75%	0.5%	0	0
$N = 50$	23.8%	56.8%	19%	0.4%	0	0
$N = 100$	22.8%	57.05%	19.5%	0.65%	0	0
$N = 200$	22.48%	56.7%	20.18%	0.65%	0	0
$N = 500$	22.1%	55.9%	21.2%	0.8%	0	0

Table 4
Percentage of nodes which have degree 1, 2, 3, 4, 5, and > 5 for the 2d LMST

Degree	1	2	3	4	5	> 5
$N = 10$	27%	55.5%	16%	1.5%	0	0
$N = 20$	20.75%	54%	24.25%	1%	0	0
$N = 50$	16.3%	58.8%	23.9%	1%	0	0
$N = 100$	16%	58.75%	24.4%	0.85%	0	0
$N = 200$	15.82%	58.7%	24.62%	0.85%	0	0
$N = 500$	15.52%	58.64%	24.82%	1.02%	0	0

Table 5
Percentage of nodes which have degree 1, 2, 3, 4, 5, and > 5 for the 3d MST

Degree	1	2	3	4	5	> 5
$N = 10$	35.5%	50%	13.5%	1%	0	0
$N = 20$	35.75%	40.75%	21.25%	2.25%	0	0
$N = 50$	31.2%	45.2%	20%	3.6%	0	0
$N = 100$	30.65%	44.3%	21.55%	3.4%	0.1%	0
$N = 200$	29.02%	46.05%	21.98%	2.8%	0.15%	0
$N = 500$	28.02%	46.07%	22.02%	3.67%	0.22%	0

Table 6
Percentage of nodes which have degree 1, 2, 3, 4, 5 and > 5 for the 3d LMST

Degree	1	2	3	4	5	> 5
$N = 10$	26%	55%	17%	2%	0%	0
$N = 20$	23.75%	47.25%	24.5%	4.25%	0.25%	0
$N = 50$	22%	46.8%	26%	5.2%	0%	0
$N = 100$	21.55%	45.95%	27.7%	4.65%	0.15%	0
$N = 200$	18.85%	47.67%	29.07%	4.17%	0.23%	0
$N = 500$	17.34%	48.12%	31.03%	3.24%	0.27%	0

more nodes were placed, the maximal transmission radius decreased when the number of nodes was increased.

Tables 3–6 show the percentages of nodes which have degree 1, degree 2, degree 3, degree 4, degree 5, and degree > 5, for the *MST* and the *LMST*. It can be observed that about half nodes have degree two, and < 2% of nodes in 2D have degree > 3 and < 5% of nodes in 3D have degree > 3.

Fig. 1 illustrate *MST* and *LMST* graphs for $n = 200$ nodes, in 2D and 3D. These figures helped us in gaining an insight on how to construct efficiently *MST* from *LMST*.

4. Wave propagation quasi-localized algorithm for finding transmission radius from the longest *LMST* edge

We adapt the wave propagation leader election algorithm [4] for the use in finding the longest *LMST* edge. Our basic idea is to substitute the node ID with the longest edge adjacent to each node in its *LMST* topology. Each node maintains a record of the longest edge it has seen so far (initially its own longest edge in its *LMST*). In each round, each node receiving larger edge in the previous round will broadcasts

its new longest edge. At the end, all the nodes will receive the same longest edge, which will be used as transmission radius.

One of drawbacks, or perhaps advantages, of given protocol is that a node does not know when the wave propagation process is finished. It is drawback in the sense that it may not use the proper transmission radius, same for all nodes, but a smaller one. However, this smaller transmission radius will still preserve network connectivity, since it is not equal to all nodes. It is advantage in the sense that it is a very simple protocol, and can be an ongoing process with dynamic ad hoc networks. It is straightforward to apply it when an *LMST* edge has increased over current transmission radius R , in which case this new value can be propagated. However, when an edge that was equal to R has decreased, the process of reducing R in the network is not straightforward, since the length of the second longest edge was not preserved with wave propagation algorithm. To address this issue, k longest *LMST* edges may be maintained, and the message to use the next smaller value is broadcast from the neighbourhood of the event. The alternative is to initiate new wave propagation from a node detecting the problem with edge that decided currently recognized R .

We did not implement this protocol, since it does not significantly differ from one in [4]. The reader can see that article about its performance. Most importantly, the number of messages per node was < 7 in all measurements, done in somewhat different settings, with denser graphs than *LMSTs*.

Therefore, we can expect much lower message count per node in our application.

5. Constructing MST from LMST in ad hoc networks

We can observe (see Fig. 1) that the only differences between the *LMST* and the *MST* are the loops that appear in the *LMST* but not in *MST*, created by edges present in *LMST* but not in *MST*. Our main idea is to somehow ‘break’ the *LMST* loops in order to obtain the *MST*.

We are now ready to describe our proposed scheme for converting *LMST* into *MST*. It consists of several iterations. Each iteration consists of two steps: traversing or eliminating dangling (tree) edges, and breaking some loops. These two steps repeat until the process ends in a node. That node is, as a byproduct, network leader, and learns longest *MST* edge in the process (it also may learn the longest *LMST* edge). The value of the longest *MST* edge can then be broadcast to other network nodes. Details of this process are as follows.

Tree step: Each leaf of *LMST* initiates a broadcast up the tree. Each node receiving such upward messages from all but one neighbor declares itself as dangling node and continues with upward messages. All edges traversed in this way belong to *MST*. Each such traversed subtree at end decides what is its candidate for R (maximal radius of *MST*). This tree advance will stop at an *LMST* loop, at a node that will be called the *breaking* node, because of its role in the sequel.

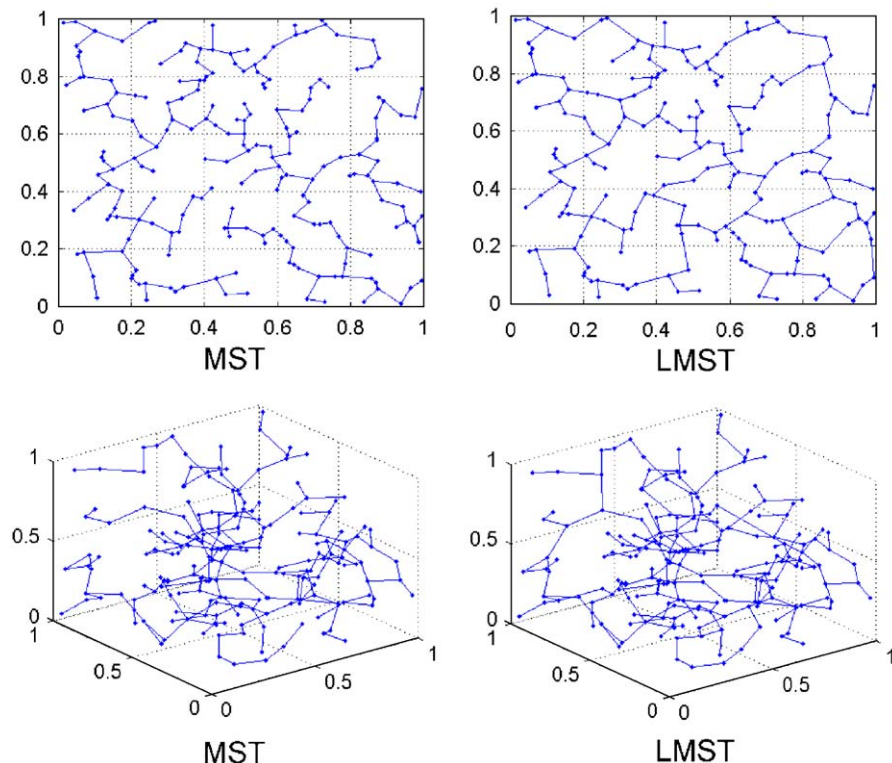


Fig. 1. MST and LMST comparison for 200 nodes.

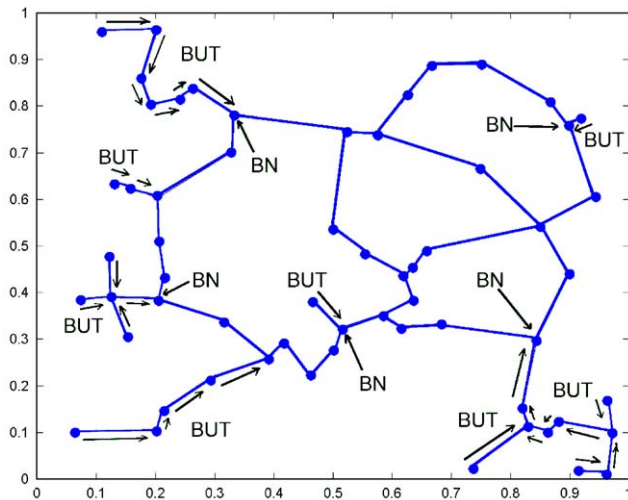


Fig. 2. Tree step in *MST* construction from *LMST* (BUT = broadcast up the tree, BN = breaking node).

Fig. 2 illustrates this tree step. Traversed dangling nodes and edges are shown by arrows. More precisely, breaking node is a node that receives at least one message from dangling node/edge and, after some predefined timeout, remains with two or more neighbors left. Breaking nodes are exactly nodes that initiate the loop step, the second of the two steps that are repeating. Note that, an alternate choice is that all nodes on any loop become breaking nodes. We did not select this option since, on average, it would generate more messages, and our goal is to reduce message count. However, in some special cases, *LMST* may not have any leaf (e.g. *LMST* being a ring). In this case, node(s) that decided to create *MST* may declare themselves as breaking nodes after certain timeout, if no related message is received in the meanwhile. In some special cases, such as sensor network trained from the air or a monitoring station, signals to create *MST* may arrive externally as part of training.

Loop step: Each of the breaking nodes from the previous step initiates the loop traversal to find and break the longest edge in the loop. Consider breaking node A on one such loop. It has, in general, two neighbors on the loop, and edges AB and AC . Note that in some special cases breaking node may also be branching node, that is, it could have more than two neighbors; in that case consider clockwise and counterclockwise neighbors with respect to incoming dangling edge. Also, in general, breaking node A belongs to two loops (that is, two faces of considered planar graph). Let $\|AB\| > \|AC\|$. Node A will select direction AC to traverse the cycle, that is, the direction of shorter edge. The message that started at A will advance using *FACE* routing algorithm [2]. Dangling edges from previous tree steps are ignored in the loop steps. If a branching node is encountered, the traversal splits into two traversals, one of each face of edges traversed so far. The advance will stop at a node D whose following neighbor E is such that $\|AB\| < \|DE\|$.

This means that a longer edge in the cycle is detected, and AB is not eliminated. Node D is then declared as the new breaking node, and starts the same longest edge verification algorithm. If the message returns to D then the longest edge DE from the loop is eliminated. Endpoints D and E of each such broken *LMST* edges follow then step 1, upward tree climbing, until they reach another *LMST* cycle. This process continues until all upward tree messages meet at a single node, which means that *MST* is constructed.

The described algorithm has several byproducts, in addition to constructing *MST*. The last node in the construction can be selected as the leader in the network, especially because it is expected to be somehow centrally positioned. Next, this leader node may, in the process, learn the longest *LMST* and longest *MST* edges, and may initiate simple broadcasting algorithm about these obtained values, which will be used as transmission radius for the whole network. This is especially needed for *MST*, as part of algorithm to find minimum transmission radius, and inform nodes about it. In case of *LMST*, we already observed that wave propagation algorithm can be used instead immediately.

Theorem 2. *The tree obtained from *LMST* by applying the described loop breakage algorithm correctly is *MST*.*

Proof. Planarity of *LMST* [6] shows that it consists of well-defined faces and therefore loop breakage process does create tree at end, by ‘opening’ up all closed faces. Theorem 1 proves that *MST* is a subset of *LMST*. The algorithm described above clearly breaks every loop, by eliminating its longest edge. Suppose that the tree obtained at the end of the process is not *MST*. Since both trees have equal number of edges, let e be the shortest *MST* edge that was not included in the tree. Edge e was eliminated at some point, since it was the longest edge in a loop of *LMST*. Subsequently, more edges from that loop may be eliminated, thus increasing the length of loop that would have been created if e was returned to the graph. However, in all these subsequent loops, including one at the very end of process, e remains the longest edge, since subsequent edges, from loops that would contain e , are originally shorter than e , but always longest among new edges that appears in the loop when they are eliminated. Since e is in *MST*, at least one other edge f from that final loop is not in *MST*. This edge f is shorter than e as explained. However, if e is replaced by f , the graph remains connected and remains a tree, with overall smaller weight than *MST*. This is a contradiction. Therefore the tree that remains at end is indeed an *MST*. \square

6. Performance evaluation of the algorithm for constructing minimal spanning trees

We consider a network of n nodes, with randomly distributed nodes over an area of 1×1 . We constructed the

Table 7
Message counts in the algorithm that constructs *MST* from *LMST*

Nodes	Average number of messages	Max number of messages	Min number of messages	Mean of messages per node	Std of messages per node	Average number of iterations
10	16.10	34	9	1.61	0.73	0.120
20	48.64	81	19	2.43	1.37	0.700
60	257.16	1083	108	4.29	3.08	1.860
100	518.94	1381	253	5.19	4.01	4.660
200	1265.25	2161	717	6.33	5.11	8.120
300	1952.35	2695	1462	6.51	5.13	8.900
500	3490.75	3957	2473	6.98	5.80	13.100

LMST for several values of n ($n = 10, 20, 60, 100, 200, 300$ and 500), with 200 generated networks for each n . The described algorithm was simulated. We measured the following characteristics:

- Average number of messages in the network, for constructing *MST* from *LMST*,
- Maximal number of messages in any of generated networks,
- Minimal number of messages in any of generated networks,
- Average number of messages per node,
- Standard deviation of message counts per node,
- Average number of iterations (that is, how many times loop step was applied).

It appears that the average number of messages per node is approximately $\log_2 n - 2$. Therefore it increases with the network size, but does it very slowly. It is not surprising since *MST* is a global structure, where change in one part of the network has impact on the decision made in other part of the network, and this happens at various levels of hierarchy (Table 7).

7. Updating *MST* after adding one node

Mobility of nodes, or changes in node activity status, will cause changes in *MST* topology. We will now design a simple algorithm for updating *MST* when a new node is added to the network. The added node first constructs its own *LMST*, that is, *MST* of itself and its 1-hop neighbors. There are two cases to consider. Simpler case is when such *LMST* contains only one edge. In this case, the edge is added to *MST*, and no further updates are needed.

The case when the *LMST* at given node contains more than one edge is non-trivial, and requires a procedure for loop breaking to find and eliminate longest edges in newly created cycles. For example, in Fig. 3, *LMST* of new node has three edges. The update procedure assumes that *MST* is organized as a tree, rooted at the leader found in the construction process. This tree can be constructed during *MST* construction algorithm, or leader can additionally construct

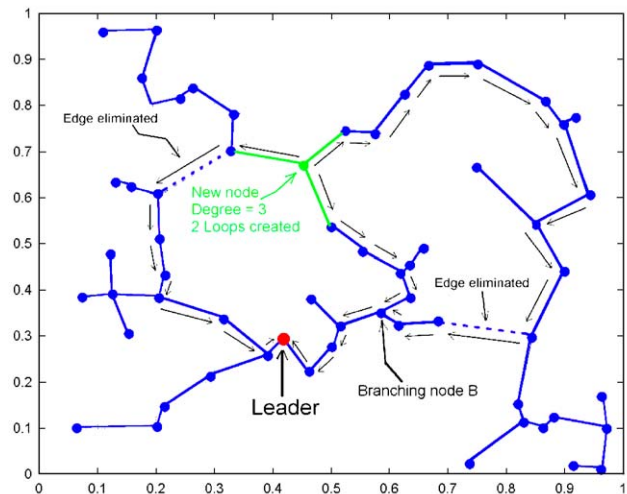


Fig. 3. Added node, traversal, and eliminated edges when *MST* is updated.

or complete the tree while informing about the length of longest edge in *MST*. All edges in *MST* are oriented toward the leader. In this way, branching is completely avoided in the traversal. The decision to use leader in the update process is made in order to avoid traversing long open *LMST* face (in *MST*, this open face contains all nodes and edges of *MST*; in fact each edge is found twice on that open face) unnecessarily and with each non-trivial node addition.

Thus, if *LMST* at added node contains $k > 1$ neighbors, k traversals toward the leader are initiated. Each of traversals records the longest edge along the path traversed. Each of the traversals stops at the leader node. However, some branching nodes may receive two copies of such traversal messages. These nodes recognize completion of a loop, and can make decision about longest loop edge to be eliminated. In example in Fig. 3, the traversal that starts to the left of new node will end at the leader node, with each intermediate node being visited once only. The other two will ‘meet’ at indicated branching node *B*. Such node *B* may forward only one of traversals toward the leader, and may stop the second incoming traversal. Starting with k traversals, $k - 1$ loops will be recognized, each at leader node or an interim

branching node. These nodes may learn the longest edge in the process, and may send backward messages toward it to ask for breaking the edge.

The described algorithm is implemented. After 100 tests we measured that the average number of messages in the network with 200 nodes for updating *MST* when one node is added was 34.8. Therefore it appears that the *MST* construction with synchronous start from *LMST*, requiring less than 7 messages per node, leads to significant communication savings.

8. Conclusions and future work

LMST is a message free localized structure in ad hoc networks, which contains *MST* as a subset and which has less than 5% additional edges not already contained in *MST*. We proposed to use the longest *LMST* edge to approximate the minimal transmission radius R , whose exact value is the longest edge in the *MST*. We compared some characteristics of *LMST* and *MST*. The average degree of the *MST* is ≈ 1.94 compared to the ≈ 2.06 obtained with the *LMST* in the 2D case. In the 3D case the degree for the *MST* is ≈ 1.94 and the degree for the *LMST* is ≈ 2.12 . Also we noted that most of the nodes had degree two or three. The longest *LMST* edge can be spread throughout the network by applying a wave propagation algorithm, previously proposed to be used as a leader election algorithm.

Existing *MST* construction algorithms were based on global knowledge of the network, or on some operations that, in distributed implementations, were not performed between neighboring nodes. The main novelty of our proposed scheme is that all the communication was restricted between neighboring nodes; therefore the message count is a realistic one. To design the new *MST* construction algorithm, we observed that the difference between *LMST* and *MST* is in some loops present in *LMST*, and that the number of these loops was not large. The construction was based on ‘breaking’ these loops in iterations, with *MST* edges being recognized between iterations. The proposed algorithm appears to have logarithmic (in number of nodes in the network) number of messages per node.

Constructing *MST* from *LMST*, following the procedure, is beneficial when the network considered is not very dynamic. Such scenarios include mesh networks for wireless Internet access, with antennas placed on the roofs of buildings. Sensor networks normally are static, but the usefulness of the construction depends on the frequency of sleep period operations. We assume that sensors are divided into groups, and that changing between active and passive states in sensors occurs inside groups, while at the same time *MST* is constructed between groups, not between individual sensors. This treatment of sensor networks justifies the application of our *MST* construction, with certain limitations regarding accuracies involved coming from changes in

active participating sensors from each group. In particular, reduced transmission range in sensor networks leads toward energy savings and prolonged network life. Reduced energy expenditure may also allow less frequent topological changes.

We described also a very simple algorithm for updating *MST* when a new node is added to the network. If an existing node is deleted from the network, its neighbors may similarly construct their new *LMSTs*, and similar procedure (with somewhat more details) can be applied.

Our proposed construction of *MST* from *LMST* works only for 2D case. It cannot be applied in 3D since the *FACE* algorithm [2] does not work in 3D. It is therefore an open problem to design an algorithm for constructing *MST* from *LMST* in 3D. Similarly, generalizing *FACE* routing with guaranteed delivery to 3D [2] remains an outstanding open problem.

Acknowledgments

This research is supported by *CONACYT* project 37017-A, *CONACYT* Scholarship for the first author, and *NSERC* grant of the second author.

References

- [1] E. Althaus, G. Calinescu, I. Mandoiu, S. Prasad, N. Tchervenski, A. Zelikovsky, Power efficient range assignment in ad-hoc wireless networks, in: Proceedings of the IEEE Wireless Communications and Networking Conference, New Orleans, USA, March 2003.
- [2] P. Bose, P. Morin, I. Stojmenovic, J. Urrutia, Routing with guaranteed delivery in ad hoc wireless networks, *ACM Wireless Networks* 7 (6) (November 2001) 609–616.
- [3] Q. Dai, J. Wu, Computation of minimal uniform transmission power in ad hoc wireless networks, 23rd International Conference on Distributed Computing Systems Workshops (ICDCSW’03), May 19–22, 2003, Providence, Rhode Island.
- [4] S. Dulman, P. Havinga, J. Jurink, Wave leader election protocol for wireless sensor networks, *MMSA Workshop*, Delft, The Netherlands, December 2002.
- [5] N. Li, J.C. Hou, L. Sha, Design and analysis of an *MST*-based topology control algorithm, in: Proceedings of the INFOCOM 2003, San Francisco, USA, 2003.
- [6] X.-Y. Li, Y. Wang, P.-J. Wan, O. Frieder, Localized low weight graph and its applications in wireless ad hoc networks, *INFOCOM*, 2004.
- [7] S. Narayanaswamy, S. Kawadia, V. Sreenivas, P. Kumar, Power control in ad hoc networks: theory, architecture, algorithm and implementation of compow protocol, *Proceedings of the European Wireless*, 2002, pp. 156–162.
- [8] M. Penrose, The longest edge of the random minimal spanning tree, *Ann. Appl. Probab.* 7 (2) (1997) 340–361.
- [9] M. Penrose, A strong law for the longest edge of the minimal spanning tree, *Ann. Probab.* 27 (1) (1999) 246–260.
- [10] M. Sanchez, P. Manzoni, Z. Haas, Determination of critical transmission range in ad hoc networks, in: Proceedings of the IEEE Multiaccess, Mobility and Teletraffic for Wireless Communication Conference, Venice, Italy, October 1999.
- [11] P. Santi, D. Blough, The critical transmitting range for connectivity in sparse wireless ad hoc networks, *IEEE Trans. Mobile Comput.* 2 (1) (2003) 1–15.



Francisco J. Ovalle-Martínez was born in Mexico City in 1977. He earned his Bachelor of Science degree in Telecommunications Engineering from the Autonomous National University of Mexico (UNAM) in 2001. He has worked as a Junior Programmer at various software and hardware firms in Mexico, such as Red de Control Corporativo and Yum International Restaurants Mexico. Currently, he is a master degree student at the University of Ottawa, and recipient of CONA-CyT scholarship. His interest areas include mobile computing, digital communications,

computer networks and networks security.



Ivan Stojmenović received his Ph.D. degree in mathematics in 1985. He held regular or visiting positions in Serbia, Japan, USA, Canada, France and Mexico. He published over 180 different papers in journals and conferences, edited 'Handbook of Wireless Networks and Mobile Computing' (Wiley, 2002), and co-edited 'Mobile Ad Hoc Networking' (IEEE Press, 2004). His current research interests include wireless ad hoc, sensor and cellular networks. He is currently editor of several journals including Journal of Multiple-Valued Logic and Soft Comput-

ing, IEEE Transactions on Parallel and Distributed Systems, Parallel Processing Letters, and Parallel Algorithms and Applications. He guest edited recently special issues in several journals including IEEE Computer Magazine (February 2004), IEEE Networks, and Wireless Communications and Mobile Computing.



Fabián García-Nocetti received a degree in electrical engineering from the Autonomous National University of Mexico (UNAM) in 1984. He received the MSc and PhD degrees in parallel computer systems engineering from the University of Wales, Great Britain in 1988 and 1991 respectively. In 1992 he joined the Institute of Research in Applied Mathematics and Systems (IIMAS) at UNAM. He is currently a professor in the Computer Systems Engineering and Automation Department at IIMAS-UNAM and Director of IIMAS. His research interests

include algorithms and architectures for high performance computer systems in real time systems applications. He is a member of IEEE.



Julio Solano-González received his degree in electrical engineering from the Autonomous National University of Mexico (UNAM), in 1984. He received his IP Diploma in 1985 from the Philips International Institute of Technological Studies, Eindhoven, The Netherlands, and his PhD. degree in parallel computer systems engineering from the University of Wales, Great Britain in 1992. During the same year, he joined the Institute of Research in Applied Mathematics and Systems (IIMAS) at UNAM. He is currently Associate Pro-

fessor and Head of the Computer Systems Engineering and Automation Department at IIMAS-UNAM. His research interests include high performance computer algorithms and architectures, evolutionary computation for signal and image processing and wireless networks.